ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

May 2026

# DJ4Oceananigans:

## A differentiable, Julia-based ocean general circulation model for gradient-based parameter calibration and online learning.

**Dhruv Apte**
Peter O'Donnell Jr. Postdoctoral Fellow, UT Austin

# Lead Contributors

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

- William Moses, Valentin Churavy, Maximilian Gelbrecht, Avik Pal, Mosè Giordano
- Joseph Kump, Sarah Williamson, Patrick Heimbach
- Gong Cheng, Mathieu Morlighem
- Sri Hari Krishna Narayanan, Michel Schanen, Alexis Montoison
- Greg Wagner, Nora Loose, Simone Silvestri

And many others…

# Ocean GCM (General Circulation Model)

The ocean state is represented as a set of spatiotemporal fields (velocity, temperature, salinity):

$$\text{State at } (\mathbf{x}, t) = [\mathbf{u}(\mathbf{x}, t), w(\mathbf{x}, t), T(\mathbf{x}, t), S(\mathbf{x}, t)]^T .$$

We solve the Boussinesq and hydrostatic approximations of incompressible Navier-Stokes:

$$\partial_t \mathbf{u} + (\mathbf{v} \cdot \nabla)\mathbf{u} + \mathbf{f} \times \mathbf{u} = -\nabla_h (p + g\eta) - \nabla \cdot \tau + \mathbf{F}_\mathbf{u} \quad \text{(momentum equation)}$$

$$\partial_z p = -\rho g \quad \text{(hydrostatic)}$$

$$\nabla_h \mathbf{u} + \partial_z w = 0 \quad \text{(Boussinesq conservation of volume)}$$

$$\rho = \rho(T, S, z) \quad \text{(equation of state)}$$

$$\frac{DT}{Dt} = \mathbf{D}_T + \mathbf{F}_T \quad \text{(temperature advection-diffusion)}$$

$$\frac{DS}{Dt} = \mathbf{D}_S + \mathbf{F}_S \quad \text{(salinity advection-diffusion).}$$

# Why Oceananigans?

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

CliMA
CLIMATE MODELING ALLIANCE

*Part of CliMA:*

- A team building a set of Earth System Models entirely in Julia.
  - Land, sea ice, atmosphere, component coupling.
- All models combine data-informed and physics-based approaches to modeling.

*Oceananigans Itself:*

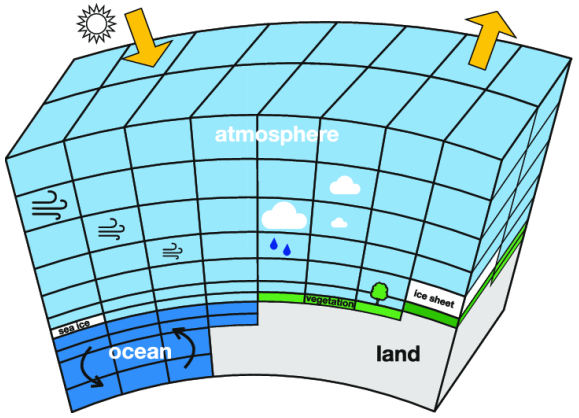
- Features ocean-flavored fluid dynamics (hydrostatic and nonhydrostatic).
- Uses finite volume simulations.
- Written with GPU support as the *primary* focus.
- Flexibility in:
  - Scale - from smaller lab-like simulations to global oceans.
  - User interface - “simple solutions easy, complex and creative simulations possible”.

TOOLBOX | 30 July 2019

**Julia: come for the syntax, stay for the speed**

**Researchers often find themselves coding algorithms in one programming language, only to have to rewrite them in a faster one. An up-and-coming language could be the answer**

Nature 572, 141-142 (2019)



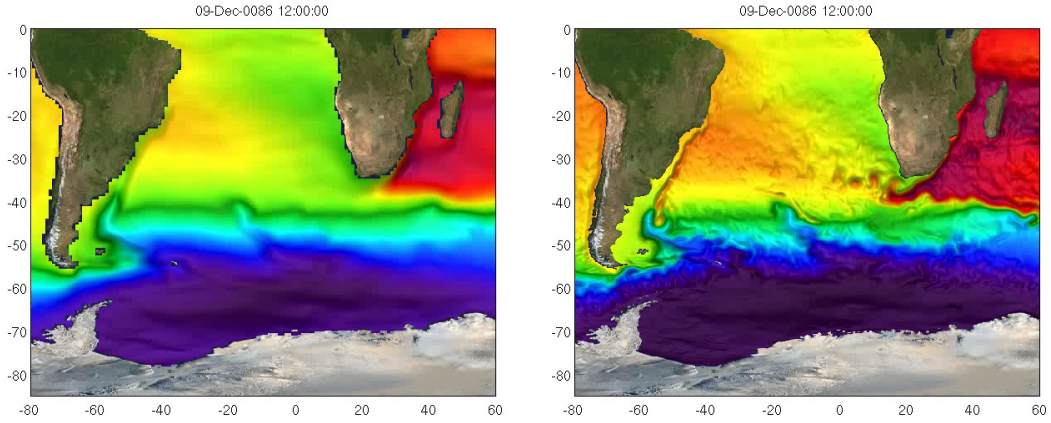

Credit: Dippe et al.

# Resolution and Scale

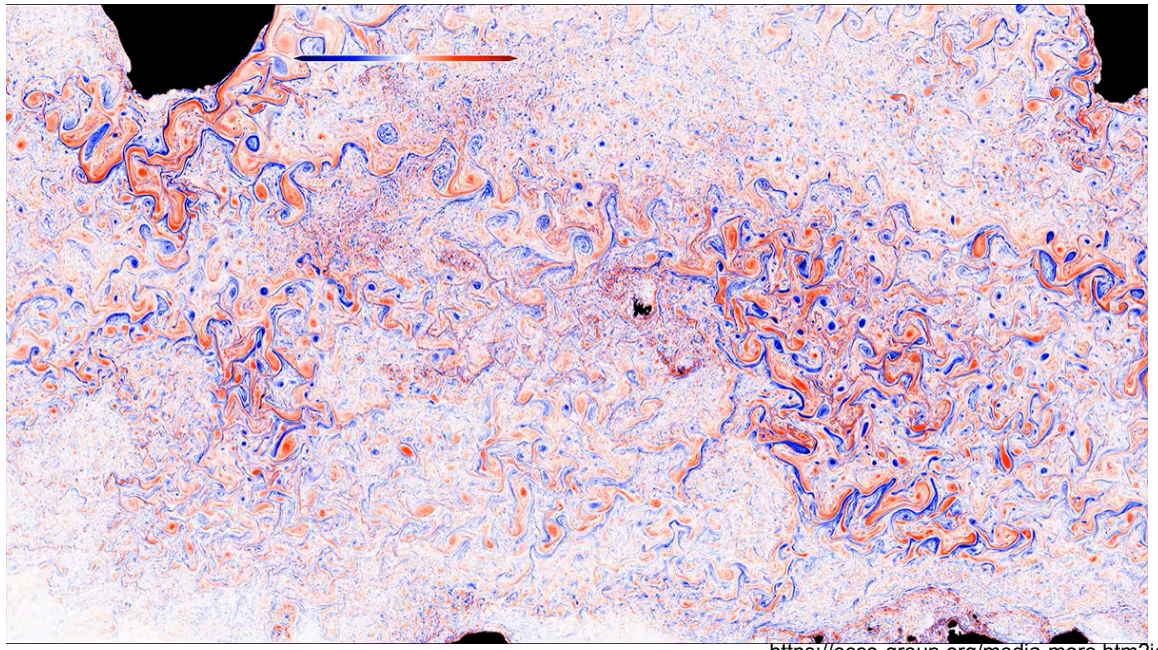
Can range from 1° (3-5m grid cells) to 1/48° (near 1b grid cells).

The central tension:

- increasing resolution may improve physical fidelity,
- *but* increases the cost of forward integration,
- and there will always be *something* that isn't physically resolved.




