Modeling and Stabilization of Transport-Dominated Flows

2026 Graduate Student Mathematical Modeling Camp
Final Report

Christopher Agesen
NJIT

Sam Bradford
Western University

Abhijnan Dikshit
Purdue University

Anshi Gupta
U. of Houston

Bhavana Morankar
NC State University

Jack Murphy
U. of Arizona

Nanda N. Raghunathan
U. of Pittsburgh

Karnav Raval
Western University

Lane H. Rogers
U. of Tennessee

Mentored by Dr. Valeria Barra
San Diego State University

July 2, 2026

Abstract

This report explores and compares numerical stabilization methods for transport-dominated flows arising in atmospheric modeling. The Streamline-Upwind (SU) and Streamline-Upwind Petrov-Galerkin (SUPG) stabilization formulations are implemented in Julia's `ClimaCore.jl` package and Python's `Firedrake` package for a test problem with slotted-cylinder initial conditions. These methods are compared for their ability to mitigate spurious oscillations while preserving sharp features against existing hyperdiffusion and quasi-monotone limiter methods, alongside various combinations. For this benchmark, quasi-monotone limiters were most effective at minimizing un-physical extrema, at the expense of diffusing the overall structure. The SUPG method was not found to improve upon the no stabilization case, for this test case in the computed error metrics. However, it performs best at preserving sharp feature and overall structure, among tested stabilized runs. Theoretical properties of SUPG are analyzed, and an asymptotics-based SUPG algorithm is proposed as future work.

Contents

1 Introduction

Accurate numerical simulations of the Earth's atmosphere are important for studying phenomena such as climate change. Among several classical modeling challenges in atmospheric flows, the transport of a passive tracer via pure advection is especially challenging due to the nature of such problems. A pure advection problem is an example of a partial differential equation (PDE) that, when discretized with a high-order Finite Element Method (FEM) or Spectral Element Method (SEM), leads to spurious numerical oscillations and instabilities.

As such, it is necessary to implement stabilization strategies that can mitigate oscillations and instabilities. Examples of such stabilization strategies include a fourth-order Laplacian operator (referred to as *hyperdiffusion*) [1], slope or flux-limiters [2], and flux-corrected transport [3]. Such methods, however, are not specific to the Finite or Spectral Element Methods (FEM or SEM), apply isotropic diffusion, symmetric in all directions, and often tend to overly smear out desired sharp gradients. Other methods such as the Streamline-Upwind Petrov Galerkin (SUPG) method [4] (and its simplified version, the Streamline-Upwind (SU) method) apply diffusion only along the streamlines of the flow and thus, can better maintain sharp gradients present in the flow.

The main aim of this project is to systematically compare different stabilization strategies using the transport of a passive tracer as a test problem. The model of passive tracer transport (via pure advection) was implemented in the `ClimaCore.jl` [5] and Firedrake [6] packages, which use the Julia and Python programming languages, respectively.

`ClimaCore.jl` [5] is an open-source library for the dynamical core (*dycore*) of a proposed advanced Earth System Model (ESM). An ESM is a massive, multiscale, multiphysics model that couples different model components and media (including the atmosphere, ocean, and land masses) to provide an accurate representation of Earth's climate systems. The proposed ESM has been developed by the Climate Modeling Alliance (CliMA), which is a coalition of scientists, engineers, and applied mathematicians from CalTech, MIT, and the NASA Jet Propulsion Laboratory. The dynamical core (also called *dycore*) of this ESM is described in this paper [7].

In the `ClimaCore.jl` library, some stabilization strategies are readily available and utilized to counteract the numerical instability of such transport-dominated flows. These include artificial diffusion, limiters, and flux-corrected transport strategies mentioned earlier.

In this project, the subject of primary interest is the FEM/SEM-specific SUPG [4] (and simplified SU method) of stabilizing numerical solutions to PDEs. We hypothesize that the SUPG method will exhibit superior performance at preserving the sharp features in transport-dominated flows (which can be of interest in problems with discontinuities or variations of scales), as this method introduces stabilization along the streamlines of the flow only—that is, anisotropically. Accordingly, we study the initial condition characterized by two high-density regions of passive tracers in the shape of

slotted cylinders, as described below in Section 2. We then proceed to compare the performance of the SU and SUPG methods to alternative methods of stabilizing transport dominated flows, such as a specific class of slope/flux-limiters and hyperdiffusion in Section 4. In this paper, we offer an extensive numerical and theoretical analysis of the mathematical basis of SU and SUPG method for reducing spurious oscillations and preserving sharp features in transport dominated flow problems.

In Section 5 (Theoretical Analysis), we distinguish global conservation of density and tracer mass from pointwise boundedness of the tracer mixing ratio. We show that the conservative semidiscrete SUPG formulation preserves total mass on the closed spherical domain, but that standard linear SUPG does not guarantee nonnegativity, a discrete maximum principle, or total-variation-diminishing behavior. We then discuss residual-based discontinuity capturing and conservative flux limiting as possible modifications for suppressing crosswind oscillations and enforcing physically admissible tracer bounds. Additionally, we propose an improved time-dependent SUPG algorithm based on an asymptotic expansion of the tracer density. This algorithm allows us to solve an implicit SUPG problem by solving a series of explicit SUPG problems instead; greatly reducing computational cost.

2 Problem Statement

This section describes the slotted cylinders benchmark problem statement. This benchmark problem is similar to the test case proposed by Nair and Lauritzen [8]. Consider the advection problem on the domain Ω

$$\begin{cases} \frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \mathbf{u}, \\ \frac{\partial Q}{\partial t} = -\nabla \cdot Q \mathbf{u} \end{cases} \tag{1}$$

Here, $\rho(\lambda, \theta, t) \in \Re$ is the fluid density, $Q(\lambda, \theta, t) = q(\lambda, \theta, t) * \rho(\lambda, \theta, t) \in \Re$ is the tracer density, and $\mathbf{u}[u, v]$ is the two-dimensional velocity field. The symbol $\nabla\cdot$ denotes the divergence operator on the surface of a sphere. Moreover, $\lambda, \theta,$ and t denote latitude, longitude, and time respectively.

For a divergent flow, $\mathbf{u}[u, v]$ is defined as

$$\mathbf{u}[u(\lambda, \theta, t), v(\lambda, \theta, t)] = \begin{bmatrix} u_0 \sin^2(\lambda) \sin(2\theta) \cos(\frac{\pi t}{T}) + (\frac{360}{T}) \cos(\theta) \\ u_0 \sin(2\lambda) \cos(\theta) \cos(\frac{\pi t}{T}) \end{bmatrix}$$

Here, $u_0 = \frac{2\pi R}{T}$, with the spherical radius of the Earth $R = 6.37122 \times 10^6$ m and $T = 12 \cdot 24 \cdot 60 \cdot 60$ s represents a total duration of twelve days in seconds.

Consider the problem characterized by two slotted-cylinders for the tracer concentration as initial condition. Let $r_0 = R/2 = 3.18561 \times 10^6$ be the radii of the two slotted cylinders, centered at $(-\pi/6, 0)$ and $(\pi/6, 0)$ respectively. The symbols r_1 and r_2 denote the radii of the two great circles centered at these respective points. Accordingly,

we initialize the tracer density $Q(\lambda, \theta, 0)$ as follows:

$$Q = \begin{cases} 1, & r_1 \le r_0 \ \text{ and } \ |\lambda - \lambda_1|\, R \ge D(r_0/6), \\ 1, & r_2 \le r_0 \ \text{ and } \ |\lambda - \lambda_2|\, R \ge D(r_0/6), \\ 1, & r_1 \le r_0 \ \text{ and } \ |\lambda - \lambda_1|\, R < D(r_0/6) \ \text{ and } \ (\phi - \phi_1)\, R < D(-5r_0/12), \\ 1, & r_2 \le r_0 \ \text{ and } \ |\lambda - \lambda_2|\, R < D(r_0/6) \ \wedge \ (\phi - \phi_2)\, R > D(5r_0/12), \\ 0.1, & \text{otherwise.} \end{cases}$$

where $D(s) = \frac{180}{\pi} s$.

