4.

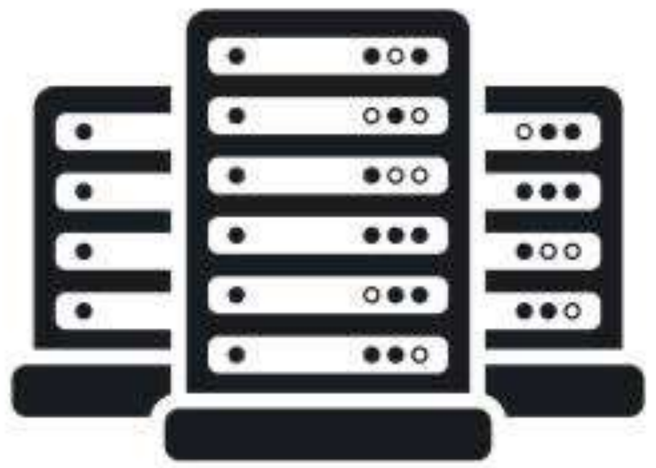
The plumbing challenges of hybrid modelling

The plumbing challenges of hybrid modelling

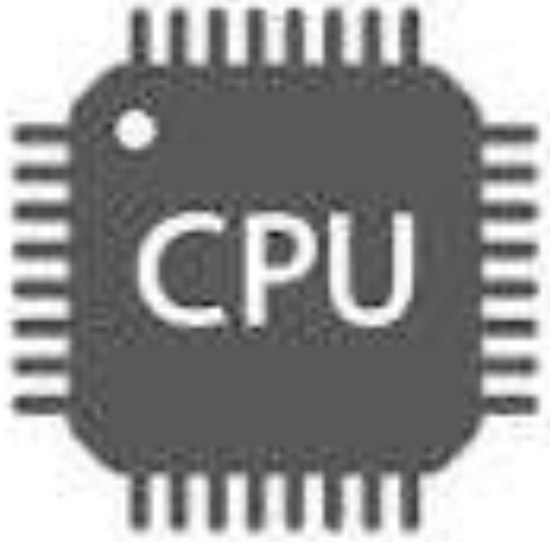
Interfacing ocean models with DL frameworks (1/3)

stable, robust, low abstraction languages

+



supercomputers

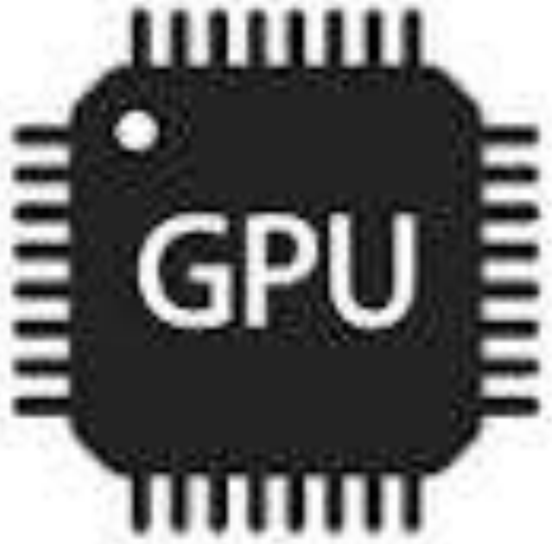


runs only on CPUs

high abstraction, fast evolving languages

+

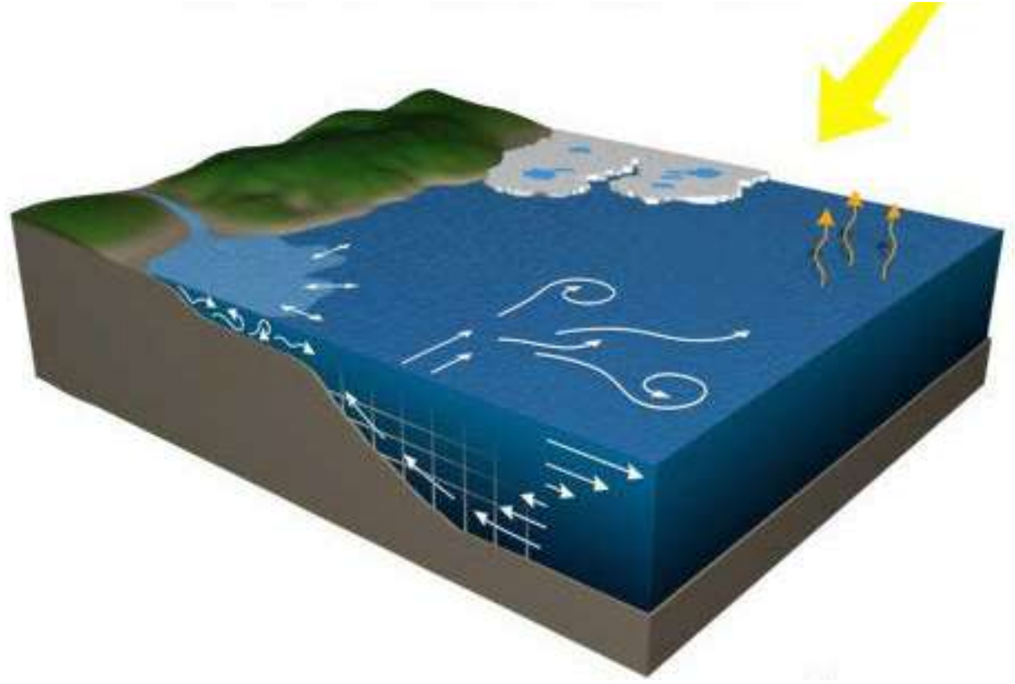
cloud ready

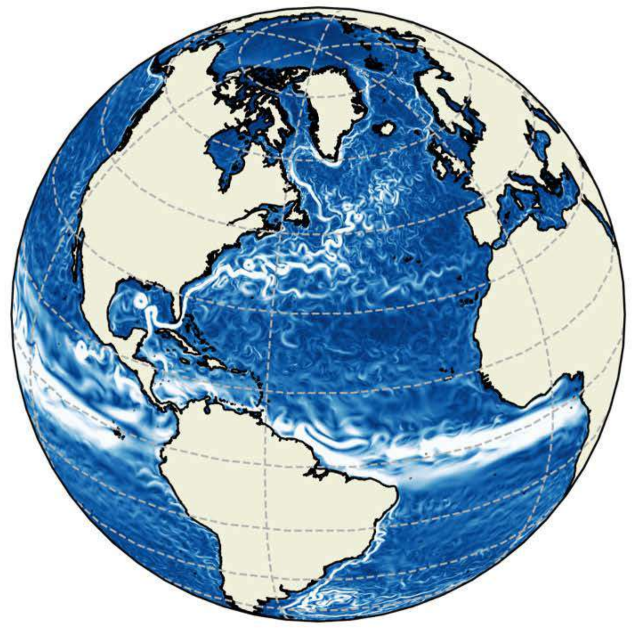


natively runs on GPUs

Interfacing ocean models with DL frameworks (2/3)

Ocean circulation models





NEMO

PyTorch

Input

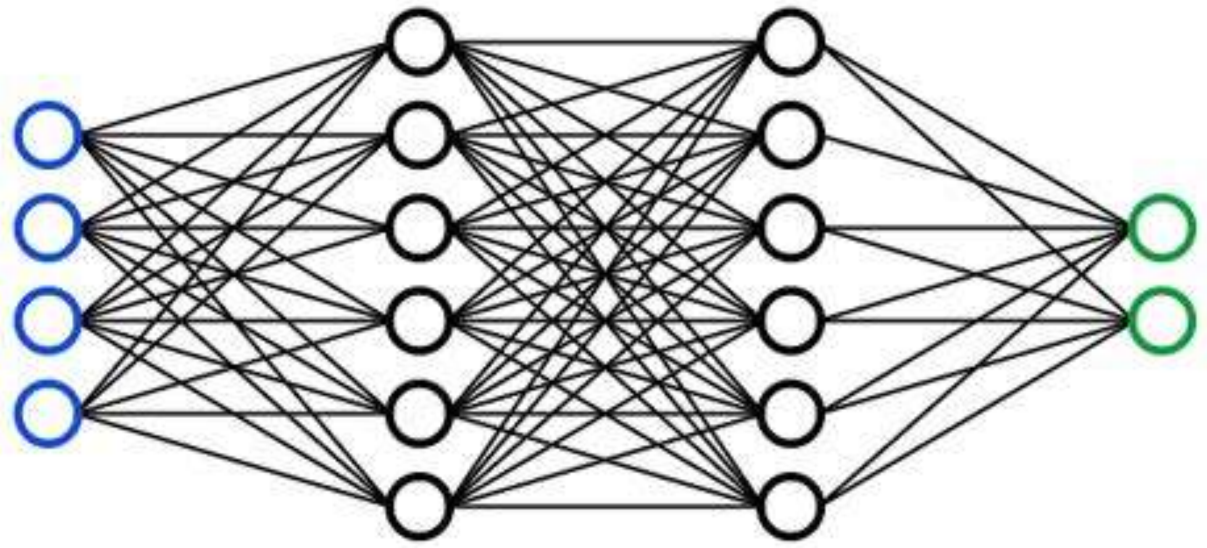
step n

+

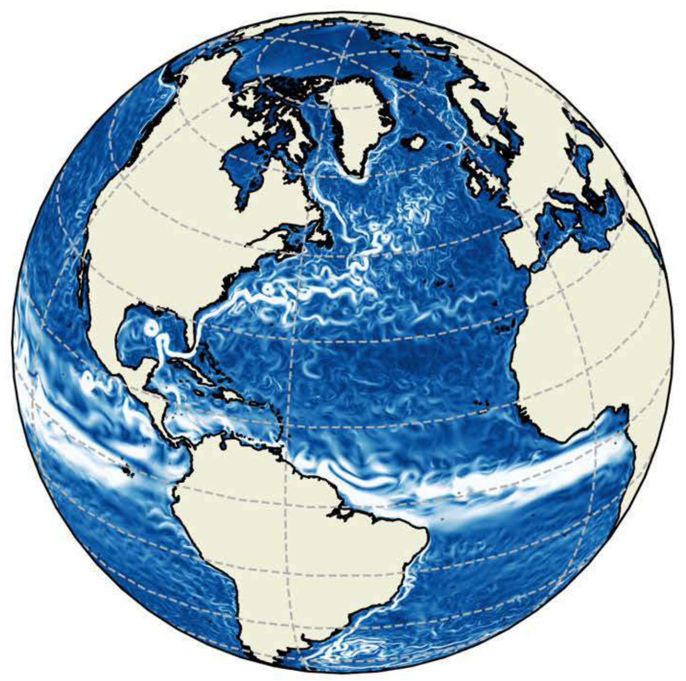
step n+1

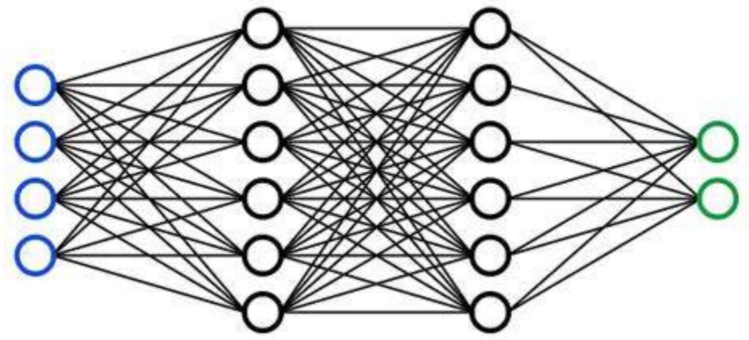
Output

Trainable components
(closures, error corrections)



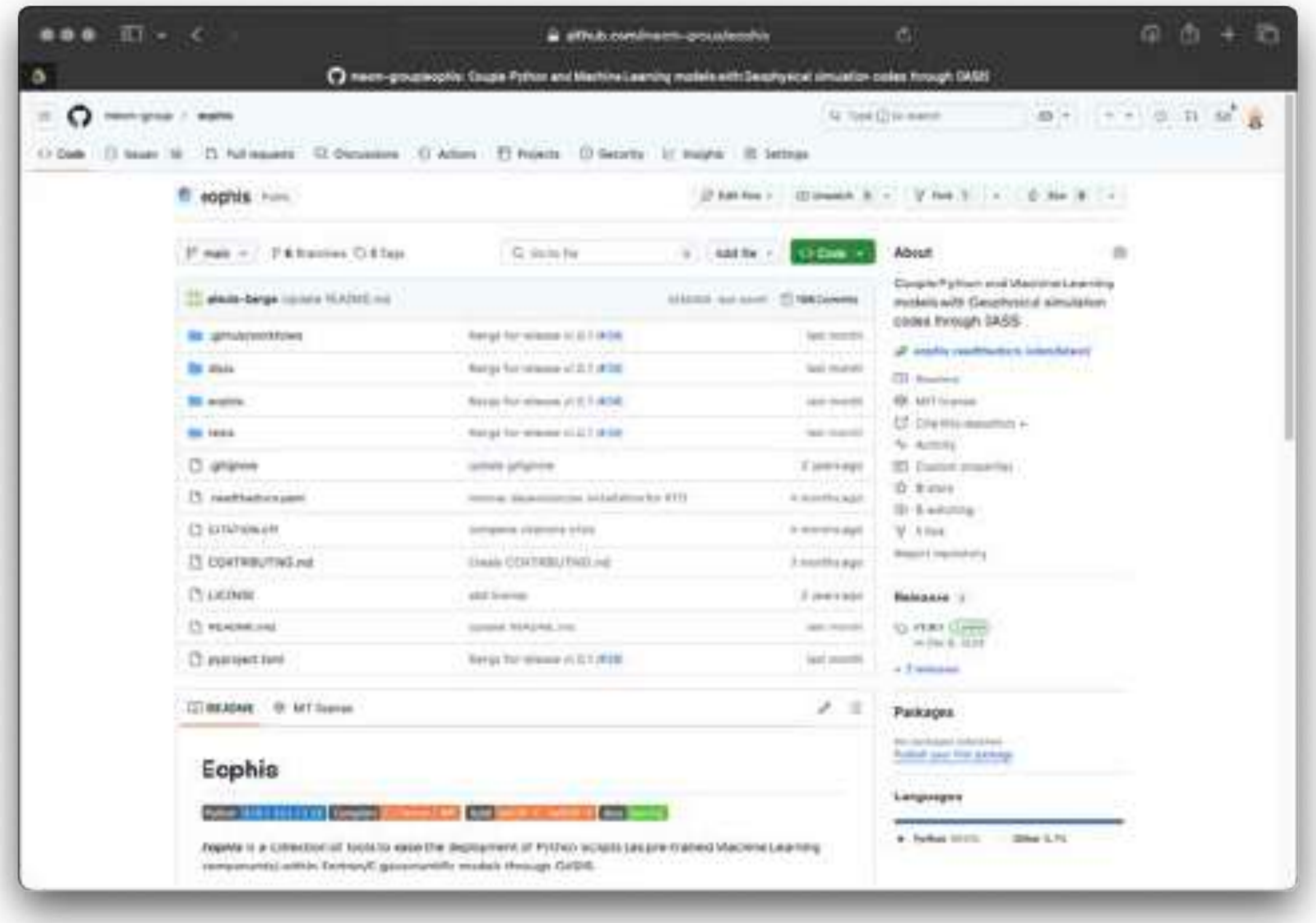
Interfacing ocean models with DL frameworks (3/3)