1° resolution 0.25° resolution



https://ecco-group.org/media-more.htm?id=31

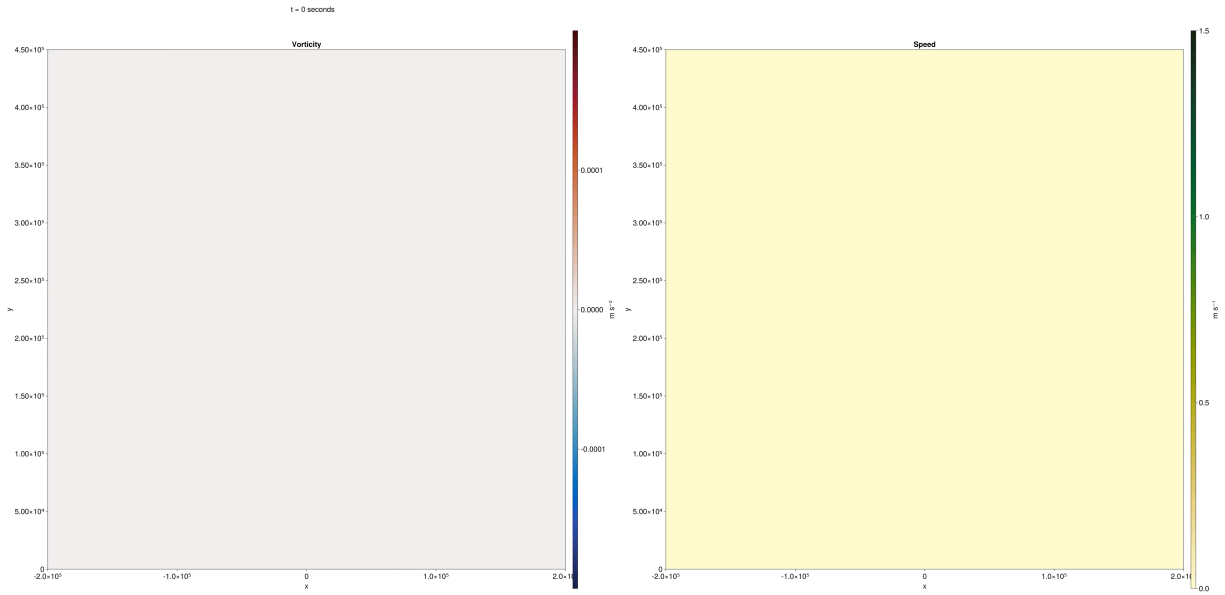
# Parameterizations

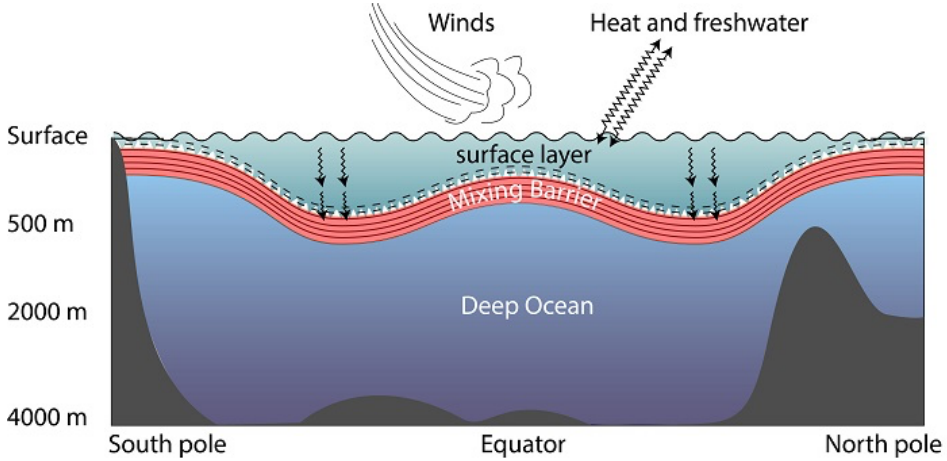
ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

*Parameterizations*: empirical or physically motivated closure models that express the aggregate effect of unresolved processes in terms of model variables at the resolved scale.

Two common use cases:

- Eddies (mesoscale)
- Vertical mixing




(Credit: cnrs)

# Use Cases for Adjoints in Ocean GCMs

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

Suppose we have a function $J(\mathbf{u})$ that measures some objective or cost of model state $\mathbf{u}(\mathbf{x}, t; \mathbf{p})$ with spatiotemporal data $\mathbf{x}$, $t$ and parameters $\mathbf{p}$. *If* we have the gradient $\nabla_{\mathbf{u}} J$ , we can do:

**Sensitivity Analysis:** $J$ is a scalar objective function, $\nabla_{\mathbf{u}} J$ gives us information on the relationship between this objective and all model inputs. FD: 2+ forward runs per scalar in $\mathbf{p}$.

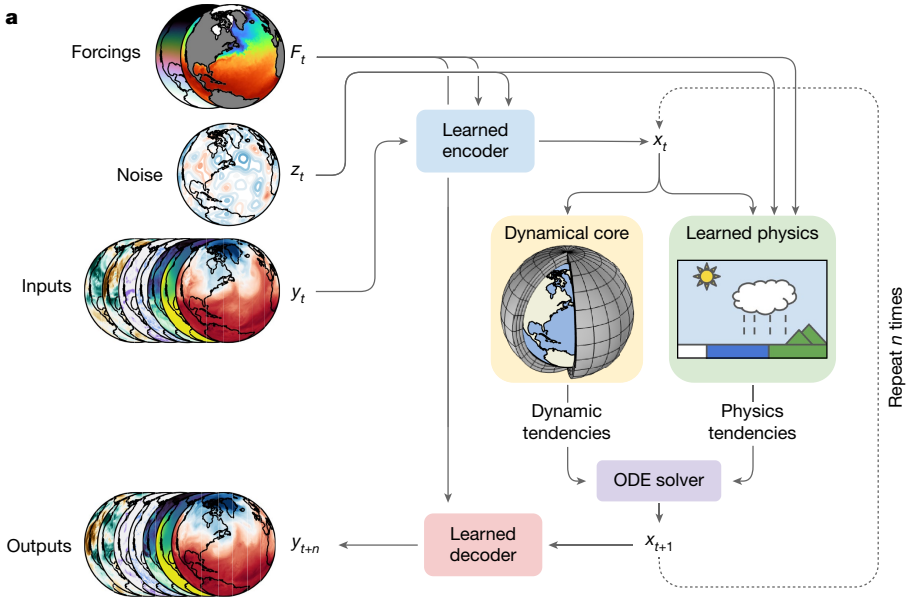
**Parameter Estimation:** $J$ measures misfit between a model output and observational or high-resolution model data. Find $\mathbf{p}$ that minimizes $J(\mathbf{u})$. Gradient-free: scales with dimension of solution space for $\mathbf{p}$.

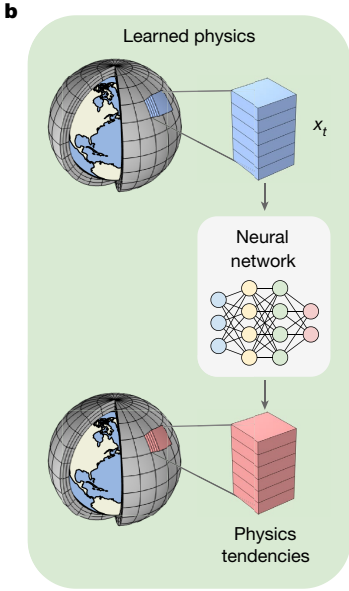
**Machine Learning:** $J$ is a cost function with a neural network (NN) embedded within a model.

While NNs can be trained on model data alone, $J$ allows us to employ *online learning* - training the NN with active runs and gradients from the original model.

# Online Learning and (Atmospheric) GCMs

- ESMs split resolved dynamics from parameterizations.
- ML parameterizations trained offline are unstable when coupled back to the dynamics
  - They treat each snapshot as independent, not accounting for previous steps' errors.
  - This is *covariate shift* / *compounding error*.
- **Online learning:** threads gradients through the differentiable solver itself, so the learned parameterization is optimized to the dynamics it actually interacts with.
  - Error can be measured across multiple timesteps at once, *penalizing drifting trajectories*.
- This requires a **differentiable dynamical core**.
- **NeuralGCM** (Google/ECMWF, 2024): stable decadal climate simulations and ECMWF-competitive ensemble forecasts, at much lower cost.



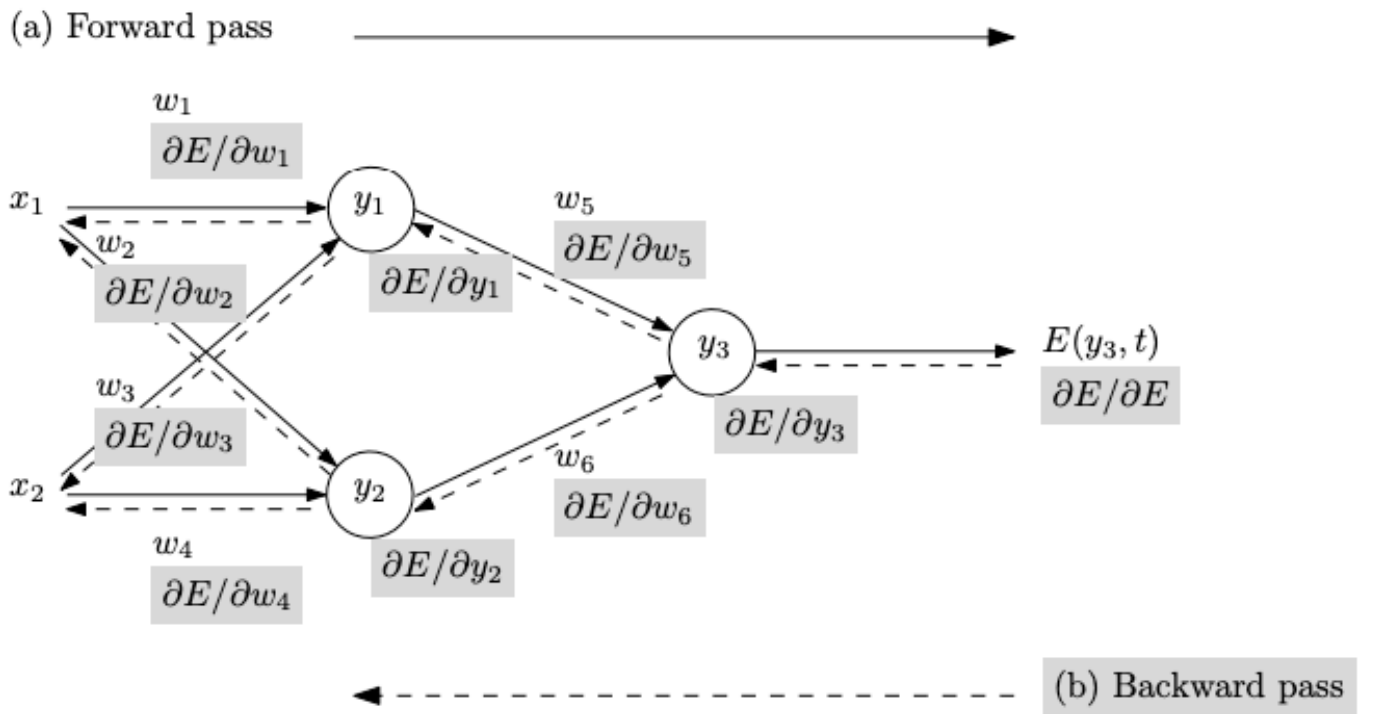




(Credit: Kochkov et al, NeuralGCM)

# Introduction to AD

In the past ocean model adjoints were written by hand, but now AD is often used to obtain $\nabla_{\mathbf{u}} J$.

*Automatic Differentiation* (AD): a set of techniques used to evaluate the derivative of a function in a computer code without manually writing an explicit rule for the derivative.




