Specification and Verification for Climate Modeling

Formalization Leading to Impactful Tooling

Alper Altuntas[1], Allison Baker[1], John Baugh[2], Ganesh Gopalakrishnan[3] and Stephen Siegel[4]

[1]NSF National Center for Atmospheric Research (NCAR),
[2]North Carolina State University, [3]University of Utah, [4]University of Delaware

May 4th, 2025

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Outline

1. Climate Modeling – A Quick Overview
2. Verification and Validation – Challenges, Current Practices
3. Toward Formal Specification for Climate Models

1. Climate Modeling – A Quick Overview

Terminology

- **Climate:** Average weather over a long period of time.
- **Climate Model:** HPC software that simulates Earth's climate.
- **Earth System:** atmosphere, ocean, land, and ice.
- **Earth System Model (ESM):** *see* Climate Model.
- **CESM:** NSF NCAR's flagship ESM.

NSF National Center for Atmospheric Research (NCAR)

A Federally Funded Research and Development Center for Earth System Science.

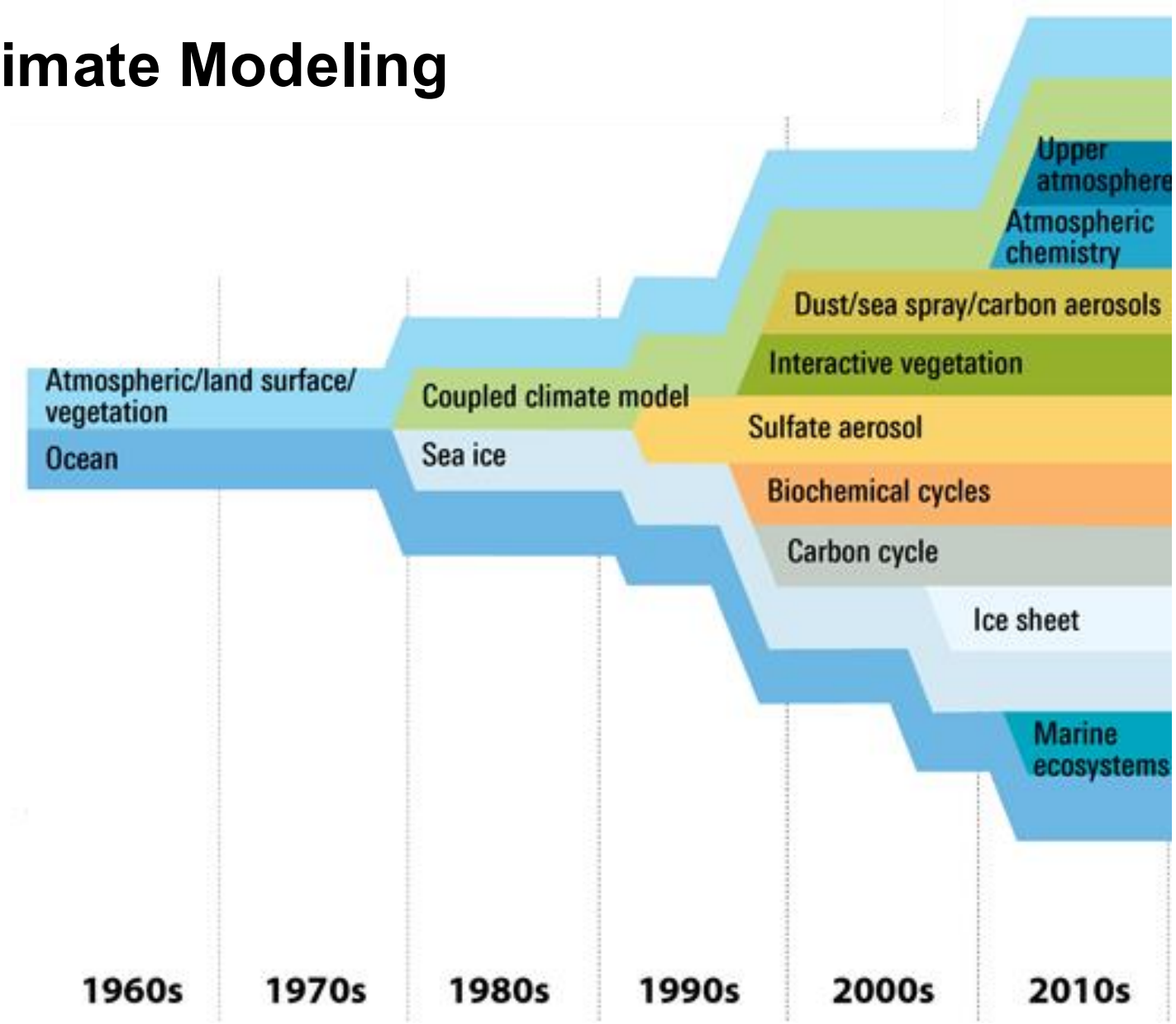
We support the research community by providing advanced computer models, observations, tools, and datasets.

Mesa Lab, Boulder CO

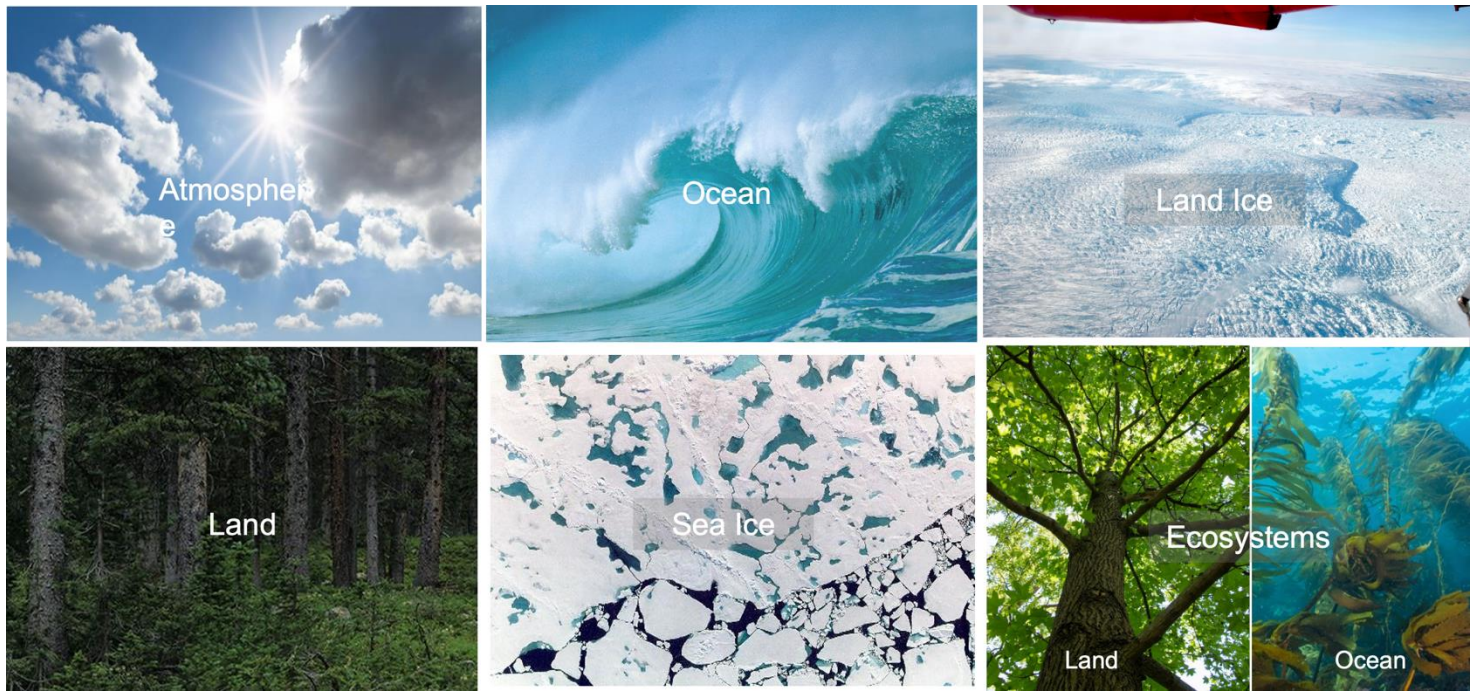
Derecho, a 20-petaflop system

Research Aviation Facility

Growth of Climate Modeling



Components of ESMs



Each component:

- Solves governing equations *(e.g., fluid dynamics, thermodynamics)*
- Exchanges data *(e.g., atmosphere <-> ocean)*

Ocean Component – Governing Equations

$$\rho_o \left[\partial_t \mathbf{u} + w\,\partial_z \mathbf{u} + (f + \zeta)\hat{\mathbf{z}} \times \mathbf{u}\right] = -\nabla_z (p + K) - \rho\,\nabla_z \Phi + \rho_o \square \quad \text{vector-inv horz velocity}$$

$$\rho\,\partial_z \Phi + \partial_z p = 0 \quad \text{hydrostatic}$$

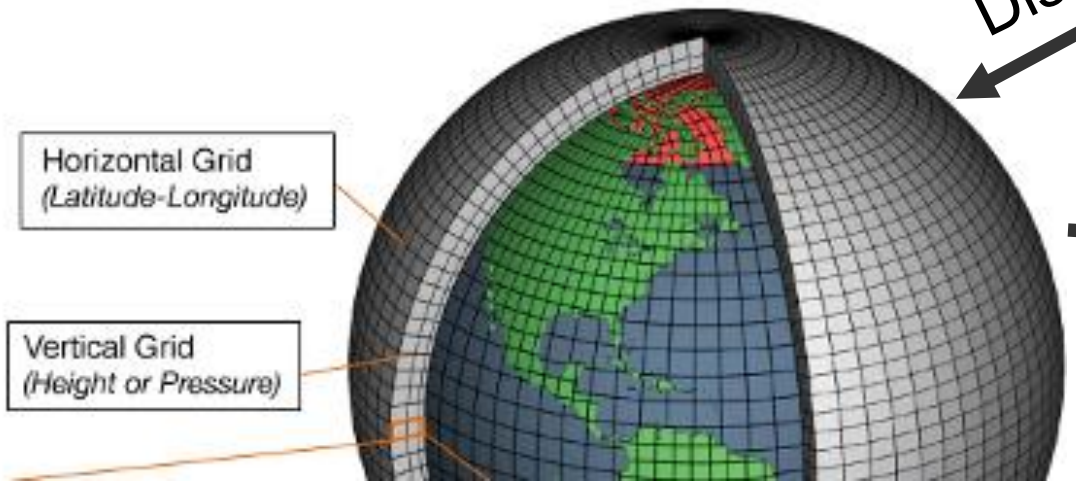
$$\nabla_z \cdot \mathbf{u} + \partial_z w = 0 \quad \text{continuity}$$

$$\partial_t \theta + \nabla_z \cdot (\mathbf{u}\,\theta) + \partial_z (w\,\theta) = \square\,{}^{\gamma}_{\theta} - \partial_z J^{(z)}_{\theta} \quad \text{potential/Cons temp}$$

$$\partial_t S + \nabla_z \cdot (\mathbf{u}\,S) + \partial_z (w\,S) = \square\,{}^{\gamma}_{S} - \partial_z J^{(z)}_{S} \quad \text{salinity}$$