A. Barge

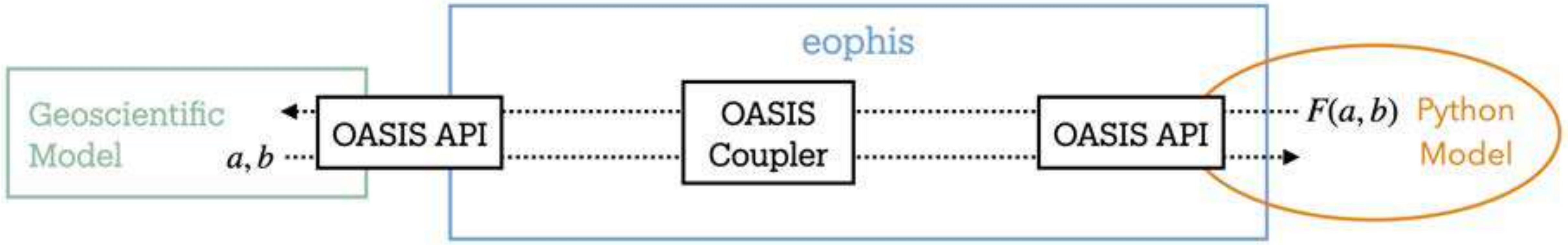
Work by Alexis Barge at IGE

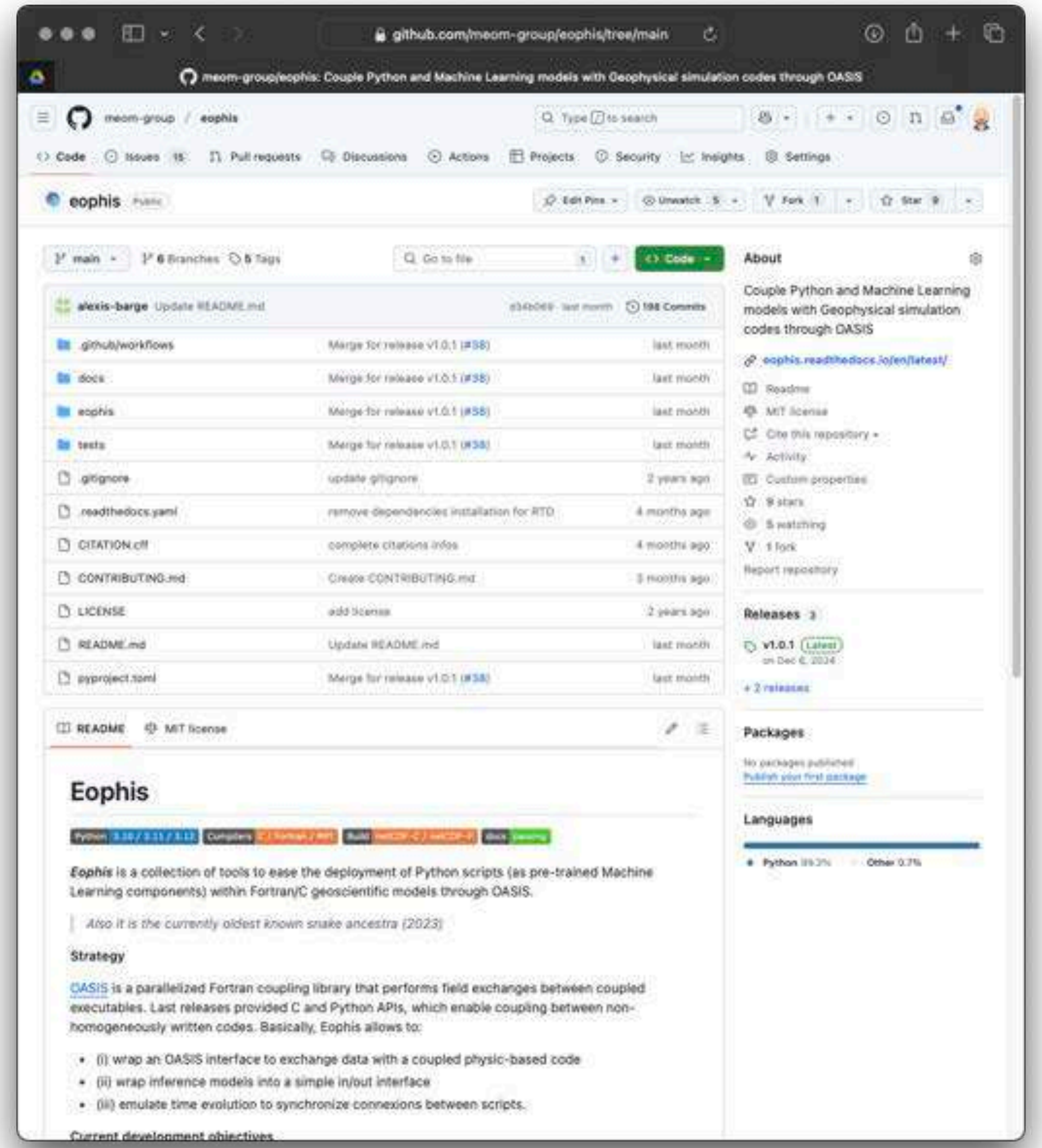


https://github.com/meom-group/eophis

- OASIS : exchange of 3D data between different codes
- Eophis : simplified deployment of ML models w/ OASIS
- Requires some change to the NEMO code
- Key : portability, domain decomposition

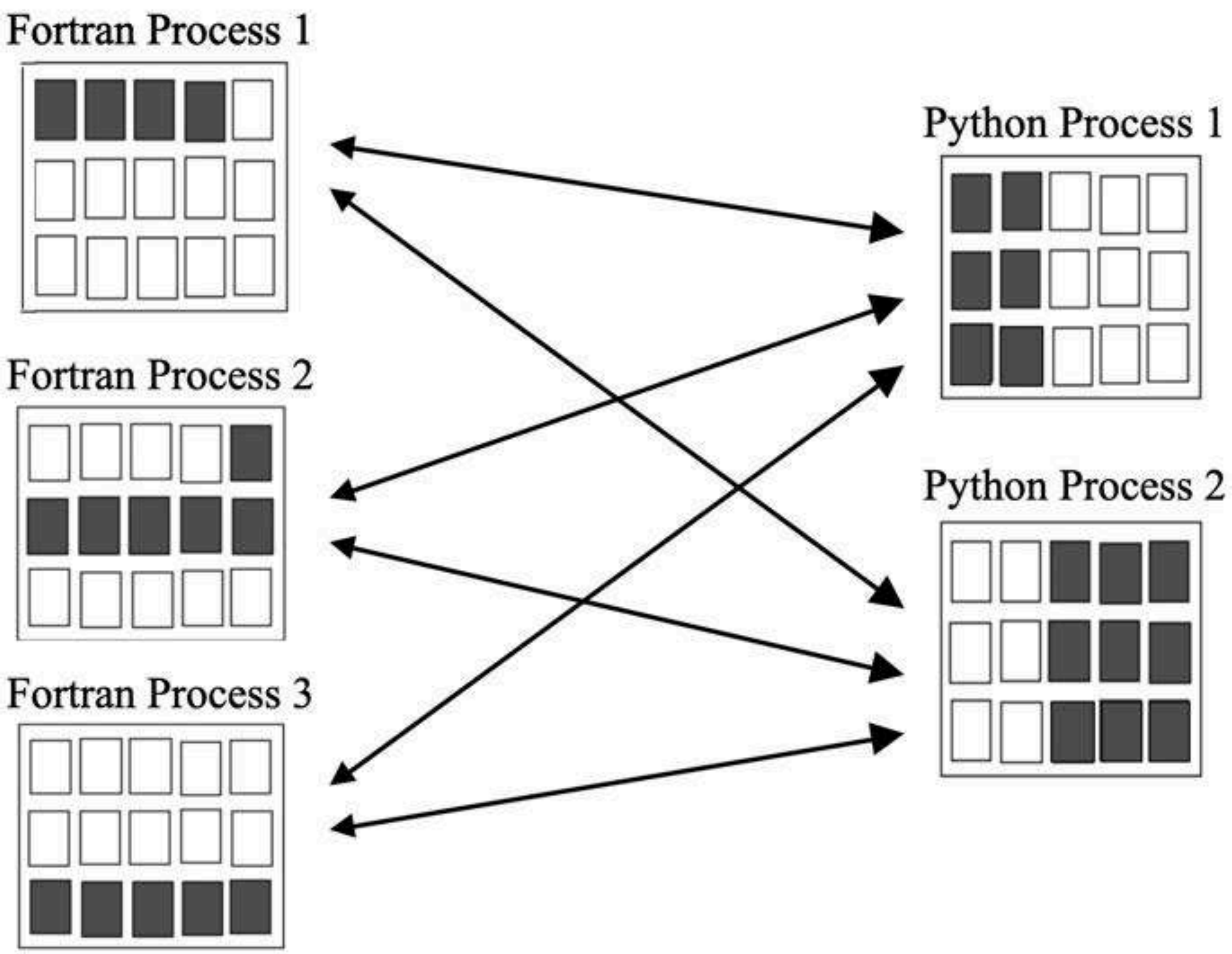
Interfacing ocean models with DL frameworks (3/3)





https://github.com/meom-group/eophis

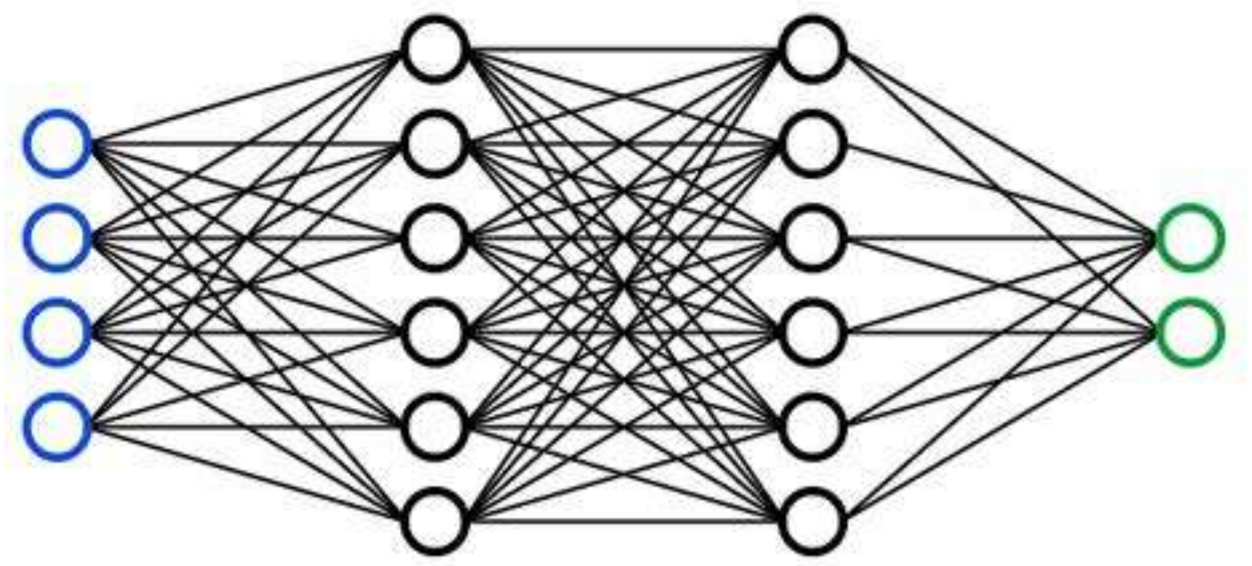
See eophis tutorial
next week



- OASIS : exchange of 3D data between different codes
- Eophis : simplified deployment of ML models w/ OASIS
- Requires some change to the NEMO code
- Key : portability, domain decomposition

The (real) challenge of online training (1/2)

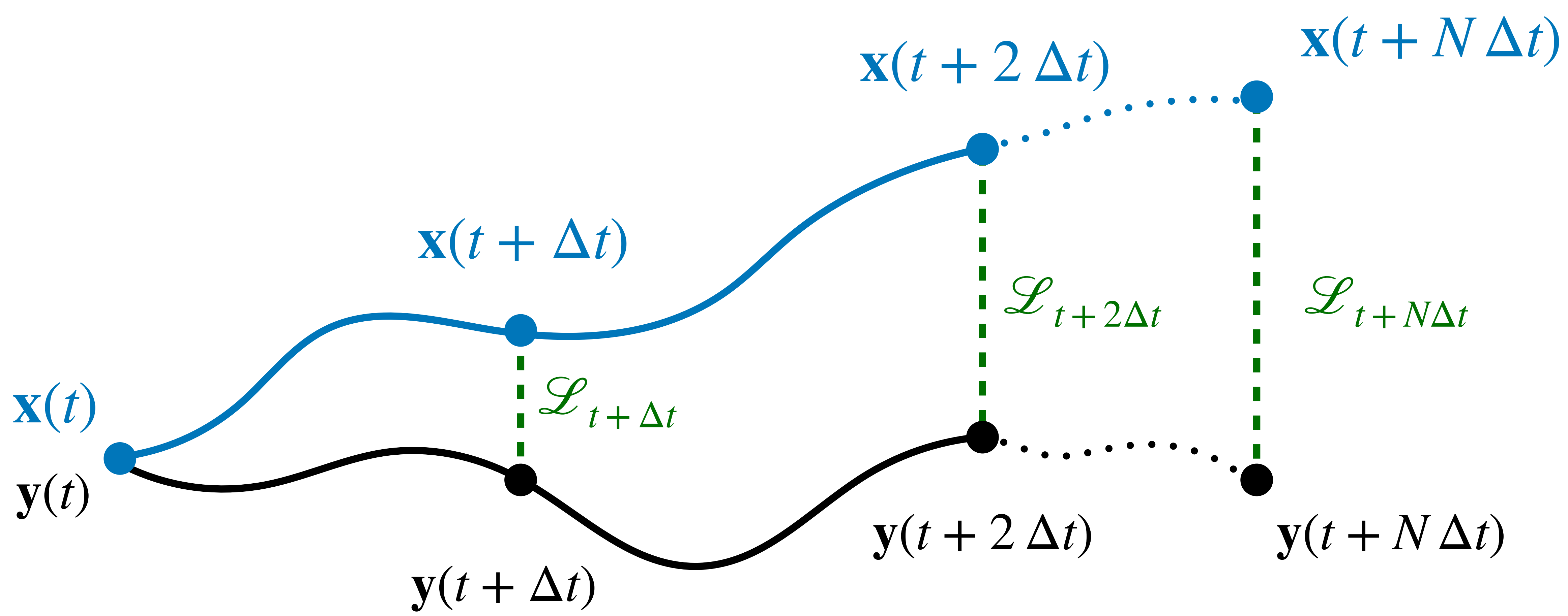
<u>offline</u> learning



mapping $\overline{\mathbf{x}} \rightarrow \overline{\mathcal{N}(\mathbf{x})}$

from pre-existing data

<u>online</u> learning



$$\partial_t \mathbf{y} + G(\mathbf{y}) + \mathscr{M}_{NN}(\mathbf{y}) = f$$

along a trajectory

(a.k.a : a posteriori, solver-in-the-loop, end-to-end, auto-regressive roll-outs)

Online training improves performance, stability, generalisation *Frezat et al. 2022; List et al. 2024*

The (real) challenge of online training (2/2)

target prediction

$$\arg\min_{\theta} \mathcal{L}(\mathbf{z}, \mathcal{M}(\mathbf{y} \mid \theta))$$

$$\frac{\partial \mathcal{L}}{\partial \theta}(\mathbf{z}, \mathcal{M}(\mathbf{y} \mid \theta)) = \frac{\partial \mathcal{M}}{\partial \theta}(\mathbf{y} \mid \theta)\frac{\partial \mathcal{L}}{\partial \mathcal{M}}$$

gradient of the loss

For time evolving problems, with

$$\mathbf{y}(t + \Delta t) = E_m \circ \cdots \circ E_1(\mathbf{y}(t))$$

$\mathcal{M} \equiv E$ Auto-regressive operator (time)

The gradient of the loss involves

$$\frac{\partial \mathcal{M}}{\partial \theta} \equiv \frac{\partial E}{\partial \theta} = \frac{\partial (E_m \circ \cdots \circ E_1)}{\partial \theta} = \frac{\partial E_m}{\partial E_{m-1}} \cdots \frac{\partial E_2}{\partial E_1}\frac{\partial E_1}{\partial \theta}$$

tricky without Automatic Differenciation (AD) !