2.1 SUPG Stabilization method

The weak form of the above advection problem may be derived by multiplying the strong-form equations presented in (1) by a test function $v \in H^1(\Omega)$ to obtain

$$\int_\Omega \frac{\partial \rho}{\partial t} v d\Omega = -\int_\Omega \left[(\nabla \cdot \rho \mathbf{u}) v\right] d\Omega, \tag{2}$$

$$\int_\Omega \frac{\partial Q}{\partial t}\, v\, d\Omega = -\int_\Omega \left[(\nabla \cdot Q\mathbf{u}) v\right] d\Omega \tag{3}$$

In order to apply integration by parts to the above form it will be necessary to let v to be in $H_0^1(\Omega)$. Since the standard Galerkin bilinear form is not coercive in the streamline direction, spurious oscillations tend to emerge near sharp gradients in convection-dominated problems. But an anisotropic stabilization may be achieved through the Streamline-Upwinding Petrov Galerkin (SUPG) method [4]. The core idea of the Petrov-Galerkin stabilization strategy is to is to replace the test function v by an alternate function, $\tilde{v} = v + \tau\, \mathbf{u} \cdot \nabla v$, which clearly indicates the flow direction. As a part of this approach the test function resides in a different function space than the trial function, unlike the classical Galerkin formulation. Thus, the SUPG weak form of (3) is given as

$$\int_\Omega \tilde{v} \left(\frac{\partial Q}{\partial t} + \nabla \cdot (Q\mathbf{u}) \right) \mathrm{d}\Omega = 0.$$

Upon expanding $\tilde{v}$ we get,

$$\underbrace{\int_\Omega v \left(\frac{\partial Q}{\partial t} + \nabla \cdot (Q\,\mathbf{u}) \right) \mathrm{d}\Omega}_{\text{standard Galerkin}} + \underbrace{\int_\Omega \tau\, (\mathbf{u} \cdot \nabla v) \left(\frac{\partial Q}{\partial t} + \nabla \cdot (Q\,\mathbf{u}) \right) \mathrm{d}\Omega}_{\text{SUPG correction}} = 0 \tag{4}$$

3 Numerical Methods

In this section, we present the discrete weak formulation underlying the ClimaCore.jl and Firedrake implementations.

3.1 SU and SUPG weak formulation: discrete implementation

The standard Galerkin formulation for the density equation is given as follows. Find $\rho_h \in V_h \subset H_0^1(\Omega)$ such that for all $v_h \in V_h$:

$$\left(\frac{\partial \rho_h}{\partial t},\, v_h\right) - (\rho_h\, \mathbf{u},\, \nabla v_h) = 0, \tag{5}$$

where $(\cdot,\cdot)$ denotes the $L^2(\Omega)$ inner product and the divergence term has been integrated by parts. Note that SUPG correction is not applied to the density. A formulation of conservative SUPG for the tracer equation is formulated in the following lines.
Find $(\rho q)_h \in V_h$ such that for all $v_h \in V_h$:

$$\begin{aligned} 0 = \underbrace{\left(\frac{\partial Q_h}{\partial t},\, v_h\right) - (Q_h\, \mathbf{u},\, \nabla v_h) + \int_{\partial\Omega} v_h\, Q_h\, (\mathbf{u}\cdot\hat{n})\, \mathrm{d}\sigma}_{\text{standard Galerkin}} \\ + \underbrace{\sum_K \int_{\Omega_K} \tau\, (\mathbf{u}\cdot\nabla v_h)\, \rho_h\, \mathcal{R}_{\text{adv}}(q_h)\, \mathrm{d}\Omega}_{\text{SUPG correction}}. \end{aligned} \tag{6}$$

Depending on the precise choice of the term $\mathcal{R}_{\text{adv}}(q_h)$, the correction may be the consistent SUPG correction or the non-consistent SU correction. The various choices of $\mathcal{R}_{\text{adv}}(q_h)$ and the corresponding corrections that they describe are charted in Table 1.

Petrov-Galerkin interpretation: To reiterate, the SU and SUPG terms may be interpreted as making use of the test function:

$$\tilde{v}_h = v_h + \tau\, \mathbf{u}\cdot\nabla v_h, \tag{7}$$

so that the trial and test function spaces differ. Consequently, we perturb the test functions so that the compact support of those functions travels along the direction of the streamlines.

Table 1: Description of corrections based on the residual equation used.

Mode	Residual $\mathcal{R}_{\text{adv}}(q_h)$	
`:SU` (Streamline Upwind only)	$\mathbf{u}\cdot\nabla q_h$	non-consistent
`:consistent` (full Brooks–Hughes SUPG [4])	$\partial_t q_h + \mathbf{u}\cdot\nabla q_h$	consistent

3.2 The Stabilization Parameter τ

The code uses the Tezduyar–Osawa [9] formula for pure advection ($\kappa = 0$):

$$\tau^{\epsilon} = \left[\left(\frac{2}{\Delta t} \right)^2 + \left(\frac{2\|\mathbf{u}\|}{h} \right)^2 \right]^{-1/2}, \tag{8}$$

where h is the average *nodal* length scale $h \sim h_e/p$ (element width divided by polynomial degree).

Reasoning behind the stabilization parameter: Recall the optimal stabilisation parameter for the 1D steady advection-diffusion problem:

$$\tau^* = \frac{h}{2\|\mathbf{u}\|}\,\xi(Pe),$$
$$\text{where } \xi(Pe) = \coth(Pe) - \frac{1}{Pe}, \text{ and } Pe = \frac{\|\mathbf{u}\|\,h}{2\varepsilon}.$$

But for an advection dominated problem($Pe \gg 1$, $\varepsilon \to 0$),

$$\coth(Pe) \to 1, \qquad \frac{1}{Pe} \to 0 \qquad \Rightarrow \qquad \xi(Pe) \to 1,$$

and hence,

$$\tau^* \;\longrightarrow\; \frac{h}{2\|\mathbf{u}\|}.$$

The UGN element length scale: UGN stands for Upwind-direction, Global, Nodal, describing how the element length scale h_{UGN} is constructed [9]:

- **U**: the length is measured in the *upwind (streamline) direction* $\hat{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|$, not isotropically.
- **G**: it is a *global* element length, using *all* nodes of the element by summing over all basis functions N_a.
- **N**: it is *nodal*, computed directly from the nodal basis functions N_a and their gradients ∇N_a.

The explicit formula from [9] is:

$$h_{\mathrm{UGN}} = 2\|\mathbf{u}\| \left(\sum_{a=1}^{n_{\mathrm{en}}} \left| \mathbf{u} \cdot \nabla N_a \right| \right)^{-1},$$

where n_{en} is the number of nodes per element. Geometrically, h_{UGN} is the distance spanned by $\mathbf{u}$ across the element in the streamline direction. SUGN stands for Stabilisation parameter, Upwind, Global, Nodal — it is simply the prefix S (stabilisation parameter) appended to UGN. The three components are [9]:

$$\tau_{\text{SUGN1}} = \frac{h_{\text{UGN}}}{2\|\mathbf{u}\|} \qquad \text{(advective timescale)},$$

$$\tau_{\text{SUGN2}} = \frac{\Delta t}{2} \qquad \text{(temporal timescale)},$$

$$\tau_{\text{SUGN3}} = \frac{h_{\text{UGN}}^2}{4\nu} \qquad \text{(diffusive timescale)}.$$

They are combined as an ℓ^2 inverse norm:

$$(\tau_{\text{SUPG}})_{\text{UGN}} = \left[\frac{1}{\tau_{\text{SUGN1}}^2} + \frac{1}{\tau_{\text{SUGN2}}^2} + \frac{1}{\tau_{\text{SUGN3}}^2}\right]^{-1/2},$$

so that τ is always bounded by the smallest of the three timescales. By removing the diffusing term, thus we get the formula defined in (8).

3.3 Implementation details for the ClimaCore.jl code

The ClimaCore.jl code used in testing was created by augmenting the existing sphere example `limiters_advection.jl` in the public repository [5] (located in `ClimaCore.jl/examples/sphere/limiters_advection.jl`). The quasi-monotone limiters and hyperdiffusion stabilization techniques were already implemented in the ClimaCore.jl API and exercised in the `limiters_advection.jl` example file. SU and SUPG were novel implementations, developed in this project. After computing the weak form of the solution, the resulting differential equation is solved using the tri-stage, third-order Strong Stability Preserving Runge-Kutta method (SSPRK33) from the `OrdinaryDiffEq.jl` package. At each time step, a weighted direct stiffness summation is applied for continuity enforcement of the continuous-Galerkin (CG) spectral elements. Table 2 below highlights choices made for key solver and discretization parameters in the code when solving the test case considered in this work.

Table 2: Discretization and solver parameters used in the ClimaCore.jl code.

Parameter	Value
Number of horizontal elements per cube panel	20
Number of quadrature nodes	4
Time step size, Δt	345.60 s
Number of time steps	3000
Hyperdiffusion coefficient	6.6×10^{14}

The example code enables the combination of different stabilization methods. SU and SUPG cannot be applied in the same run, but either can be paired with limiters hyperdiffusion, or both limiters and hyperdiffusion. The implementation of the continuous equations and residual calculation for SUPG is described below.