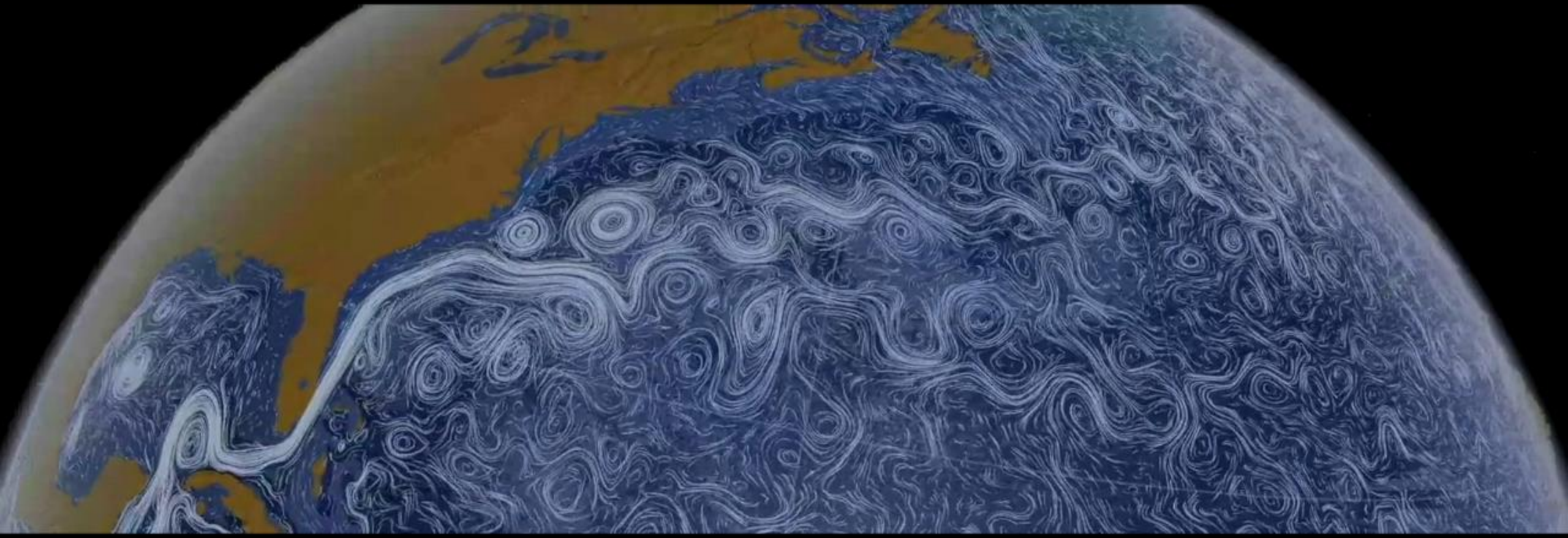
$$\rho = \rho(S, \theta, z) \quad \text{equation of state.}$$

Discretize



+ parameterizations
+ computational routines

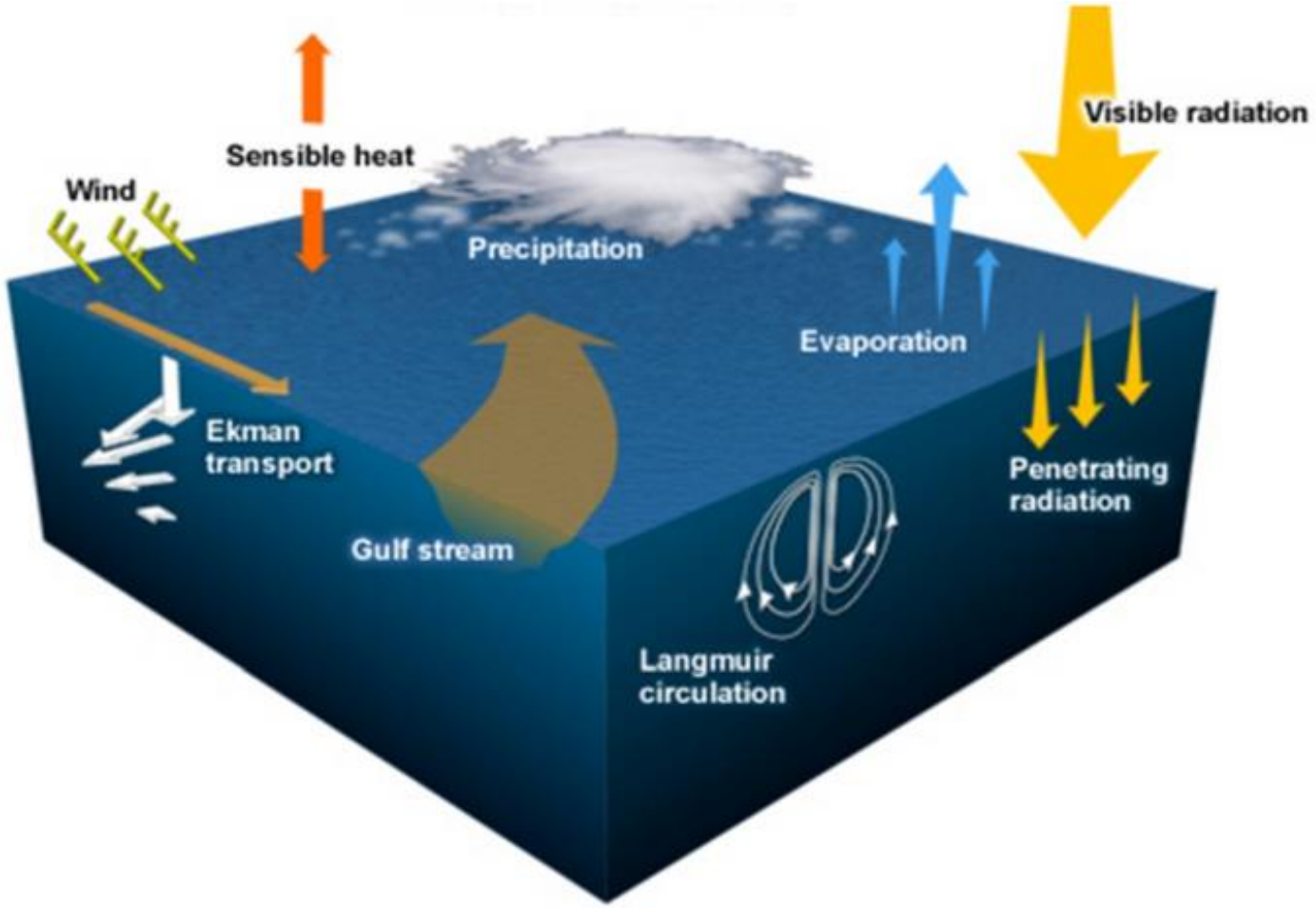
MOM6
160K LOC Fortran



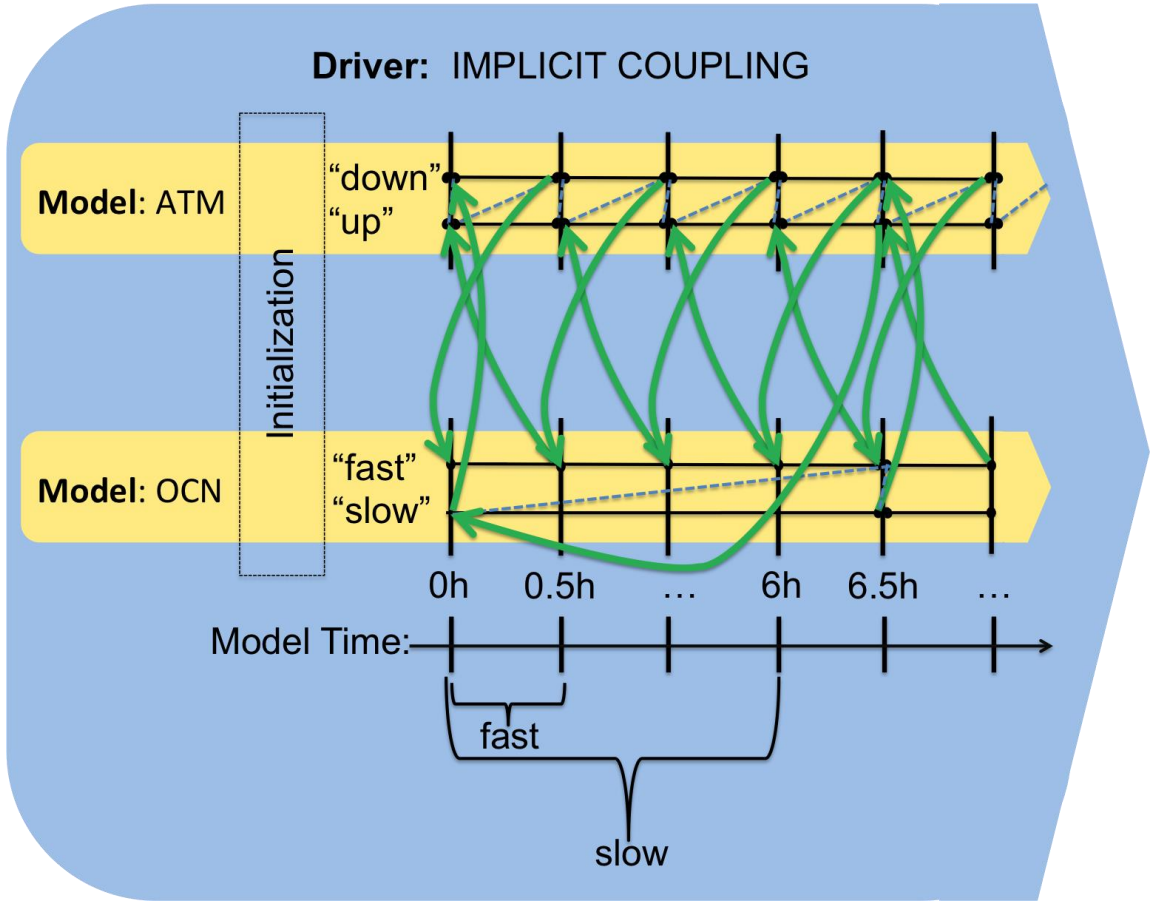
Visualization of eddy-resolving ocean model output (ECCO2) showing Gulf Stream currents and heat transport.