The challenge of online training strategies (2/2)

target prediction

$$\arg\min_{\theta} \mathscr{L}(\mathbf{z}, \mathscr{M}(\mathbf{y} \mid \theta))$$

AD is readily available in some language

For time evolving problems, with

$$\mathbf{y}(t + \Delta t) = E_m \circ \cdots \circ E_1(\mathbf{y}(t))$$

$\mathscr{M} \equiv E$ temporal evolution operator

The gradient of the loss involves

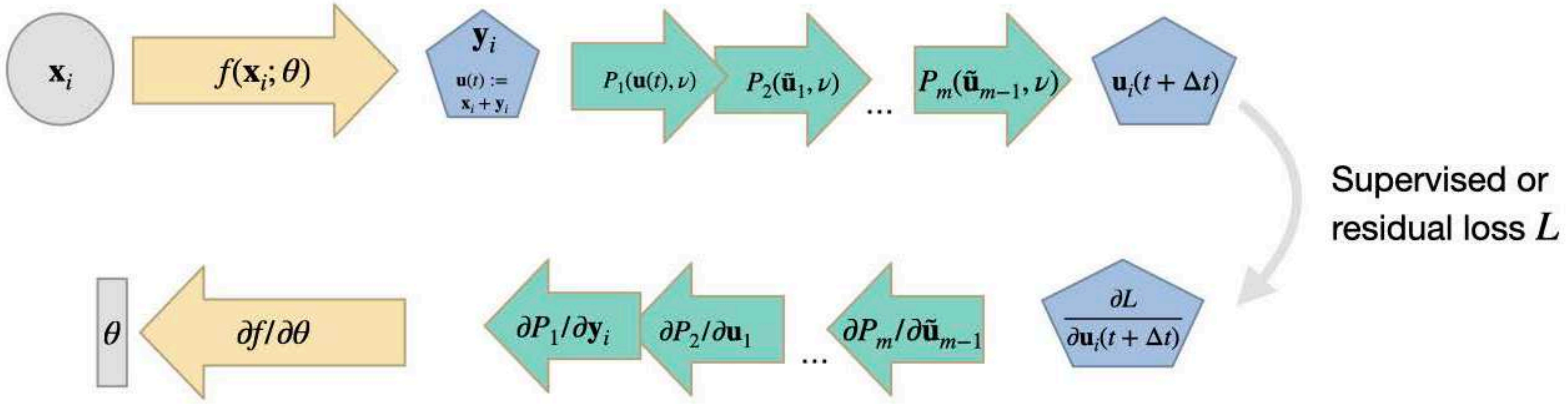
$$\frac{\partial \mathscr{M}}{\partial \theta} \equiv \frac{\partial E}{\partial \theta} = \frac{\partial (E_m \circ \cdots \circ E_1)}{\partial \theta} = \frac{\partial E_m}{\partial E_{m-1}} \cdots \frac{\partial E_2}{\partial E_1} \frac{\partial E_1}{\partial \theta}$$

tricky without Automatic Differenciation (AD) !

The challenge of online training strategies (2/2)

target prediction

$$\underset{\theta}{\arg\min}\ \mathscr{L}(\mathbf{z}, \mathscr{M}(\mathbf{y} \mid \theta))$$



See eg Sapienza et al. 2024
https://arxiv.org/abs/2406.09699

AD is readily available in some language

But AD used yet in climate models…

Differentiable programming

- programs composed of differentiable building blocks
- building blocks : trainable and procedural code components
- trainable end-to-end with gradient based optimisation

AI-native hybrid geoscientific models


Article

Neural general circulation models for weather and climate

https://doi.org/10.1038/s41586-024-07744-y
Received: 13 November 2023
Accepted: 15 June 2024
Published online: 22 July 2024
Open access
Check for updates

Dmitrii Kochkov[1,6,✉], Janni Yuval[1,6,✉], Ian Langmore[1,6], Peter Norgaard[1,6], Jamie Smith[1,6], Griffin Mooers[1], Milan Klöwer[2], James Lottes[1], Stephan Rasp[1], Peter Düben[3], Sam Hatfield[3], Peter Battaglia[4], Alvaro Sanchez-Gonzalez[4], Matthew Willson[4], Michael P. Brenner[1,5] & Stephan Hoyer[1,6,✉]

General circulation models (GCMs) are the foundation of weather and climate prediction[1,2]. GCMs are physics-based simulators that combine a numerical solver for large-scale dynamics with tuned representations for small-scale processes such as cloud formation. Recently, machine-learning models trained on reanalysis data have achieved comparable or better skill than GCMs for deterministic weather forecasting[3,4]. However, these models have not demonstrated improved ensemble forecasts, or shown sufficient stability for long-term weather and climate simulations. Here we present a GCM that combines a differentiable solver for atmospheric dynamics with machine-learning components and show that it can generate forecasts of deterministic weather, ensemble weather and climate on par with the best machine-learning and physics-based methods. NeuralGCM is competitive with machine-learning models for one- to ten-day forecasts, and with the European Centre for Medium-Range Weather Forecasts ensemble prediction for one- to fifteen-day forecasts. With prescribed sea surface temperature, NeuralGCM can accurately track climate metrics for multiple decades, and climate forecasts with 140-kilometre resolution show emergent phenomena such as realistic frequency and trajectories of tropical cyclones. For both weather and climate, our approach offers orders of magnitude computational savings over conventional GCMs, although our model does not extrapolate to substantially different future climates. Our results show that end-to-end deep learning is compatible with tasks performed by conventional GCMs and can enhance the large-scale physical simulations that are essential for understanding and predicting the Earth system.

Solving the equations for Earth's atmosphere with general circulation models (GCMs) is the basis of weather and climate prediction[1,2]. Over the past 70 years, GCMs have been steadily improved with better numerical methods and more detailed physical models, while exploiting faster computers to run at higher resolution. Inside GCMs, the unresolved physical processes such as clouds, radiation and precipitation are represented by semi-empirical parameterizations. Tuning GCMs to match historical data remains a manual process[5], and GCMs retain many persistent errors and biases[6–8]. The difficulty of reducing uncertainty in long-term climate projections[9] and estimating distributions of extreme weather events[10] presents major challenges for climate mitigation and adaptation[11].

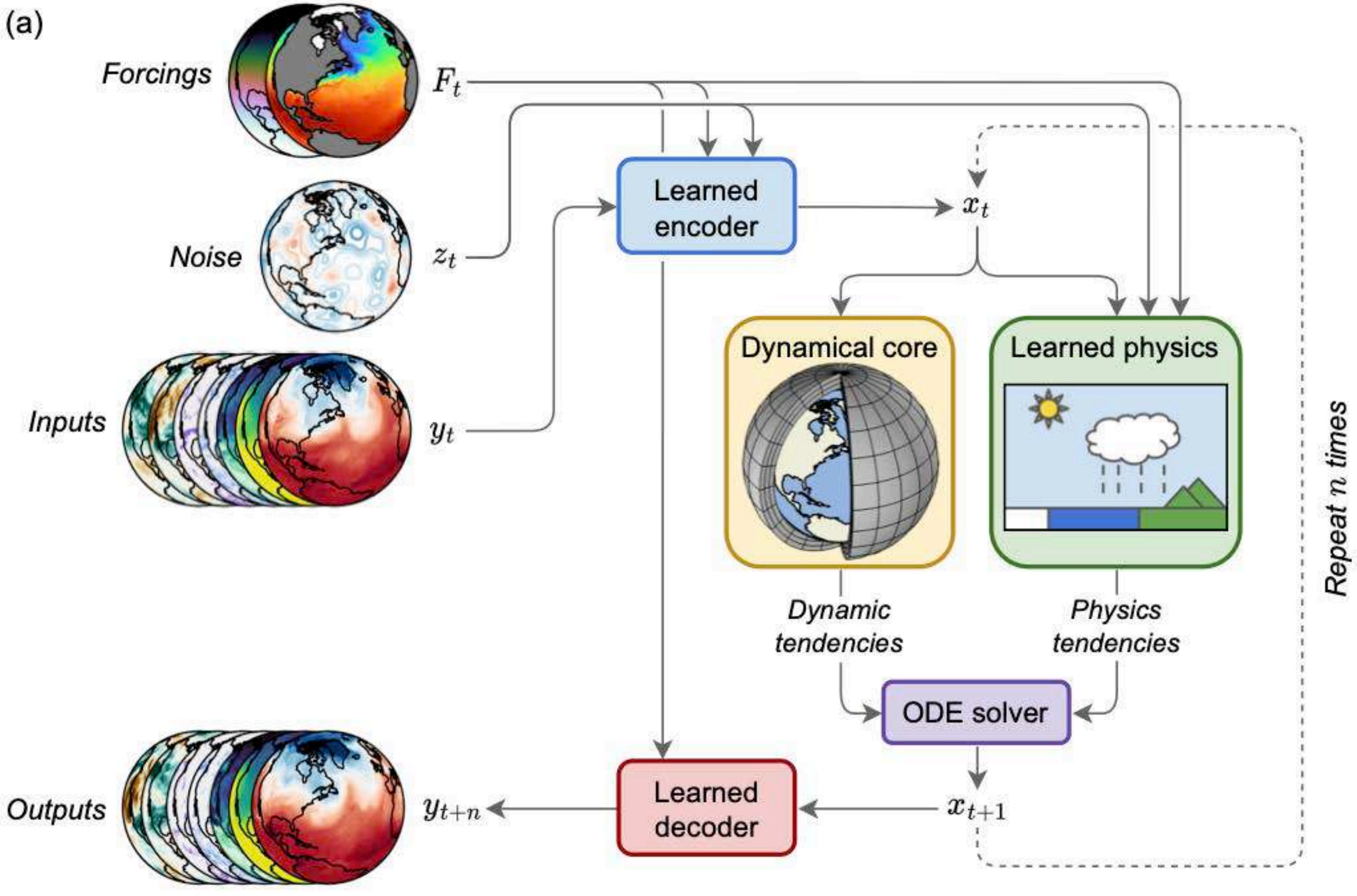
Recent advances in machine learning have presented an alternative for weather forecasting[3,4,12,13]. These models rely solely on machine-learning techniques, using roughly 40 years of historical data from the European Center for Medium-Range Weather Forecasts (ECMWF) reanalysis v5 (ERA5)[14] for model training and forecast initialization. Machine-learning methods have been remarkably successful, demonstrating state-of-the-art deterministic forecasts for 1- to 10-day weather prediction at a fraction of the computational cost of traditional models[3,4]. Machine-learning atmospheric models also require considerably less code, for example GraphCast[3] has 5,417 lines versus 376,578 lines for the National Oceanic and Atmospheric Administration's FV3 atmospheric model[15] (see Supplementary Information section A for details).

Nevertheless, machine-learning approaches have noteworthy limitations compared with GCMs. Existing machine-learning models have focused on deterministic prediction, and surpass deterministic numerical weather prediction in terms of the aggregate metrics for which they are trained[3,4]. However, they do not produce calibrated uncertainty estimates[4], which is essential for useful weather forecasts[1]. Deterministic machine-learning models using a mean-squared-error loss are rewarded for averaging over uncertainty, producing unrealistically blurry predictions when optimized for multi-day forecasts[3,13]. Unlike physical models, machine-learning models misrepresent derived (diagnostic) variables such as geostrophic wind[16]. Furthermore,

[1]Google Research, Mountain View, CA, USA. [2]Earth, Atmospheric and Planetary Sciences, Massachusetts Institute of Technology, Cambridge, MA, USA. [3]European Centre for Medium-Range Weather Forecasts, Reading, UK. [4]Google DeepMind, London, UK. [5]School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. [6]These authors contributed equally: Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Stephan Hoyer. ✉e-mail: dkochkov@google.com; jannyuval@google.com; shoyer@google.com

https://doi.org/10.1038/s41586-024-07744-y

Kochkov et al. (2024)



https://github.com/google-research/dinosaur

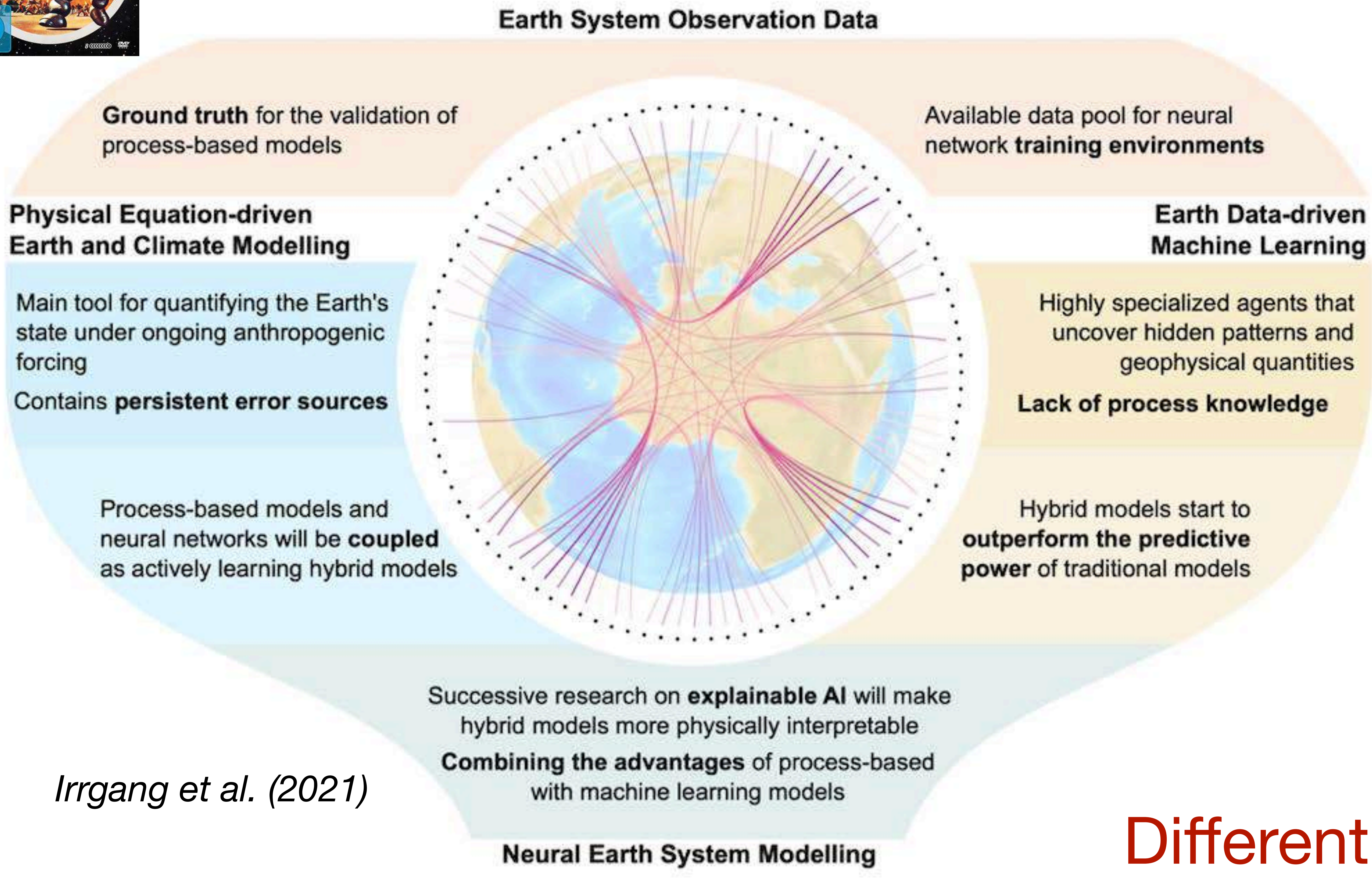
https://github.com/google-research/neuralgcm

5.

