Emergent nonlinearities and fast solvers for fluids from small to global scales

Valeria Barra, Ph.D.
Research Software Engineer
valeriabarra.org

California Institute of Technology

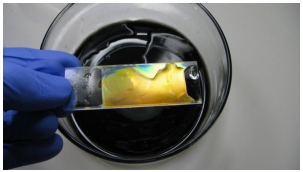
SDSU

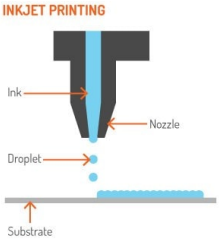
Caltech

Overview

PhD research: Viscoelastic fluids

PhD in Applied Math from NJIT on numerical simulations of thin films (long-waves) of viscoelastic fluids





Viscoelastic materials:

- hysteresis:
  loop in stress-strain rate curve
- stress relaxation:
  constant $\epsilon \Rightarrow$ decreasing σ
- creep:
  constant $\sigma \Rightarrow$ increasing ϵ

Caltech

Mechanical system analogs

- Hookean: elastic solids

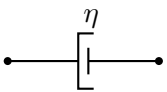
$\sigma_{ij} = 2G\epsilon_{ij}$

G shear elastic modulus

- Newtonian: viscous fluids

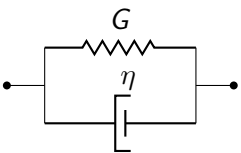
$\sigma_{ij} = 2\eta\dot{\epsilon}_{ij}$

η dynamic (shear) viscosity



- Kelvin-Voigt: linear viscoelastic solids

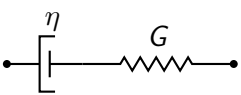
$\sigma_{ij} = 2G\epsilon_{ij} + 2\eta\dot{\epsilon}_{ij}$



- Maxwell: linear viscoelastic fluids

$\sigma_{ij} + \lambda_1 \partial_t \sigma_{ij} = 2\eta\dot{\epsilon}_{ij}$

λ_1 relaxation time, s. t. $\lambda_1 = \eta/G$.



Caltech

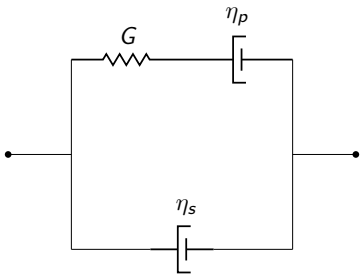
Mechanical system analog for Jeffreys

Jeffreys Model: linear viscoelastic fluids

$$\sigma_{ij} + \lambda_1 \partial_t \sigma_{ij} = 2\eta \left(\dot{\epsilon}_{ij} + \lambda_2 \partial_t \dot{\epsilon}_{ij} \right) ,$$

with $\lambda_2 = \lambda_1 \frac{\eta_s}{\eta_s + \eta_p}$, and $\eta = \eta_s + \eta_p \Rightarrow \lambda_1 \geq \lambda_2$. With η_s and η_p viscosity of Newtonian solvent and polymeric solute, respectively.

λ_1 relaxation time, λ_2 retardation time.



Governing equations

Conservation laws:

$$\rho\left(\partial_t \mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla(p+\Pi) + \nabla\cdot\boldsymbol{\sigma} + \mathbf{F_b}\ , \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0\,, \tag{2}$$

where, in $2D$, $\mathbf{u} = (u(x,y,t), v(x,y,t))$, is the vector velocity field, $\nabla = (\partial_x, \partial_y)$, p is the pressure, Π is the disjoining pressure due to the van-der-Waals interaction (attraction/repulsion) force, and $\mathbf{F_b} = (\rho g \sin\alpha, -\rho g \cos\alpha)$ body force.

Jeffreys' model:

$$\boldsymbol{\sigma} + \lambda_1 \partial_t \boldsymbol{\sigma} = 2\eta(\dot{\boldsymbol{\epsilon}} + \lambda_2 \partial_t \dot{\boldsymbol{\epsilon}})$$

Schematic

Setup and boundary conditions:

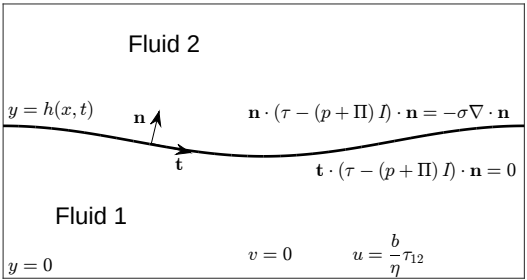


Figure: Schematic of the fluid interface and boundary conditions.

Kinematic BC: $Df/Dt = f_t + \mathbf{u} \cdot \nabla f = 0$, with $f(x, y, t) = y - h(x, t)$.