NASA/Goddard Space Flight Center Scientific Visualization Studio

Atmosphere – Ocean Coupling



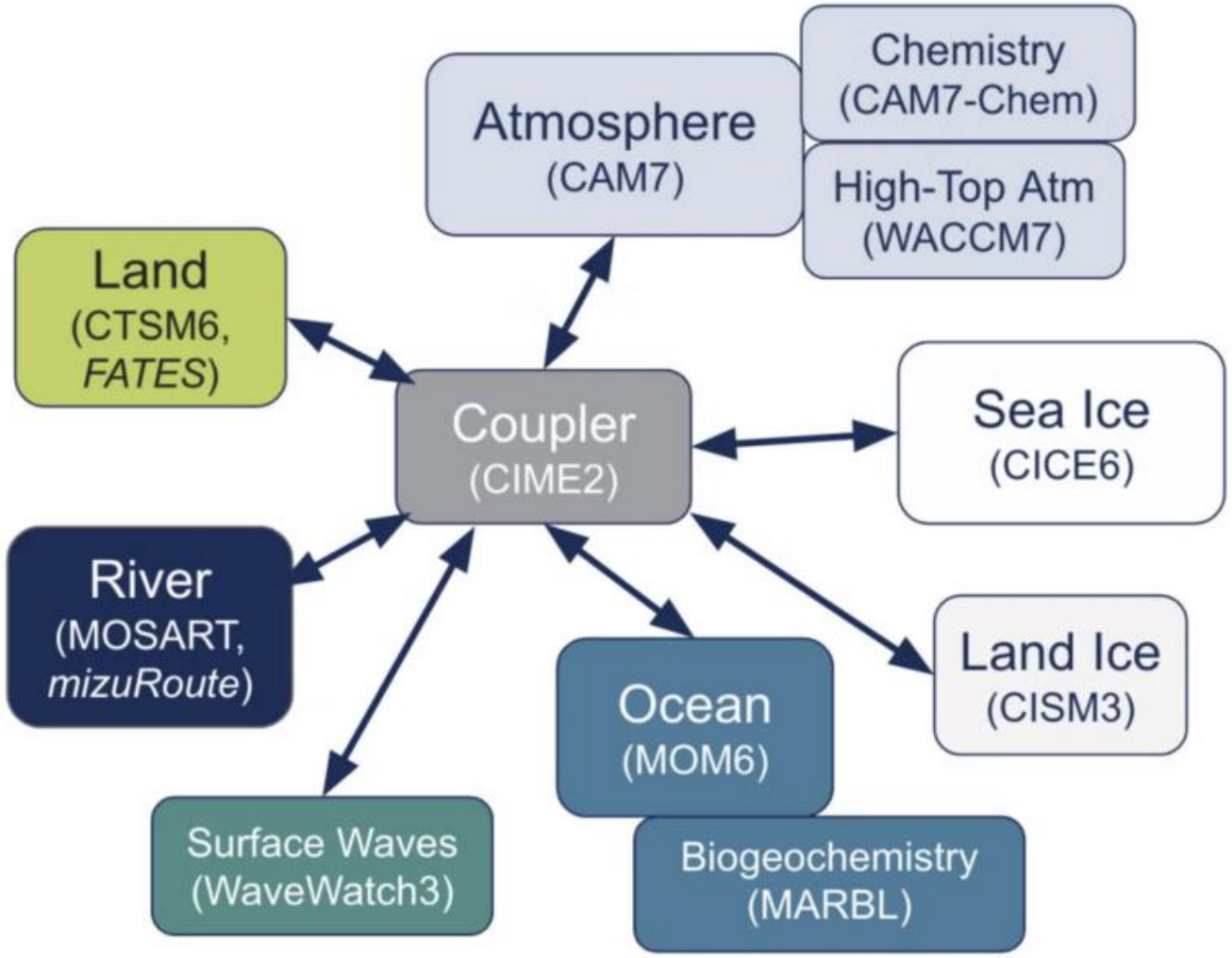
Atmosphere – Ocean Coupling, Data Flow



After each atmospheric "down" sweep, fast ocean processes must run before the "up" sweep. This repeats until slower ocean processes are triggered, which rely on ATM fields.

https://earthsystemmodeling.org/nuopc/

CESM Architecture



2. Verification and Validation – Challenges and Current Practices

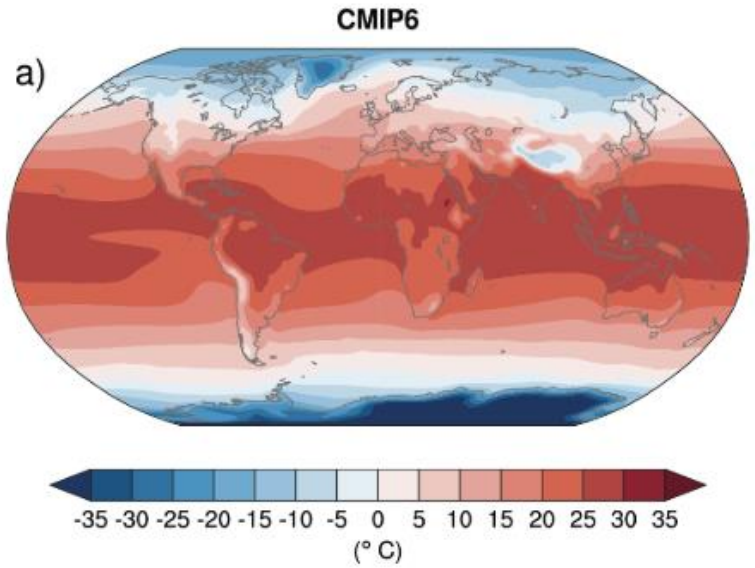
(a) Scientific, (b) Software

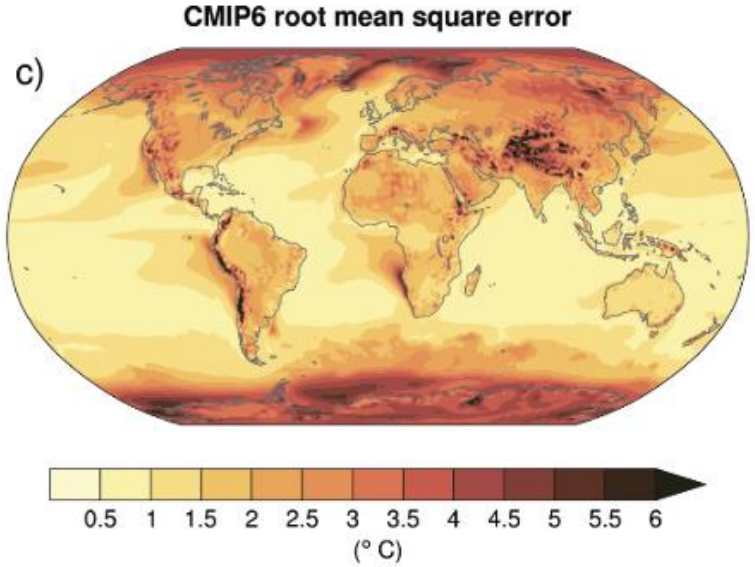
Scientific Validation

- Hindcasting
- Analysis
- Benchmarks
- Intercomparisons e.g., CMIP6

These methods generally require substantial resources and/or manual expert analysis (subjective).