Towards AI-native hybrid climate models ?

Towards AI-native hybrid climate models ?

AI-native hybrid climate models ?



Irrgang et al. (2021)

Betting harnessing **observations** & hi-fidelity **simulations**

… for optimising

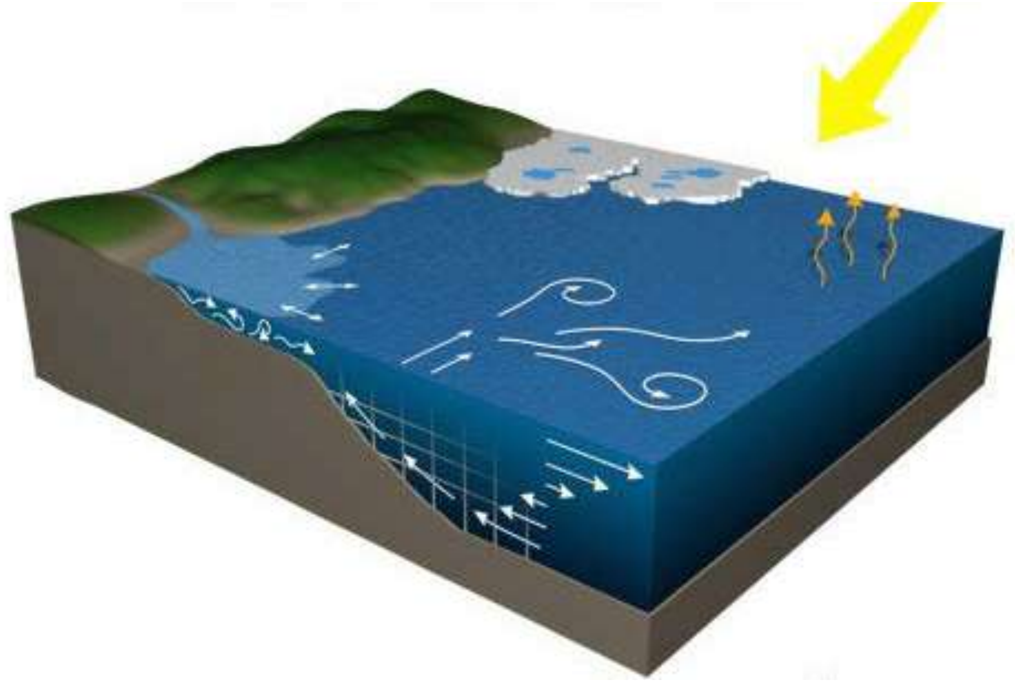
- model **parameters**
- **numerical** schemes
- subgrid **closures**
- …

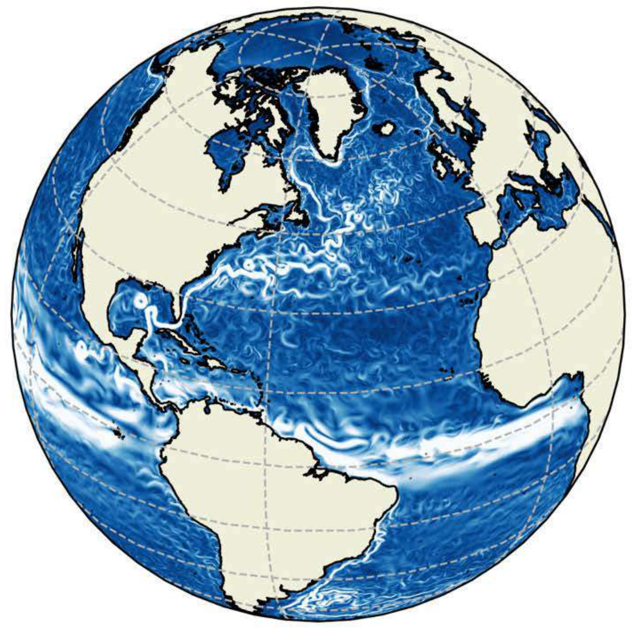
Differentiable programming in earth system models ?

Current generation hybrid models

Input

Ocean circulation models





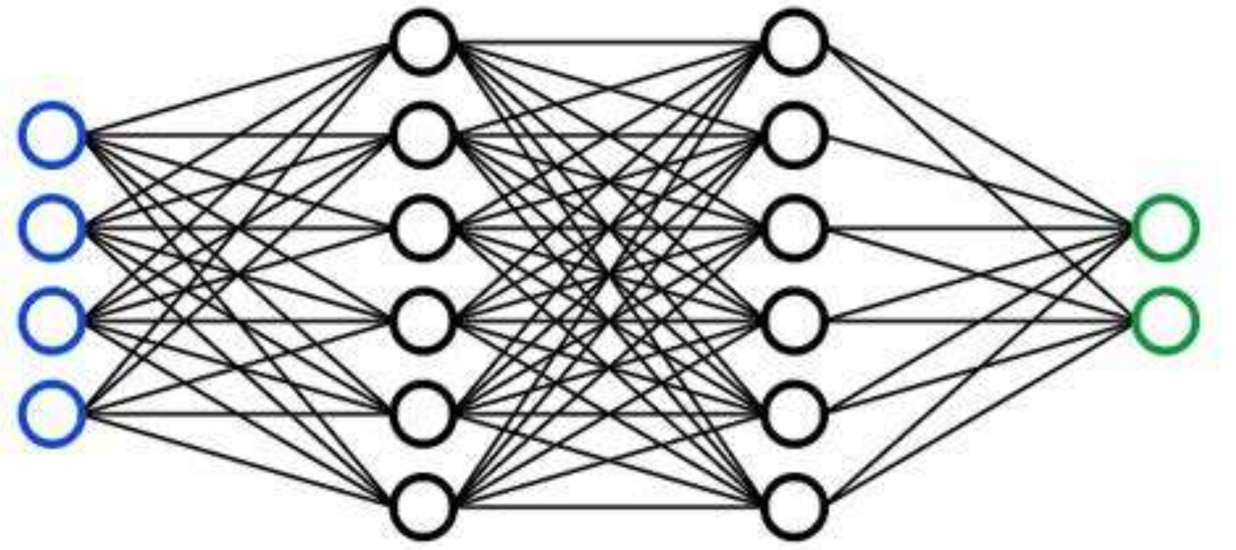
step n

+

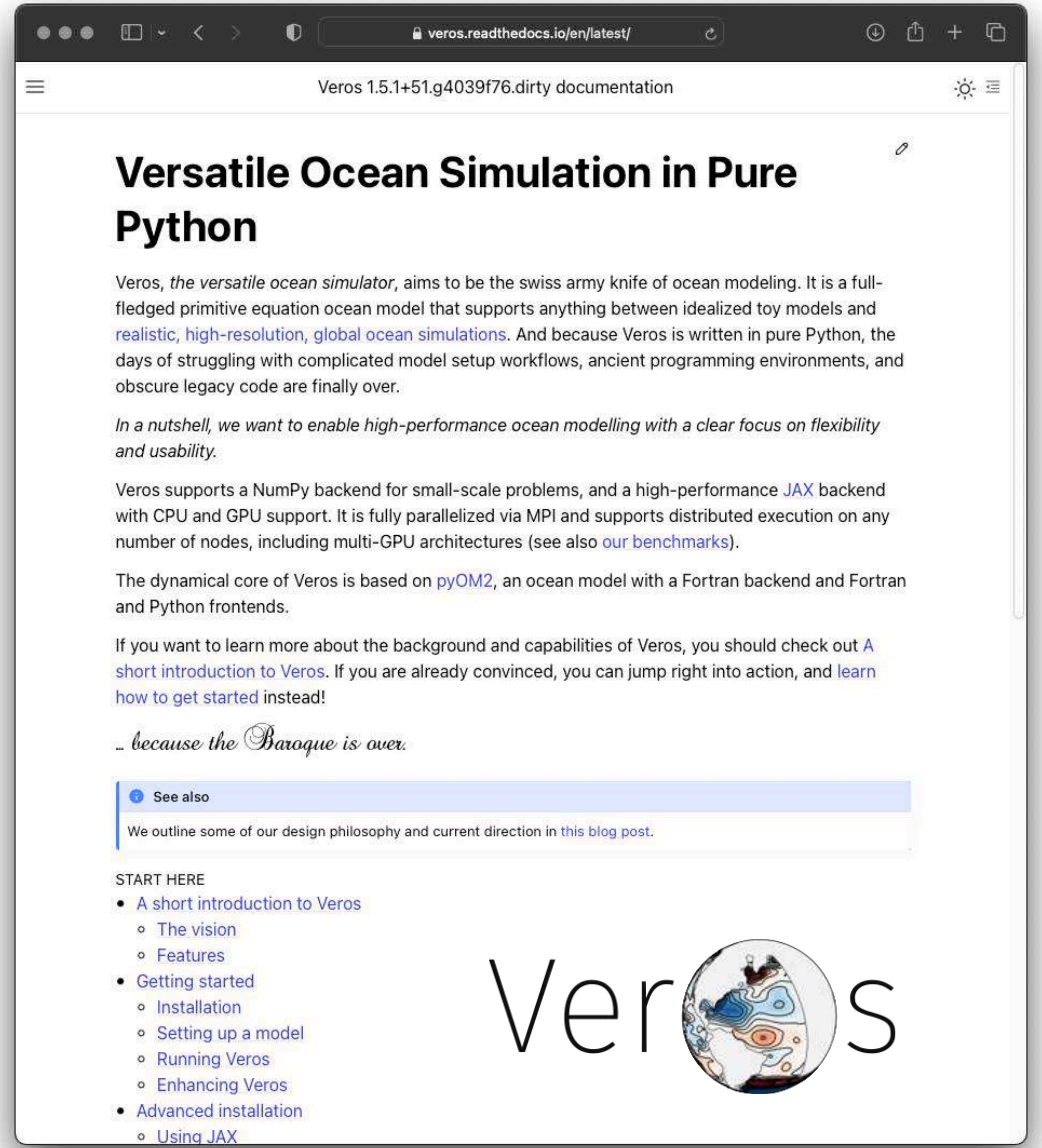
step n+1

Output

Trainable components
(closures, error corrections)