Nondimensionalization

Scalings:

$$x = Lx^*\,,\ (y, h, h_\star, b) = H(y^*, h^*, h_\star^*, b^*)\,,\ (p, \Pi) = P(p^*, \Pi^*)\,,$$
$$u = Vu^*\,,\ v = \varepsilon V v^*\,,\ (t, \lambda_1, \lambda_2) = T(t^*, \lambda_1^*, \lambda_2^*)\,,\ \gamma = \frac{V\eta}{\varepsilon^3}\gamma^*\,,$$

$$\begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix} = \frac{\eta}{T} \begin{pmatrix} \sigma_{11}^* & \frac{\sigma_{12}^*}{\varepsilon} \\ \frac{\sigma_{21}^*}{\varepsilon} & \sigma_{22}^* \end{pmatrix},$$

where $H/L = \varepsilon \ll 1$ is the small parameter. Pressure is scaled with $P = \eta/(T\varepsilon^2)$, and time with $T = L/V$.

Dimensionless governing equations

Long-wave approximation in two spatial dimensions:

$$(1+\lambda_2\partial_t)h_t + \frac{\partial}{\partial x}\left\{(\lambda_2-\lambda_1)\left(\frac{h^2}{2}Q - hR\right)h_t + \left[(1+\lambda_1\partial_t)\frac{h^3}{3} + (1+\lambda_2\partial_t)bh^2\right]\frac{\partial}{\partial x}\left(\frac{\partial^2 h}{\partial x^2} + \Pi(h)\right)\right\} = 0\,,$$

$$Q + \lambda_2 Q_t = -\frac{\partial}{\partial x}\left(\frac{\partial^2 h}{\partial x^2} + \Pi(h)\right),$$

$$R + \lambda_2 R_t = -h\frac{\partial}{\partial x}\left(\frac{\partial^2 h}{\partial x^2} + \Pi(h)\right).$$

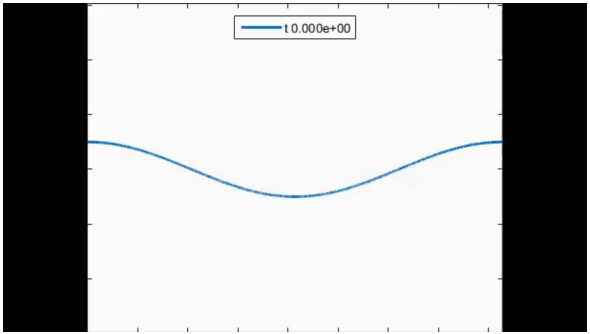
disjoining pressure: $\Pi(h) = \frac{\gamma(1-cos\theta_e)}{Mh_\star}\left[\left(\frac{h_\star}{h}\right)^n - \left(\frac{h_\star}{h}\right)^m\right]$,

θ_e contact angle, $M = 0.5$, $(n = 3, m = 2)$, $h_\star$ precursor film thickness.

Jeffreys' constitutive law:

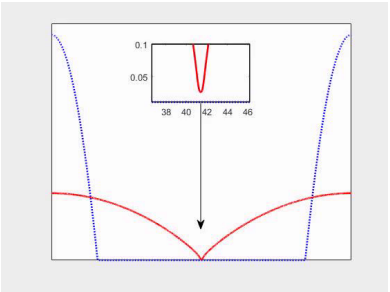
$$\boldsymbol{\sigma} + \lambda_1\partial_t\boldsymbol{\sigma} = 2\eta(\dot{\boldsymbol{\epsilon}} + \lambda_2\partial_t\dot{\boldsymbol{\epsilon}})$$

Dewetting film

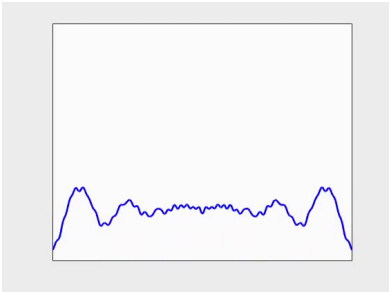


A viscoelastic dewetting film exhibits secondary satellite droplets in the dewetting region that viscous films do not exhibit.

Dewetting films with slippage and on an inverted plane

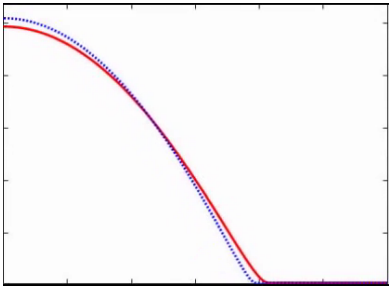


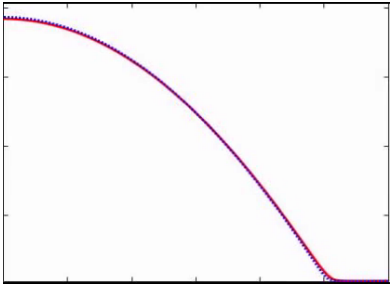
A higher slippage with the substrate suppresses formation of satellite droplets



Rayleigh-Taylor instabilities in the case of an inverted plane

Spreading and receding drops

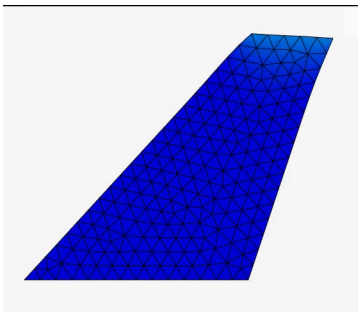


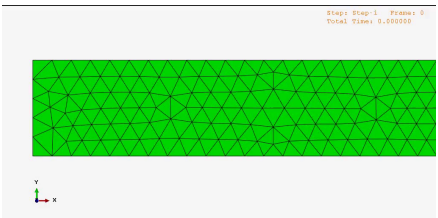


The viscoelastic drop spreads faster and recedes slower compared to the Newtonian one