(Credit: Gowri Shankar)

# Enzyme and Reactant

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

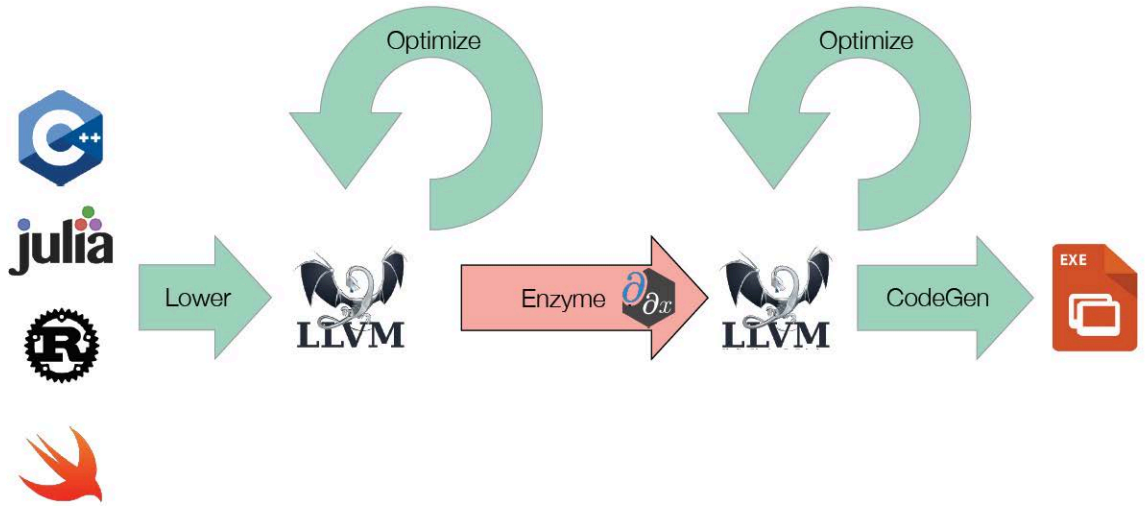
**Enzyme** (multi-language AD Package):

- Take existing code as LLVM IR (low-level virtual machine intermediate representation), integrate LLVM's optimization profile, then differentiate.
- Three pass architecture: type analysis, activity analysis, gradient synthesis.

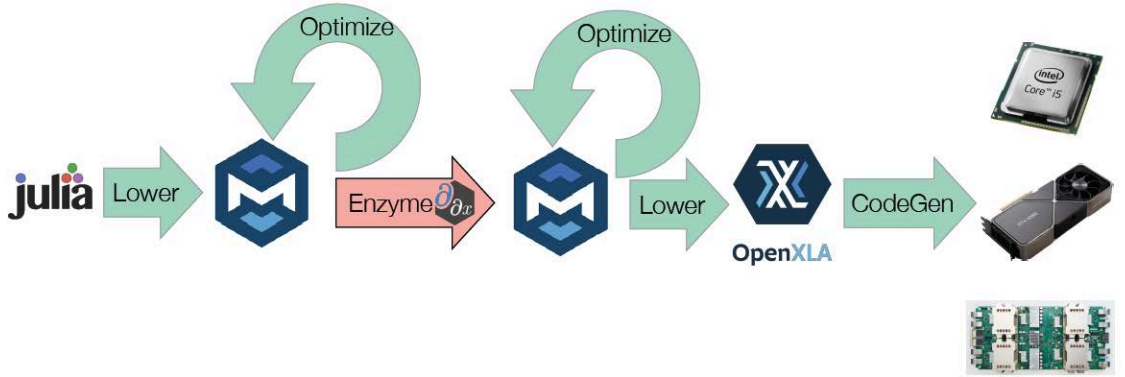
**Reactant.jl** (Julia-specific compiler tool):

- Compile Julia into MLIR (multi-level IR), run optimizations on top of it, then create executables via XLA (accelerated linear algebra).
- Enzyme can interface with MLIR before XLA compile.
- *Optimized MLIR is type-stable, and easier to differentiate than Julia-LLVM alone.*

Enzyme Pipeline:




Reactant Pipeline (with Enzyme included):



# Case Study: Re-entrant Channel Model

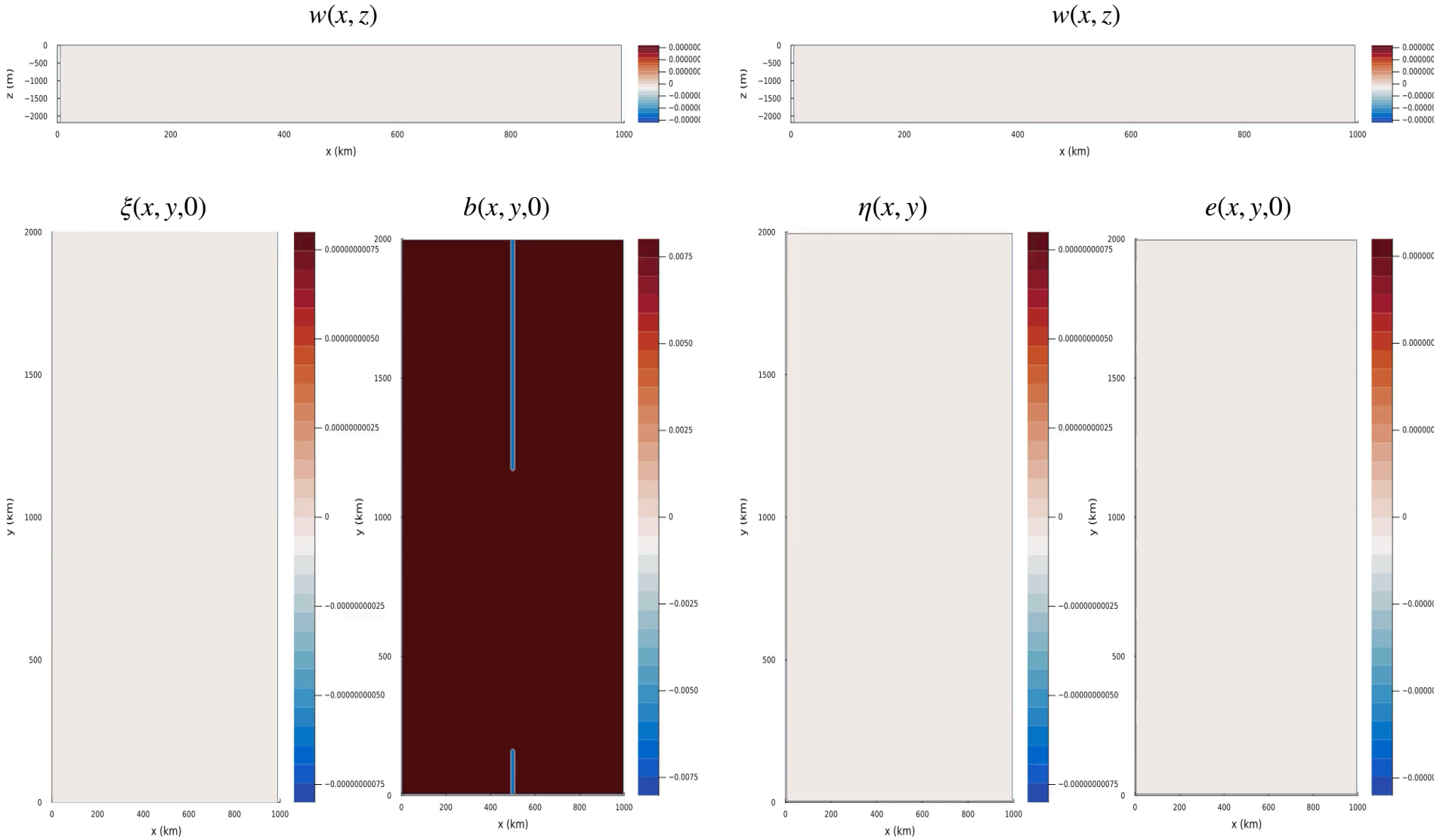
An idealization of the Southern Ocean (based on configuration in Abernathy et al. 2011)

Addition: a ridge topography with a gap (Drake passage).


Credit: National Geographic

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

**From Moses et al., 2026, "DJ4Earth: Differentiable, and Performance-portable Earth System Modeling via Program Transformations", in JAMES.**



# Case Study: Re-entrant Channel Model

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

A useful objective function is the zonal volume transport across the gap in model topography:

$$J\left(u(x_0, y, z, t)\right) = U(x_0, t) = \sum_{y,z} u(x_0, y, z, t)\Delta y \Delta z$$

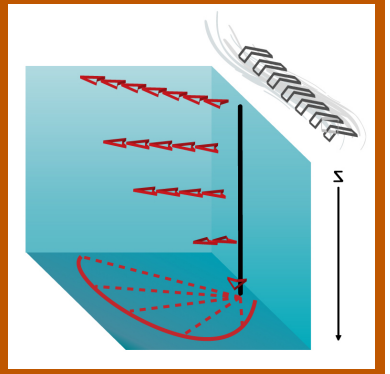

How do different model components impact this transport? Including:

- Boundary conditions (like wind stress)
- Initial state (like the initial temperature field)
- Model parameters (like diffusivity coefficients).

*We ran the model with Enzyme + Reactant AD for 14 days, obtaining sensitivities.*

# Case Study: Re-entrant Channel Model

Sensitivity field for the effect of meridional wind stress on zonal transport.



(Credit: public domain/ wiki)

Northward wind $(\tau_y > 0) \rightarrow$ westward Ekman transport

Surface divergence and upwelling of dense anomalies $(\rho' > 0)$ along ridge - *this is stronger away from the gap*.