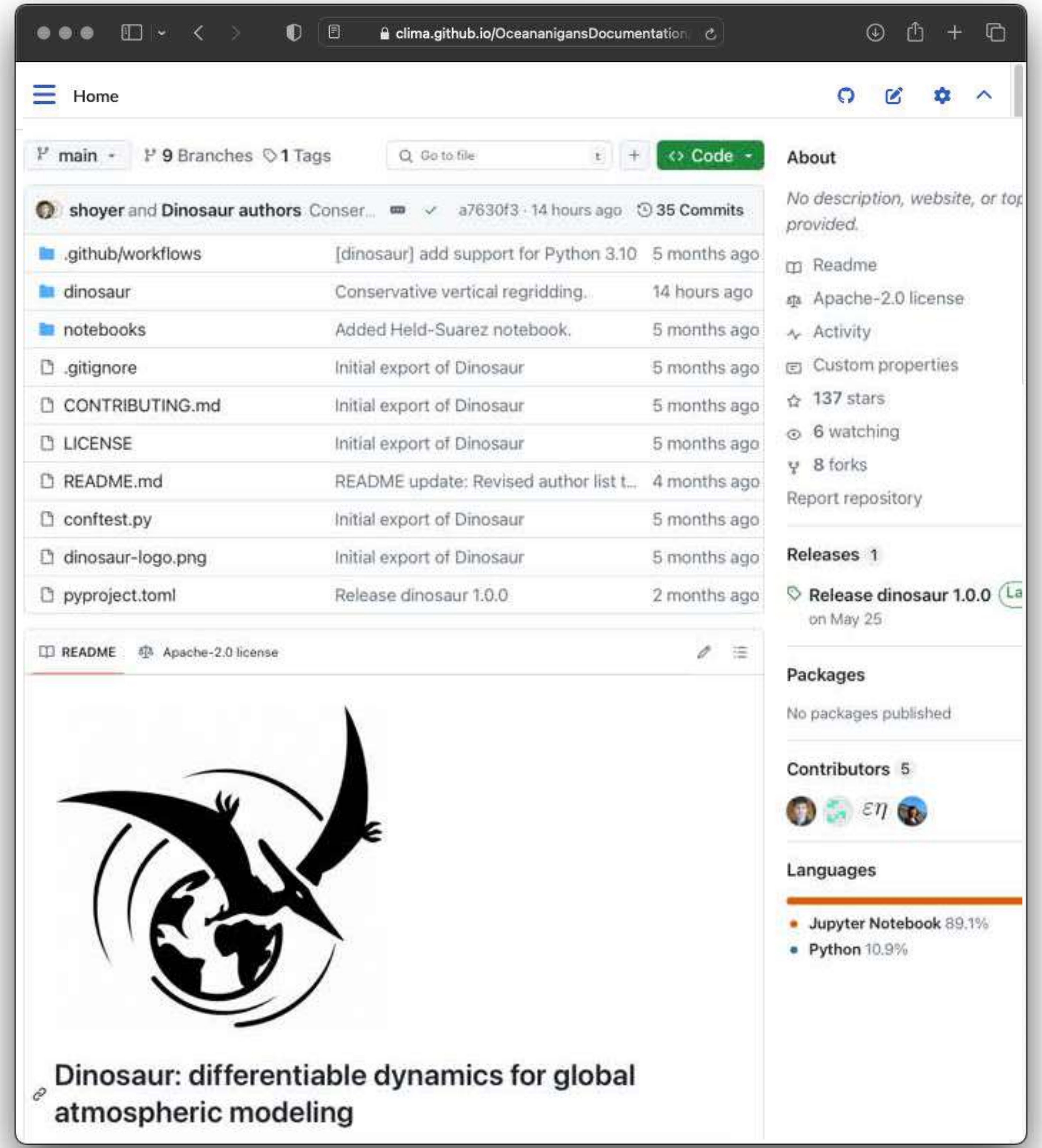


A new generation of geoscientific models



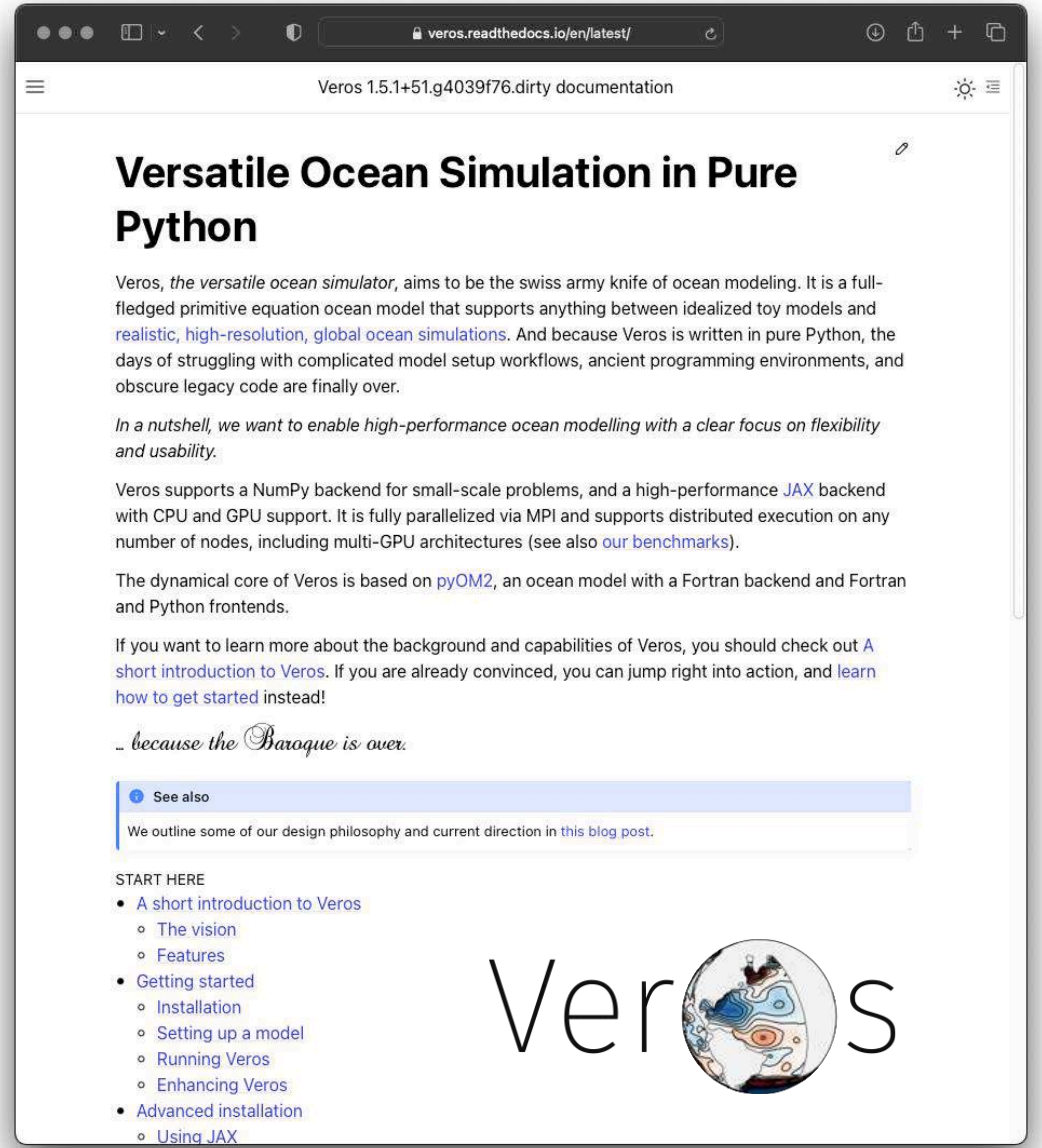
Atmos →

← Ocean



Modern code and compute : simple to write, scales, runs on any hardware

A new generation of geoscientific models


veros.readthedocs.io/en/latest/

Veros 1.5.1+51.g4039f76.dirty documentation

Versatile Ocean Simulation in Pure Python

Veros, *the versatile ocean simulator*, aims to be the swiss army knife of ocean modeling. It is a full-fledged primitive equation ocean model that supports anything between idealized toy models and realistic, high-resolution, global ocean simulations. And because Veros is written in pure Python, the days of struggling with complicated model setup workflows, ancient programming environments, and obscure legacy code are finally over.

In a nutshell, we want to enable high-performance ocean modelling with a clear focus on flexibility and usability.

Veros supports a NumPy backend for small-scale problems, and a high-performance JAX backend with CPU and GPU support. It is fully parallelized via MPI and supports distributed execution on any number of nodes, including multi-GPU architectures (see also our benchmarks).

The dynamical core of Veros is based on pyOM2, an ocean model with a Fortran backend and Fortran and Python frontends.

If you want to learn more about the background and capabilities of Veros, you should check out A short introduction to Veros. If you are already convinced, you can jump right into action, and learn how to get started instead!

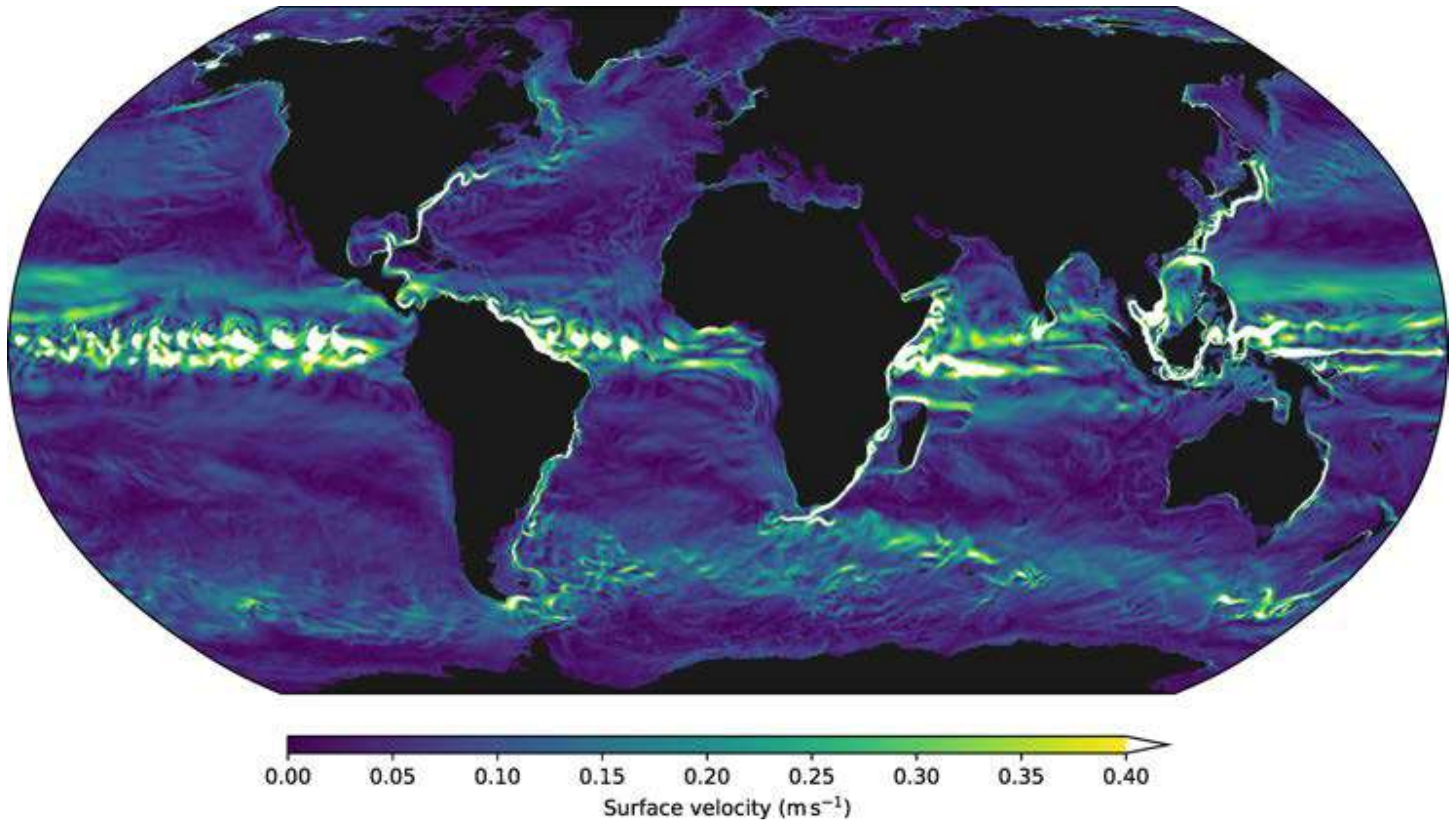
… because the Baroque is over.

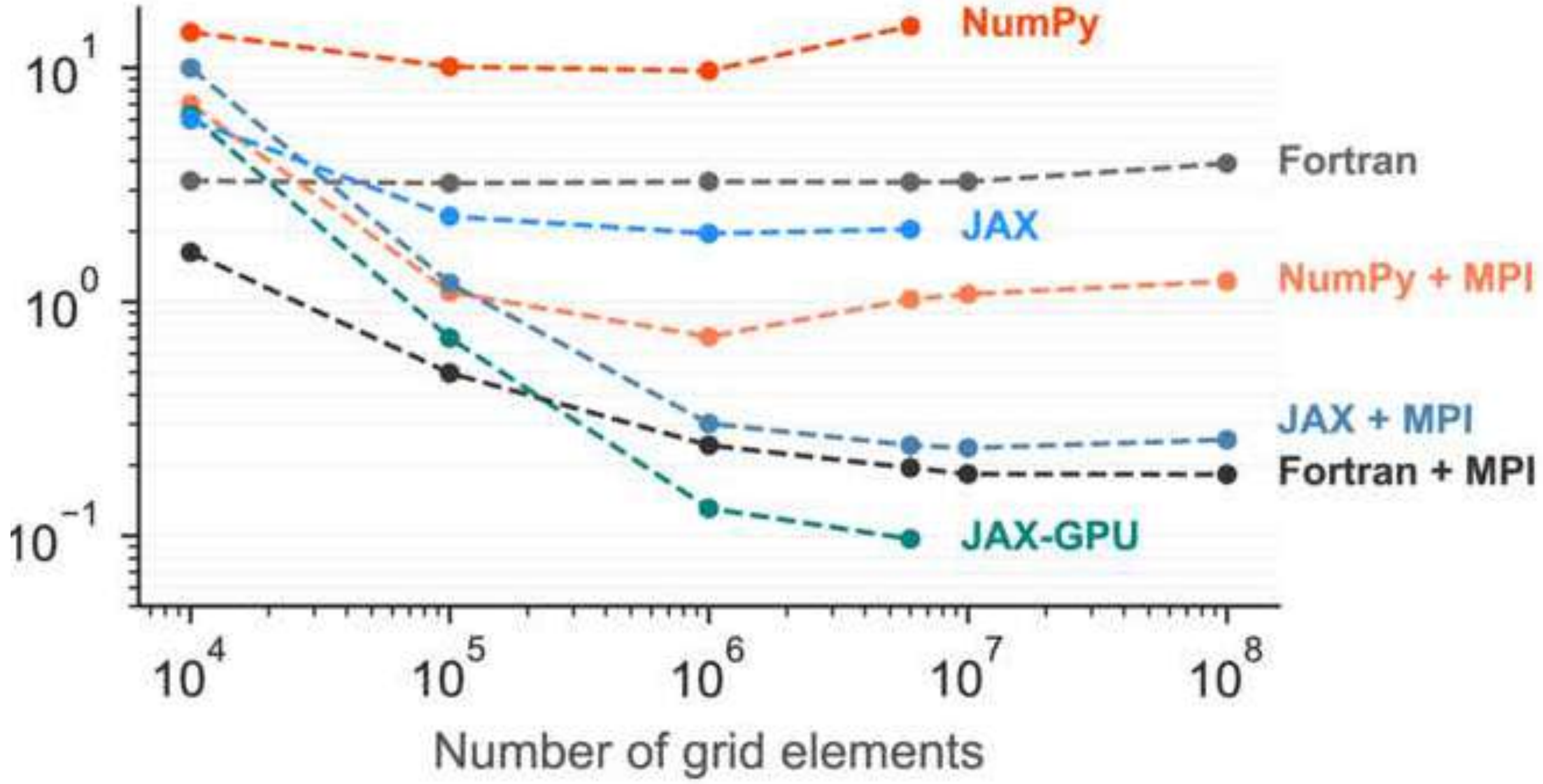
See also

We outline some of our design philosophy and current direction in this blog post.

START HERE

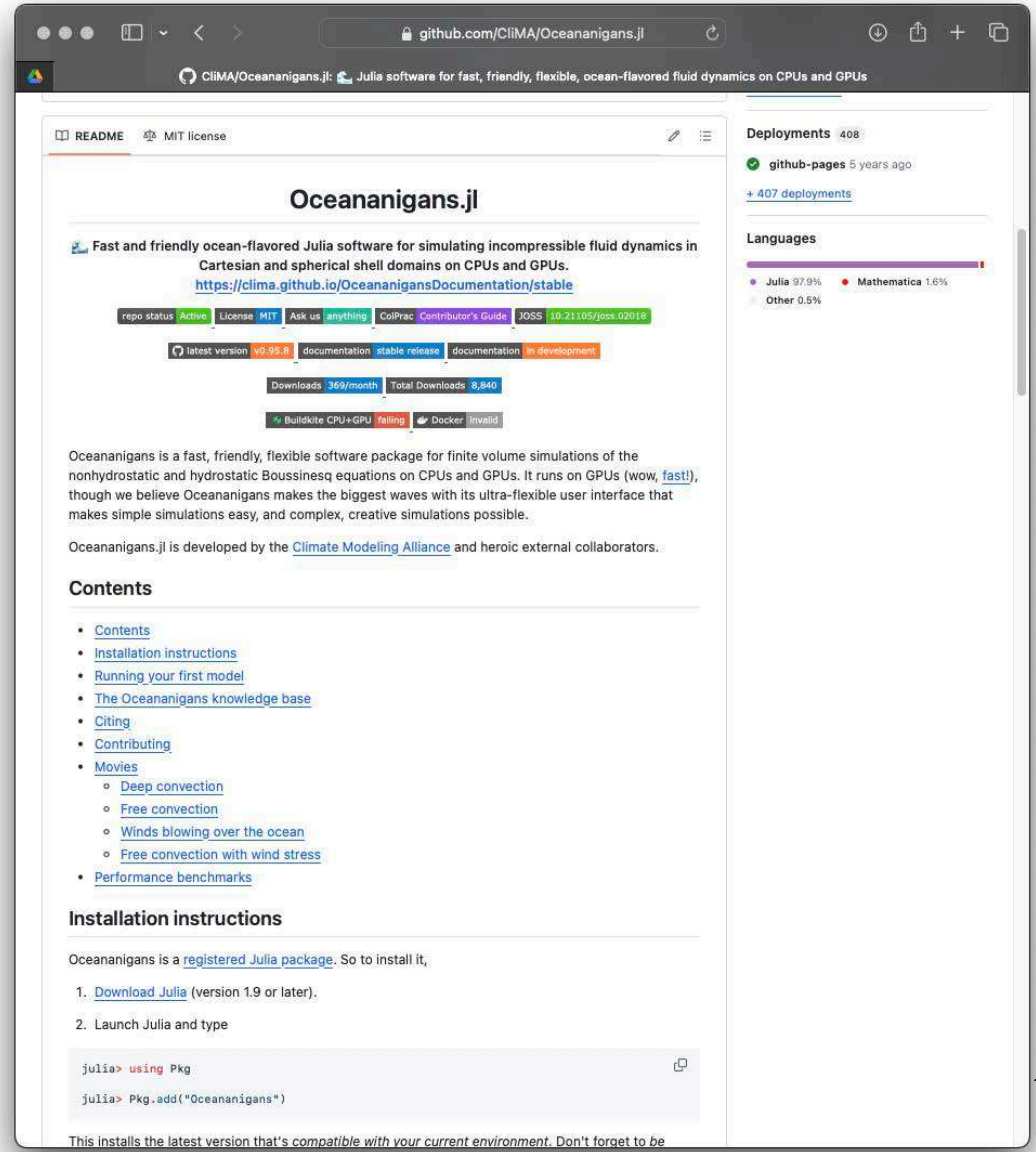
- A short introduction to Veros
  - The vision
  - Features
- Getting started
  - Installation
  - Setting up a model
  - Running Veros
  - Enhancing Veros
- Advanced installation
  - Using JAX





