NSF CSSI 2103942: Convergence of Bayesian inverse methods and scientific machine learning in Earth system models through universal differentiable programming

Patrick Heimbach (Lead PI, UT Austin), Karen Willcox (co-PI, UT Austin), Alan Edelman (PI, MIT)
Chris Hill (Co-PI, MIT), Nora Loose (PI, CU Boulder), Chris Rackauckas (JuliaHub), William S. Moses (UIUC)
Mathieu Morlighem (PI, Dartmouth), Michel Schanen (PI, UChicago), Sri Hari Krishna Narayanan (PI, UChicago)

DJ4Earth

Climate Change in the Ocean & Cryosphere

Understanding and mitigating climate changes is a global challenge.

- >90% of the Earth's net energy imbalance (EEI) goes into the ocean.
- ~25% of the anthropogenic emission of CO_2 ends up in ocean, leading to ocean acidification.
- Increased freshwater input to the ocean from melting of polar ice sheets (Greenland, Antarctica) and ice caps raises sea level.

Modeling

Simulation requires:

- Physics at a variety of levels
- Increasing detail (resolution) and complexity (process representation)
- Creates vast output (e.g., 10-100TB/simulation, 10PB for CMIP6)
- Arctic simulation at 1-2 km resolution takes >7 million core hours, 1.1PB output, running on 10,000 cores

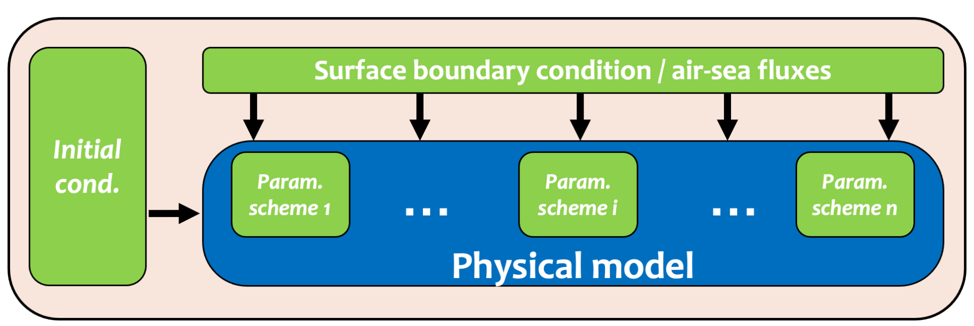


Figure 1. Overview of ocean simulations. Initial inputs, and boundary conditions are time evolved through a model that involves uncertain constitutive laws and subgrid-scale parameterizations.

Significant uncertainties remain, including:

- Uncertain inputs (initial and boundary conditions)
- Parametric and structural model uncertainties (requiring calibration)
- Remote influences limiting regional simulations

Data

Real-world data is very sparse and heterogeneous. Given both incomplete observations and simulations, how can we derive useful results, like:

- Casual, dynamical attribution
- Detection of small, residual signals in noisy system
- Computing comprehensive uncertainties
- Informing efficient observing strategies

By combining domain science with computational & computer science!

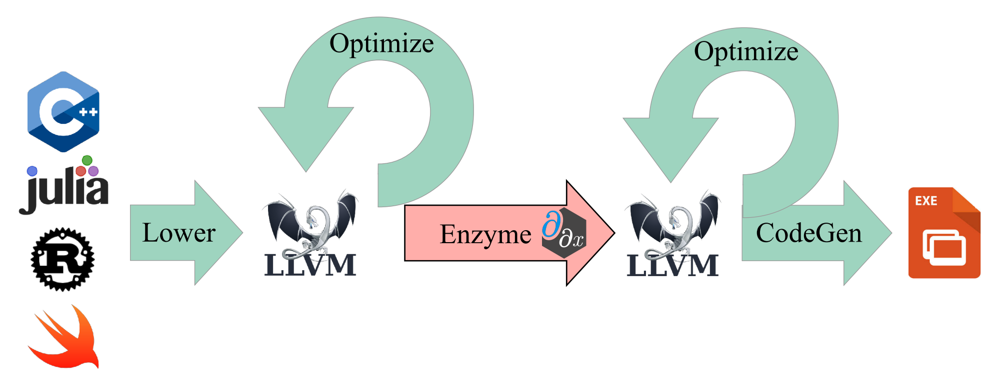
Frameworks: Need for Differentiable Programming

We need fast & scalable derivatives for full-model learning from data:

1. Differentiate w.r.t. boundary conditions to explore forcing sensitivites.
2. Differentiate w.r.t. initial conditions to produce optimal forecast.
3. Differentiate w.r.t. parameters to calibrate the model to data.
4. Substitute parameterized schemes with a surrogate neural network.

Algorithms: Enzyme Automatic Differentiation

The Enzyme tool computes derivatives in a common compiler. This enables Enzyme to differentiate any LLVM-based language (C, C++, Fortran, Rust, Swift, Julia, Python, JaX, MLIR, PyTorch) *AND* leverage compiler analyses and optimizations for performance.



- First AD tool to differentiate existing GPU kernels, where new optimizations enabled orders of magnitude speedups.
- Efficient scaling support for multiple parallel paradigms (AMD, NVIDIA, OpenMP, MPI, and more) won best student paper at SC'22.
- In use at MIT, Harvard, Facebook, Smithsonian, AMD, Google, ANL, LLNL, UT Austin, NASA, Dartmouth, CU Boulder, TU Munich, University of Washington, Adobe, Toronto, and several startups.