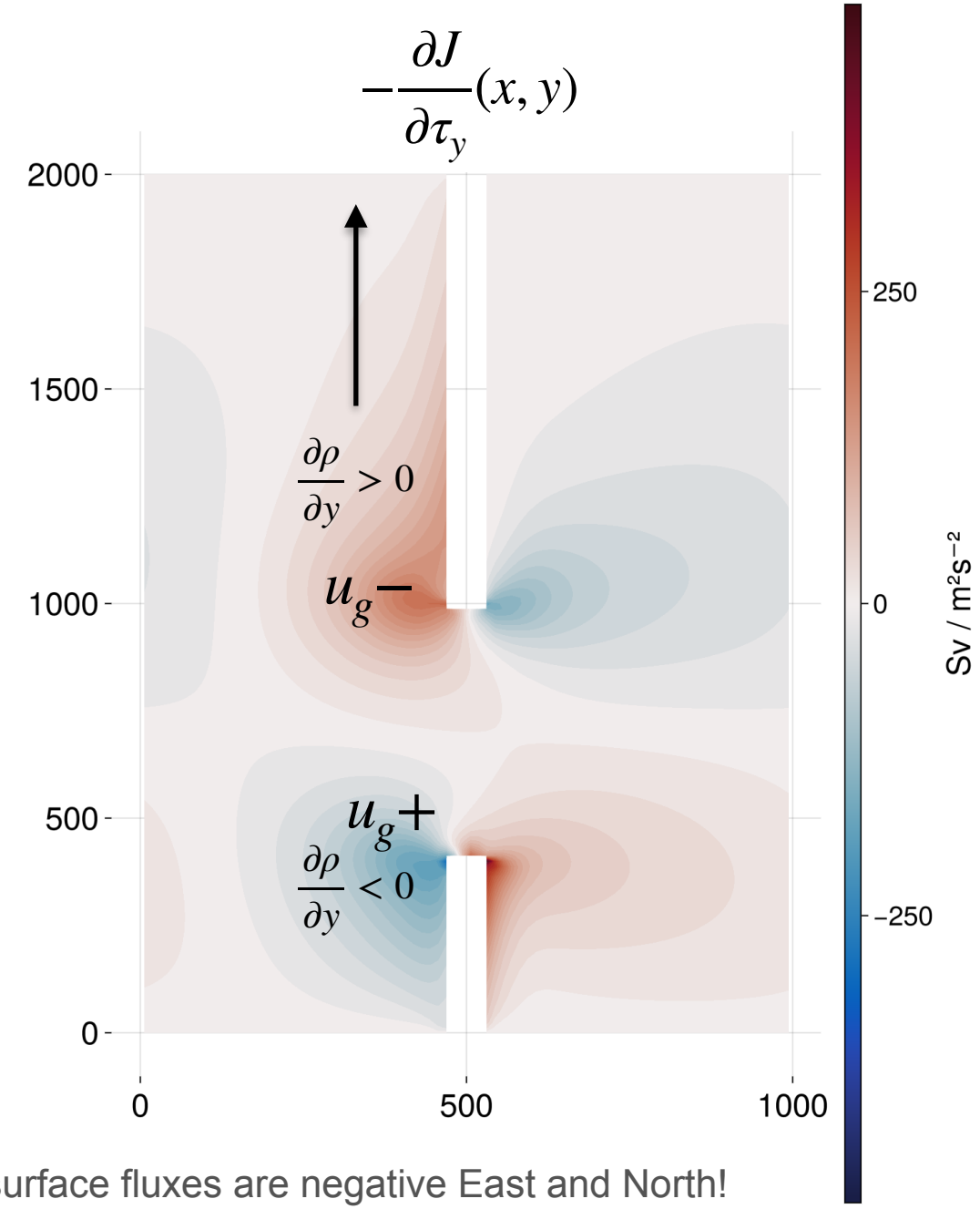
Since $\rho'$ increases *away from the gap*, $\frac{\partial \rho}{\partial y} > 0$ north and $\frac{\partial \rho}{\partial y} < 0$ south.

Thermal wind balance:

$$\frac{\partial u_g}{\partial z} = \frac{g}{\rho_0 f}\frac{\partial \rho}{\partial y}$$

Southern hemisphere, $f < 0$:

$$\frac{g}{\rho_0 f} < 0 \implies \frac{\partial u_g}{\partial z} > 0 \text{ if } \frac{\partial \rho}{\partial y} < 0$$




Oceananigans sign conventions at surface fluxes are negative East and North!

## Case Study: Performance Benefits of Reactant

The forward model sees massive improves on CPU, more modest on GPU.

Since Reactant statically allocates all data, it has a much lower memory cost for forward code.

(times are in seconds)

| Backend | Configuration | Number of Timesteps | | | | |
|---|---|---|---|---|---|---|
| | | **25** | **100** | **400** | **1600** | **8100** |
| CPU (NVIDIA Grace Superchip) | Oceananigans (primal) | 3.787 | 14.472 | 57.414 | 217.565 | 1105.209 |
| | + Reactant (primal) | 0.599 | 2.419 | 9.533 | 37.890 | 191.038 |
| | + Reactant (AD) | 26.534 | 103.609 | 396.494 | 1610.118 | 8282.979 |
| GPU (NVIDIA Grace + H200) | Oceananigans (primal) | 0.432 | 0.602 | 1.198 | 3.968 | 17.113 |
| | + Reactant (primal) | 0.0367 | 0.144 | 0.573 | 2.319 | 11.692 |
| | + Reactant (AD) | 0.758 | 2.910 | 11.340 | 15.946 | 22.011 |
| TPU (Google Cloud v6e Trillium) | Oceananigans (primal) | — | — | — | — | — |
| | + Reactant (primal) | 0.245 | 0.975 | 3.895 | 6.681 | |
| | + Reactant (AD) | 2.115 | 4.688 | 5.490 | 5.975 | |

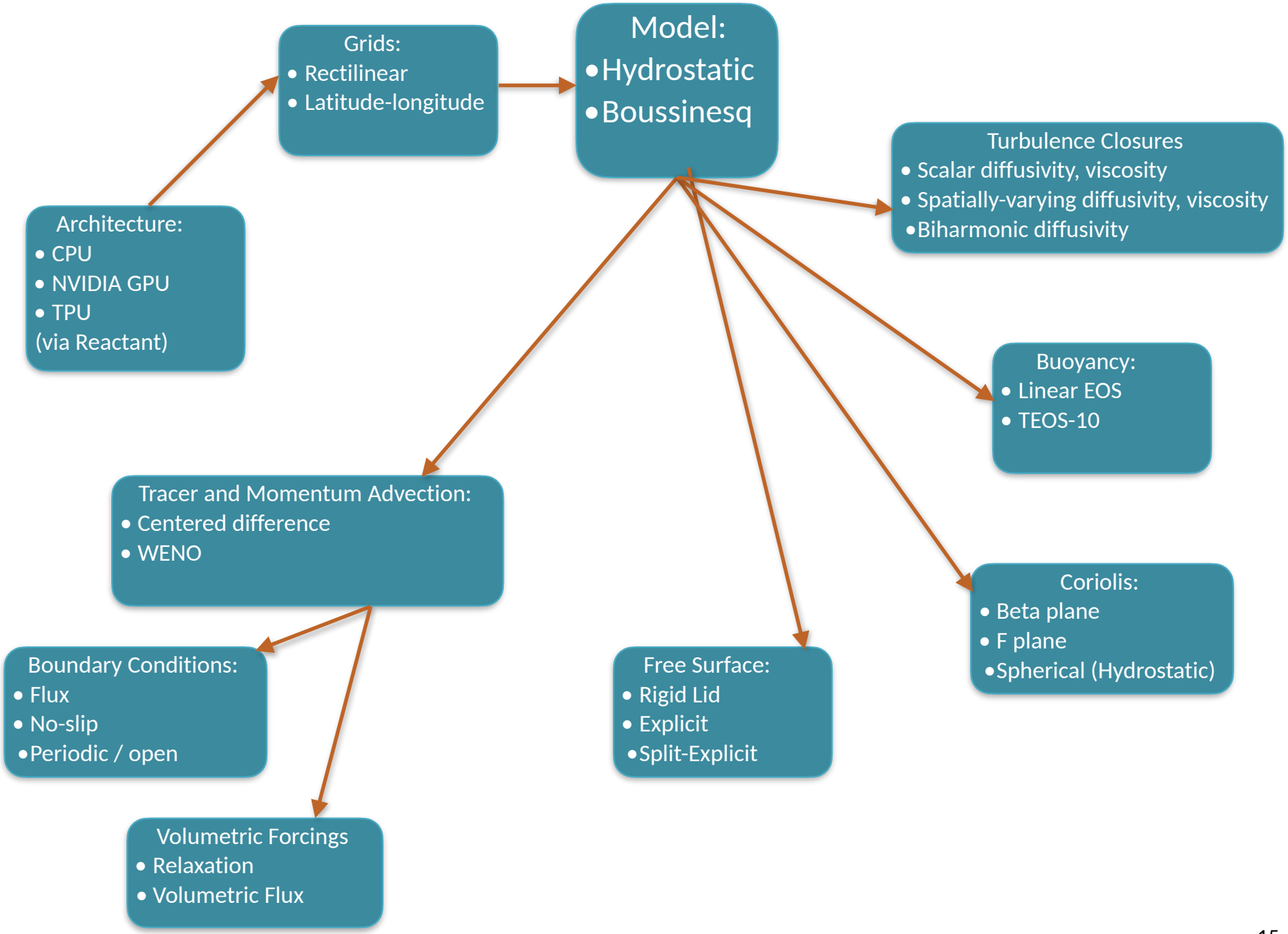
(~5.7x speedup)
(AD ~40x)

(~40% speedup)
(AD ~2x)

| | Timesteps | | | | |
|---|---|---|---|---|---|
| **Metric** | **25** | **100** | **400** | **1600** | **8100** |
| Reactant heap - primal (GiB) | 0.3406 | 0.3405 | 0.3314 | 0.3405 | 0.3405 |
| Reactant heap - AD (GiB) | 9.650 | 10.62 | 12.55 | 16.40 | 26.05 |
| AD / primal memory ratio | 28.33 | 31.18 | 37.86 | 48.16 | 76.51 |
| Default memory estimate (GiB) | 0.0904 | 0.3406 | 1.34 | 5.35 | 27.04 |
| Default allocations ($\times 10^3$) | 520 | 1489 | 5366 | 20874 | 104905 |
| Default GC (%) | 1.04 | 3.10 | 6.36 | 7.96 | 9.84 |

# Current Progress with AD on Oceananigans

We have successfully differentiated a large portion of the Oceananigans code base with Enzyme + Reactant.