Near Surface Air Temperature (1995-2014) – Ensemble Mean





Software Verification & Validation

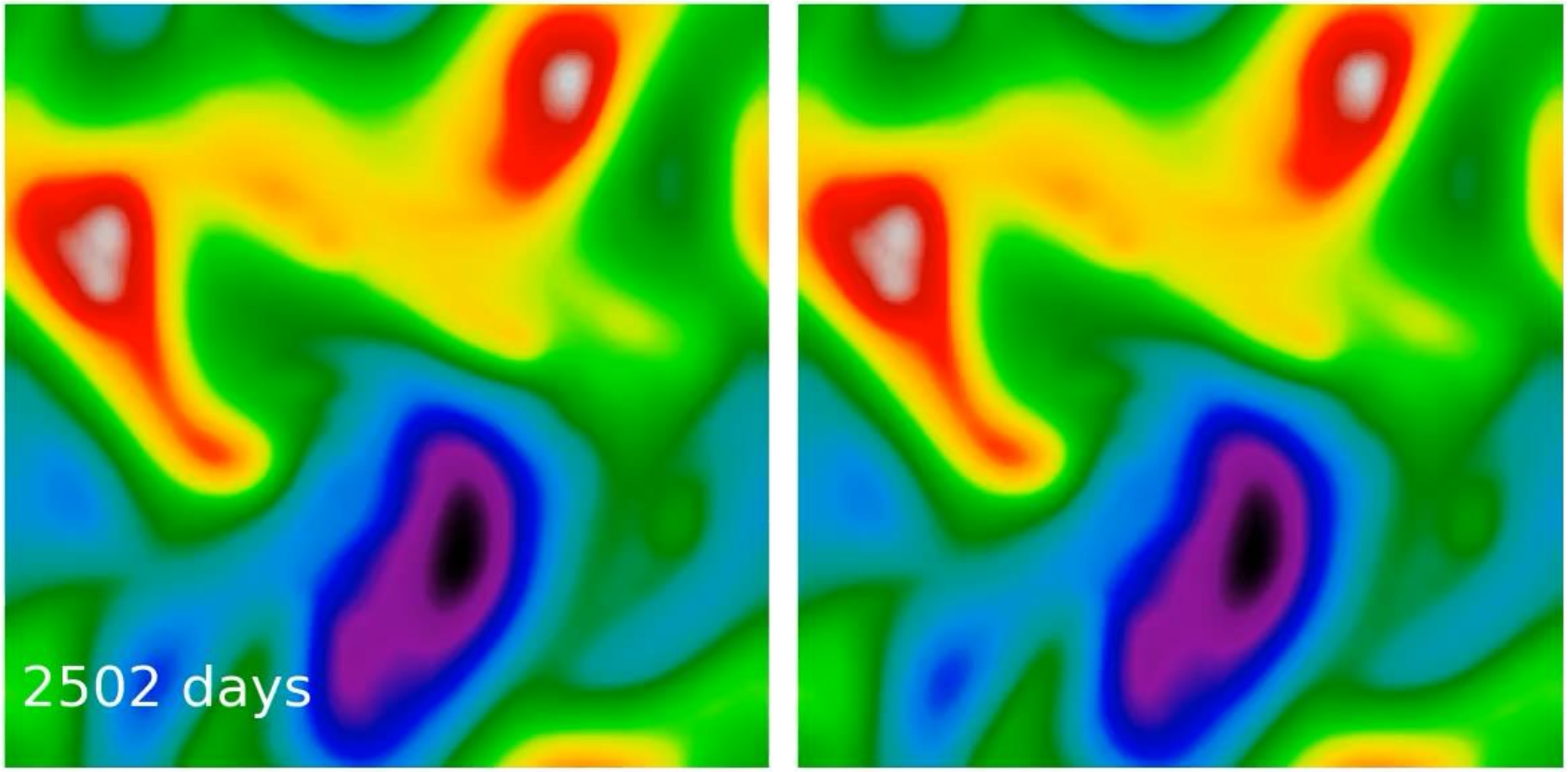
Backbone: Regression tests

- Run test suite to check various aspects and different combination of model configurations and software environments.

We seek:

- Bitwise identical results (to the extend possible).
- Exact restart reproducibility.
- No answer changes across different processor counts.

Bitwise Reproducibility – Why?



Marshall Ward, “The MOM6 Development Cycle” (2023)

Bitwise Reproducibility – How?

Some methods used in MOM6 (ocean component):

Addition

Addition operations must be done in pairs. When more than one addition is required, the order should be specified using parentheses.

- This is bad:
  - `z = a + b + c`
- This is good:
  - `z = (a + b) + c`

We avoid the Fortran `sum()` function since the result is dependent on the order of operations within the summation. Using explicit loops allows us to define the order of summation. So

```
a = sum(b(:))
```

should be

```
a = 0.
do k = 1, nz
  a = a + b(k)
```

Global summation

Floating point operations across MPI ranks are volatile, since the order can change depending on the state of the network. Functions such as `MPI_Reduce` will not generally be reproducible when used for floating point arithmetic.

When performing summations over MPI ranks, use the `reproducing_sum` function.

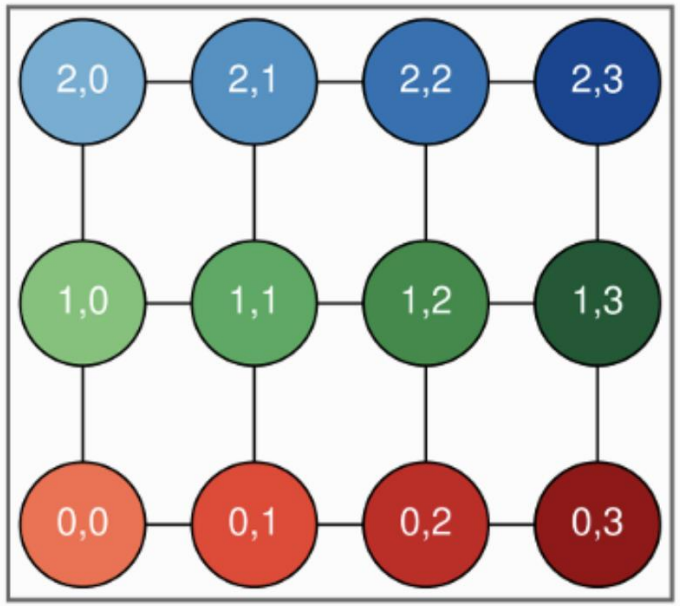
```
use MOM_coms, only: reproducing_sum
...
sum = reproducing_sum(array(:,:))
```

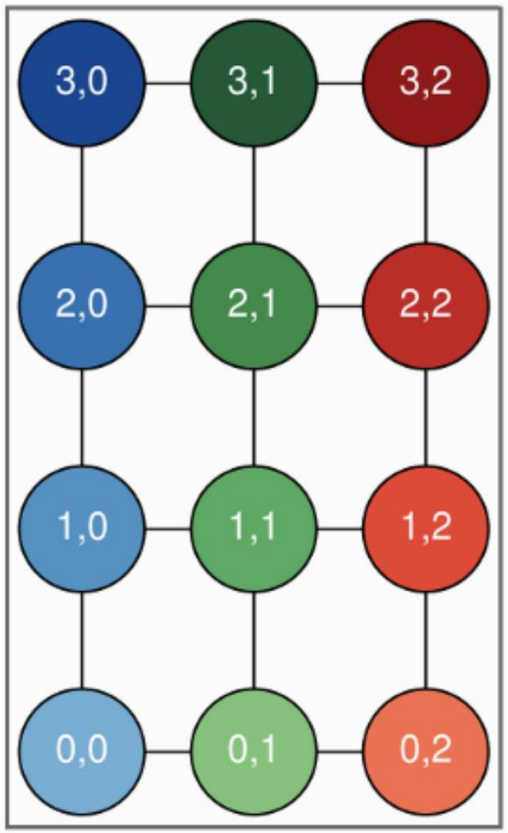
Trascendental functions

Use of transcendental functions, such as trigonometric functions, non-integer powers, and logarithms, are often implementation-dependent and should be avoided when possible.

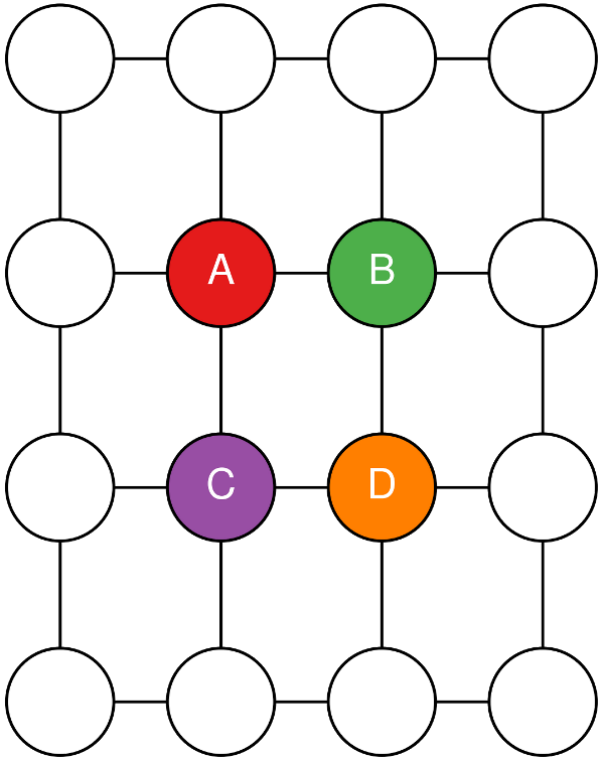
Rotational Symmetry Test in MOM6

By setting the runtime option ROTATE_INDEX to True, the model rotates the domain by some number of 90 degree turns. This option can be used to look for bugs in which east-west operations do not match north-south operations. It changes the order of array elements as shown here:





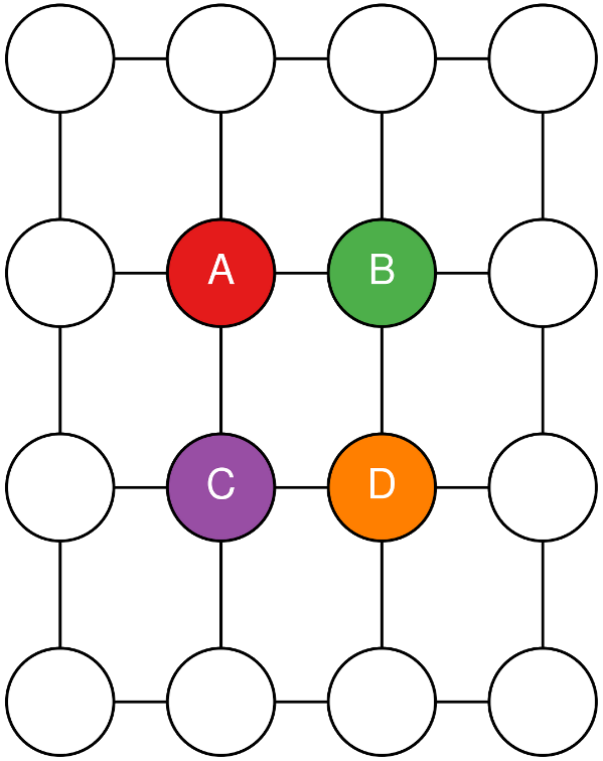
Rotational Symmetry Test in MOM6



$$\phi_{i,j}^{(c)} = \frac{1}{4}(\phi_A + \phi_B + \phi_C + \phi_D)$$

$$\phi_{i,j}^{(c)} = \frac{1}{4}((\phi_A + \phi_B) + (\phi_C + \phi_D))$$

Rotational Symmetry Test in MOM6



$$\phi_{i,j}^{(c)} = \frac{1}{4}(\phi_A + \phi_B + \phi_C + \phi_D)$$

$$\phi_{i,j}^{(c)} = \frac{1}{4}((\phi_A + \phi_B) + (\phi_C + \phi_D))$$

$$\phi_{i,j}^{(c)} = \frac{1}{4}((\phi_A + \phi_D) + (\phi_B + \phi_C))$$

(*)Add parentheses for FMA rotational symmetry #1634

Merged **Hallberg-NOAA** merged 30 commits into `mom-ocean:main` from `Hallberg-NOAA:FMA_rotational_symmetry_main` on Aug