Let Ω be the sphere. The state consists of density ρ and tracer q (e.g. concentration). The strong form of the system is:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\, \mathbf{u}) = 0 \tag{9}$$

$$\frac{\partial (\rho q)}{\partial t} + \nabla \cdot (\rho q\, \mathbf{u}) = 0 \tag{10}$$

where $\mathbf{u}$ is a prescribed, time-dependent velocity field. Because the continuity equation (9) holds, the tracer equation (10) can be rewritten via the product rule:

$$\frac{\partial (\rho q)}{\partial t} + \nabla \cdot (\rho q\, \mathbf{u}) = \rho \underbrace{\left(\frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q \right)}_{\mathcal{R}_{\text{adv}}(q)} + q \underbrace{\left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\, \mathbf{u}) \right)}_{=0 \text{ by } (9)} = \rho\, \mathcal{R}_{\text{adv}}(q).$$

Hence at the *continuous* level the conservative residual of (10) factors as:

$$R(\rho q) = \rho\, \mathcal{R}_{\text{adv}}(q), \qquad \text{where} \qquad \mathcal{R}_{\text{adv}}(q) = \frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q.$$

At the discrete level, if ρ_h does not satisfy (9) exactly (which it generally will not), then the conservative and advective forms decouple.

3.4 Implementation details for the Firedrake code

The Firedrake library [6] implementation utilized the SU stabilization method for solving the advection problem. Originally, the SUPG stabilization implemented in the Gusto library [10] was utilized. However, further exploration uncovered that stabilization is only implemented in Gusto for planar meshes rather than spherical meshes, which are of interest in this work, since the pure advection problem is modeled on a cubed spherical mesh in this project. As such, the SU stabilization was directly implemented through the Firedrake library using the weak form of the governing equations. The Gusto library example for the slotted cylinders benchmark [10] was adapted to test the Firedrake implementation.

Firedrake allows the user to directly specify the test function space (whereas this is indirectly implemented in ClimaCore.jl and not part of the public-facing API). This allowed an easy specification of the SU correction test function shown in (7). This test function can then be used to specify the linear and bilinear forms of FEM as follows. Casting $\partial Q/\partial t + \mathbf{u} \cdot \nabla Q = 0$ as a method-of-lines problem for the rate $k \equiv \partial Q/\partial t$, each stage solves: find k with

$$\underbrace{\int_\Omega v_h\, k\, dx}_{a(w_h,k)} = \underbrace{-\int_\Omega \tilde{v}_h\, (\mathbf{u} \cdot \nabla Q)\, dx}_{L(w_h)}, \quad \tilde{v}_h = v_h + \tau\, \mathbf{u} \cdot \nabla v_h, \quad \forall\, v_h \in V_h. \tag{11}$$

Since the SU correction is applied to the advection term only and not the temporal term, $a(\phi, k)$ is the standard Galerkin mass matrix, independent of $\mathbf{u}$ and τ. It is therefore assembled and factorised once (`constant_jacobian=True`) and reused across all stages, unlike the full SUPG, where the mass matrix must be reassembled whenever $\mathbf{u}$ changes; the trade-off is that the stabilization is only $\mathcal{O}(h)$ consistent. The stabilization parameter follows the Tezduyar formula as given in (8). To calculate the stabilization parameter, $h = \sqrt{|K|}$ is computed from the local cell area, $|K|$, and stored as a $DG0$ field in Firedrake.

Subsequently, the `LinearVariationalSolve` functionality from Firedrake can be used to solve the weak form of the equations. To advance the solution in time, the SSPRK3 scheme [11] is implemented, which solves the weak form of the equations and updates the solution at each time step. In this way, the Firedrake implementation can apply the SU stabilization to an unsteady advection problem.
With $k(D, t)$ the rate from (11) at field D and velocity time t, the step $D^{(n)} \to D^{(n+1)}$ is

$$\begin{aligned} D^{(1)} &= D^{(n)} + \Delta t\, k\left(D^{(n)},\, t^{(n)}\right), \\ D^{(2)} &= \tfrac{3}{4} D^{(n)} + \tfrac{1}{4} D^{(1)} + \tfrac{1}{4} \Delta t\, k\left(D^{(1)},\, t^{(n)} + \Delta t\right), \\ D^{(n+1)} &= \tfrac{1}{3} D^{(n)} + \tfrac{2}{3} D^{(2)} + \tfrac{2}{3} \Delta t\, k\left(D^{(2)},\, t^{(n)} + \tfrac{1}{2}\Delta t\right), \end{aligned} \tag{12}$$

with the velocity re-evaluated at the stage abscissae $c = [0,\, 1,\, \frac{1}{2}]$. Each stage is one LU solve of the pre-factored mass matrix; being strong-stability-preserving, SSPRK3 adds no oscillations beyond those in the spatially stabilized operator. Table 3 below details choices for important discretization and solver parameters used in the Firedrake code. The refinement level is an indication of the spatial grid size. A refinement level of n indicates that there are 2^n segments along each edge of a panel in the discretization.

4 Results and Discussion

We now compare numerical solutions of the passive-tracer advection system (1) under various stabilization (and combinations thereof) methods, using both the `ClimaCore.jl`

Table 3: Discretization and solver parameters used in the Firedrake code.

Parameter	Value
Refinement level	4
Time step size, Δt	345.60 s
Number of time steps	3000
Polynomial degree of elements	3
Quadrature degree	6

[5] and Firedrake [6] packages. We compare the simulations both qualitatively through the tracer fields and quantitatively using particular error metrics of interest.

We solve the advection system with the slotted-cylinder test case, with the initial condition as shown in Figure 1. Following the examples in `ClimaCore.jl` [5], this is chosen such that the tracer field returns to its initial configuration after one cycle $t \in [0, T]$. As such, we choose two classes of metrics to quantify the error between the

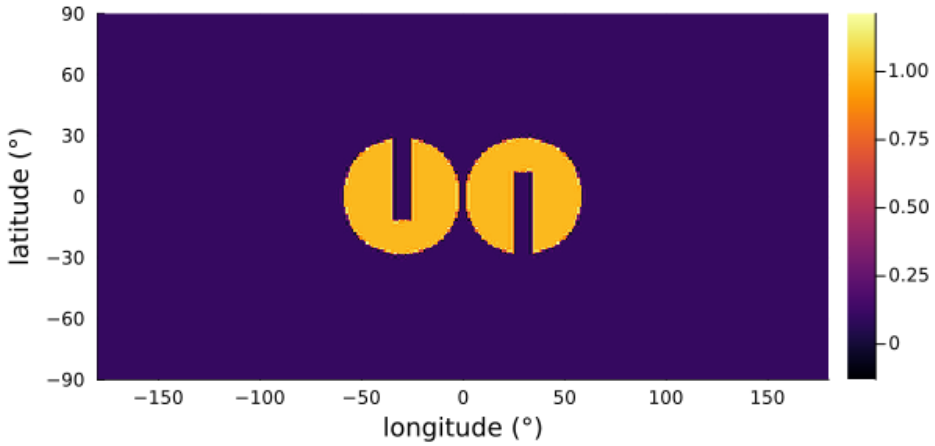


Figure 1: Initial condition for all simulations showing two slotted cylinders, plotted on a latitude-longitude grid.

initial and final field. In order to evaluate if the tracer field remains physically bounded (no spurious undershoots/overshoots), we compute

$$q_{\text{over}} = \frac{\max_{\lambda,\theta} q_T - \max_{\lambda,\theta} q_0}{\Delta q_0}, \qquad q_{\text{under}} = \frac{\min_{\lambda,\theta} q_T - \min_{\lambda,\theta} q_0}{\Delta q_0}, \tag{13}$$

where q_0 and q_T denote the initial and final tracer fields (respectively) and the initial tracer range is defined as

$$\Delta q_0 = \max_{\lambda,\theta} q_0 - \min_{\lambda,\theta} q_0. \tag{14}$$

While (13) measure the appearance of local nonphysical extrema, they do not measure the preservation of the overall shape of the tracer field. Thus, we also compute the global errors between the initial and final tracer fields

$$\ell_1 = \frac{I[|q_T - q_0|]}{I[|q_0|]}, \quad \ell_2 = \left(\frac{I[(q_T - q_0)^2]}{I[q_0^2]}\right)^{1/2}, \quad \ell_\infty = \frac{\max_{\lambda,\theta} |q_T - q_0|}{\max_{\lambda,\theta} |q_0|}, \tag{15}$$

where

$$I[f] = \frac{1}{4\pi} \int_0^{2\pi} \int_{-\pi/2}^{\pi/2} f(\lambda, \theta) \cos\theta \, \mathrm{d}\theta \, \mathrm{d}\lambda \tag{16}$$

denotes the spherical area average.

We numerically solve the passive-tracer advection system (1) for the slotted-cylinder test case, using hyperdiffusion, quasi-monotone limiting, SU, SUPG, and selected combinations of these stabilization methods. The error metrics (13) and (15) are computed for each case, and shown in Table 4.