Membranes

Shear and extensional free-boundary flows of viscoelastic membranes





Linear finite elements with plane stress formulation. Different constitutive models considered: elastic, viscous, viscoelastic (Maxwell)

Publications

V. Barra, S. Afkhami, L. Kondic, *Mathematical and numerical modeling of thin viscoelastic films of Jeffreys type subjected to the van der Waals and gravitational forces*, EPJE, **42**, 1 – 14 (2019)

V. Barra, S. A. Chester, S. Afkhami, *Numerical Simulations of Nearly Incompressible Viscoelastic Membranes*, Computers & Fluids, **175**, 36 – 47 (2018)

V. Barra, S. Afkhami, L. Kondic, *Interfacial dynamics of thin viscoelastic films and drops*, J. Non-Newt. Fluid Mech., **237**, 26 – 38 (2016)

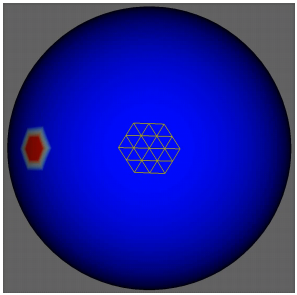
B. Adeyemi, P. Jadhawar, L. Akanji, **V. Barra**, *Effects of fluid–fluid interfacial properties on the dynamics of bounded viscoelastic thin liquid films*, J. Non-Newt. Fluid Mech., **309**, 104893 (2022)

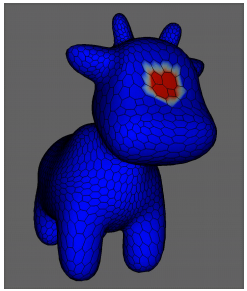
Caltech

Other projects: Computer Graphics applications

PIXAR

Developed 2 proprietary libraries in C++ for viscous fluid simulations on curved surfaces

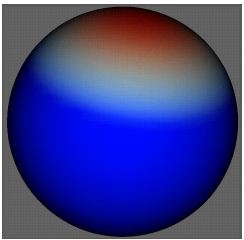


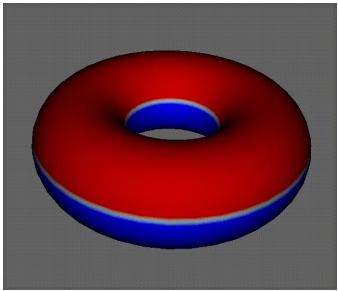


A vorticity-formulation 2D Navier-Stokes solver with fluid-structure interactions, using Discrete Exterior Calculus (DEC) in a finite volume discretization

Other projects (cont'ed)

A thin film (long-wave) solver on curved surfaces, with arbitrary topology and element shapes





Prototyped a plugin for the 3D graphics software package Houdini

Overview

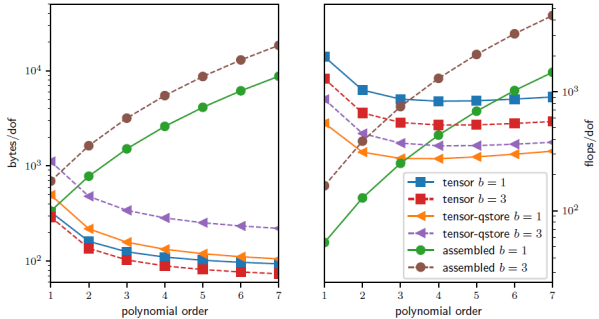
Fast algebra for high-order element-based discretizations: libCEED

Postdoc project supervised by Jed Brown at

Caltech

libCEED Overview

- High-order methods have been considered too expensive for decades because relied on sparse matrices assembly, which results in $O(p^d)$ storage and $O(p^{2d})$ compute per degree of freedom (DoF) in d dimensions, for basis polynomial order p
- On the other hand, optimized spectral element implementations can achieve $O(1)$ storage and $O(p)$ compute per DoF

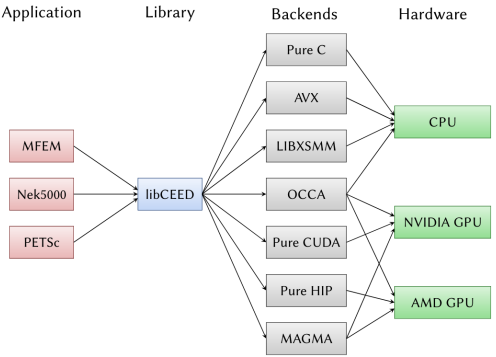


[Courtesy: Jed Brown]