Algorithms: Checkpointing.jl

Time evolution creates iterative algorithms with millions of iterations. Even on modest-sized models, this may requires infeasible amounts of memory. Checkpointing.jl provides a trade-off between re-computation and storage by transforming loop iterations.

$$\begin{aligned} \mathbf{x}_{t+1} &= f(\mathbf{x}_t) \\ \bar{\mathbf{x}}_t &= \bar{f}(\mathbf{x}_t, \bar{\mathbf{x}}_{t+1}) \\ &= \frac{\partial f(\mathbf{x}_t)}{\partial \mathbf{x}_t} \bar{\mathbf{x}}_{t+1} \end{aligned}$$

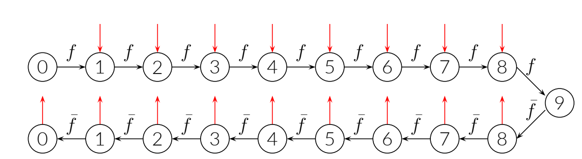


Figure 2. Evaluation process of iteratively applying function f for $t = 1:9$ iterations, f is called with state x_t as the input and state x_{t+1} as the output. The adjoint function $\bar{f}$ of f computes state $\bar{x}_t$ with respect to state $\bar{x}_{t+1}$ and x_t. The red down and up arrows mark a stored and restored state, respectively.

Science Application: Ocean Model

Development of a differentiable barotropic gyre (single layer ocean model with wind-driven circulation). Below is the adjoint of spatially averaged kinetic energy at the final time with respect to the initial displacement field η. Computed using Checkpointing.jl and Enzyme.jl. Energy is computed as

$$\mathcal{E}(t_f, u, v) = \frac{\sum_{x,y} u^2(t_f, x, y) + v^2(t_f, x, y)}{n_x n_y}$$

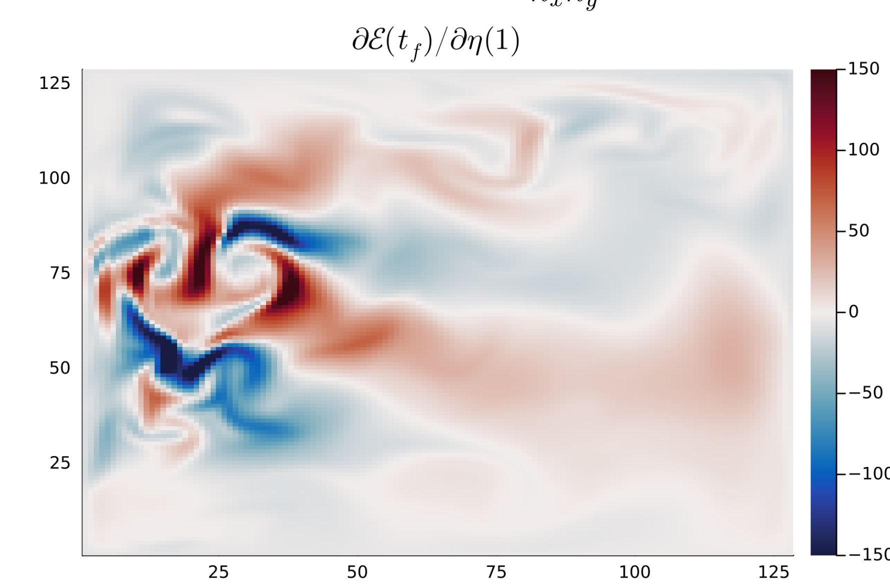


Figure by Sarah Williamson (UT Austin)

Science Application: Ice Sheet Model

Development of dJUICE.jl, a differentiable ice sheet model in Julia. Below is a snapshot of a transient simulation of ice velocity of Helheim glacier, southeast Greenland. We are additionally developing a differentiable sea ice model for the Julia-based ocean model Oceananigans.jl.

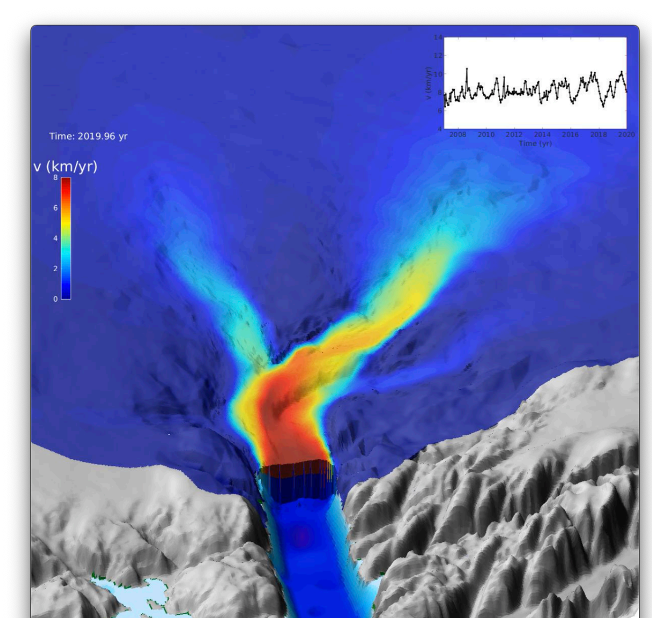


Figure by Cheng Gong (Dartmouth)

To learn more, please visit:
`dj4earth.github.io`