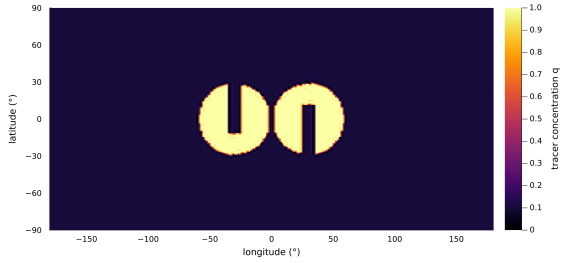
Table 4: Comparison of simulations with different stabilization choices, including combinations. All results were computed using ClimaCore.jl except for the labeled Firedrake run. Here H, L, SU, and SUPG denote hyperdiffusion, the quasi-monotone limiter, streamline upwind, and streamline-upwind Petrov–Galerkin stabilization, respectively.

Stabilization	q_{over}	q_{under}	ℓ_1	ℓ_2	ℓ_∞
None	9.43×10^{-3}	-1.37×10^{-2}	9.60×10^{-4}	4.39×10^{-3}	1.34×10^{-1}
H	1.24×10^{-1}	-1.42×10^{-1}	1.62×10^{-1}	5.99×10^{-1}	6.82
L	4.69×10^{-7}	-7.32×10^{-8}	9.86×10^{-2}	4.48×10^{-1}	6.30
SU	3.99×10^{-2}	-7.36×10^{-2}	9.39×10^{-2}	4.36×10^{-1}	4.50
SU (Firedrake)	6.57×10^{-2}	-7.83×10^{-2}	7.92×10^{-2}	1.70×10^{-1}	6.17×10^{-1}
SUPG	5.96×10^{-2}	-4.61×10^{-2}	1.43×10^{-2}	2.75×10^{-2}	6.25×10^{-1}
H + L	1.41×10^{-7}	-7.32×10^{-8}	1.43×10^{-1}	5.98×10^{-1}	7.01
L + SU	4.69×10^{-7}	-7.32×10^{-8}	1.21×10^{-1}	5.14×10^{-1}	5.52
L + SUPG	4.69×10^{-7}	-7.32×10^{-8}	9.86×10^{-2}	4.48×10^{-1}	6.30
H + L + SU	1.38×10^{-7}	-7.32×10^{-8}	1.63×10^{-1}	6.34×10^{-1}	6.58
H + L + SUPG	1.41×10^{-7}	-7.32×10^{-8}	1.43×10^{-1}	5.98×10^{-1}	7.01

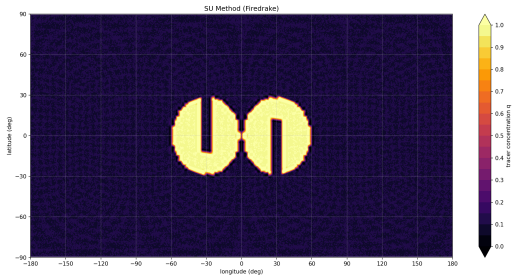
First, we turn our attention to the solutions computed using the SU stabilization method, as this was the case computed by both the ClimaCore.jl and Firedrake implementations. The solutions at the initial, intermediate ($t = T/2$) and final ($t = T$) times are shown in Figure 2. Observe that the solutions computed using different implementations qualitatively match with one another over the time domain, with visibly comparable smearing of the sharp tracer features by the final time. This is also reflected in the error results shown in Table 4, where all of the metrics match up to the order of magnitude, except for ℓ_∞. This comparison provides a useful consistency check, as the main behavior of the SU method matches between two independent implementations. However, the discrepancy in ℓ_∞ indicates that they are not numerically identical, and more work is required to find the source of this difference.

In Figure 3 we show the tracer field at the final time point computed using `ClimaCore.jl` for four stabilization methods, including the case without stabilization[1]. Observe that the solution without stabilization is able to retain the sharp features of the slotted

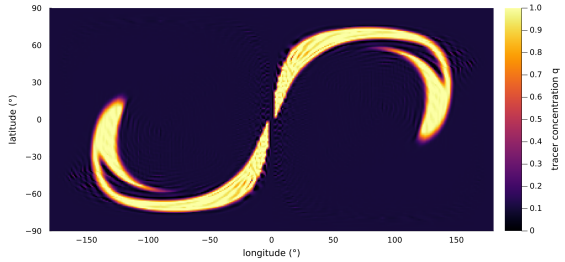
[1]For completeness, the tracer fields at both $t = T/2$ and $t = T$ for all ten ClimaCore.jl stabilization configurations are shown in Appendix A.



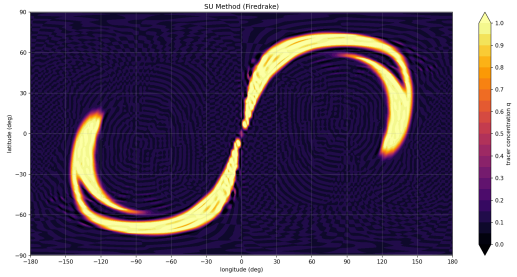
(a) ClimaCore.jl, $t = 0$.



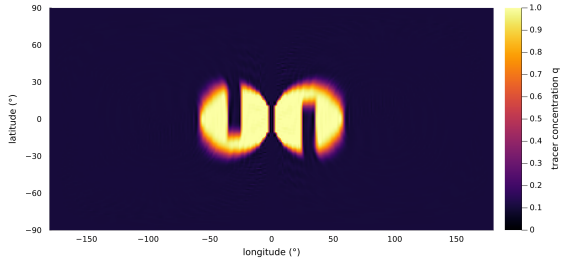
(b) Firedrake, $t = 0$.



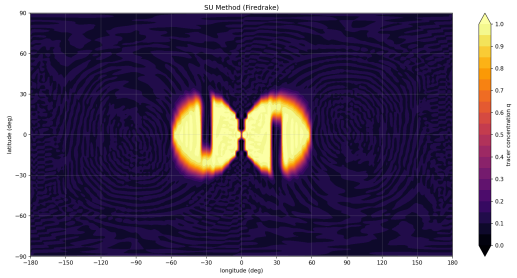
(c) ClimaCore.jl, $t = T/2$.



(d) Firedrake, $t = T/2$.



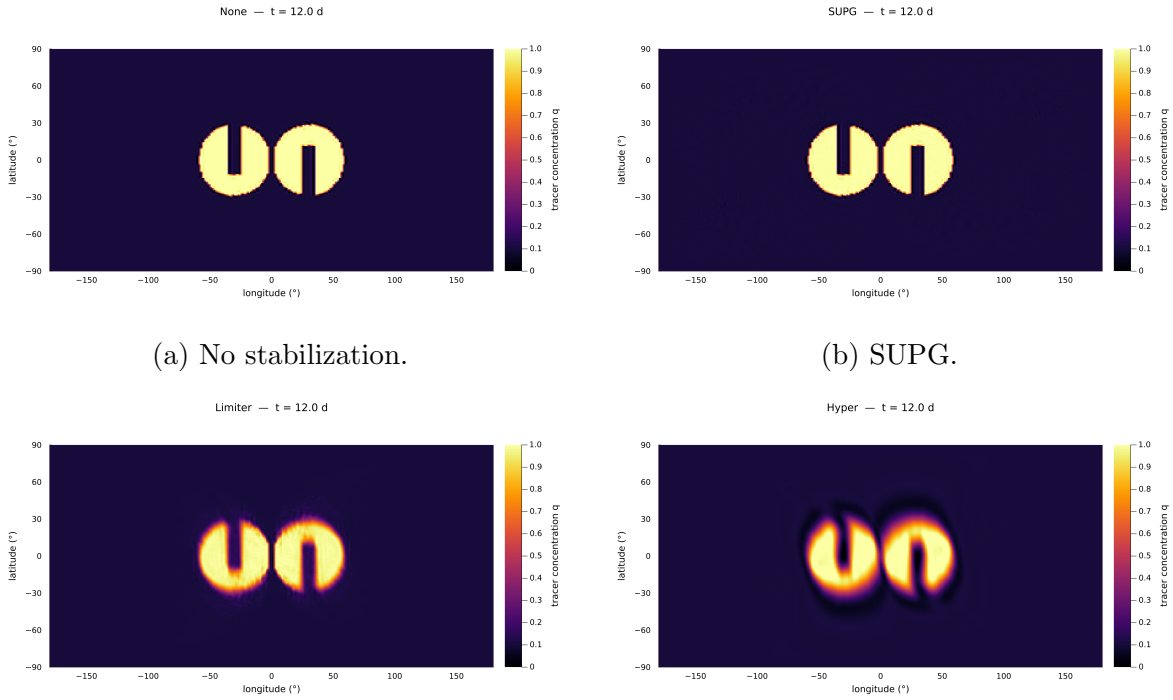
(e) ClimaCore.jl, $t = T$.



(f) Firedrake, $t = T$.

Figure 2: Comparison of the streamline-upwind (SU) solutions computed with ClimaCore.jl and Firedrake. The top row confirms the common initial tracer configuration at $t = 0$; the middle and bottom rows show the tracer fields at $t = T/2$ and $t = T$, respectively.

cylinders quite well and the solution with SUPG is not visually different than that with no stabilization. As expected, we see more blurring/smearing around the sharp features of the cylinders in the case of the quasi-monotone limiter and even more in the hyperdiffusion case.