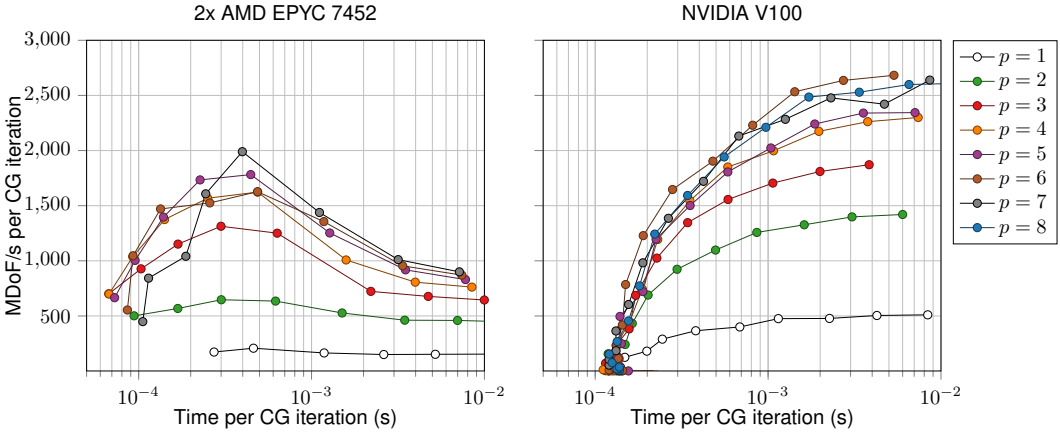
[https://github.com/CEED/libCEED/]

libCEED: the Code for Efficient Extensible Discretization

- libCEED uses a matrix-free operator description, based on a purely algebraic interface, where user only specifies the action of weak form operators
- Primary target: high-order finite/spectral element methods (FEM/SEM) exploiting tensor-product structure
- Open-source (BSD-2 license) C library with Fortran, Python, Julia and Rust interfaces
- libCEED is light-weight and performance-portable via run-time selection of specialized implementations (backends) optimized for CPUs and GPUs

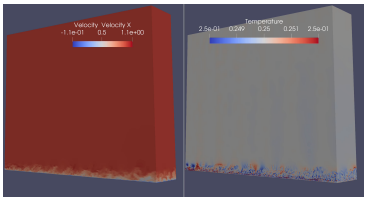


Performance

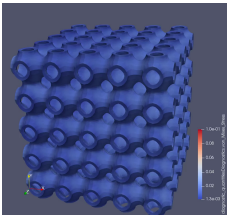


Application examples

Integration of libCEED with the Portable, Extensible Toolkit for Scientific Computation (PETSc) for examples in fluids and solid mechanics



DNS of a flat plate synthetic turbulence generator.



A compressed Schwarz periodic minimal surface.
Applications: additive manufacturing, soft robotics

In preparation: J. Brown, **V. Barra**, N. Beams, L. Ghaffari, M. Knepley, W. Moses, R. Shakeri, K. Stengel, J. Thompson, J. Zhang, *Performance-Portable Solid Mechanics via Matrix-Free p-Multigrid*, in preparation for SC 23
ArXiv: arXiv:2204.01722v3

Caltech

Publications

A. Abdelfattah, **V. Barra**, N. Beams et al., *GPU algorithms for Efficient Exascale Discretizations*, Parallel Computing, **108**, 102841 (2021)

T. Kolev et al., *Efficient exascale discretizations: High-order finite element methods*, Int J. High. Perform. Comput. Appl., **6**, 527-552 (2021)

J. Brown, A. Abdelfattah, **V. Barra** et al., *libCEED: Fast algebra for high-order element-based discretizations*, JOSS **63**, 2945 (2021)

V. Barra, J. Brown, J. Thompson, Y. Dudouit, *High-performance operator evaluations with ease of use: libCEED's Python interface*, Proceedings of the 19th Python in Science Conference 85 - 90 (2020)

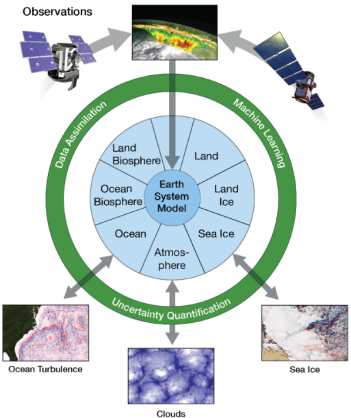
Overview

About CliMA

The Climate Modeling Alliance (CliMA) is a coalition of scientists, engineers, and applied mathematicians from **Caltech**, **MIT**, and the **NASA Jet Propulsion Laboratory**. We are building the first Earth System Model (ESM) in the Julia programming language that automatically learns from diverse data sources to produce more accurate climate predictions with quantified uncertainties.

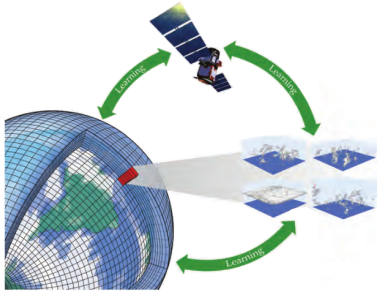
Caltech

Goals



[Source: courtesy of Tapio Schneider (Caltech)]

- The Earth System Model (ESM) will be grounded in physics (using sub-grid scale, cloud-resolving modeling) and designed for automated calibration of parameters using machine learning.
- High-resolution Large-Eddy Simulations (LES) are used to inform parametrizations of the global circulation model (GCM), which in turn, can be used for large-scale forcings to force the LES.