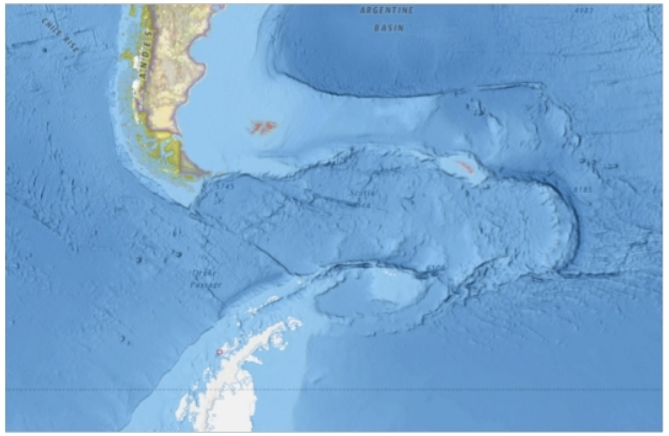

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

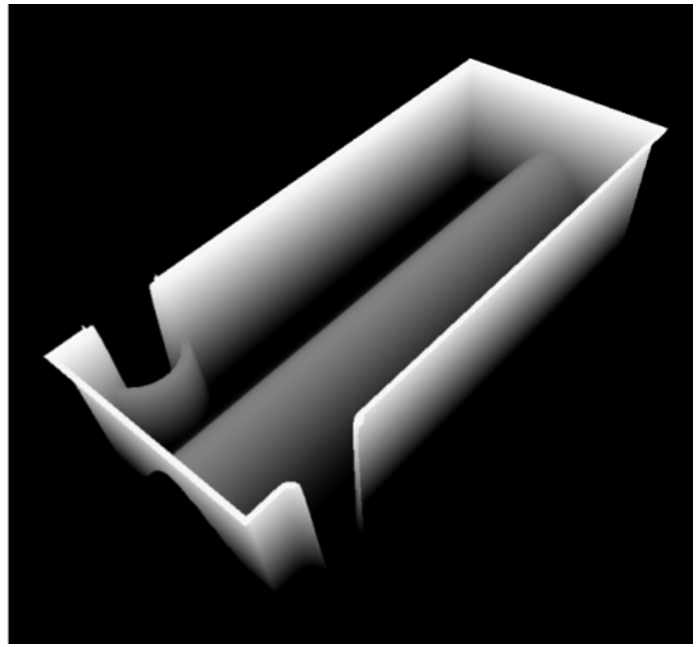
# Next Steps: Parameter Estimation

Recall the cost function $J$ measuring the misfit between some model output and observational or high-res data.

We plan on using this framework to invert for parameterization coefficients. Specifically **mesoscale eddies** and **vertical mixing**.

This can be done on our regional re-entrant channel. Another configuration of interest is the Diabatic NeverWorld2 Ocean (DINO).



https://dl.acm.org/doi/fullHtml/10.1145/3569951.3603640

# Some Parameterization Schemes

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

**GM/Redi** for mesoscale eddies *(Gent-McWilliams & Redi schemes - established in 1990s)*

- Represents tracer advection-diffusion from eddies, using flows along density-neutral slopes (isopycnals).

$$\nabla \cdot \left( \kappa_\rho \mathbf{K}_{\text{Redi}} + \kappa_{\text{GM}} \mathbf{K}_{\text{GM}} \right) \nabla \tau, \mathbf{K}_{\text{Redi}} = \begin{bmatrix} 1 & 0 & S_x \\ 0 & 1 & S_y \\ S_x & S_y & S_x^2 + S_y^2 \end{bmatrix}, \mathbf{K}_{\text{GM}} = \begin{bmatrix} 0 & 0 & -S_x \\ 0 & 0 & -S_y \\ S_x & S_y & 0 \end{bmatrix}$$

**CATKE** for turbulent mixing *(Convective-Adjustment Turbulent Kinetic Energy - devised in 2023)*

- Uses a prognostic TKE tracer $e$ and diagnostic mixing length scale $\ell$ that features dynamic convective adjustment.

$$\frac{\partial}{\partial t}\tau = -\frac{\partial}{\partial z}\left(\overline{w'\tau'}\right), \qquad \overline{w'\tau'} \approx -\kappa_\tau \frac{\partial}{\partial z}\tau \text{ where } \kappa_\tau = \ell_\tau \sqrt{e}.$$

- Trained on gradient-free Ensemble Kalman Inversion (EKI).

*Ongoing effort:* determining ideal way to train parameters ($\kappa_\rho$, $\kappa_{\text{GM}}$, $\sqrt{e}$, $\ell_\tau$, resulting free parameters in their derivations).

# Parameter Estimation

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

$$J(\mathbf{u}) = \|\mathscr{M}\left(\mathbf{u}(\mathbf{x}, t, \mathbf{p})\right) - \mathscr{D}\|_{\text{misfit}}.$$

We plan on using this framework to invert for parameterization coefficients in GM/Redi or CATKE, such as:

- $\kappa_\rho$
- $\kappa_{\text{GM}}$
- $e$ (or its free parameters)
- $\ell_\tau$ (or its free parameters)

The CATKE free parameters were trained on 21 models using gradient-free EKI.

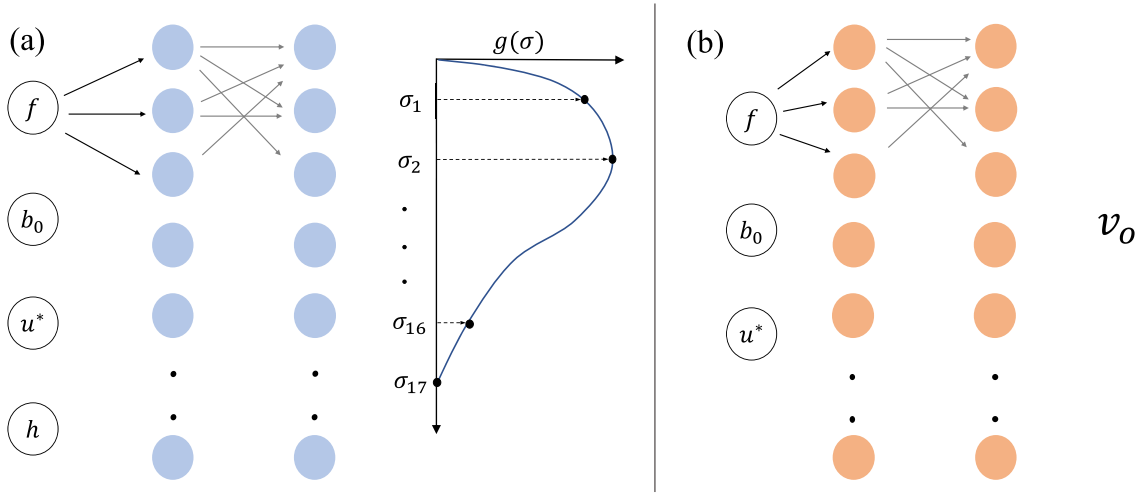
Now that we have gradients and AD, we can use them (likely with more robust Quasi-Newton methods) to train their free parameters on a much larger regional model and in problem-specific contexts.

# Parameter Estimation *and ML*

Incorporate a small neural network (NN) into the parameterization scheme. Either…

- **Augment** using a small NN inside the traditional physics-based parameterization to train some or all free parameters in the scheme:
  - *Zanna et al.* uses dimensional analysis to derive non-dimensional parameters for this task.
  - These NNs can be "small" (≈ 20-1000 nodes total).
- **Replace** the entire tendency introduced by the parameterization with one given by a (larger) NN.

Key advantage: Julia has ML packages (Lux.jl, Flux.jl) which interact with Enzyme + Reactant. Thus we can train NN weights *online*.



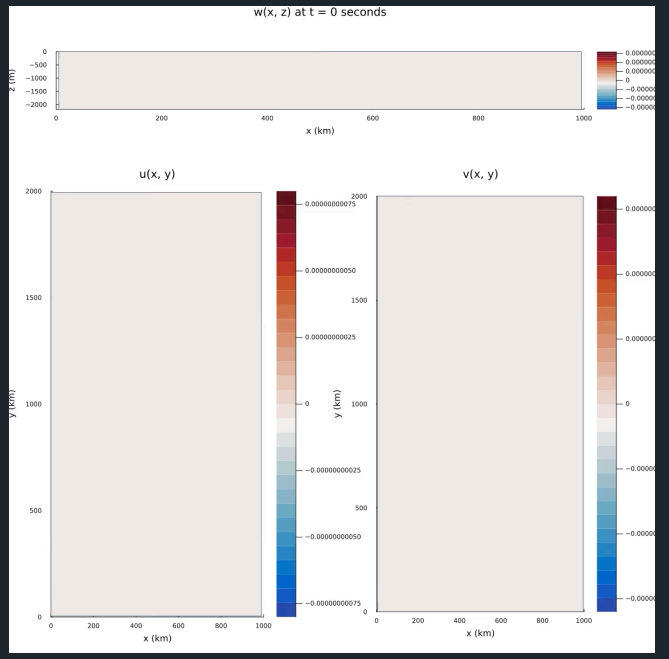

(Credit: Zanna et al, 2023)

This is similar to NeuralGCM, but with a more targeted focus on individual parameterizations and shift to ocean modeling.

# Image citations

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

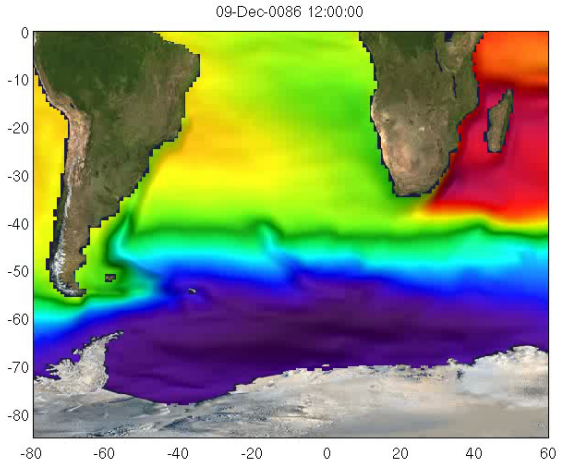

1. https://www.researchgate.net/publication/262972565_Analysis_of_three_heavy_rain_events_of_the_2009_season_as_simulated_by_the_WRF_model
2. https://bg.copernicus.org/articles/18/3189/2021/bg-18-3189-2021.html
3. https://arxiv.org/pdf/2309.06662
4. https://www.clivar.org/sites/default/files/documents/wgomd/ws2014/06_highres_bentsen.pdf
5. https://dl.acm.org/doi/fullHtml/10.1145/3569951.3603640
6. https://www.researchgate.net/figure/Schematic-of-a-Coupled-General-Circulation-Model-CGCM-On-the-most-basic-level-the_fig2_327314089




**Thank you!**

**Check out our recent paper by scanning the QR code!**

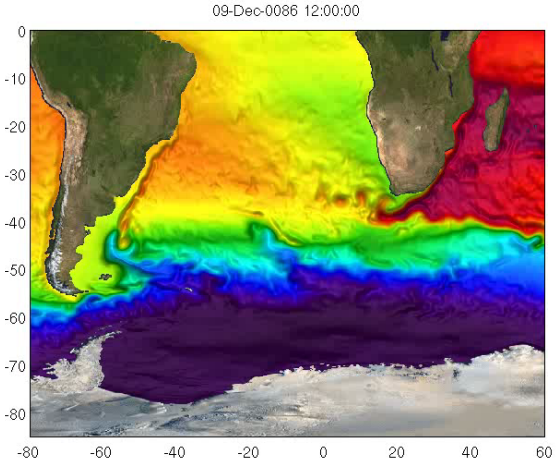
# Resolution and Scale




1° resolution

Computational scale of an ocean GCM → horizontal resolution (# grid cells, timestep from CFL).