Conversation 11 | Commits 30 | Checks 10 | Files changed 54

Hallberg-NOAA commented on Jul 30, 2024 • edited — Collaborator

This series of 30 commits adds parentheses in 54 files throughout the MOM6 code base, or (in a few cases) revises the order of sums in expressions for various u- and v-component calculations, so that all answers exhibit rotational symmetry when fused-multiply-adds (FMAs) are enabled and appropriate parameter settings are chosen.

```
726          do j=js,je ; do I=Isq,Ieq
727            CAuS(I,j,k) = 0.25 * &
728 -                (qS(I,J) * (vh(i+1,J,k) + vh(i,J,k)) + &
729 -                 qS(I,J-1) * (vh(i,J-1,k) + vh(i+1,J-1,k))) *
      G%IdxCu(I,j)
730          enddo ; enddo
```

```
726          do j=js,je ; do I=Isq,Ieq
727            CAuS(I,j,k) = 0.25 * &
728 +                ((qS(I,J) * (vh(i+1,J,k) + vh(i,J,k))) + &
729 +                 (qS(I,J-1) * (vh(i,J-1,k) + vh(i+1,J-1,k))))
      * G%IdxCu(I,j)
730          enddo ; enddo
```

Reproducibility vs Performance

- **We limit compiler optimizations like FMA**
  To preserve exact results, we restrict reordering of expressions.
- **Manual control of arithmetic operations**
  Parentheses placement and explicit order of operations.
- **Reproducing Sums and Reductions!**
- **Our priority: correctness/reproducibility over performance**
  Bit-for-bit reproducibility takes precedence in our workflows.

When Bitwise Reproducibility isn't Feasible

How to check correctness when exact reproducibility isn't possible:

- New scientific modules
- Updated compilers or toolchains
- New hardware / systems
- Aggressive optimizations
- Any other major changes

Ensemble Consistency Test (ECT)

Question: Is the new answer correct? (not well-defined)

Alternative question: Is it statistically distinguishable from the original?

ECT approach: evaluate in the context of the climate model's variability

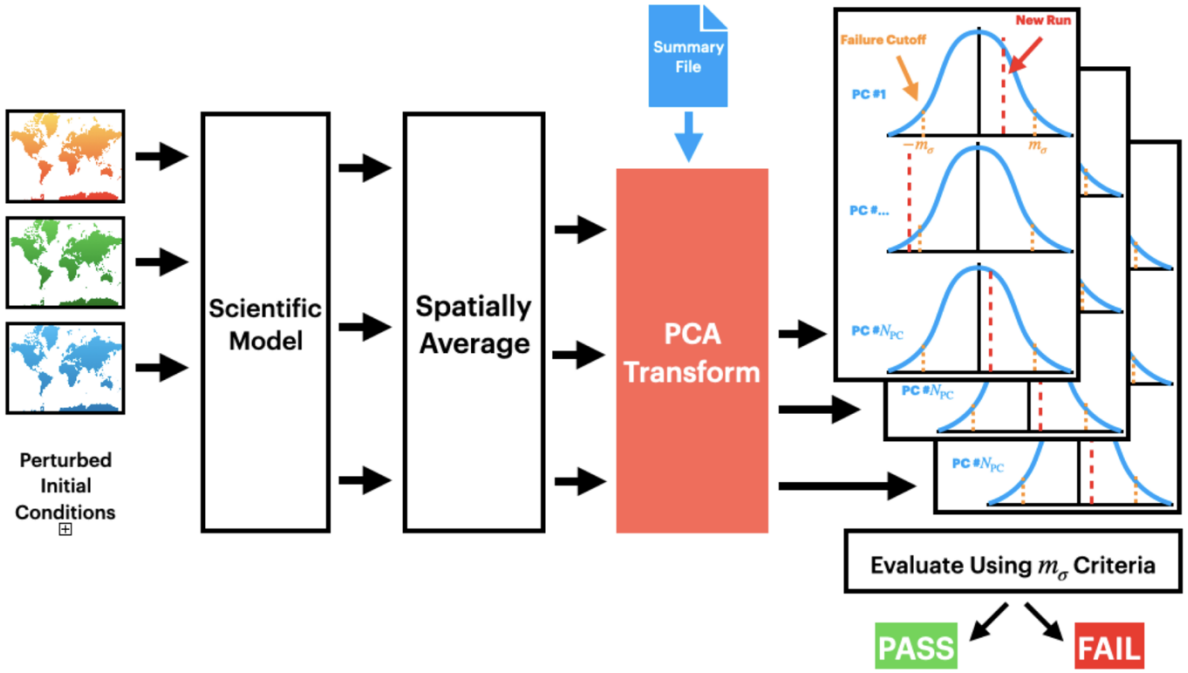
(climate scientists think of uncertainty through the use of ensembles)

https://github.com/NCAR/PyCECT

Ensemble Consistency Test (ECT)

1. **Build a Reference Ensemble**
    1. Use a trusted hardware/software stack.
    2. Apply small perturbations to initial conditions ($O(10^{-14})$).
    3. Perform short model runs (e.g., 9 timesteps).

2. **Test New Configuration**
    1. Run the updated model / environment.
    2. Compare output against ensemble.
    3. Check if results are **statistically distinguishable.**



https://github.com/NCAR/PyCECT

3. Toward Formal Specification for ESMs

and a case study

Toward Formal Specification for ESMs

Motivation:

- Traditional testing is computationally expensive and incomplete.
- Manual validation is labor-intensive and subjective.
- ECT is very helpful for validating porting and optimizations, but necessitates a baseline.

Goal:

- A domain-aware, relational spatio-temporal logic for ESMs to allow property checking, along the lines of lightweight formal methods.

Targets for Specification and Verification

1. Model Coupling, Concurrency, and Parallelism
2. **Continuous and Instantaneous Processes**
3. **Spatial Representations**

Continuous and Instantaneous Processes

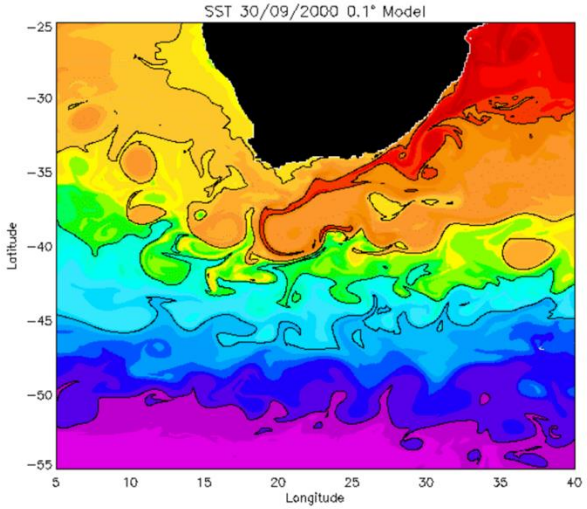
A key aspect of our proposed specification approach is the distinction and the interplay between continuous and instantaneous processes:

1. **Dynamical core**, i.e., governing equations (PDEs)
2. **Parameterizations** to represent unresolved processes
3. **Computational routines**

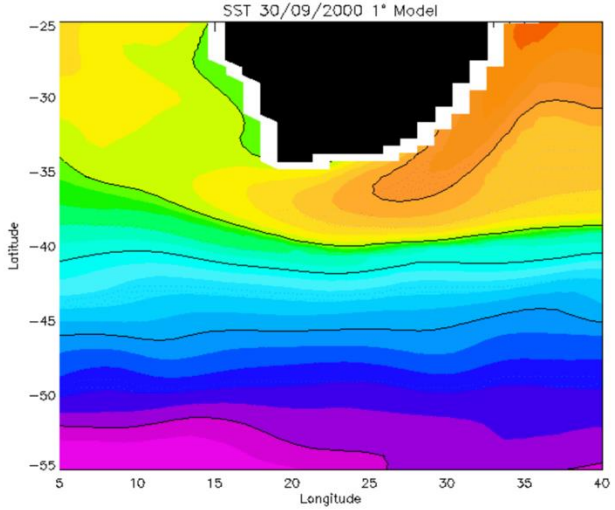
continuous

discrete

Δx = 0.1 degree

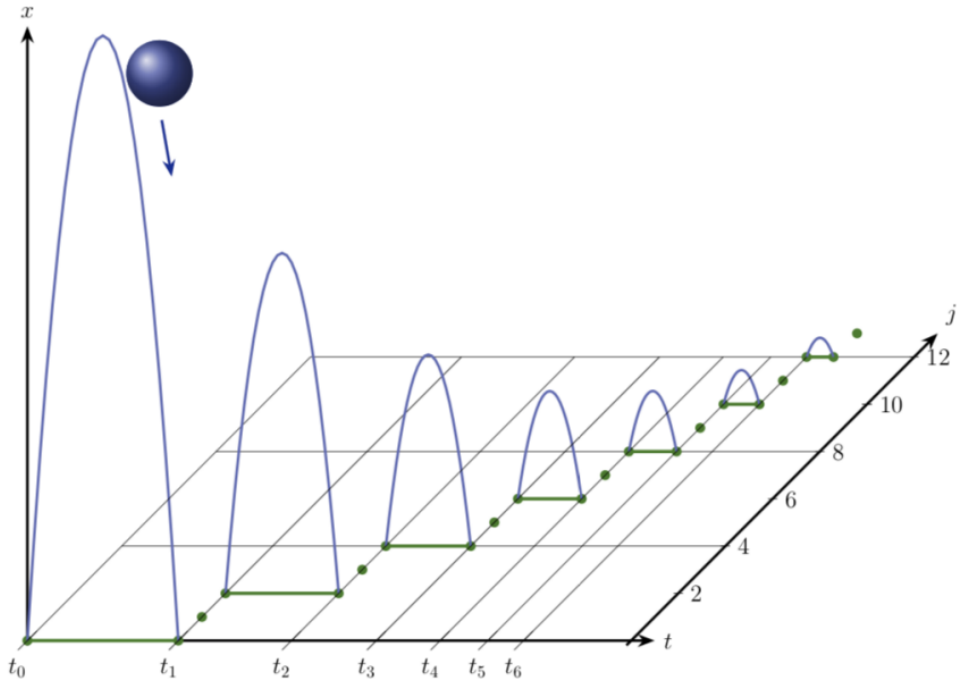


Δx = 1.0 degree



Continuous and Instantaneous Processes

To specify and reason about continuous and instantaneous processes, we draw inspiration from cyber-physical systems (CPS) domain, where continuous evolution is represented by ordinary differential equations (ODEs) while abrupt changes are represented by discrete assignments.