(a) No stabilization.

(b) SUPG.

(c) Quasi-monotone limiter.

(d) Hyperdiffusion.

Figure 3: Tracer fields at final time point $t = T$ for four stabilization methods implemented in `ClimaCore.jl`.

Now turning our attention to Table 4, we can better assess the methods' performance in preventing unphysical extrema while retaining the overall tracer field. Observe that compared to the no stabilization case, the quasi-monotone limiter drastically reduces final-time overshoot and undershoot, with q_{under} and q_{over} reducing down to 10^{-7}–10^{-8}. However, this is done at the expense of significantly larger global errors. Hyperdiffusion performs poorly across all of the computed metrics compared to the no stabilization case. Among the ClimaCore.jl stabilized solutions, the SUPG method performs best at preserving the tracer structure (as is evident by the smallest ℓ_1, ℓ_2, and ℓ_∞ errors), however it does not improve on the unstabilized baseline. Finally, in the cases where the limiter is included in a combined method, the extrema errors remain in the same 10^{-7}–10^{-8} range, showing that the behavior is analogous to that when the limiter method used alone.

Overall, these results illustrate the tradeoff between preserving global structure and suppressing local overshoots and undershoots. The quasi-monotone limiter is most effective at preventing spurious oscillation, but does so at the expense of global errors.

The SUPG method is better than the limiters at preserving the global structure, but has larger q_{under} and q_{over} errors. No single stabilization method of those tested is best across all metrics – and most notably, the SUPG method does not improve on the no stabilization baseline for any of the metrics.

It should be noted that the metrics chosen here for the definition of the errors, namely equations (13) and (15), only account for the difference between the final time solutions and the initial conditions. They do not track the temporal evolution over time. For a test case such as the one analyzed in this work, where the flow distorts the initial tracer configuration and then reverts it back to match its initial condition, tracking the history of the errors at every time step would give more information on the performance of the different stabilization methods in the transient stages. Finally, higher-order polynomial degree tests should be considered as they would highlight the benefits of the stabilization methods compared to the no stabilization results.

5 Theoretical Analysis

5.1 Conservation, boundedness, and variation control

The exact transport system possesses two different kinds of structure. The conservative variables ρ and $Q = \rho q$ satisfy global conservation laws, whereas the mixing ratio q satisfies a pointwise transport equation. Indeed, expanding the tracer-density equation gives

$$0 = \frac{\partial(\rho q)}{\partial t} + \nabla \cdot (\rho q \mathbf{u}) \tag{17}$$

$$= \rho \left(\frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q \right) + q \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right). \tag{18}$$

Consequently, wherever $\rho > 0$, the density equation implies

$$\frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q = 0. \tag{19}$$

Thus q is constant along characteristics of the exact velocity field. In particular, the continuous solution satisfies

$$\min_{\Omega} q(\cdot, 0) \leq q(\mathbf{x}, t) \leq \max_{\Omega} q(\cdot, 0) \qquad \text{for all } t \tag{20}$$

for which the characteristic flow remains well defined. For the present slotted-cylinder problem, the physically relevant interval is therefore $0.1 \leq q \leq 1$.

Semidiscrete conservation

The conservative SUPG formulation in (6) preserves total tracer mass independently of whether it preserves the pointwise bounds in (20). To see this, choose the constant test

function $w_h = 1$. This choice is admissible because the computational domain is the closed sphere and the finite element space contains constants. Since $\nabla 1 = 0$, both the advective weak term and the SUPG correction vanish. There is also no boundary-flux contribution on a closed surface. Hence,

$$\left(\frac{\partial Q_h}{\partial t}, 1 \right) = 0, \tag{21}$$

which gives

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_\Omega Q_h \, \mathrm{d}\Omega = 0. \tag{22}$$

The same argument applied to the density equation yields

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_\Omega \rho_h \, \mathrm{d}\Omega = 0. \tag{23}$$

These identities hold at the semidiscrete level, up to quadrature and linear solver tolerances. A Runge–Kutta method also preserves these linear invariants provided that the residual at every stage has zero integral.

The factor ρ_h in the tracer SUPG correction is nevertheless important. It makes the stabilization act on the advective residual of the mixing ratio while the prognostic variable $Q_h = \rho_h q_h$ remains conservative. This mass-weighted construction supports consistency between the density and tracer equations, including preservation of a spatially constant mixing ratio. It does *not*, by itself, imply $q_h \geq 0$ or prevent the creation of new extrema.

Why standard SUPG is not bound preserving or TVD

The SUPG term adds residual-based dissipation primarily in the streamline direction [4]. This improves stability for advection-dominated problems, but the resulting high-order finite element operator is not generally monotone and does not satisfy a discrete maximum principle. In particular, the consistent mass matrix and the off-diagonal entries of the transport operator need not have the sign structure required for a positivity-preserving update [12]. This distinction is visible in Table 4: the SUPG solution conserves the transported quantity but still produces nonzero overshoot and undershoot metrics, whereas the quasi-monotone limiter reduces both metrics to nearly machine precision.

This limitation is consistent with the classical order barrier identified by Godunov: within the usual linear one-step setting, a method that is monotone for linear advection cannot be more than first-order accurate [13]. The theorem does not apply verbatim to every multidimensional finite element formulation, but it identifies the basic reason that a consistent residual-based stabilization such as SUPG (which is linear when applied to a linear PDE, and nonlinear otherwise) cannot simultaneously provide sharp resolution and unconditional monotonicity. A nonlinear limiting or nonlinear diffusion mechanism is required.

For a one-dimensional grid, the discrete total variation is commonly defined by

$$\mathrm{TV}(q^n) = \sum_j |q^n_{j+1} - q^n_j|, \tag{24}$$

and a method is TVD if $\mathrm{TV}(q^{n+1}) \le \mathrm{TV}(q^n)$ [14]. Strict TVD should be interpreted cautiously for the present spherical deformational-flow benchmark. Even the exact multidimensional transport map can stretch interfaces and increase $\int_\Omega |\nabla q| \, \mathrm{d}\Omega$ during the cycle. The more appropriate requirements here are preservation of the physical interval $[0.1, 1]$, absence of numerically generated extrema, and control of excess variation relative to the exact transported field. Since the flow returns the tracer to its initial configuration at $t = T$, the ratio $\mathrm{TV}(q_h(T))/\mathrm{TV}(q_h(0))$ can still be used as an end-of-cycle diagnostic, but a per-step TVD inequality is not expected from the exact multidimensional dynamics.

Modifications needed for sharper and bound-preserving transport

A first extension is a residual-dependent discontinuity-capturing or crosswind-diffusion term. One representative form is

$$\mathcal{S}_{\mathrm{DC}}(q_h, w_h) = \sum_K \int_K \nu_{\mathrm{DC},K} \, (\mathbf{P}_\perp \nabla q_h) \cdot (\mathbf{P}_\perp \nabla w_h) \, \mathrm{d}\Omega, \tag{25}$$

where

$$\mathbf{P}_\perp = \mathbf{I} - \widehat{\mathbf{u}} \otimes \widehat{\mathbf{u}}, \qquad \widehat{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|}, \tag{26}$$

and $\nu_{\mathrm{DC},K} \ge 0$ is activated by a local residual or gradient sensor. At points where $\|\mathbf{u}\|$ is zero or sufficiently small, the projector must be regularized, for example by replacing $\|\mathbf{u}\|$ with $\max(\|\mathbf{u}\|, \varepsilon_u)$ for a small positive tolerance ε_u. The original "beyond SUPG" construction adds precisely this type of discontinuity-capturing mechanism to control strong internal and boundary layers [15]. Such a term damps crosswind oscillations that streamline diffusion alone cannot see and can substantially reduce ringing near the slots. However, a generic shock-capturing term is not automatically TVD or maximum-principle preserving; its coefficient and nonlinear sensor must be designed with those properties in mind.

A stronger route is conservative algebraic flux correction or convex limiting. In algebraic form, write the high-order semidiscrete method as

$$\mathbf{M}_C \dot{\mathbf{q}} + \mathbf{K}_H \mathbf{q} = 0, \tag{27}$$

where $\mathbf{M}_C$ is the consistent mass matrix and $\mathbf{K}_H$ contains the Galerkin and SUPG contributions. A bound-preserving low-order method is first constructed using a lumped mass matrix $\mathbf{M}_L$ and sufficient graph viscosity:

$$\mathbf{M}_L \dot{\mathbf{q}} + \mathbf{K}_L \mathbf{q} = 0. \tag{28}$$

The low-order update is chosen to satisfy a local maximum principle under an appropriate CFL restriction. The difference between the high- and low-order operators is then decomposed into pairwise antidiffusive fluxes $f_{ij} = -f_{ji}$. Replacing each flux by $\alpha_{ij} f_{ij}$, with $0 \leq \alpha_{ij} \leq 1$, restores as much high-order accuracy as the local bounds permit while retaining conservation. This is the finite element flux-correction framework developed in [16]; related invariant-domain and maximum-principle preserving continuous finite element constructions are given in [17].