Modern code and compute : simple to write, scales, runs on any hardware

A new generation of geoscientific models

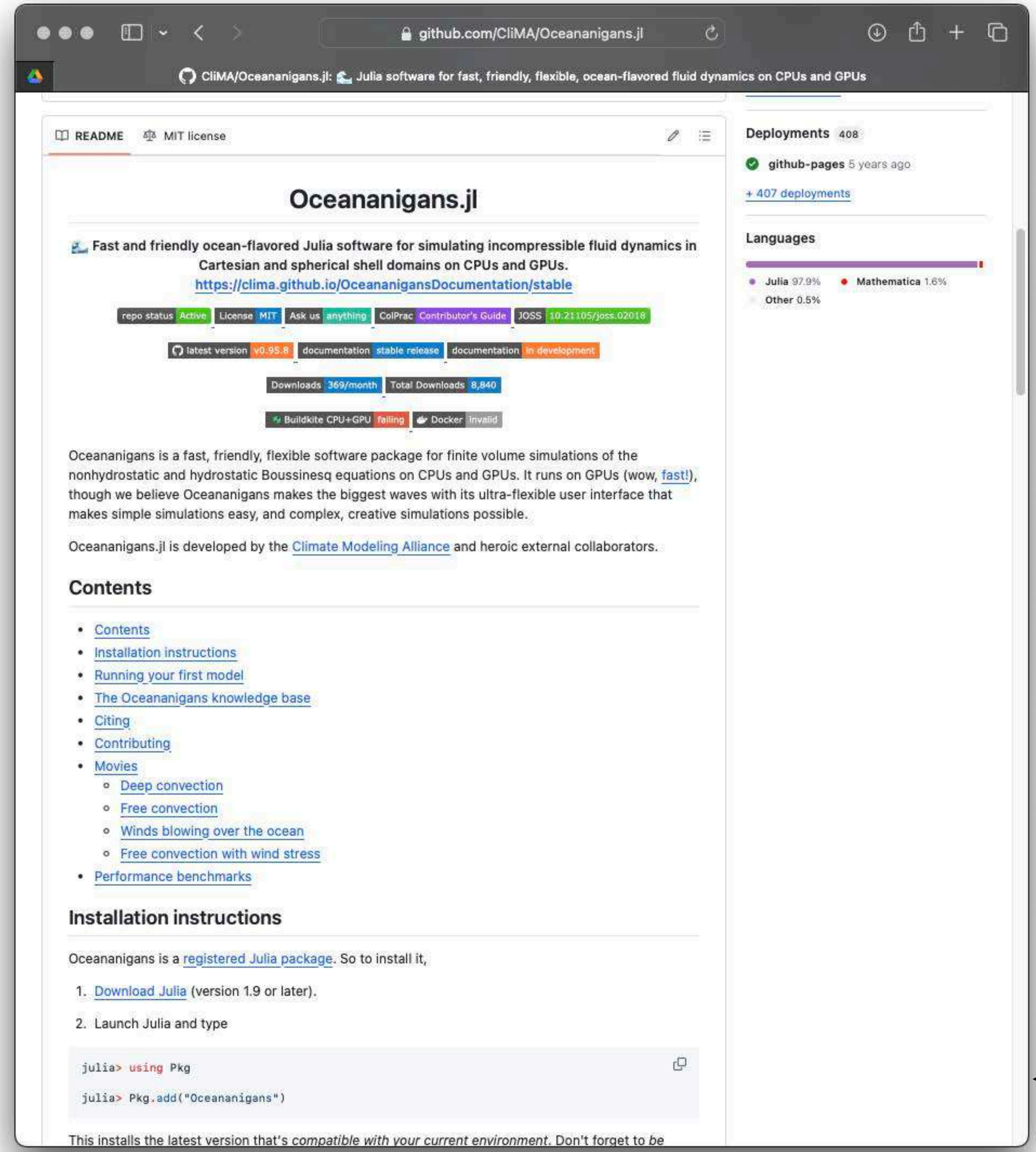


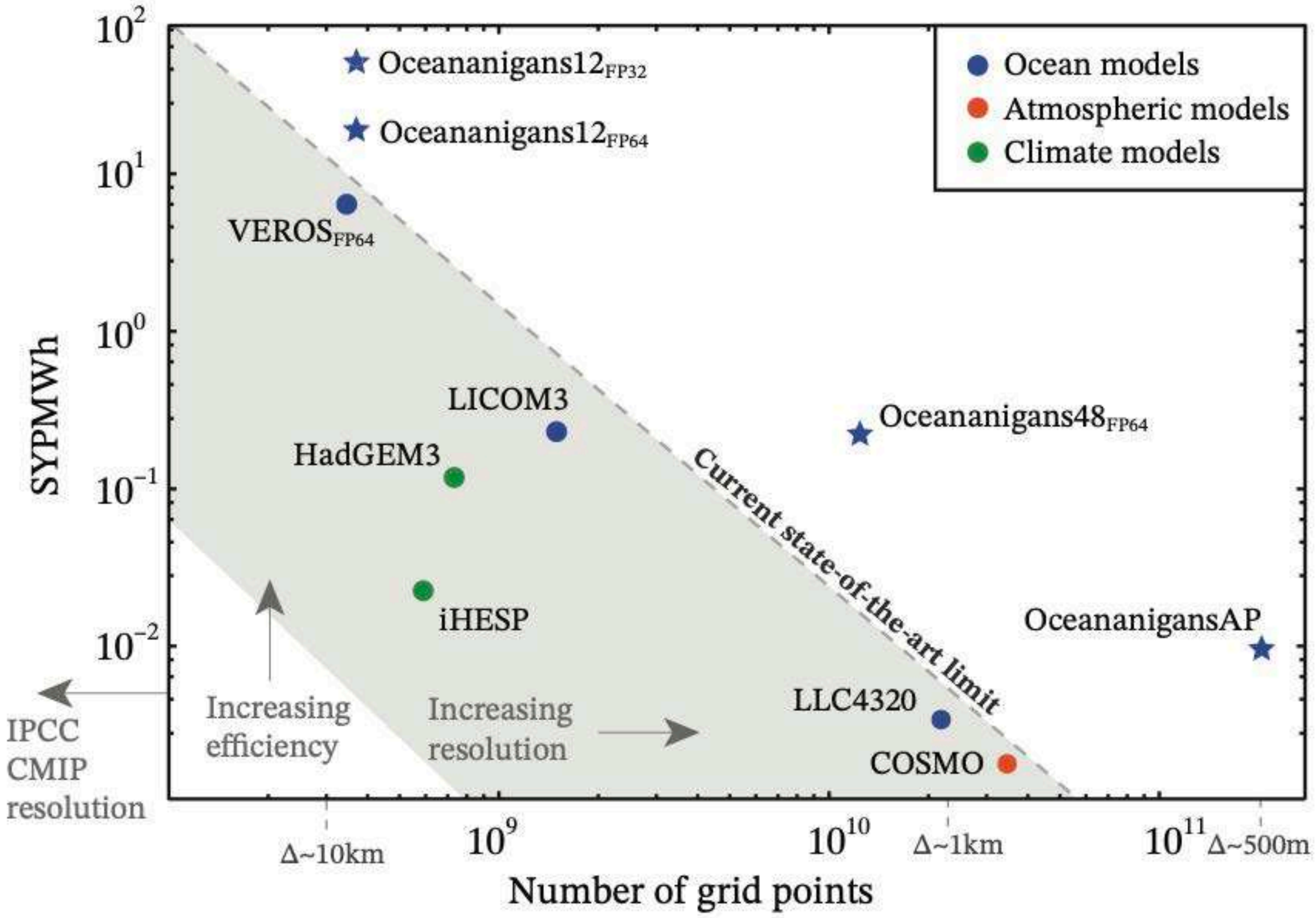
julia

Ocean

Modern code and compute : simple to write, scales, runs on any hardware

A new generation of geoscientific models

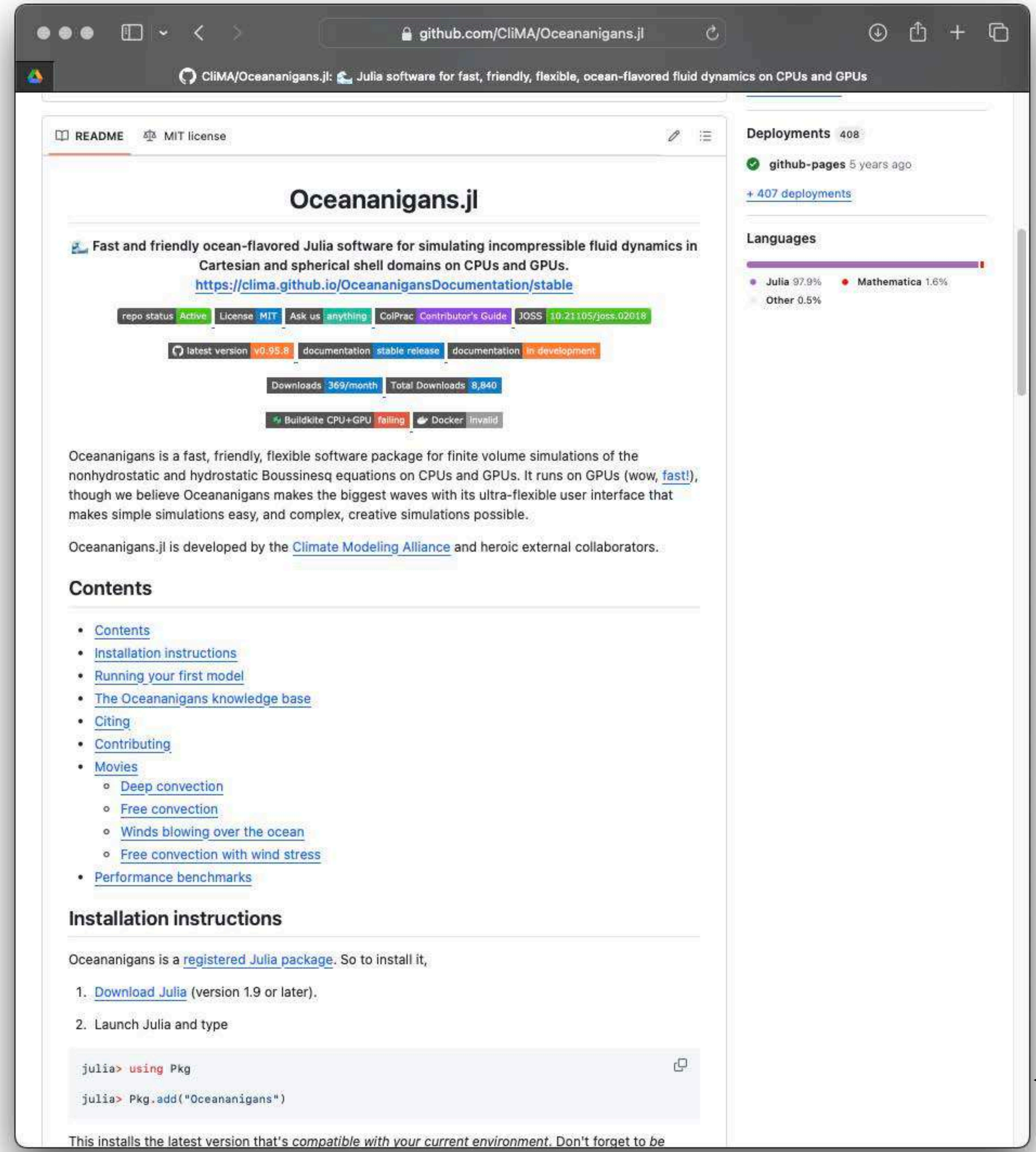




Ocean

Modern code and compute : simple to write, scales, runs on any hardware

A new generation of geoscientific models



github.com/CliMA/Oceananigans.jl

CliMA/Oceananigans.jl: Julia software for fast, friendly, flexible, ocean-flavored fluid dynamics on CPUs and GPUs

README | MIT license

Oceananigans.jl

Fast and friendly ocean-flavored Julia software for simulating incompressible fluid dynamics in Cartesian and spherical shell domains on CPUs and GPUs. https://clima.github.io/OceananigansDocumentation/stable

repo status Active | License MIT | Ask us anything | ColPrac Contributor's Guide | JOSS 10.21105/joss.02018

latest version v0.95.8 | documentation stable release | documentation in development

Downloads 369/month | Total Downloads 8,840

Buildkite CPU+GPU failing | Docker invalid

Oceananigans is a fast, friendly, flexible software package for finite volume simulations of the nonhydrostatic and hydrostatic Boussinesq equations on CPUs and GPUs. It runs on GPUs (wow, fast!), though we believe Oceananigans makes the biggest waves with its ultra-flexible user interface that makes simple simulations easy, and complex, creative simulations possible.

Oceananigans.jl is developed by the Climate Modeling Alliance and heroic external collaborators.

Contents

- Contents
- Installation instructions
- Running your first model
- The Oceananigans knowledge base
- Citing
- Contributing
- Movies
  - Deep convection
  - Free convection
  - Winds blowing over the ocean
  - Free convection with wind stress
- Performance benchmarks

Installation instructions

Oceananigans is a registered Julia package. So to install it,

1. Download Julia (version 1.9 or later).
2. Launch Julia and type

```
julia> using Pkg

julia> Pkg.add("Oceananigans")
```

This installs the latest version that's *compatible with your current environment*. Don't forget to be

Deployments 408

github-pages 5 years ago

+ 407 deployments

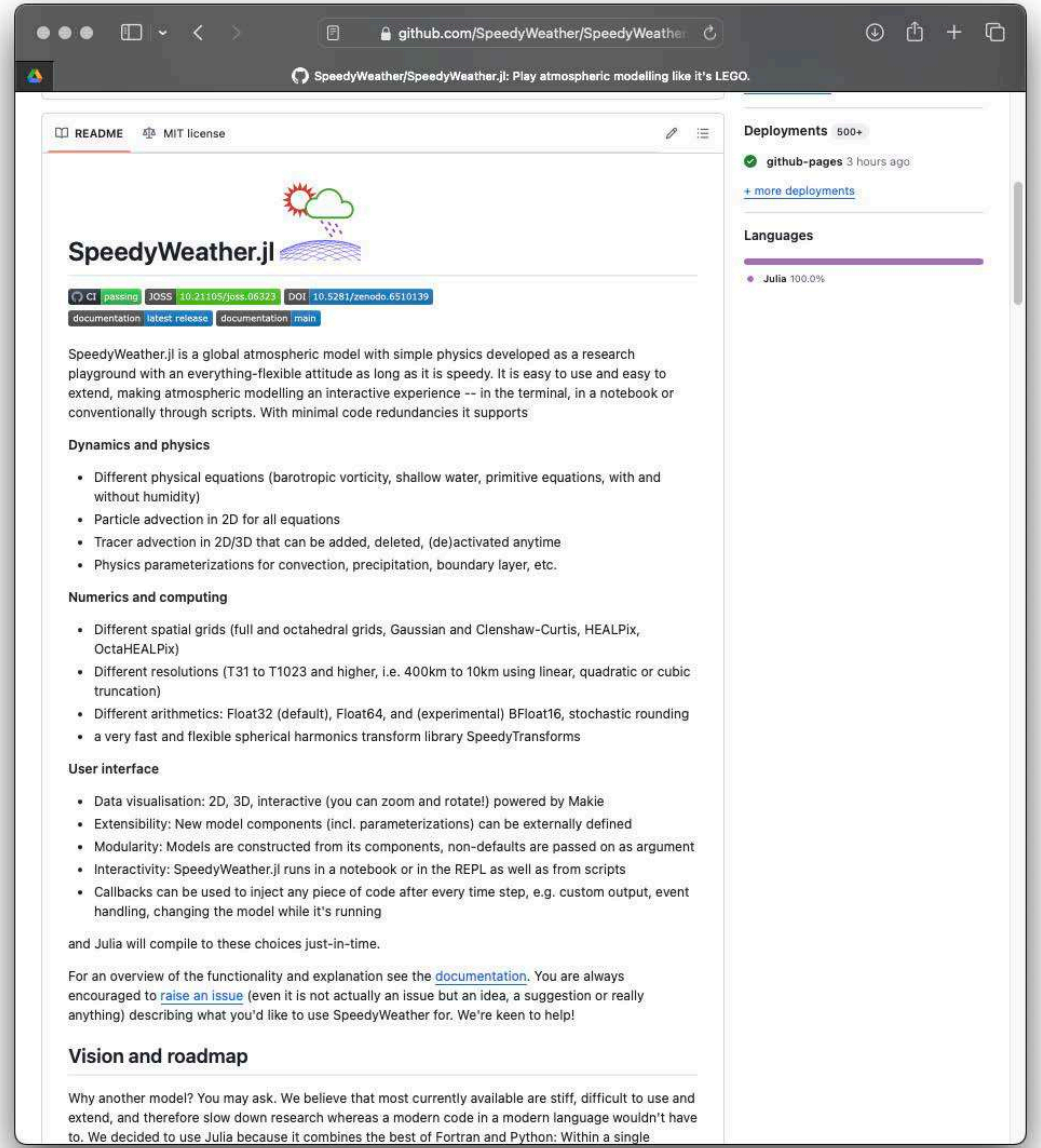
Languages

Julia 97.9% | Mathematica 1.6% | Other 0.5%

julia

Atmos →

← Ocean



github.com/SpeedyWeather/SpeedyWeather

SpeedyWeather/SpeedyWeather.jl: Play atmospheric modelling like it's LEGO.