Platzer, A. "Logical Foundations of Cyber-Physical Systems." (2018)

- Continuous evolution:

$$\frac{\partial x}{\partial t} = v, \frac{\partial v}{\partial t} = -g$$

- Discrete change:

$$v := -v$$

Hybrid programming (CPS)

A hybrid model of bouncing ball (in KeYmaera X)

$$
\begin{array}{lll}
 & (0 \leq x \leq x_{max}) \rightarrow & \textit{initial conditions} \\
 & \left[\{\right. & \textit{execution begins} \\
\text{over some } \Delta t \left\{\right. & \quad \{x' = v, v' = -g \quad \&(x \geq 0)\ \} & \textit{continuous evolution} \\
\text{instantaneous} \left\{\right. & \quad ?(x = 0)\ v := -cv; & \textit{discrete assignment} \\
 & \left.\}^*\right] & \textit{loop or terminate} \\
 & (0 \leq x \leq x_{max}) & \textit{postcondition}
\end{array}
$$

where x is altitude, v is velocity, a is acceleration.

Altuntas and Baugh (2018)

Hybrid programming for numerical software

We argue that numerical models can be viewed as cyberphysical systems so as to abstract away from certain aspects and instead focus on higher-level algorithmic correctness.

Take, shallow water equations:

$$\eta' = -\frac{\partial uh}{\partial x}, \ldots$$

- In an **actual numerical model**, discretize both in time and space:

$$\frac{\eta_i^{n+1} - \eta_i^n}{\Delta t} = \left((uh)_{i+1}^n - (uh)_{i-1}^n\right)/(\Delta x), \ldots$$

- In a **hybrid verification model**, discretize in space only:

$$\eta' = \left((uh)_{i+1} - (uh)_{i-1}\right)/(\Delta x), \ldots$$

where η is water elevation, h is water height, u is velocity.

Altuntas and Baugh (2018)

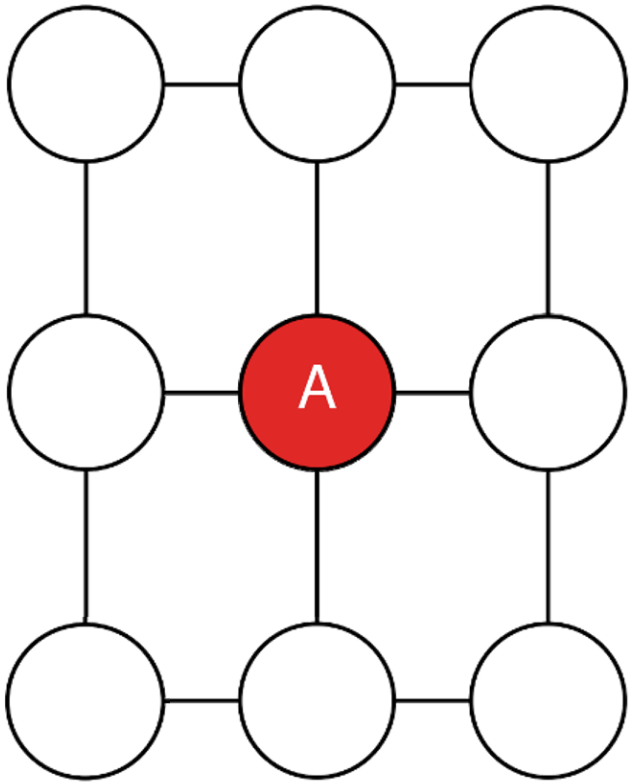
Mesh Representations

Specify the smallest necessary discrete grid:

- Key idea: Limit grid to local domain of dependence
- Focus: A single grid cell and its immediate neighbors

Why Minimal Grid Works:

- PDEs have limited domain of dependence (CFL condition)
- Nondeterminism and compositional reasoning can be utilized for external inputs/behavior.



Case Study - A bug in a MOM6-CESM parameterization scheme

The KPP scheme

- The continuous evolution of a scalar quantity λ over a vertical water column:

$$\frac{\partial \overline{\lambda}}{\partial t} = \frac{\partial}{\partial z}(\overline{w'\lambda'} + \overline{w}\,\overline{\lambda})$$

- The unresolved turbulent flux parameterized as a diffusive process (Griffies et al., 2015):

$$\overline{w'\lambda'} = -K_\lambda(\frac{\partial \overline{\lambda}}{\partial z} + \gamma_\lambda)$$

- The diffusivity K_λ at depth d within the OBL (Large et al., 1994):

$$K_\lambda = h \cdot w_\lambda(\sigma) \cdot G_\lambda(\sigma)$$

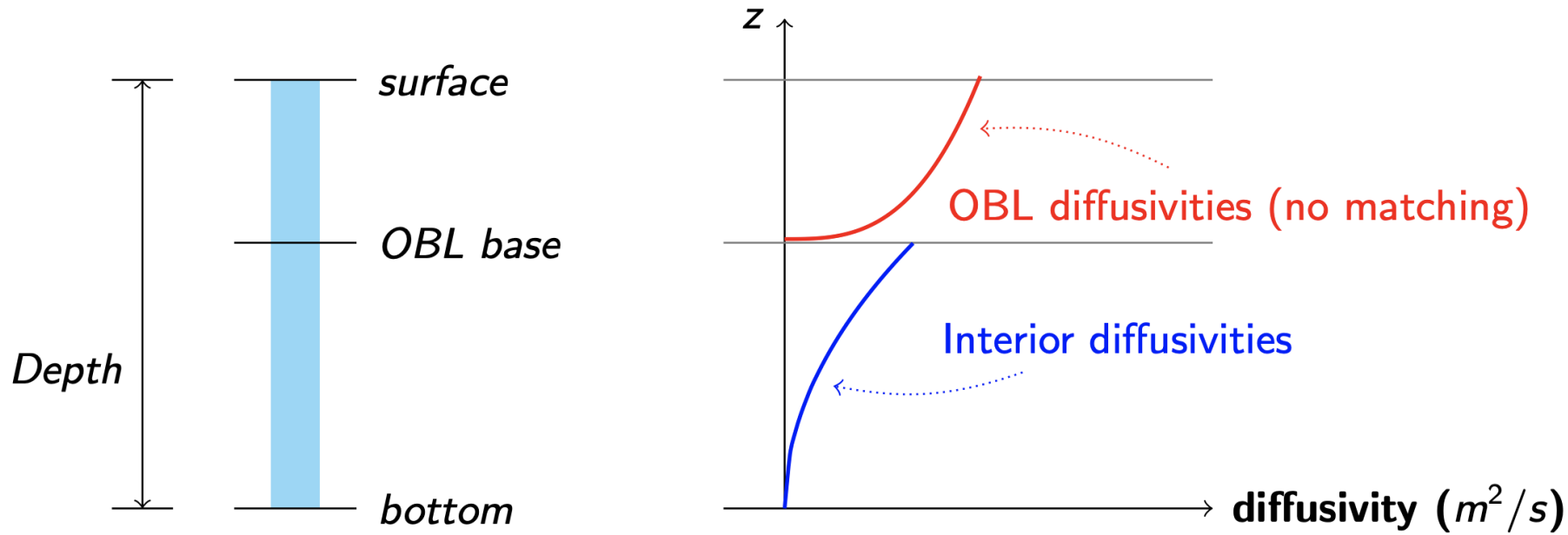
- Shape function for the OBL diffusivities:

$$G_\lambda(\sigma) = a_0 + a_1\sigma + a_2\sigma^2 + a_3\sigma^3$$

Case Study

The KPP scheme

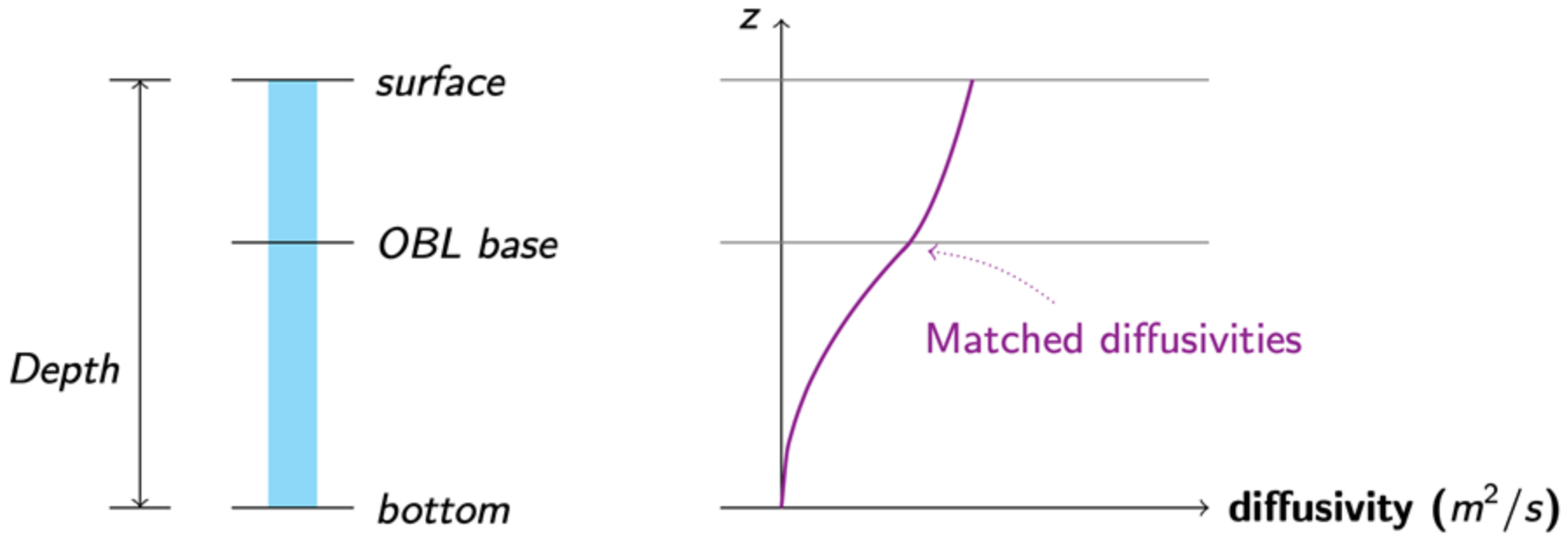
- Compute the ocean boundary layer (OBL) depth.
- Compute the diffusivities within ocean interior.
- Compute the OBL diffusivities (no matching).