For the coupled density–tracer system, the conservative variable Q_h should continue to be updated in mass form, but the admissible bounds should be imposed on $q_h = Q_h/\rho_h$. The density and tracer fluxes must therefore be limited consistently. Pointwise clipping of q_h after a time step would restore nonnegativity but generally destroy tracer-mass conservation, whereas pairwise conservative flux limiting can enforce the bounds and preserve (22). When an SSP Runge–Kutta method is used, the limiter must be applied at every forward-Euler stage so that the fully discrete update remains a convex combination of bound-preserving stages [11].

These additions have clear benefits: they prevent physically impossible negative tracer concentrations, suppress spurious ringing, and make the extrema metrics in (13) controlled properties of the algorithm rather than empirical outcomes. The tradeoffs are additional nonlinearity, a CFL restriction for explicit bound preservation, and some extra diffusion near discontinuities. A useful practical hierarchy is therefore SUPG for streamline stability, residual-based discontinuity capturing for unresolved crosswind gradients, and conservative flux limiting only where the physical bounds would otherwise be violated.

5.2 Asymptotic Approach to Time-Dependent SUPG

In Section 3, we explored the efficacy of the time-independent SUPG implementation using the stabilizing term

$$\tau^e \int_\Omega (u \cdot \nabla v) \left(\nabla \cdot (Qu) + \frac{\partial Q}{\partial t} \right)$$

Where the term $\frac{\partial Q}{\partial t}$ is computed from the *previous* time step as a proxy for the $\frac{\partial Q}{\partial t}$ field at the current time step. While this allows us to set up an explicit method rather than an implicit method, it suffers from inaccuracy for large time steps. Rather than pursuing a computationally costly implicit SUPG method, we take an asymptotic expansion based approach to improve our proxy for $\frac{\partial Q}{\partial t}$ at the current time step.

Consider the SUPG stabilizing term (assuming no source, i.e. $s = 0$)

$$\tau^e \int_\Omega (u \cdot \nabla v) \left(\nabla \cdot (Qu) + \frac{\partial Q}{\partial t} \right)$$

$$= \tau^e \int_\Omega \nabla v \cdot \left[u \left(\nabla \cdot (Qu) + \frac{\partial Q}{\partial t} \right) \right]$$

And since $\nabla \cdot (AB) = (\nabla A) \cdot B + A(\nabla \cdot B)$, this can be written as

$$= \tau^e \int_\Omega \nabla \cdot \left\{ v \left[u \left(\nabla \cdot (Qu) + \frac{\partial Q}{\partial t} \right) \right] \right\} - \tau^e \int_\Omega v \left\{ \nabla \cdot \left[u \left(\nabla \cdot (Qu) + \frac{\partial Q}{\partial t} \right) \right] \right\}$$

By the divergence theorem,

$$\tau^e \int_\Omega \nabla \cdot \left\{ v \left[u \left(\nabla \cdot (Qu) + \frac{\partial Q}{\partial t} \right) \right] \right\} = \tau^e \oint_{\partial\Omega} v \left[u \left(\nabla \cdot (Qu) + \frac{\partial Q}{\partial t} \right) \right] \cdot \hat{\mathbf{n}} = 0,$$

since v vanishes on $\partial\Omega$. Thus, the SUPG stabilizing term is

$$-\tau^e \int_\Omega v \left\{ \nabla \cdot \left[u \left(\nabla \cdot (Qu) + \frac{\partial Q}{\partial t} \right) \right] \right\}$$

Then the tracer dynamics are described by

$$\int_\Omega v \left(\frac{\partial Q}{\partial t} \right) = - \int_\Omega v(\nabla \cdot (Qu)) - \tau^e \int_\Omega v \left\{ \nabla \cdot \left[u \left(\nabla \cdot (Qu) + \frac{\partial Q}{\partial t} \right) \right] \right\}.$$

We now perform a nondimensionalization of this problem. Take

$$Q = c\bar{Q}, \quad u = U\bar{u}, \quad t = T\bar{t}, \quad \nabla\cdot = \frac{1}{L}\bar{\nabla}\cdot$$

so we have

$$\begin{aligned} \frac{c}{T} \int_\Omega v \left(\frac{\partial \bar{Q}}{\partial \bar{t}} \right) = &- \frac{cU}{L} \int_\Omega v(\bar{\nabla} \cdot (\bar{Q}\bar{u})) - \tau^e \frac{cU^2}{L^2} \int_\Omega v\{\bar{\nabla} \cdot [\bar{u}(\bar{\nabla} \cdot (\bar{Q}\bar{u})]\} \\ &- \tau^e \frac{cU}{LT} \int_\Omega v \left\{ \bar{\nabla} \cdot \left[\bar{u} \left(\frac{\partial \bar{Q}}{\partial \bar{t}} \right) \right] \right\}. \end{aligned}$$

By the substitution $T = L/U$, we get

$$\begin{aligned} \int_\Omega v \left(\frac{\partial \bar{Q}}{\partial \bar{t}} \right) = &- \int_\Omega v(\bar{\nabla} \cdot (\bar{Q}\bar{u})) - \tau^e \frac{U}{L} \int_\Omega v\{\bar{\nabla} \cdot [\bar{u}(\bar{\nabla} \cdot (\bar{Q}\bar{u})]\} \\ &- \tau^e \frac{U}{L} \int_\Omega v \left\{ \bar{\nabla} \cdot \left[\bar{u} \left(\frac{\partial \bar{Q}}{\partial \bar{t}} \right) \right] \right\} \end{aligned}$$

In our atmospheric simulation of the Earth, $U \approx 10^2[km/hr]$, $L \approx 6 \cdot 10^3[km]$. We have $\tau^e \approx 2.5 \cdot 10^{-1}[hr]$. Thus,

$$\tau^e \frac{U}{L} \approx 4 \cdot 10^{-3} \ll 1.$$

Letting $\varepsilon := \tau^e \frac{U}{L}$, we have

$$\begin{aligned} \int_\Omega v \left(\frac{\partial \bar{Q}}{\partial \bar{t}} \right) = &- \int_\Omega v(\bar{\nabla} \cdot (\bar{Q}\bar{u})) \\ &- \varepsilon \int_\Omega v(\bar{\nabla} \cdot \{\bar{u}[\bar{\nabla} \cdot (\bar{Q}\bar{u})]\}) - \varepsilon \int_\Omega v \left\{ \bar{\nabla} \cdot \left[\bar{u} \left(\frac{\partial \bar{Q}}{\partial \bar{t}} \right) \right] \right\} \end{aligned} \tag{29}$$

From this point on, we will neglect the overbar notation (e.g. $\bar{Q}$) in the non-dimensionalization for simplicity. Consider the asymptotic expansion

$$Q = Q_0 + \varepsilon Q_1 + ...$$

Substituting into (19) and expanding up to $\mathcal{O}(1)$, we get the leading term relation

$$\int_\Omega v\left(\frac{\partial Q_0}{\partial t}\right) = -\int_\Omega v(\nabla \cdot (Q_0 u)) \tag{30}$$

Expanding up to $\mathcal{O}(\varepsilon)$, we have

$$\begin{aligned}\int_\Omega v\left(\frac{\partial Q_1}{\partial t}\right) = &-\int_\Omega v(\nabla \cdot (Q_1 u)) - \int_\Omega v\{\nabla \cdot [u(\nabla \cdot (Q_0 u))]\} \\ &- \int_\Omega v\left\{\nabla \cdot \left[u\left(\frac{\partial Q_0}{\partial t}\right)\right]\right\}.\end{aligned} \tag{31}$$

If we consider the time discretization

$$\frac{\partial Q_i^n}{\partial t} \approx \frac{Q_i^n - Q_i^{n-1}}{\Delta t}, \tag{32}$$

we can construct an algorithm based on solving consecutive pairs (Q_0^{n+1}, Q_1^{n+1}) from (Q_0^n, Q_1^n). First, we initialize the start system based on the initial concentration Q. We can set

$$(Q_0^0, Q_1^0) = (Q, 0)$$

Then, given some pair (Q_0^n, Q_1^n) at the n^{th} time step, we follow the diagram in Figure 4 and the algorithm outlined below.