[Source: Physics Today - June 2021, pg. 44-51]

Technical and Scientific Aims

- Support CPUs and GPUs using a common open-source code base written in the high-level, dynamic Julia programming language (familiar syntax, similar to Python and Matlab).
- Julia has an interactive REPL, is Just-In-Time (JIT) compiled (triggered by first evaluation of function). Allows polymorphism via multiple dispatch (at compile or run time).
- Can write generic code, compiler will specialize on types of calling arguments, e.g., `f(x::AbstractArray)` where `AbstractArray` can be `Array` of `Float32`, `Float64` or a `CuArray`.
- Be accessible and extensible by a mixture of users.

- For the atmosphere model, support both Large-Eddy Simulation (LES) and General Circulation Model (GCM) configurations (i.e., Cartesian and spherical geometries).
- Allow specification of any governing equations and boundary conditions by composing operators.
- Support non-uniform unstructured meshes.

ClimateMachine.jl: a first codebase

- Supports only Discontinuous Galerkin (DG) discretization. Same in each direction (horizontal/vertical) but allows different polynomial order. No staggered grids supported
- Can prescribe PDEs only in conservation form $\partial_t \boldsymbol{Q} + \nabla \cdot \boldsymbol{F}(\boldsymbol{Q}) = S(\boldsymbol{Q})$
- Operator volume/face kernels written in KernelAbstractions.jl (a unified programming model, similar to OpenCL/SYCL, which allows for single-source code for CPUs & GPUs, but primarily "*a GPU code which runs on CPUs*")
- Overlaps computation & communication: distributed via MPI.jl.
- Efficient, but somewhat inflexible

A. Sridhar et al., *Large-eddy simulations with ClimateMachine v0.2.0: a new open-source code for atmospheric simulations on GPUs and CPUs*, Geoscientific Model Development, **15**, 6259–6284 (2022)

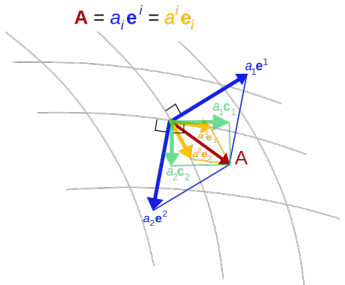
A. Souza, J. He, T. Bischoff, M. Waruszewski, L. Novak, **V. Barra**, T. Gibson, A. Sridhar et al., *The Flux-Differencing Discontinuous Galerkin Method Applied to an Idealized Fully Compressible Nonhydrostatic Dry Atmosphere*, in review for JAMES

ClimaCore.jl

ClimaCore.jl — a new dynamical core (*dycore*).

A library (suite of tools) for constructing flexible space discretizations.

- Geometry:
  - Supports different geometries (Cartesian & spherical).
  - Supports covariant/contravariant vector representation for curvilinear, non-orthogonal systems and Cartesian vectors for Euclidean spaces.
- Space Discretizations:
  - Horizontal: Support both Continuous Galerkin (CG) and Discontinuous Galerkin (DG).
  - Vertical: staggered Finite Differences (FD).



Caltech

ClimaCore.jl: API

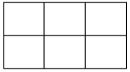
- API objects:
  - Domain, Mesh, Topology, Space, Field.
- `Field` abstraction:
  - Scalar, Vector or Struct-valued.
  - Stores values, geometry, and mesh info.
  - Flexible memory layouts.
  - Useful overloads: `sum` (integral), `norm`, `mean`.
  - Compatible with DifferentialEquations.jl time integrators.

```
domain = Domains.SphereDomain(radius)
mesh = Meshes.EquiangularCubedSphere(domain, Ne)
grid_topology = Topologies.Topology2D(mesh)
quad = Spaces.Quadratures.GLL{Nq}()
space = Spaces.SpectralElementSpace2D(grid_topology, quad)
```

```
julia> sum(ones(space))
4.618802153517008
```

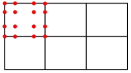
Domain

Mesh

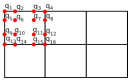


Topology

Space



Field



Caltech

ClimaCore.jl: API (cont'ed)

Flexible data layout:

- Support for different memory layouts: Array-of-Structs (AoS), Struct-of-Arrays (SoA), Array-of-Struct-of-Arrays (AoSoA).
- Common interface: `slab` for extracting 2D horizontal field slices; `column` for 1D vertically-aligned nodes.
- Add element node size dimensions to type domain (i.e., specialize on polynomial degree, useful for loop unrolling; important for kernel performance).
- Flexible memory layouts allow for flexible threading models:
  - CPU thread over elements.
  - GPU thread over nodes/node columns (ongoing).