README | MIT license

SpeedyWeather.jl

CI passing | JOSS 10.21105/joss.06323 | DOI 10.5281/zenodo.6510139

documentation latest release | documentation main

SpeedyWeather.jl is a global atmospheric model with simple physics developed as a research playground with an everything-flexible attitude as long as it is speedy. It is easy to use and easy to extend, making atmospheric modelling an interactive experience -- in the terminal, in a notebook or conventionally through scripts. With minimal code redundancies it supports

Dynamics and physics

- Different physical equations (barotropic vorticity, shallow water, primitive equations, with and without humidity)
- Particle advection in 2D for all equations
- Tracer advection in 2D/3D that can be added, deleted, (de)activated anytime
- Physics parameterizations for convection, precipitation, boundary layer, etc.

Numerics and computing

- Different spatial grids (full and octahedral grids, Gaussian and Clenshaw-Curtis, HEALPix, OctaHEALPix)
- Different resolutions (T31 to T1023 and higher, i.e. 400km to 10km using linear, quadratic or cubic truncation)
- Different arithmetics: Float32 (default), Float64, and (experimental) BFloat16, stochastic rounding
- a very fast and flexible spherical harmonics transform library SpeedyTransforms

User interface

- Data visualisation: 2D, 3D, interactive (you can zoom and rotate!) powered by Makie
- Extensibility: New model components (incl. parameterizations) can be externally defined
- Modularity: Models are constructed from its components, non-defaults are passed on as argument
- Interactivity: SpeedyWeather.jl runs in a notebook or in the REPL as well as from scripts
- Callbacks can be used to inject any piece of code after every time step, e.g. custom output, event handling, changing the model while it's running

and Julia will compile to these choices just-in-time.

For an overview of the functionality and explanation see the documentation. You are always encouraged to raise an issue (even it is not actually an issue but an idea, a suggestion or really anything) describing what you'd like to use SpeedyWeather for. We're keen to help!

Vision and roadmap

Why another model? You may ask. We believe that most currently available are stiff, difficult to use and extend, and therefore slow down research whereas a modern code in a modern language wouldn't have to. We decided to use Julia because it combines the best of Fortran and Python: Within a single

Deployments 500+

github-pages 3 hours ago

+ more deployments

Languages

Julia 100.0%

Modern code and compute : simple to write, scales, runs on any hardware

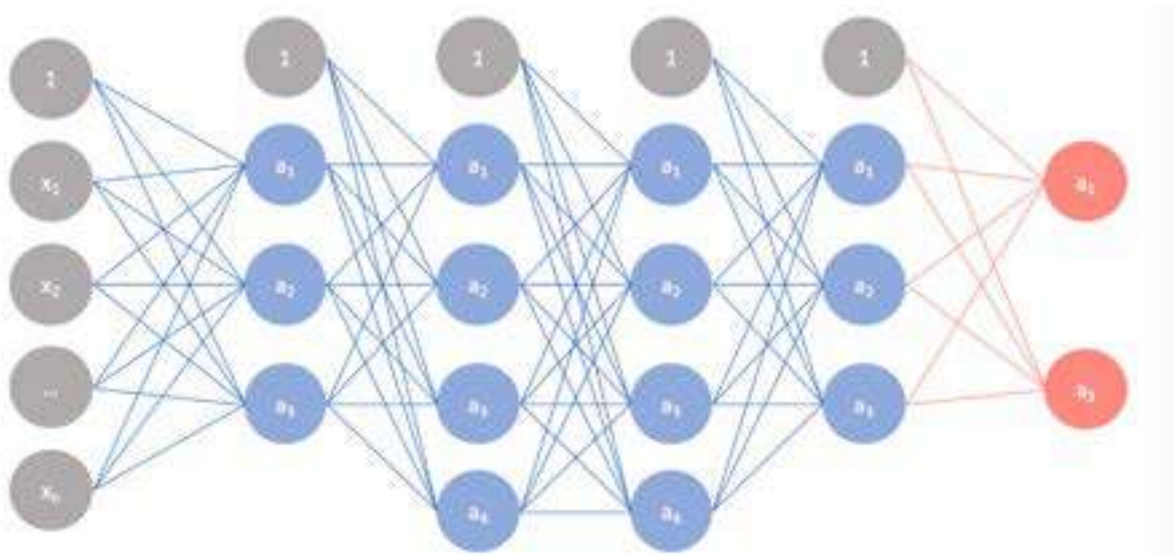
Allowing a seamless integration with AI

stable, robust, low abstraction languages

high abstraction, fast evolving languages

Allowing a seamless integration with AI

Data-driven modelling



Physics-based modelling

$$\frac{\mathrm{D}u}{\mathrm{D}t} = \frac{uv\tan\phi}{r} - \frac{uw}{r} + fv - f'w - \frac{c_p\theta}{r\cos\phi}\frac{\partial\Pi}{\partial\lambda} + \mathrm{D}(u),$$

$$\frac{\mathrm{D}v}{\mathrm{D}t} = -\frac{u^2\tan\phi}{r} - \frac{vw}{r} - uf - \frac{c_p\theta}{r}\frac{\partial\Pi}{\partial\phi} + \mathrm{D}(v),$$

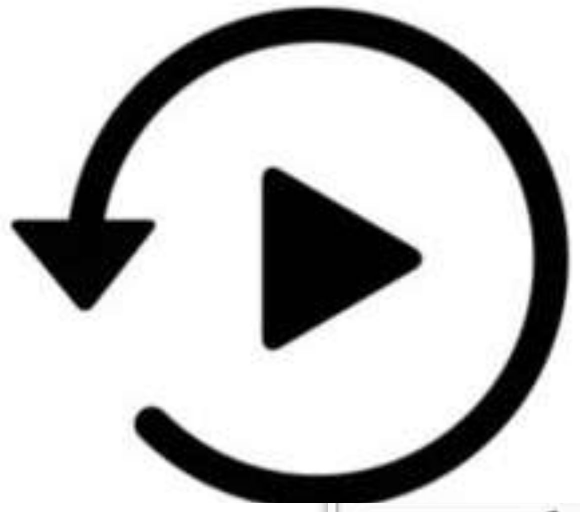
$$\delta\frac{\mathrm{D}w}{\mathrm{D}t} = \frac{u^2+v^2}{r} + uf' - g(r) - c_p\theta\frac{\partial\Pi}{\partial r},$$

PyTorch

high abstraction, fast evolving languages

PyTorch

high abstraction, fast evolving languages



AI-native hybrid climate models ?

Neural general circulation models for weather and climate

https://doi.org/10.1038/s41586-024-07744-y

Received: 13 November 2023

Accepted: 15 June 2024

Published online: 22 July 2024

Open access

Check for updates

Dmitrii Kochkov[1,6,✉], Janni Yuval[1,6,✉], Ian Langmore[1,6], Peter Norgaard[1,6], Jamie Smith[1,6], Griffin Mooers[1], Milan Klöwer[2], James Lottes[1], Stephan Rasp[1], Peter Düben[3], Sam Hatfield[3], Peter Battaglia[4], Alvaro Sanchez-Gonzalez[4], Matthew Willson[4], Michael P. Brenner[1,5] & Stephan Hoyer[1,6,✉]

General circulation models (GCMs) are the foundation of weather and climate prediction[1,2]. GCMs are physics-based simulators that combine a numerical solver for large-scale dynamics with tuned representations for small-scale processes such as cloud formation. Recently, machine-learning models trained on reanalysis data have achieved comparable or better skill than GCMs for deterministic weather forecasting[3,4]. However, these models have not demonstrated improved ensemble forecasts, or shown sufficient stability for long-term weather and climate simulations. Here we present a GCM that combines a differentiable solver for atmospheric dynamics with machine-learning components and show that it can generate forecasts of deterministic weather, ensemble weather and climate on par with the best machine-learning and physics-based methods. NeuralGCM is competitive with machine-learning models for one- to ten-day forecasts, and with the European Centre for Medium-Range Weather Forecasts ensemble prediction for one- to fifteen-day forecasts. With prescribed sea surface temperature, NeuralGCM can accurately track climate metrics for multiple decades, and climate forecasts with 140-kilometre resolution show emergent phenomena such as realistic frequency and trajectories of tropical cyclones. For both weather and climate, our approach offers orders of magnitude computational savings over conventional GCMs, although our model does not extrapolate to substantially different future climates. Our results show that end-to-end deep learning is compatible with tasks performed by conventional GCMs and can enhance the large-scale physical simulations that are essential for understanding and predicting the Earth system.

Solving the equations for Earth's atmosphere with general circulation models (GCMs) is the basis of weather and climate prediction[1,2]. Over the past 70 years, GCMs have been steadily improved with better numerical methods and more detailed physical models, while exploiting faster computers to run at higher resolution. Inside GCMs, the unresolved physical processes such as clouds, radiation and precipitation are represented by semi-empirical parameterizations. Tuning GCMs to match historical data remains a manual process[5], and GCMs retain many persistent errors and biases[6–8]. The difficulty of reducing uncertainty in long-term climate projections[9] and estimating distributions of extreme weather events[10] presents major challenges for climate mitigation and adaptation[11].

Recent advances in machine learning have presented an alternative for weather forecasting[3,4,12,13]. These models rely solely on machine-learning techniques, using roughly 40 years of historical data from the European Center for Medium-Range Weather Forecasts (ECMWF) reanalysis v5 (ERA5)[14] for model training and forecast initialization. Machine-learning methods have been remarkably successful, demonstrating state-of-the-art deterministic forecasts for 1- to 10-day weather prediction at a fraction of the computational cost of traditional models[3,4]. Machine-learning atmospheric models also require considerably less code, for example GraphCast[3] has 5,417 lines versus 376,578 lines for the National Oceanic and Atmospheric Administration's FV3 atmospheric model[15] (see Supplementary Information section A for details).

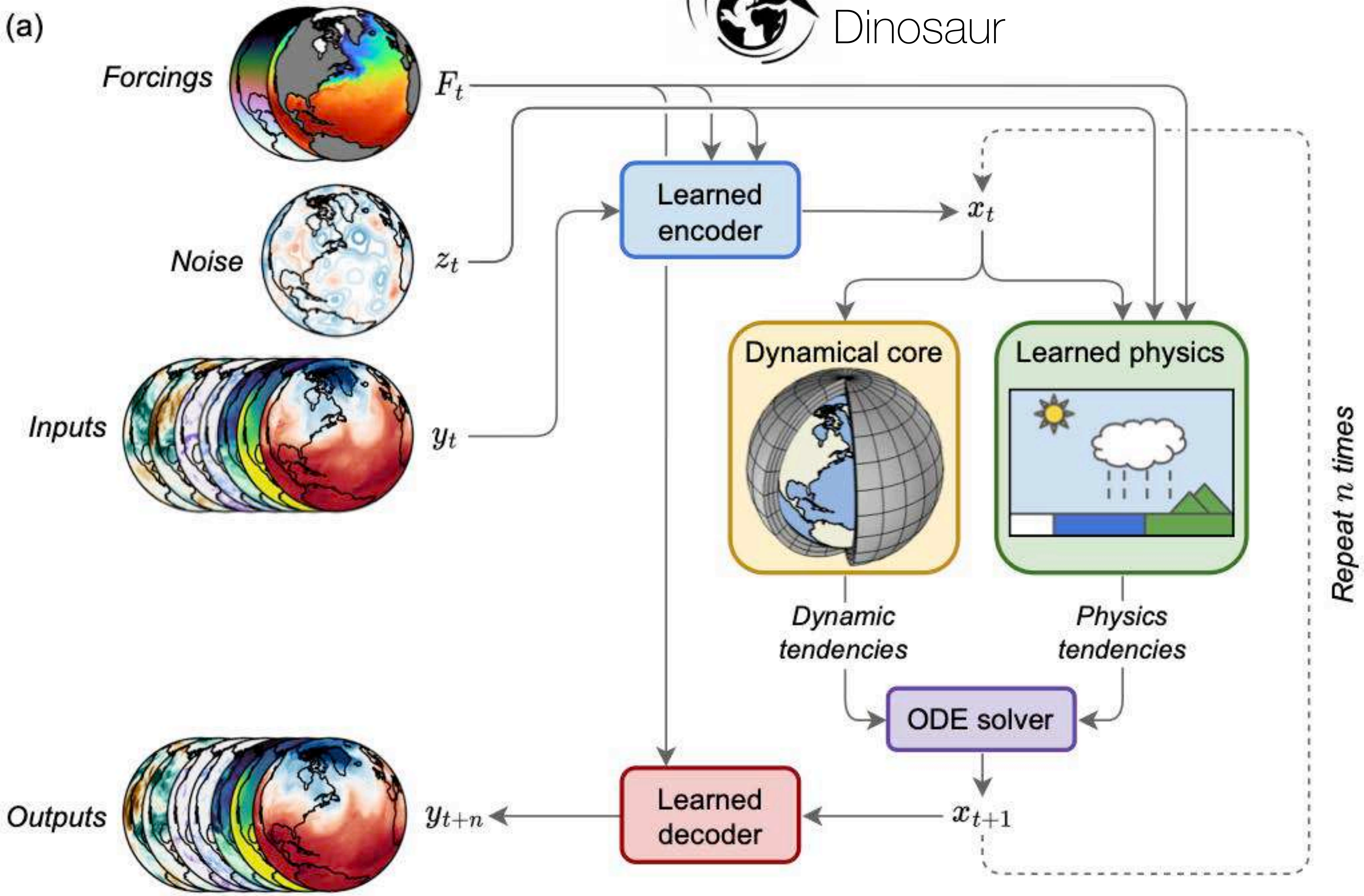
Nevertheless, machine-learning approaches have noteworthy limitations compared with GCMs. Existing machine-learning models have focused on deterministic prediction, and surpass deterministic numerical weather prediction in terms of the aggregate metrics for which they are trained[3,4]. However, they do not produce calibrated uncertainty estimates[4], which is essential for useful weather forecasts[1]. Deterministic machine-learning models using a mean-squared-error loss are rewarded for averaging over uncertainty, producing unrealistically blurry predictions when optimized for multi-day forecasts[3,13]. Unlike physical models, machine-learning models misrepresent derived (diagnostic) variables such as geostrophic wind[16]. Furthermore,