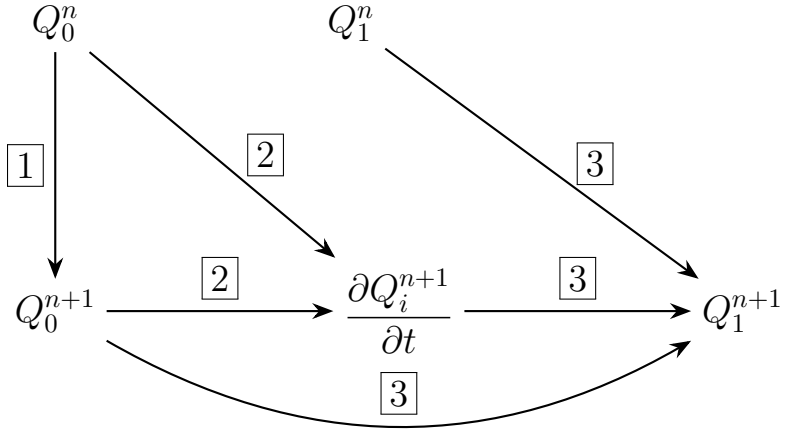


Figure 4: Diagrammatic representation of the solution algorithm for the asymptotic approach of the pure advection problem with SU stabilization.

1. Use the RK4 algorithm to compute Q_0^{n+1} from Q_0^n within the standard Galerkin method in (30)

2. Numerically compute $\frac{\partial Q_0^{n+1}}{\partial t}$ from Q_0^n and Q_0^{n+1} in (32).

3. Use the SU method to compute Q_1^{n+1} from Q_1^n, Q_0^{n+1}, and $\frac{\partial Q_0^{n+1}}{\partial t}$ in (31).

4. Set $n = n + 1$, compute the tracer concentration $Q^n = Q_0^n + \varepsilon Q_1^n$, and repeat from step 1.

While this algorithm involves solving two discrete Galerkin-type problems at each time step (standard Galerkin for Q_0^{n+1}, SU for Q_1^{n+1}), it allows us to include a non-delayed time-dependent term in the strong-form residual in the stabilizing term.

It should be noted that our asymptotic expansion $Q = Q_0 + \varepsilon Q_1$ is arbitrary, and could contain more terms. The algorithm detailed in this section is but a member of a class of algorithms; the algorithm where the number of expansion terms in $n = 2$. For example, with the addition of a third term Q_2 scaled by ε^2, we could detail an algorithm which requires solving three discrete Galerkin-type problems at each time step. Computation scales linearly with the number of expansion terms, as n terms requires n Galerkin-type problems to be solved at each time step. The choice for the number of expansion terms used should be based on comparisons with implicit SUPG methods, as well as computational constraints.

6 Conclusions and Future Work

In this work, we have implemented the Streamline Upwind (SU) and Streamline Upwind Petrov-Galerkin (SUPG) stabilization methods for finite element methods (FEM) using the `ClimaCore.jl` library [5]. An implementation for SU is also developed in Firedrake [6] to compare and contrast it with the implementation in ClimaCore.jl. Other stabilization methods, such as hyperdiffusion and flux-limiters, are also implemented in ClimaCore.jl and we conduct a cross-comparison of these four stabilization methods.

The comparison between the stabilization methods was performed using the slotted-cylinders deformation flow benchmark problem. For this problem, we found that the solution obtained without numerical stabilization is able to resolve the sharp features of the cylinders quite well and gave the smallest global errors. Stabilization using quasi-monotone limiters greatly improved at reducing spurious extrema at the expense of global accuracy, leading to smearing over the course of the simulation. The hyperdiffusion method performed relatively poorly across all metrics. When methods were combined, the flux-limiter method was shown to largely determine the extrema behavior, with hyperdiffusion, SU, and SUPG having minimal additive impact. In a comparison of implementation, both Python's Firedrake and Julia's ClimaCore.jl produced errors at the same order of magnitude, with the notable exception of the

ℓ_∞-error, where they differ by a single order of magnitude. The SUPG method was not found to improve upon the no stabilization baseline, for the metrics tested in this work. However, among the stabilized runs, the SUPG method resulted in the lowest global errors.

Following these results, multiple directions for this research may be pursued. For instance, additional test cases with a variety of tracer configurations and fluid regimes would provide a more comprehensive comparison of the stabilization methods. Moreover, additional error tracking measuring the error over time (rather than only at the final time) would give a more complete view. Finally, our implementations of the the SU and SUPG methods would benefit from different sets of parameter studies.

The theoretical analysis also clarifies that conservation and bound preservation are distinct properties. The conservative semidiscrete SUPG formulation preserves total density and tracer mass on the closed spherical domain, but standard linear SUPG is not generally maximum-principle preserving or total variation diminishing. The overshoots and undershoots observed numerically are therefore consistent with the structure of the method. A future bound-preserving implementation could combine SUPG with residual-based crosswind discontinuity capturing and a conservative flux or convex limiter applied consistently to the density and tracer fluxes.

An asymptotics based SUPG algorithm based on the SU and delayed SUPG methods was developed, but remains to be implemented. For potential future research, this algorithm could be implemented and compared with implicit SUPG methods, and the impact of the number of asymptotic expansion terms on accuracy could be studied.

7 Acknowledgments

We would like to gratefully acknowledge the support of SIAM and the University of Delaware for providing the resources to conduct this work. Additionally, a special thank you to Dr. Valeria Barra for her excellent mentorship and support.

8 AI Use Disclosure

The code development for the Firedrake implementation was aided by Claude Opus 4.8.

Bibliography

[1] Paul A. Ullrich et al. “Impact and importance of hyperdiffusion on the spectral element method: A linear dispersion analysis”. In: *Journal of Computational Physics* 375 (Dec. 2018), pp. 427–446. ISSN: 10902716. DOI: 10.1016/j.jcp.2018.06.035.

[2] Oksana Guba, Mark Taylor, and Amik St-Cyr. "Optimization-based limiters for the spectral element method". In: *Journal of Computational Physics* (2010). DOI: https://doi.org/10.1016/j.jcp.2014.02.029.

[3] Steven T Zalesak. "Fully multidimensional flux-corrected transport algorithms for fluids". In: *Journal of Computational Physics* 31.3 (1979), pp. 335–362. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(79)90051-2. URL: https://www.sciencedirect.com/science/article/pii/0021999179900512.

[4] Alexander N. Brooks and Thomas J.R. Hughes. "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations". In: *Computer Methods in Applied Mechanics and Engineering* (1982). DOI: https://doi.org/10.1016/0045-7825(82)90071-8.

[5] CliMA. *ClimaCore.jl*. Version v0.14.51. Apr. 2025. URL: https://github.com/CliMA/ClimaCore.jl.git.

[6] David A. Ham et al. *Firedrake User Manual*. First edition. Imperial College London et al. May 2023. DOI: 10.25561/104839.

[7] Dennis Yatunin et al. "The Climate Modeling Alliance Atmosphere Dynamical Core: Concepts, Numerics, and Scaling". In: *Journal of Advances in Modeling Earth Systems* 18.3 (2026), e2025MS005014. DOI: https://doi.org/10.1029/2025MS005014. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2025MS005014.

[8] Ramachandran D. Nair and Peter H. Lauritzen. "A class of deformational flow test cases for linear transport problems on the sphere". In: *Journal of Computational Physics* (2010). DOI: https://doi.org/10.1016/j.jcp.2010.08.014.

[9] Tayfun Tezduyar and Sunil Sathe. "Stabilization Parameters in SUPG and PSPG Formulations". In: *Journal of Computational and Applied Mathematics* (2003).

[10] The Gusto Development Team. *Gusto: A library for dynamical cores using compatible finite element discretisations*. https://github.com/firedrakeproject/gusto. 2026.

[11] Chi-Wang Shu and Stanley Osher. "Efficient implementation of essentially non-oscillatory shock-capturing schemes". In: *Journal of computational physics* (1988). DOI: https://doi.org/10.1016/0021-9991(88)90177-5.

[12] Jean-Luc Guermond, Bojan Popov, and Yong Yang. "The Effect of the Consistent Mass Matrix on the Maximum-Principle for Scalar Conservation Equations". In: *Journal of Scientific Computing* 70.3 (2017), pp. 1358–1366. DOI: 10.1007/s10915-016-0285-7.

[13] Sergei K. Godunov. "A Difference Method for Numerical Calculation of Discontinuous Solutions of the Equations of Hydrodynamics". In: *Matematicheskii Sbornik* 47.3 (1959). In Russian; English translation published as U.S. Joint Publications Research Service report JPRS 7226 (1969), pp. 271–306.