ClimaCore.jl: API (cont'ed)

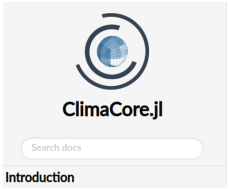
ClimaCore.jl's composable `Operators` and Julia broadcasting:

- Julia broadcasting:
  - apply a vectorized function point-wise to an array. Scalar values are "broadcast" over arrays; Fusion of multiple operations.
  - User-extensible API: can be specialized for custom functions or argument types (e.g., `CuArray` compiles and applies a custom CUDA kernel).
- `Operators` (`grad`, `div`, `interpolate`) are "pseudo-functions": act like functions when broadcasted over a `Field`, but can't be called on a single value; can be composed and fused w/ function calls. Matrix-free, i.e., no assembly; specify action of operator.

```
# apply f to each element of X
f.(X)


# fuse multiple operations
# and assign to existing array
# without intermediate temporaries
Y .= X0 .+ ϵ .* f.(X)


# expression internally calls
materialize!(Y,
  broadcasted(+, X0,
    broadcasted(*, ϵ,
      broadcasted(f, X))))
```

```
grad = Operators.Gradient()
wdiv = Operators.WeakDivergence()
diff = @. -wdiv(grad(u))
```

Some personal contributions



Operators

Edit on GitHub

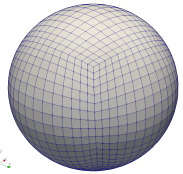
Operators

Operators can compute spatial derivative operations.

- for performance reasons, we need to be able to "fuse" multiple operators and function applications

I have worked in adding support for

- Different "cubed-sphere" meshes (Equiangular, Equidistant, Conformal)
- High-order differential operators and flux limiters
- Unit tests, integration tests and examples
- Docs, tutorials, CliMAWorkshops (https://github.com/CliMA/ClimaWorkshops)



Examples: Shallow-water equations

The shallow water equations
(in vector-invariant form):

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\boldsymbol{u}) = 0 \tag{3a}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla(\Phi + \tfrac{1}{2}\|\boldsymbol{u}\|^2) = (\boldsymbol{u} \times (f + \nabla \times \boldsymbol{u})) \tag{3b}$$

where f is the Coriolis term and $\Phi = g(h + h_s)$.

Written in terms of a curvilinear,
non-orthogonal basis:

$$\frac{\partial h}{\partial t} + \frac{1}{J}\frac{\partial}{\partial \xi^j}\left(hJu^j\right) = 0 \tag{4a}$$

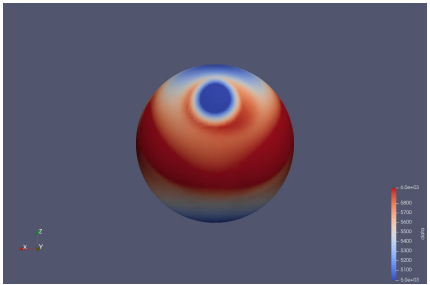
$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial \xi^i}(\Phi + \tfrac{1}{2}\|\boldsymbol{u}\|^2) = E_{ijk}u^j(f^k + \omega^k) \tag{4b}$$

```
∇ = Operators.Gradient()
∇· = Operators.WeakDivergence()
curl = Operators.Curl()

@. begin
    dydt.h = - ∇·(y.h * y.u)
    dydt.u = -∇(g * (y.h + h_s) - norm(y.u)^2 / 2) + y.u × (f + curl(y.u))
end
```

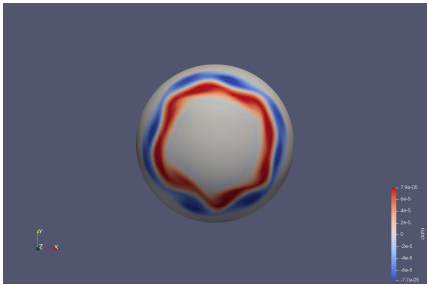
Caltech

Shallow-water equation Test Cases

ClimaCore.jl/examples/sphere/shallow_water.jl



Shallow-water equations suite, Test Case 5 [**Williamson1992**]. Zonal flow over an isolated mountain.



Shallow-water equations suite, barotropic instability test case [**Galewsky2004**]. Zonal jet with compact support at mid-latitude. A small height disturbance is then added, which causes the jet to become unstable and collapse into a highly vortical structure.

Caltech

Examples: Advection (transport) problems

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \boldsymbol{u}, \tag{5a}$$

$$\frac{\partial Q}{\partial t} = -\nabla \cdot Q \boldsymbol{u}, \tag{5b}$$

Transport of a passive tracer, with $Q = \rho q$, where q denotes tracer concentration (i.e., mixing ratio or mass of tracer per mass of dry air, in dry problems, or mass of tracer per mass of moist air, in moist problems) per unit mass, and ρ fluid density.