Case Study

The KPP scheme

- Compute the ocean boundary layer (OBL) depth.
- Compute the diffusivities within ocean interior.
- Compute the OBL diffusivities **(matching)**.



Case Study

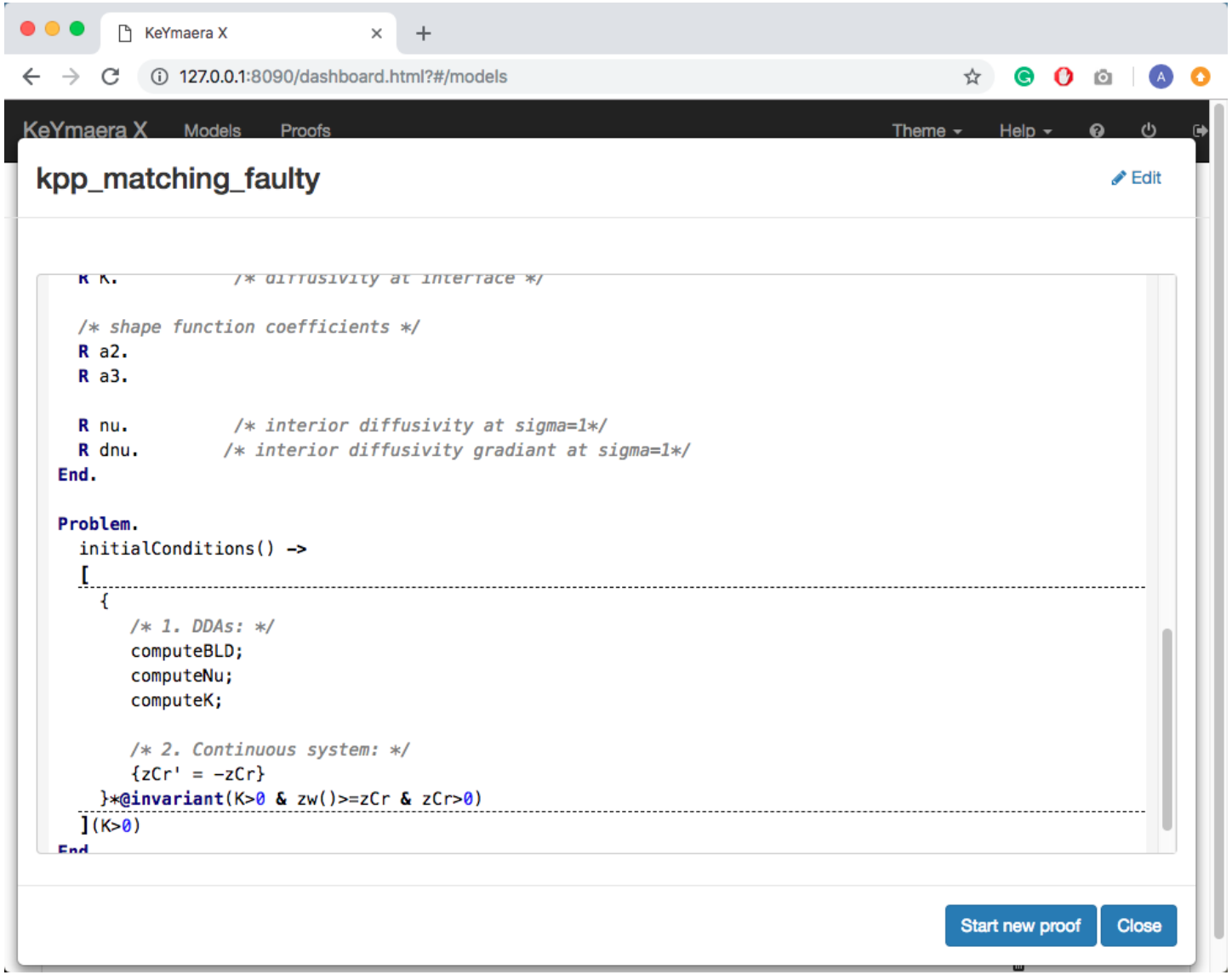
Undesired behavior:
- Negative diffusivities encountered in OBL.

Debugging:
- Could only reproduce with the full-scale global configuration.
- Pinpointed issue via an interactive debugger.
- Took several days and many thousands of CPU hours.

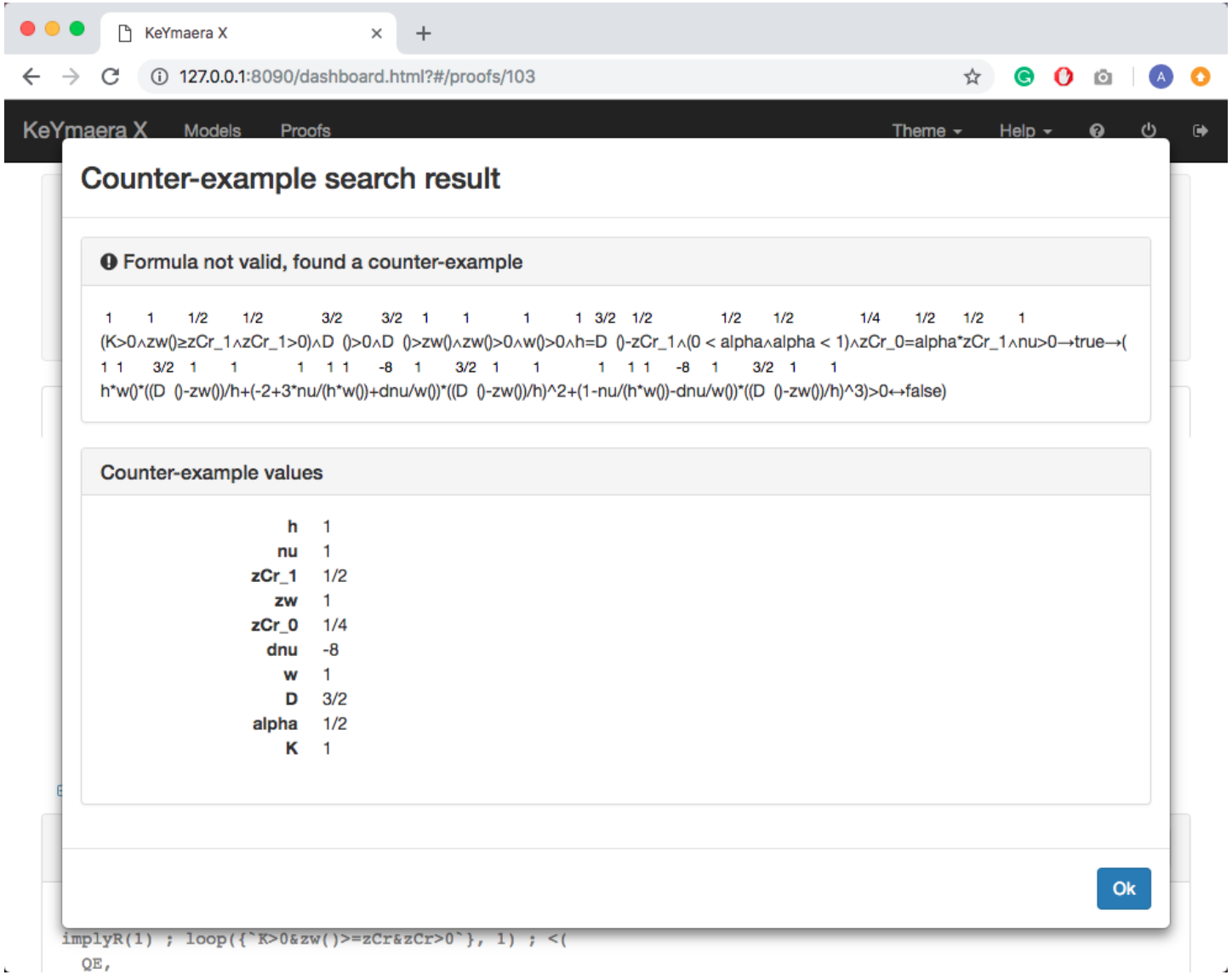
Fix:
- Modify the matching algorithm for cases where the interior diffusivity gradient is negative.

Verification:
- A KeYmaera X model of the KPP scheme.