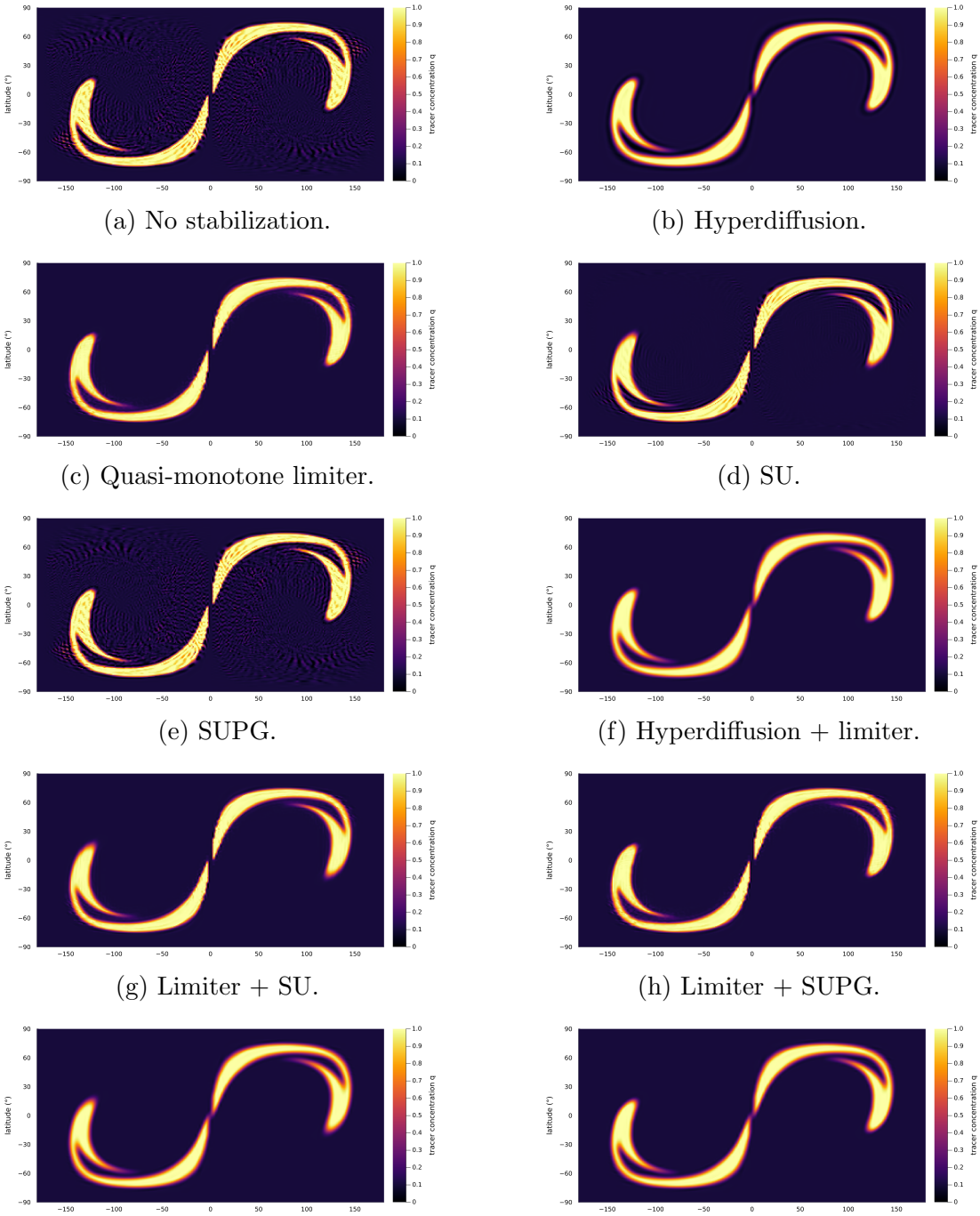
[14] Ami Harten. "High Resolution Schemes for Hyperbolic Conservation Laws". In: *Journal of Computational Physics* 49.3 (1983), pp. 357–393. DOI: 10.1016/0021-9991(83)90136-5.

[15] Michel Mallet Thomas J.R. Hughes and Akira Mizukami. "A new finite element formulation for computational fluid dynamics: II. Beyond SUPG". In: *Computer Methods in Applied Mechanics and Engineering* (1986). DOI: https://doi.org/10.1016/0045-7825(86)90110-6.

[16] Dmitri Kuzmin and Stefan Turek. "Flux Correction Tools for Finite Elements". In: *Journal of Computational Physics* 175.2 (2002), pp. 525–558. DOI: 10.1006/jcph.2001.6955.

[17] Jean-Luc Guermond and Bojan Popov. "Invariant Domains and Second-Order Continuous Finite Element Approximation for Scalar Conservation Equations". In: *SIAM Journal on Numerical Analysis* 55.6 (2017), pp. 3120–3146. DOI: 10.1137/16M1106560.

A Tracer Fields for All Stabilization Configurations



(a) No stabilization.

(b) Hyperdiffusion.

(c) Quasi-monotone limiter.

(d) SU.

(e) SUPG.

(f) Hyperdiffusion + limiter.

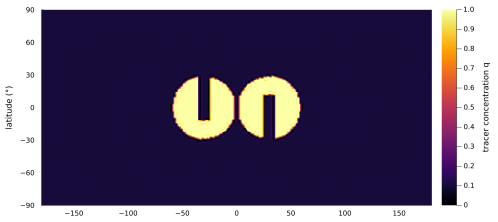
(g) Limiter + SU.

(h) Limiter + SUPG.

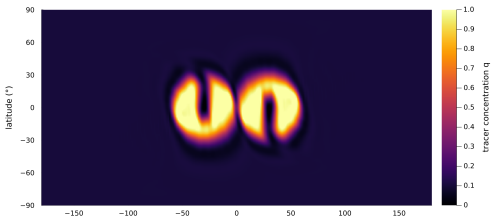
(i) Hyperdiffusion + limiter + SU.

(j) Hyperdiffusion + limiter + SUPG.

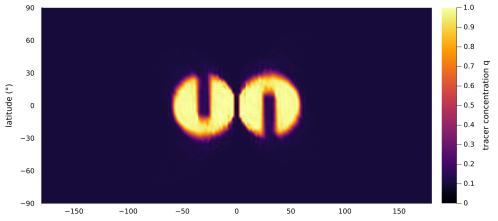
Figure 5: Tracer fields at the intermediate time $t = T/2$ for all stabilization choices implemented in ClimaCore.jl.



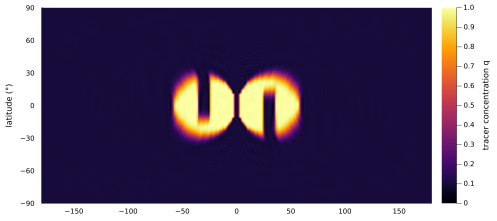
(a) No stabilization.



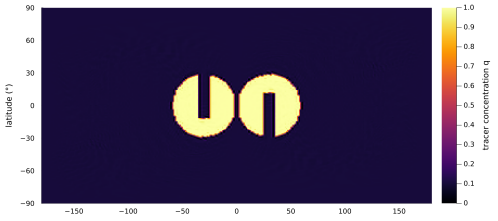
(b) Hyperdiffusion.



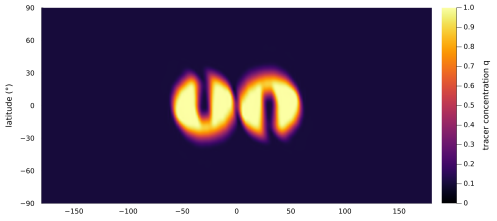
(c) Quasi-monotone limiter.



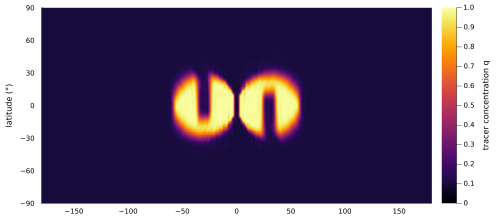
(d) SU.



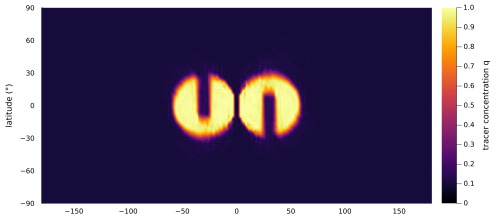
(e) SUPG.



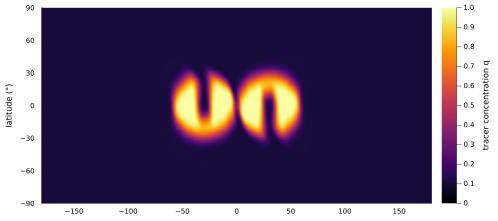
(f) Hyperdiffusion + limiter.



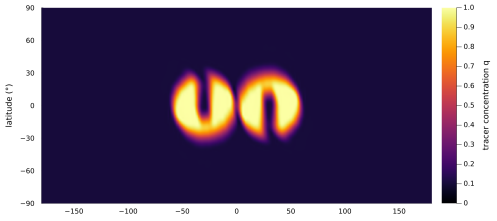
(g) Limiter + SU.



(h) Limiter + SUPG.



(i) Hyperdiffusion + limiter + SU.



(j) Hyperdiffusion + limiter + SUPG.

Figure 6: Tracer fields at the final time $t = T$ for all stabilization choices implemented in ClimaCore.jl.