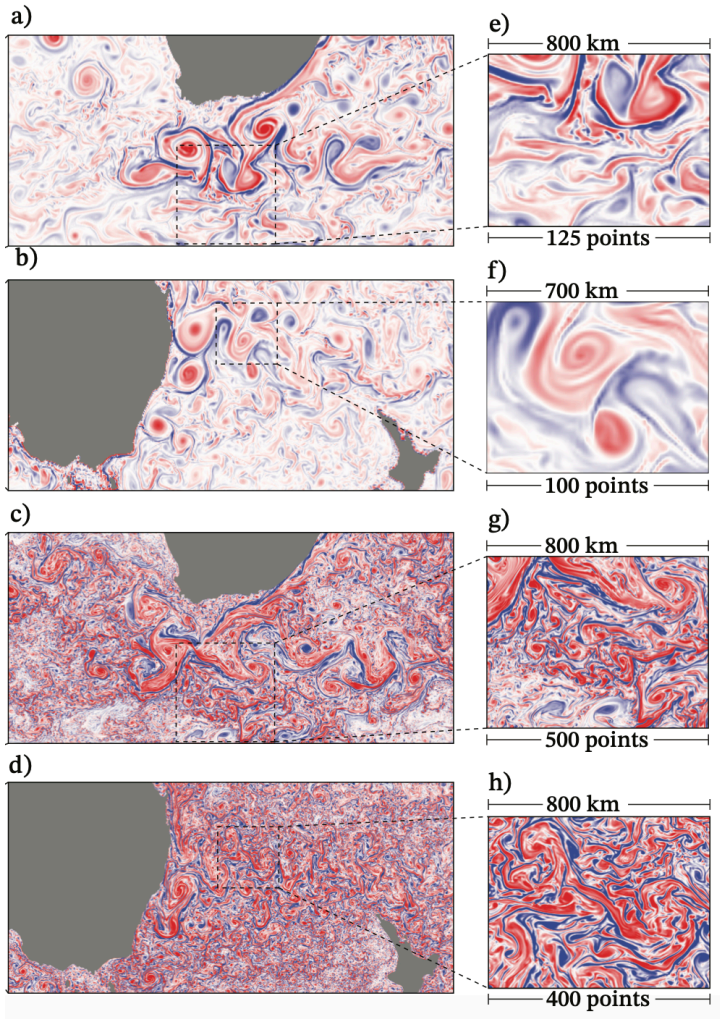
1° (100 km near equator)




0.25° resolution

Feasible on modest HPC.

1/4° (25 km, eddy-permitting)




1/12° (8 km, eddy-resolving)

Require massive resources, though GPUs help.

1/48° (2 km)

https://arxiv.org/pdf/2309.06662

https://www.clivar.org/sites/default/files/documents/wgomd/ws2014/06_highres_bentsen.pdf

# Enzyme and Reactant

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

**Enzyme** (multi-language AD Package):

- Take existing code as LLVM IR, integrate LLVM's optimization profile, then differentiate.
- Three pass architecture:
  - *Type analysis*: use a type tree to describe the known type at any given byte offset.
  - *Activity analysis*: determine which values are differentially active.
  - *Gradient synthesis*: "differentiate".

*Compared to other AD tools:*

LLVM (formerly low-level virtual machine) produces an IR (intermediate representation) for languages like C and Julia. It offers a powerful compiler infrastructure. By using it Enzyme can differentiate several programming languages, and take advantage of compiler optimizations before computing the gradient.

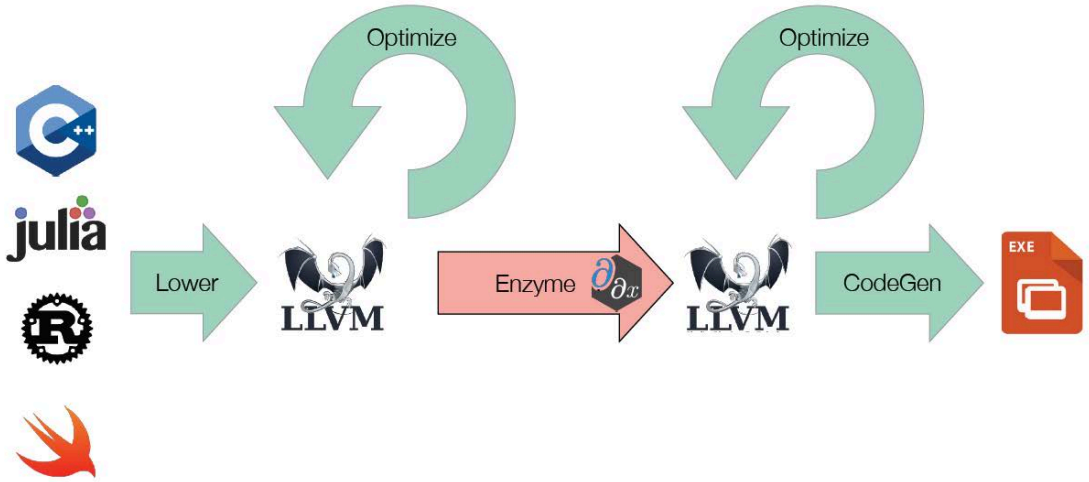
```
float mag(const float* x);//Compute magnitude in O(N)
void norm(float* out, float* in) {
    // float res = mag(in); code motion optimization can move outside the loop
    for(int i=0; i<N; i++) { out[i] = in[i]/mag(in); }
}
```

```
// LICM, then AD, O(N)
void ∇norm(float* out, float* d_out,
           float*  in, float* d_in) {
  float res = mag(in);
  for (int i=0; i<N; i++) {
    out[i] = in[i]/res;
  }
  float d_res = 0;
  for (int i=0; i<N; i++) {
    d_res += -in[i]*in[i]/res * d_out[i];
    d_in[i] += d_out[i]/res;
  }
  ∇mag(in, d_in, d_res);
}
```

```
// AD, then LICM O(N^2)
void ∇norm(float* out, float* d_out,
           float*  in, float* d_in) {
  float res = mag(in);
  for (int i=0; i<N; i++) {
    out[i] = in[i]/res;
  }
  for (int i=0; i<N; i++) {
    float d_res = -in[i]*in[i]/res \
                  * d_out[i];
    d_in[i] += d_out[i]/res;
    ∇mag(in, d_in, d_res);
  }
}
```

Figure 1: ***Top:*** An $O(N^2)$ function `norm` which normalizes a vector. Running loop-invariant-code-motion (LICM) [45, Sec. 13.2] moves the $O(N)$ call to `mag` outside the loop, reducing `norm`'s runtime to $O(N)$. ***Left:*** An $O(N)$ $\nabla$`norm` resulting from running LICM before AD. Both `mag` and its adjoint $\nabla$`mag` are outside the loop. ***Right:*** An $O(N^2)$ $\nabla$`norm` resulting from running LICM after AD. $\nabla$`mag` remains inside the loop as it uses a value computed inside the loop, making LICM illegal.

https://proceedings.neurips.cc/paper_files/paper/2020/file/9332c513ef44b682e9347822c2e457ac-Paper.pdf