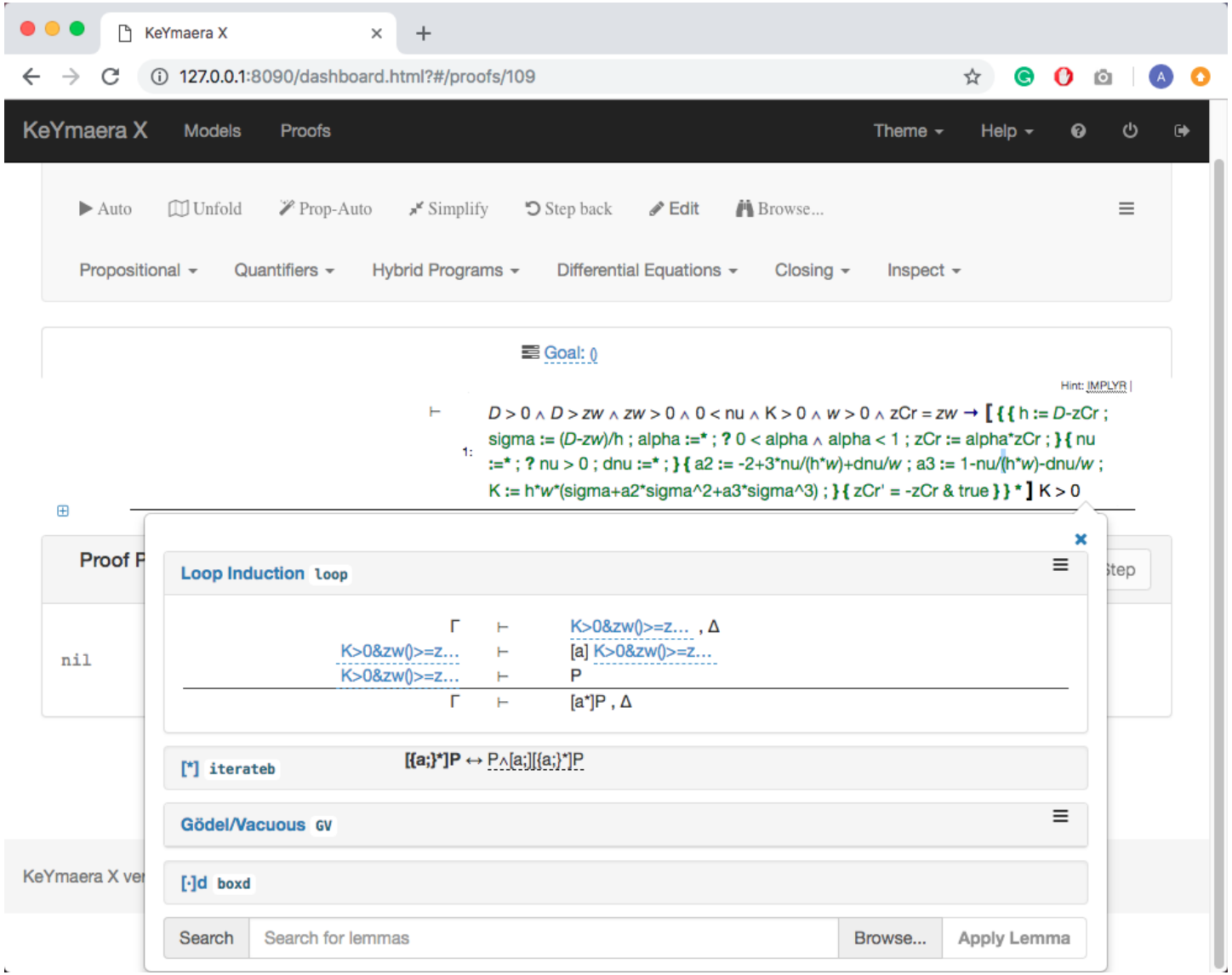
Developed the hybrid model of KPP in KeYmaera X



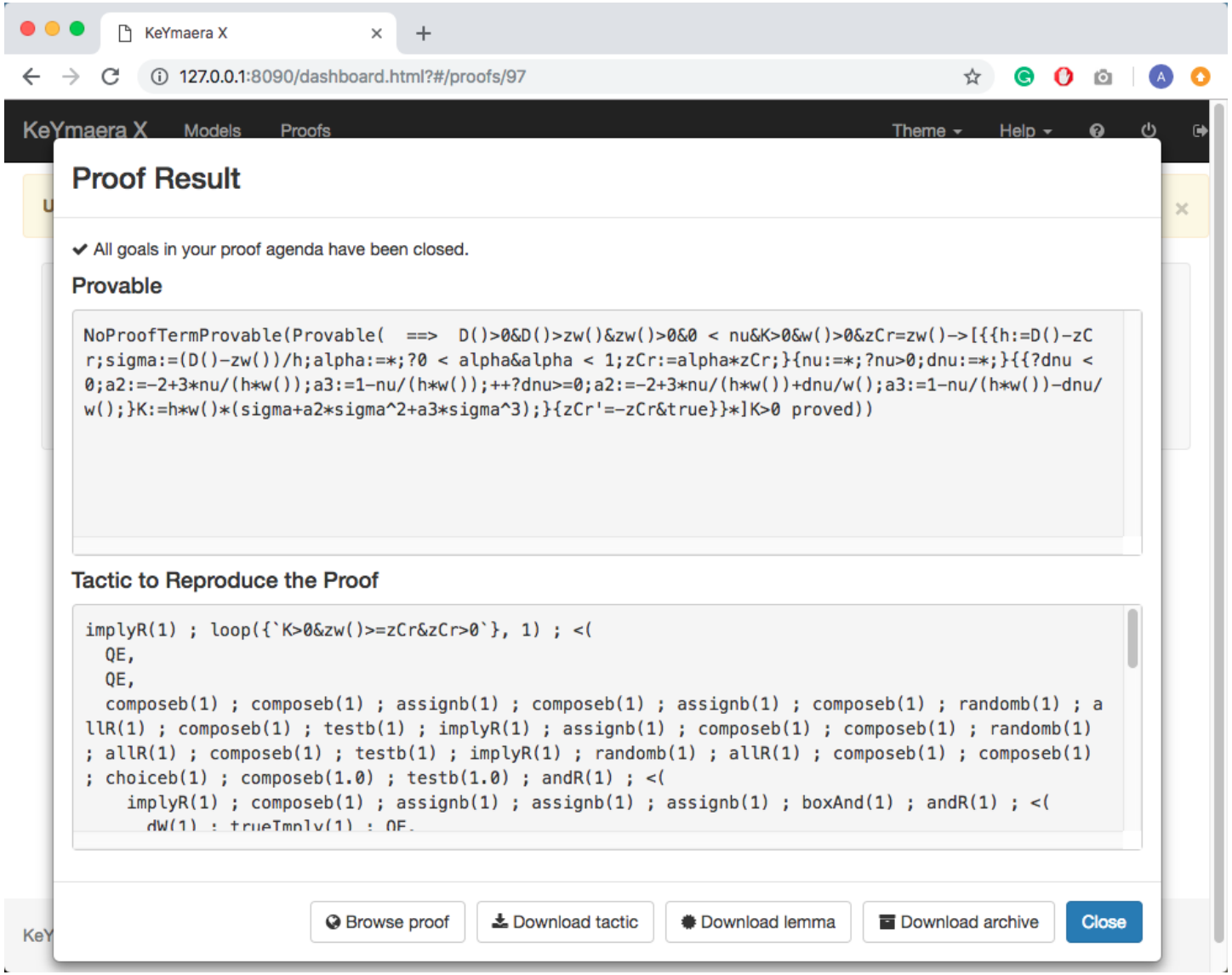
Reproduced the erroneous behavior



Applied a fix, and constructed a manual proof



Closed the proof, confirming that the fix is valid for all possible cases.



Hybrid Theorem Proving for Numerical Models

Complementary Technique:

- Provides higher confidence through formal verification.

Limitations to Widespread Use:

- Requires expertise in formal methods and theorem proving.
- Manual proof construction is nontrivial, especially with differential equations.

Practical Implication:

- Valuable for targeted, high-stakes verification,
  but unlikely to become a routine day-to-day tool for ESM modelers.

An Alternative: Lightweight Formal Methods with CIVL-C

- More potential to become practical & scalable.
- Developer-friendly syntax.
  - CIVL resembles imperative languages.
  - Familiar to scientific software developers.

- Efficient Verification Workflow:
  - Automated feedback on defects.
  - Enables rapid iteration during development.
  - No need for full formal proofs at each step.

CIVL-C model of the KPP defect:

```
1  $input int N; // number of outer iterations
2  $input int M; // max number inner iterations
3  $input double dt; // delta_t
4  $assume(0<dt && dt<1);
5  $input double zw, D, w;
6  $assume(D>0 && D>zw && zw>0 && w>0);
7  double t=0.0; // time
8  double nu, dnu, h, sigma, alpha, zCr, K, a2, a3;
9
10 double G(double sigma, double a2, double a3) {
11   return sigma +
12     a2*$pow(sigma,2) + a3*$pow(sigma,3);
13 }
14
15 void computeNu(void) {
16   $havoc(&nu);
17   $assume(nu>0);
18   $havoc(&dnu);
19 }
20
21 void computeBLD(void) {
22   h = D - zCr;
23   sigma = (D - zw)/h;
24   $havoc(&alpha);
25   $assume(0<alpha && alpha<1);
26   zCr = alpha*zCr;
27 }
28
29 void computeK(void) {
30   a2 = -2 + 3*nu/(h*w) + dnu/w;
31   a3 = 1 - nu/(h*w) - dnu/w;
32   K = h*w * G(sigma, a2, a3);
33 }
34
35 void invariant(void) {
36   $assert(K>0);
37   $assert(zw>=zCr);
38   $assert(zCr>0);
39 }
40 void initialConditions() {
41   $havoc(&nu);
42   $havoc(&dnu);
43   $havoc(&h);
44   $havoc(&sigma);
45   $havoc(&alpha);
46   $havoc(&zCr);
47   $havoc(&K);
48   $havoc(&a2);
49   $havoc(&a3);
50   $assume(0<nu && K>0);
51   $assume(zCr == zw);
52 }
53
54 void printState(void) { ... }
55
56 int main(void) {
57   printState();
58   initialConditions();
59   for (int i=0; i<N; i++) {
60     printState();
61     invariant();
62     computeBLD();
63     computeNu();
64     computeK();
65     // zCr'=-zCr ...
66     int m = $choose_int(M);
67     for (int j=0; j<m; j++) {
68       t += dt;
69       zCr += -zCr*dt;
70     }
71   }
72   printState();
73   invariant();
74 }
```

Our conclusion from the paper:

Using this model, we are able to find the same defect reported in the earlier work. This manifests as a violation of the assertion $K > 0$ (line 36) found with $N = 2$ and $M = 1$ in less one second. The model checker also prints the symbolic values of all variables that lead to the violation. When function `computeK` is replaced with the corrected version, no violations are found. For $N = M = 3$, verification takes 1.5 seconds. This time includes 31 calls to the automated theorem prover Z3 [44], which easily discharges the assertions. The CIVL approach provides strong evidence for the correctness of the fix, though not as strong as a formal proof. Nevertheless, it was straightforward to write the model and the verification process itself is fully automated.

An Alternative: Lightweight Formal Methods with CIVL-C

Limitations of CIVL-C:

- No native support for continuous dynamics or ODEs:
  - Continuous evolution must be approximated with discrete schemes
- Limited mathematical rigor compared to theorem provers
  - Provides strong evidence, but not formal guarantees.

Bottom Line:

- Strikes a balance between rigor and usability.

Announcement

https://ncar.github.io/correctness-workshop/

Thanks!

altuntas@ucar.edu