[1]Google Research, Mountain View, CA, USA. [2]Earth, Atmospheric and Planetary Sciences, Massachusetts Institute of Technology, Cambridge, MA, USA. [3]European Centre for Medium-Range Weather Forecasts, Reading, UK. [4]Google DeepMind, London, UK. [5]School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. [6]These authors contributed equally: Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Stephan Hoyer. ✉e-mail: dkochkov@google.com; janniyuval@google.com; shoyer@google.com

https://doi.org/10.1038/s41586-024-07744-y

Kochkov et al. (2024)

Dinosaur



https://github.com/google-research/dinosaur

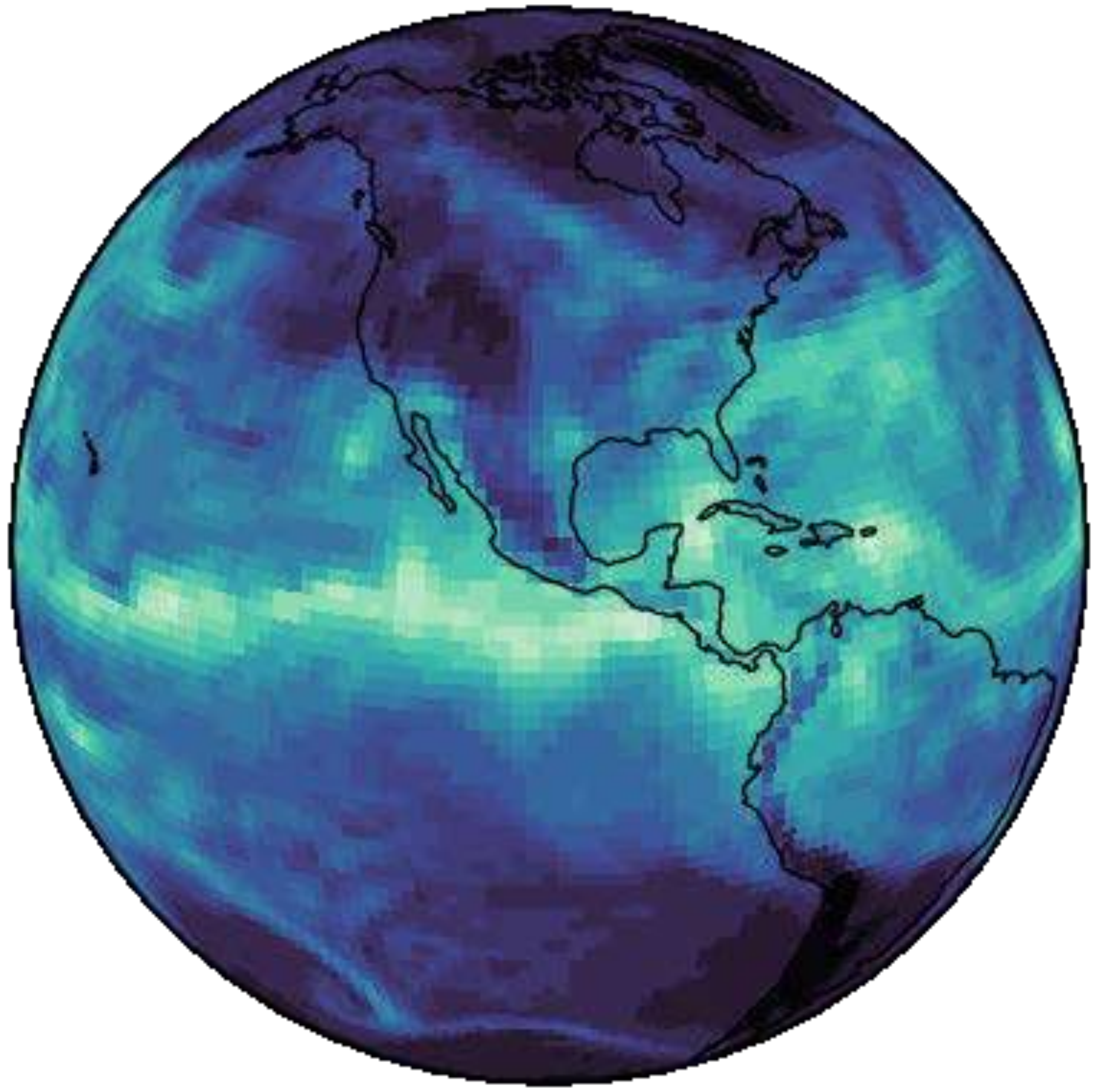
https://github.com/google-research/neuralgcm

AI-native hybrid climate models ?

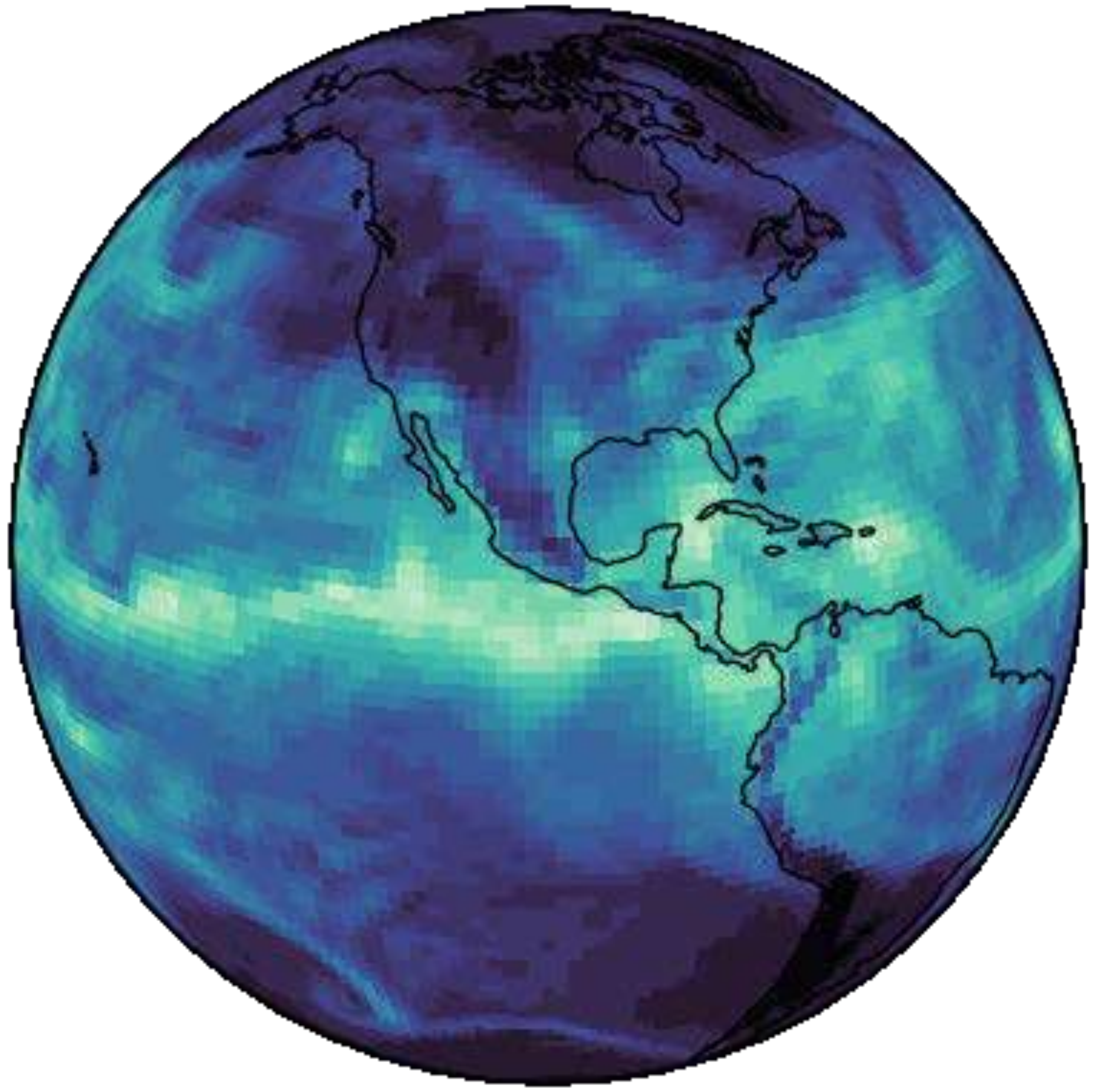
Kochkov et al. (2024)

https://arxiv.org/abs/2311.07222

Total column water, 0-15 days



ERA5

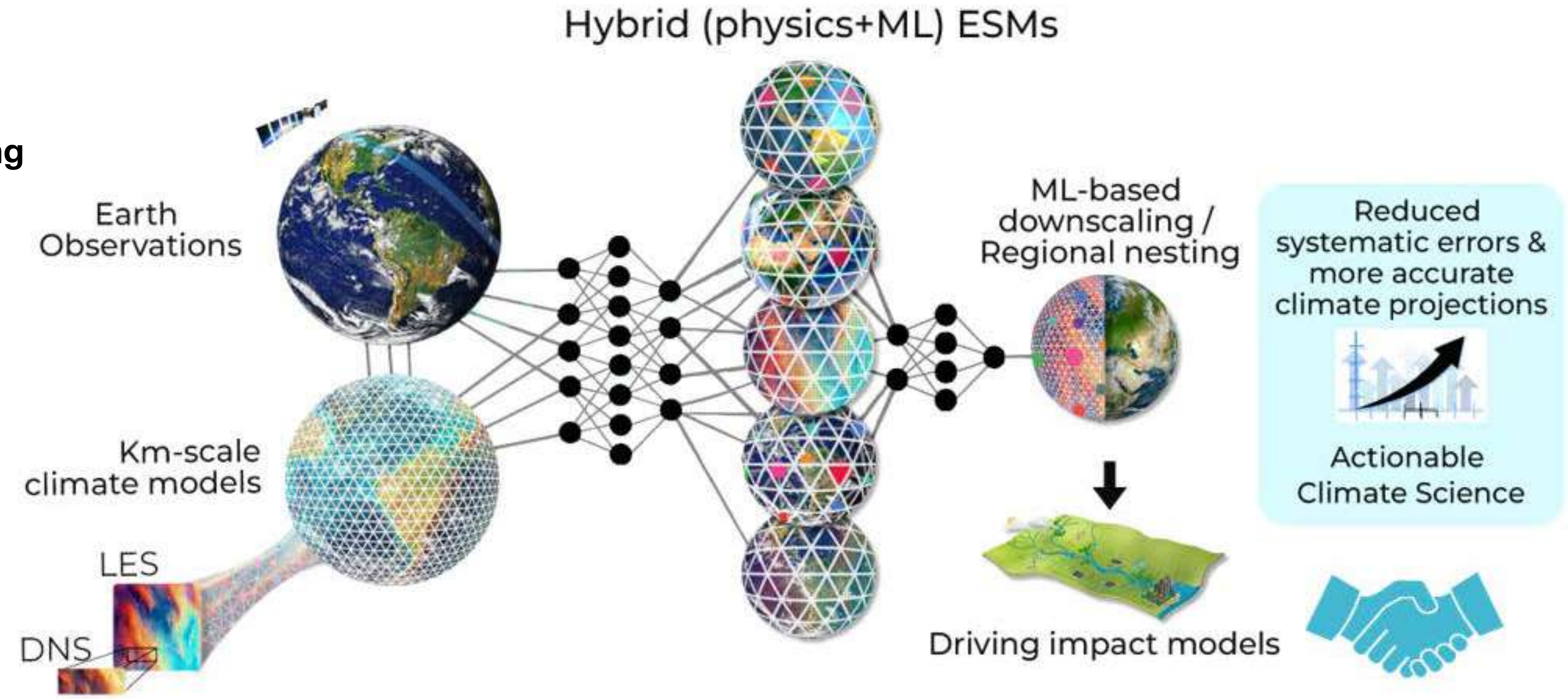


NeuralGCM

Hybrid w/ online : non-blurry forecast + stable simulators (runs ~10 years)

AI-native hybrid climate models ?

See P. Gentine's keynote during TRACCS General Assembly



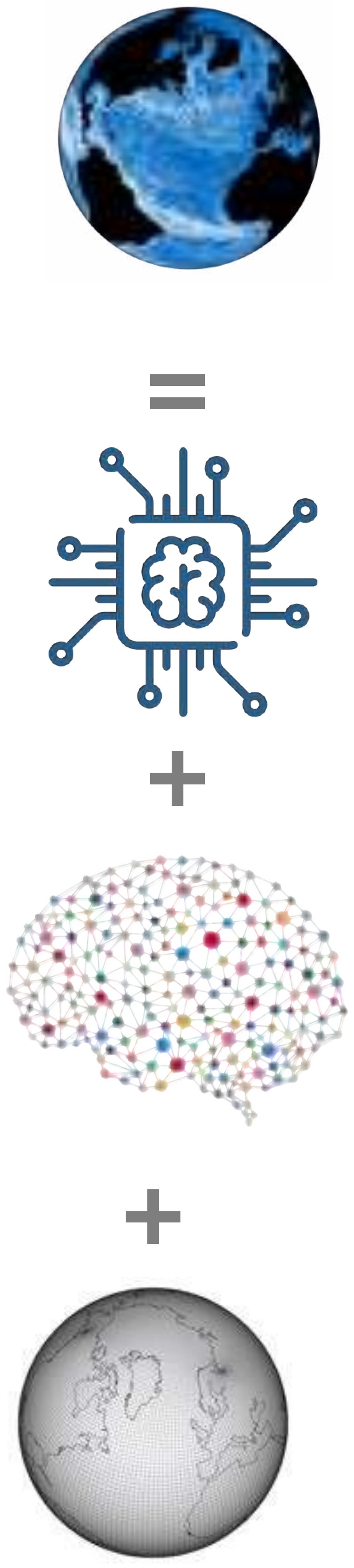
Eyring, Gentine, et al., 2024 https://doi.org/10.1038/s41561-024-01527-w

LEGO land :
Versatile models allowing exploration of advanced ML workflows

Harnessing global observations !

Differentiable programming
in earth system models

Summary



- Why we are **augmenting ocean models** w/ trained comp.
- Described how this can be **done in practice** today
- Further progress require **deeper recast** of our models
- Exploration w/ new generation of **AI-native hybrid** models
- Exciting time for **cross-disciplinary** investigations !

AI4PEX M2LINES

References

- Frezat et al. (2021). Physical invariance in neural networks for subgrid-scale scalar flux modeling. https://doi.org/10.1103/PhysRevFluids.6.024607

- Frezat et al. (2022) A posteriori learning for quasi-geostrophic turbulence parametrization, JAMES. 14. https://doi.org/10.1029/2022MS003124

- Frezat et al. (2024) Gradient-free online learning of subgrid-scale dynamics with neural emulators, sub., https://doi.org/10.48550/arXiv.2310.19385

- Yan (2024). Adjoint-based online learning of two-layer quasi-geostrophic baroclinic turbulence. arXiv. https://doi.org/10.48550/arXiv.2411.14106