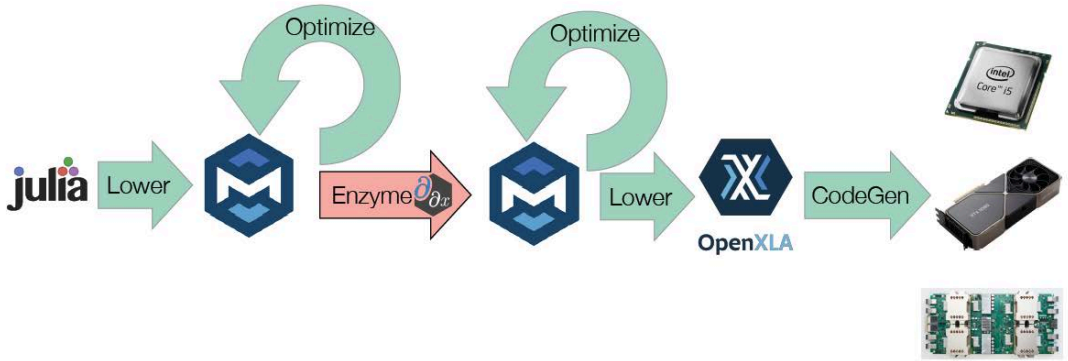


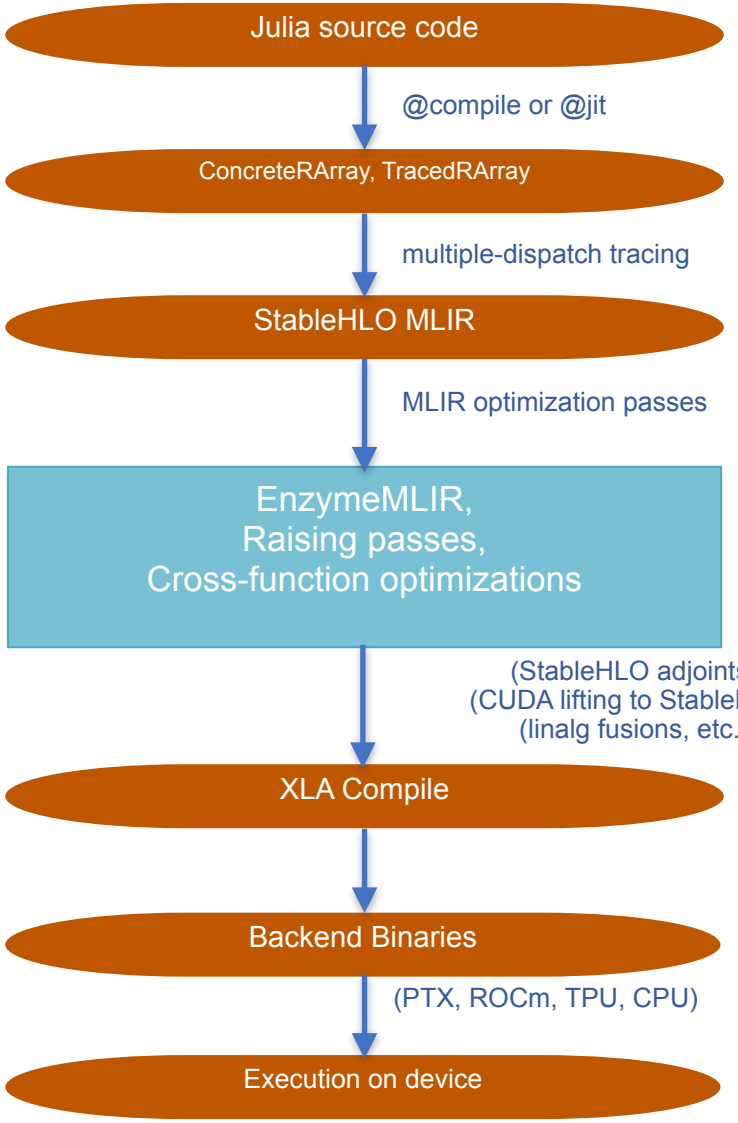
# Enzyme and Reactant

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

**Reactant.jl** (Julia-specific compiler tool - *experimental*):

- Compile Julia into MLIR (multi-level IR), run optimizations on top of it, then create executables via XLA.
- Relies on StableHLO - operation set with ~100 parallel and linear algebra operations.
- XLA is the backend that compiles StableHLO into hardware-specific code (used in JAX, PyTorch, etc.).
- Front end is a tracing system similar to JAX jit, but using Julia's own type system and multiple dispatch instead.
    - Every data structure has a ConcreteRArray (stores actual data) and a TracedRArray (symbolic stand-in to compile MLIR/StableHLO).
- Enzyme can interface with MLIR between StableHLO and XLA compile.
- *Optimized MLIR is type-stable, and easier to differentiate than Julia-LLVM alone.*

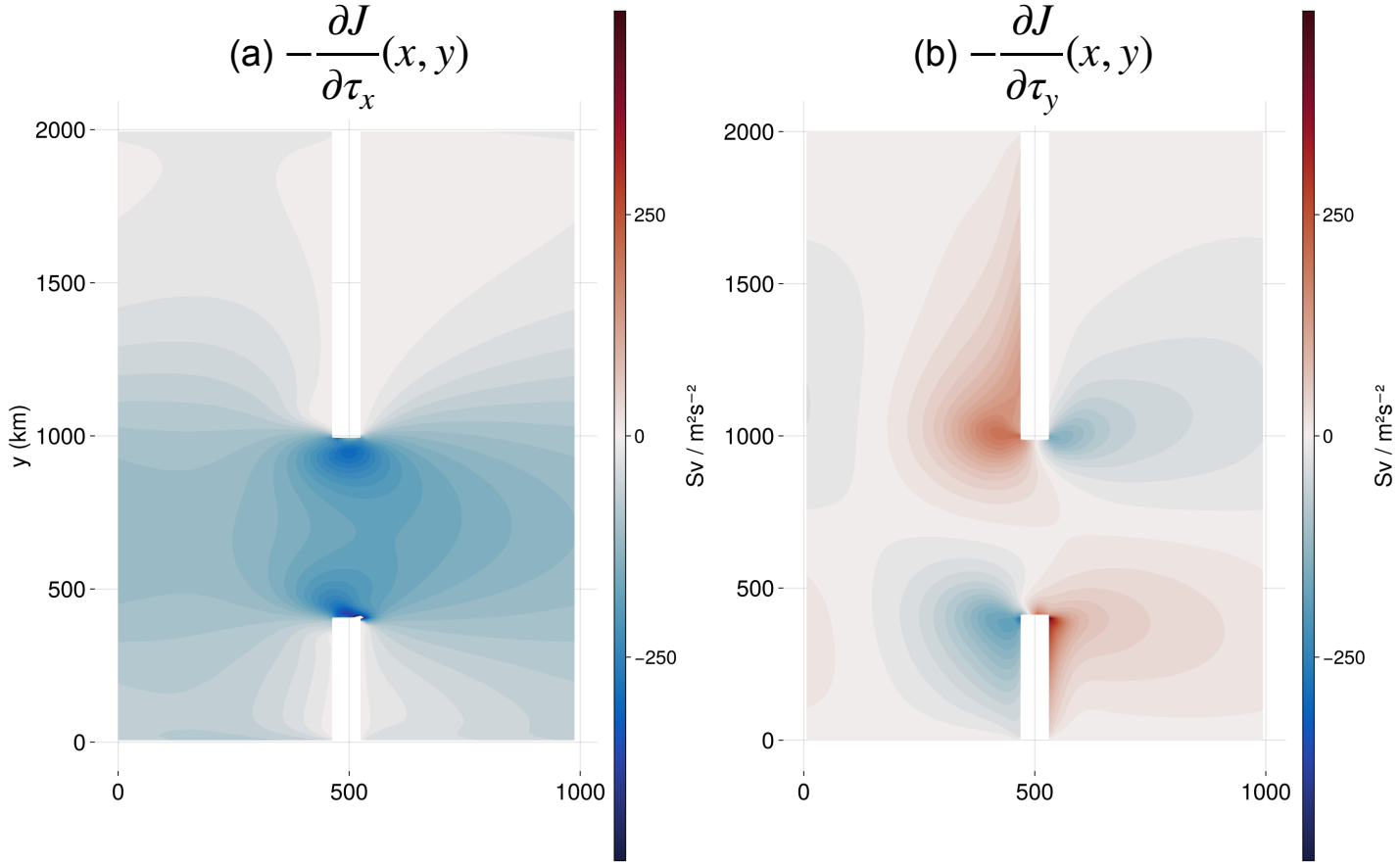


# Case Study: Re-entrant Channel Model

Sensitivities for wind stress' effects on zonal transport:

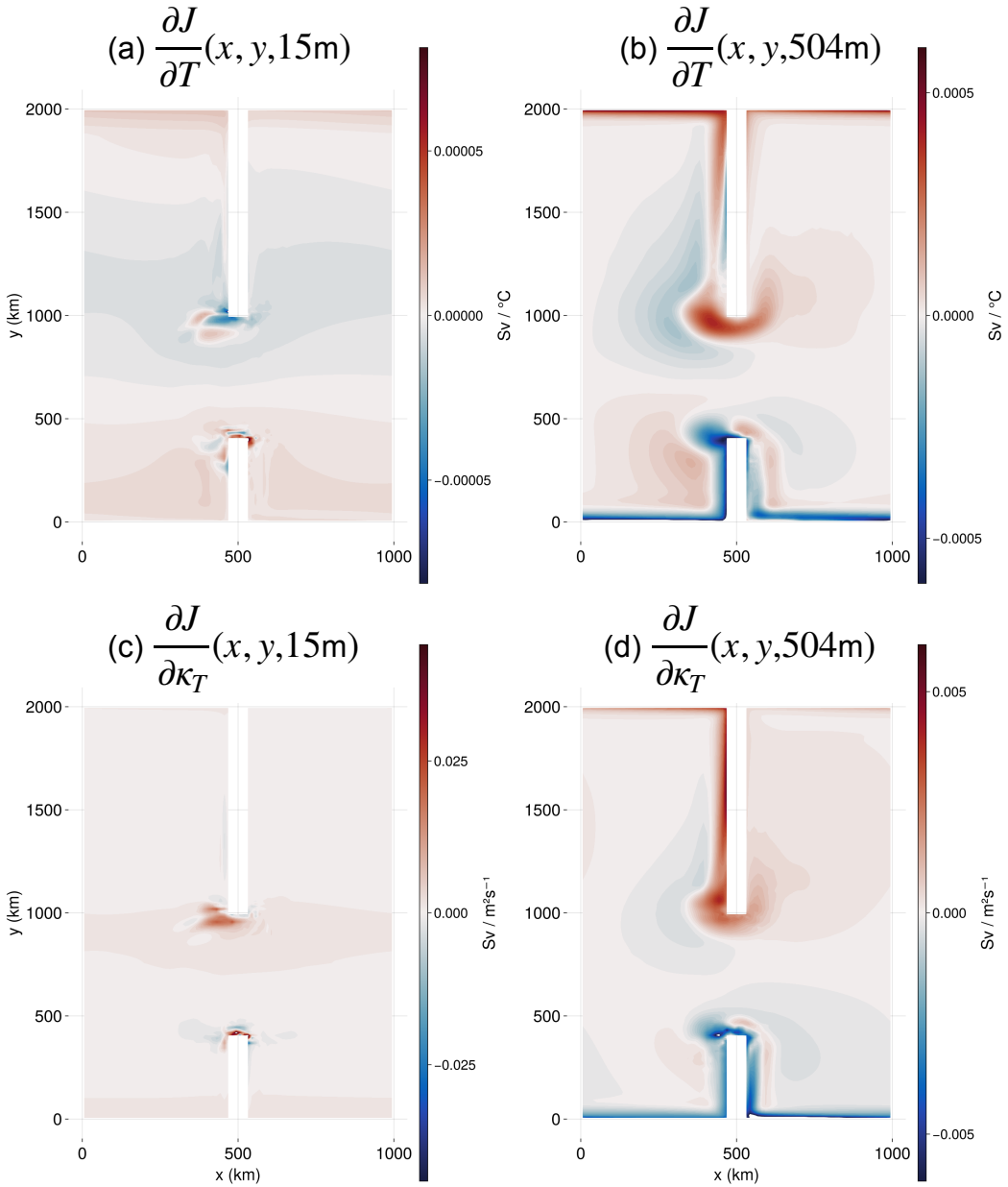
- (a) is **zonal wind stress** $\tau_x$ which alters the meridional density gradient through southern Ekman transport. By thermal wind balance this increases geostrophic velocity $u_g$.
- (b) is **meridional wind stress** $\tau_y$ which controls the pressure difference across the gap via Ekman transport and surface map divergence.