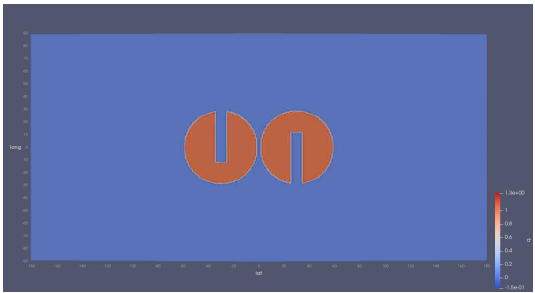
```
∇· = Operators.WeakDivergence()

@. dydt.ρ =  - ∇·(y.ρ * u)  # contintuity equation
@. dydt.ρq = - ∇·(y.ρq * u) # advection of tracer equation
```

Quasimonotone flux limiters

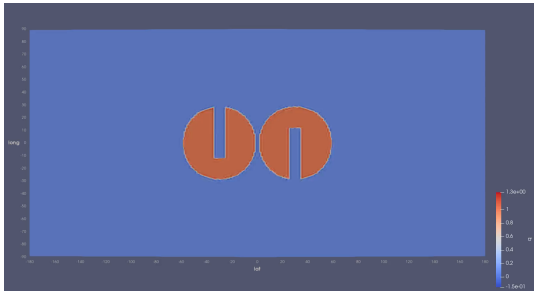
- Traditional SEM advection operator is oscillatory but due to its mimetic properties it is locally conservative and has a monotone property with respect to element averages
- We use a class of optimization-based locally conservative quasimonotone (monotone with respect to the spectral element nodal values) limiters that prevent all overshoots and undershoots at the element level [**GubaOpt2014**]
- It also maintains quasimonotonicity even with the addition of a dissipation term such as viscosity or hyperviscosity
- This involves solving a constrained optimization problem (a weighted least square problem) that is local to each element. That is, we need to solve a standard quadratic program (QP). We have to minimize a quadratic objective function subject to linear constraints
- The only additional interelement communication introduced is in determining the suitable minimum and maximum constraints

Flux limiter test case: slotted cylinders on a 2D sphere

$p = 6$, $ne = 20 \times 20 \times 6$ (effective resolution 0.75° at equator.)



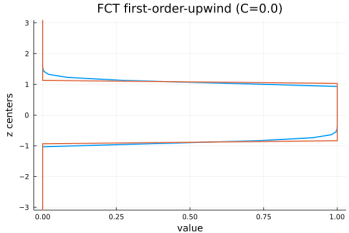
No limiter.



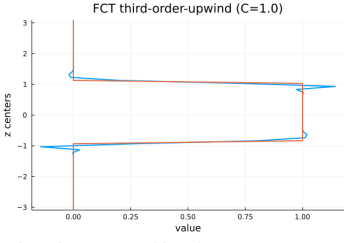
With limiter.

Flux-Corrected Transport

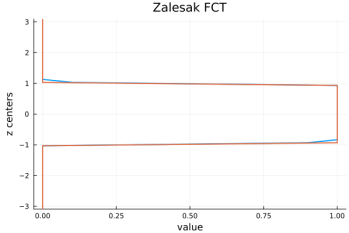
Flux-corrected transport (FCT) was proposed by Boris and Book (1973) [**BorisBook1973**] as a way of approximating a conservation law with a high-order scheme in regions where the solution is smooth while using a low-order monotone scheme where the solution is poorly resolved or discontinuous. The concept of FCT and the algorithms for its implementation were generalized by Zalesak (1979) [**Zalesak1979**].



Dissipative and very dispersive



Less dissipative and less dispersive (but over/under-shoots)



No over/under-shoots and limited dispersion.

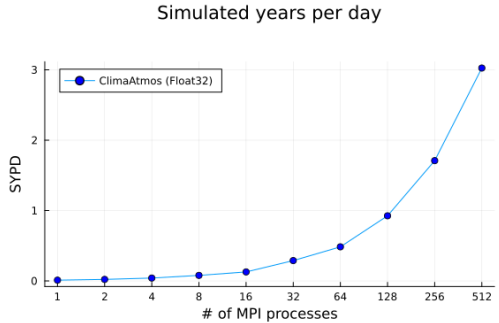
Performance/optimization efforts

Three current streams of work to reduce time-to-solution:

1. Mathematical/numerical:
   - Use SSP HEVI/IMEX time integrators to overcome stringent CFL condition, otherwise exacerbated by elements aspect ratio $\sim 1 : 10^4$
2. Julia Optimizations:
   - Disable bounds checking to facilitate vectorized (SIMD) instructions via `@inbounds`
   - Ensure type stability using tools, e.g., JET.jl that allows to do static type checking
   - Eliminate dynamic memory allocations
3. Profiling:
   - Our group developed the NVTX.jl package for instrumenting the code for use with Nvidia Nsight profiler, which supports profiling MPI-enabled code. The profiling is included in our automated scaling tests, uploaded after each run (performance CI)
   - Other Julia profiling tools: `@time`, `-track-allocation`

Parallelization

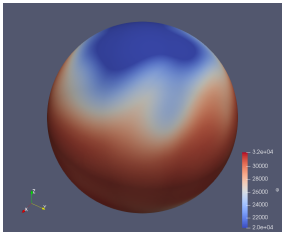
- ClimaComms.jl wrapper library that supports generic distributed and shared computing paradigms. Currently using MPI.jl and CUDA.jl in the backend for Distributed Topology on CPUs and GPUs.
- I/O: NetCDF and parallel HDF5 support.
- Different threading across blocks/iteration patterns



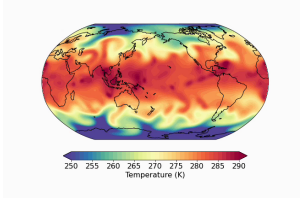
Preliminary scaling results.