(Oceananigans sign conventions at surface are negative East and North, recall periodic BC.)

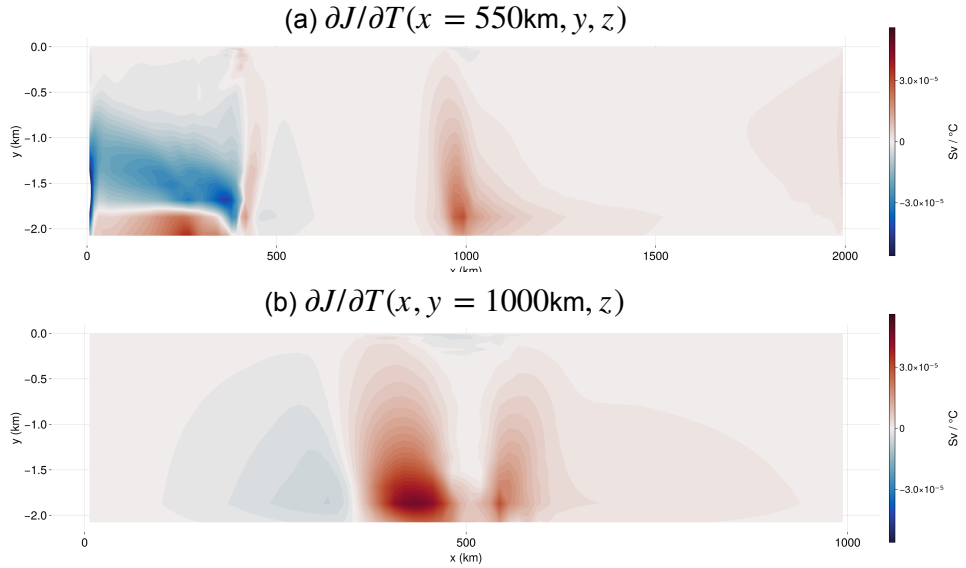
# Case Study: Re-entrant Channel Model

Sensitivities for Temperature $T$ (a and b) and vertical diffusivity $\kappa_T$ (c and d).




How temperature $T$ affects $J$:

- Local warming → steeper meridional density gradient → vertical shear (thermal wind).
- Density gradient raises steric height, which changes horizontal pressure gradients.
- With topography, same horizontal pressure change → larger bottom pressure change, stresses that affect momentum balance.




(Vertical diffusivity $\kappa_T$ closely resembles $T$ over smaller timescales.)

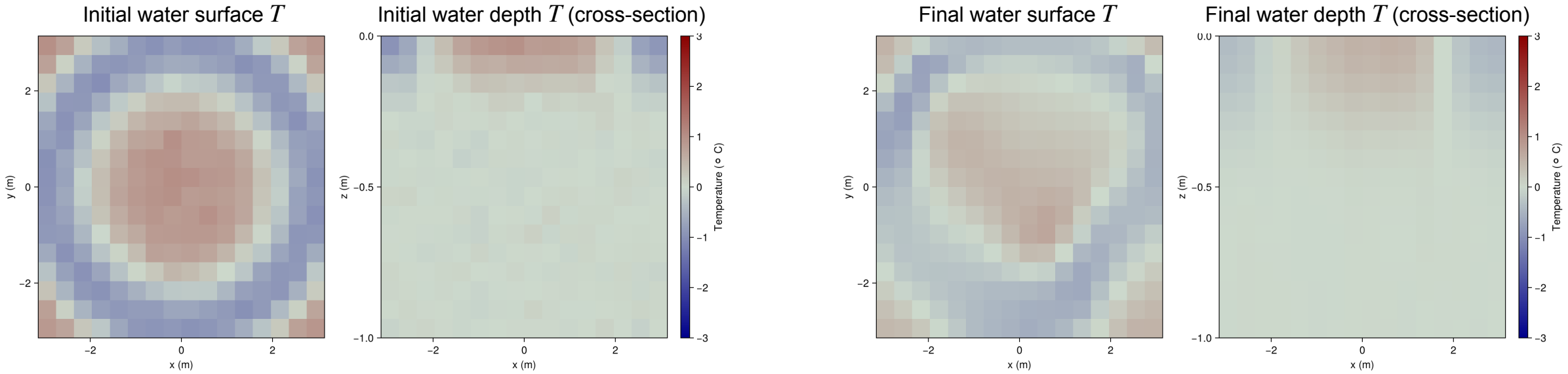
# Case Study: Parameter Estimation Minitutorial

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

- This was a proof-of-concept case for parameter estimation in ocean modeling.
- Used a simpler configuration - barotropic only.
- Included an optional ice floe with thermodynamics and thickness $h$.

Goal: reconstruct the initial temperature field $T_i$ given the final temperature and sea ice thickness at time step $n$ as cost functions, using AD results with a gradient descent:

$$J(\mathbf{u}(\mathbf{x}; T_i)) = \|T_n - T_{\text{true}}\|^2 + \|h_n - h_{\text{true}}\|^2, \text{ with next guess } [T_i]_{k+1} = [T_i]_k - \gamma \nabla J([T_i]_k).$$



# Wait, why is AD *faster* than forward-only for TPUs?

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

| Mode | 25 steps | 100 steps | 400 steps | 1600 steps |
|---|---|---|---|---|
| Primal | $7.258 \times 10^{11}$ | $2.901 \times 10^{12}$ | $1.160 \times 10^{13}$ | $1.991 \times 10^{13}$ |
| AD | $4.433 \times 10^{12}$ | $1.029 \times 10^{13}$ | $1.454 \times 10^{13}$ | $1.716 \times 10^{13}$ |

Table 2: Raw FLOPs performed for Oceananigans primal and reverse-mode AD runs on TPU v6e. The primal scales evenly with timesteps until 400 to 1600 where it increases by only $\approx 72\%$, while the AD run exhibits sublinear scaling and dips below the primal by 1600 steps.

- The v6e TPU has a matrix-multiply-unit (MXU) of $256 \times 256$,
- so we need data that can be chunked into tiles of size 256, or its aggressively padded, which adds wasted FLOPs.
- This is especially true for AD runs with checkpoints steps, backpropagation, redundant forward ops (much more potential loop unrolling).
- $1600 \times$ (size of each model field) is likely divisible by 256 since grid size is $80 \times 160 \times 32$.
- 25, 100, 400 are *not*.

# Some Parameterization Schemes

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

**GM/Redi** for mesoscale eddies *(Gent-McWilliams & Redi schemes - established in 1990s)*

- Represents tracer advection-diffusion from eddies.
- The Redi Scheme: mixes tracer properties along isopycnals (surfaces where the density remains constant).
  - Uses a diffusion operator oriented along the isopycnal.
  - Introduces a symmetric diffusive tensor $\mathbf{K}_{\text{Redi}}$ that projects the tracer onto the surface with coefficient $\kappa_\rho$, using isoneutral slopes $S_x, S_y$.
- The GM scheme: applies the advective effect of eddies. Based on the rotation of a streamfunction specified by the isopycnals.

When combined, GM/Redi's tendency for tracer $\tau$ is

$$\nabla \cdot \left( \kappa_\rho \mathbf{K}_{\text{Redi}} + \kappa_{\text{GM}} \mathbf{K}_{\text{GM}} \right) \nabla \tau, \mathbf{K}_{\text{Redi}} = \begin{bmatrix} 1 & 0 & S_x \\ 0 & 1 & S_y \\ S_x & S_y & S_x^2 + S_y^2 \end{bmatrix}, \mathbf{K}_{\text{GM}} = \begin{bmatrix} 0 & 0 & -S_x \\ 0 & 0 & -S_y \\ S_x & S_y & 0 \end{bmatrix}$$

*Ongoing effort*: determining ideal choice of coefficients $\kappa_\rho, \kappa_{\text{GM}}$. Originally these were scalars, now spatially-varying scalar fields are considered optimal.

# Some Parameterization Schemes

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES

**CATKE** for turbulent mixing *(Convective-Adjustment Turbulent Kinetic Energy - devised in 2023)*

- Uses a prognostic TKE tracer and diagnostic mixing length scale the features dynamic convective adjustment.
- Predicts depth range and timescale over which mixing occurs.

Start with a column model for a horizontally-averaged tracer (horizontal velocities $u, v$ are similar but with a Coriolis term):

$$\frac{\partial}{\partial t}\tau = -\frac{\partial}{\partial z}\left(\overline{w'\tau'}\right)$$

Here $w', \tau'$ are fluctuations from the averaged quantity - the change in $\tau$ is based on vertical fluctuations only. Then we model the horizontally-averaged vertical fluxes with

$$\overline{w'\tau'} \approx -\kappa_\tau \frac{\partial}{\partial z}\tau \text{ where } \kappa_\tau = \ell_\tau \sqrt{e},$$

$e$ is TKE, $\sqrt{e}$ is turbulent velocity scale, $\ell_\tau$ is mixing length.

- TKE is modeled analogously to kinetic energy.
- Mixing length modeled after shear or convective turbulence depending on GCM.
- Separate vertical flux models for velocities, tracers, and TKE. All added as tendencies to these components.

CATKE originally used *gradient-free* Ensemble Kalman Inversion to train 18 total free parameters, concurrently running 21 large eddy simulations on a GPU, horizontally-averaged into column models. It was written for Oceananigans.jl, *with GPU implementation in mind*.

*Ongoing effort:* determining ideal way to train parameters ($\sqrt{e}$, $\ell_\tau$, resulting free parameters in their derivations).