Conclusions

- ClimaCore.jl is the new open-source dycore for the atmosphere and land components of the CliMA's proposed Earth System Model (ESM), entirely written in the Julia dynamic language
- We introduced ClimaCore.jl's API for flexible discretizations and high-performance composable solvers
- We showed examples of applications for atmospheric flows and flux limiters to overcome oscillation challenges for the high-order SEM advection operator



[Held-Suarez 180-day simulation.]



[AMIP (w/o EDMF and topography) simulation.]

Acknowledgements

Many thanks to all ClimateMachine.jl's and ClimaCore.jl's users and contributors.
In particular, for the latest CliMA dycore efforts a special mention goes to:

- Tapio Schneider[1] (PI), Paul Ullrich[2], Oswald Knoth[3], Simon Byrne[1], Jake Bolewski[1], Charles Kawczynski[1], Sriharsha Kandala[1], Zhaoyi Shen[1], Jia He[1], Kiran Pamnany[1], Ben Mackay[1], Akshay Sridhar[1], Dennis Yatunin[1], Lenka Novak[1], Toby Bischoff[1], Daniel (Zhengyu)Huang[1], Andre Souza[4], Yair Cohen[1]
  1: Caltech, 2: UC Davis, 3: TROPOS, 4: MIT

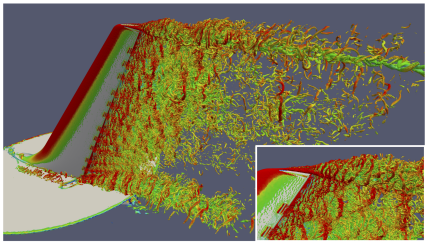
Our funders:

SCHMIDT FUTURES NSF DARPA HEISING-SIMONS FOUNDATION Audi Environmental Foundation CHARLES TRIMBLE

Outlook and future directions

Future work:

- Explore other stabilization methods for hyperbolic problems
  - In particular, Streamline-Upwind Petrov-Galerkin (SUPG) methods for numerical weather and climate simulations
  - Why are these important: These methods ensure correct energy behavior, conservation and stability (other classical stabilized formulations can create undesired artificial energy)
  - Collaboration with Ken Jansen, CU Boulder, creator of the Parallel-Hierarchic-Adaptive-Stabilized-Transient-Analysis (PHASTA)

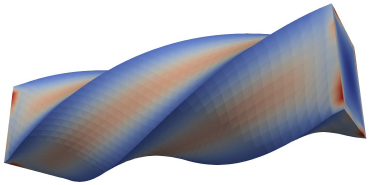


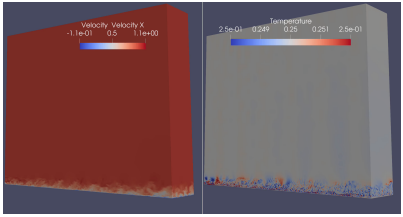
[Source: https://phasta.scigap.org/]

Outlook and future directions (cont'd)

More future work:

- Use of multigrid strategies for fluid problems where it can be beneficial:
  - Extend collaboration with Jed Brown, CU Boulder – our work showed efficiency and scalability of matrix-free p-multigrid methods with algebraic multigrid coarse solvers through large deformation hyperelastic simulations of solids
  - Use this in fluid problems with transition from turbulent to viscous regions





DNS of a flat plate synthetic turbulence generator.

Mentoring and Engagement

- Mentoring experience (GSMM Camp)
- Continue efforts in Diversity, Equity, Inclusion and Belonging
- Advocacy for (invisible/visible) disabilities in STEM, women in STEM, international students and scholars, upward mobility, systemic exclusion of underrepresented groups
- Active in industrial workshops (MPI)
- Students success and professional development: professional associations, e.g., US-RSE, WHPC, SIAM, etc

Thank you!

References

[1] David L. Williamson, John B. Drake, James J. Hack, Rüdiger Jakob, Paul N. Swarztrauber, *A standard test set for numerical approximations to the shallow water equations in spherical geometry*, Journal of Computational Physics, 102(1), 211-224, 1992.

[2] Joseph Galewsky, Richard K. Scott, Lorenzo M. Polvani, *An initial-value problem for testing numerical models of the global shallow-water equations*, Tellus A: Dynamic Meteorology and Oceanography, 56(5), 429-440, 2004.

[3] Oksana Guba, Mark Taylor, Amik St-Cyr, *Optimization-based limiters for the spectral element method*, ournal of Computational Physics, 267, 176-195, 2014.

[4] Jay P Boris, David L Book, *Flux-corrected transport. I. SHASTA, a fluid transport algorithm that works*, Journal of Computational Physics, 11(1), 38-69, 1973.

[5] Steven T Zalesak, *Fully multidimensional flux-corrected transport algorithms for fluids*, Journal of computational physics, 31(3), 335–362, 1979.