# ESSBP Overview

Alex Bercik

September 2, 2021

This is a brief introduction to ESSBP, a tensor-product Entropy-Stable Summation By Parts solver for the numerical solution of PDEs in Python.

## Contents

# 1 Code Organization

The code is structured into 2 main sections: `Driver` and `Source`.

## 1.1 `Driver`

This directory contains the files required to run the code in `Source`. These files define parameters, then call the classes defined in `Source` to initialize problems with the parameters provided. Each instance of the class defines a new problem, and so if multiple problems are to be run at once, one only need to define several instances in the driver file. One can also perform analysis here in the driver file.

## 1.2 `Source`

Holds the source code for the solver. `DiffEq` holds classes for each type of equation to be solved, as well as the base class `DiffEqBase`. Here is where information on how to solve the specific equation is stored, such as fluxes. `Disc` stores information for spatial discretization, such as definition of operators and the mesh. `Solvers` holds information specific to the method used, such as Finite Difference, SBP, or DG. This calls `Disc` classes and to complete the methods written in a more general form. `TimeMarch` defines methods to time march the resulting semi-discretized ODE and is called by the `solve` function in `Solvers/PdeSolver`. Finally `Methods` contains specific functions that are useful throughout the code. These can be plotting tools, animation tools, and convergence tests in `Methods/Analysis`, debugging tools in `Methods/DebugTools`, or various functions such as 3D matrix multiplication compiled with Numba to speed up performance in `Methods/Functions`.

# 2 Example Problem: SBP Linear Convection

We begin in `Driver/RunLCE` by defining parameters. We choose one of the 4 SBP finite difference choices for `disc_type`, noting that since we are solving the problem in 1 dimension, `'Rdn1'` and `'R0'` will result in the same operators.

We initialize a class instance `diffeq` for our problem with

```
from Source.DiffEq.LinearConv import LinearConvSbp
diffeq = LinearConvSbp(para, obj_name, q0_type)
```

where `para` is the wave speed, `obj_name=None` means we have no objective function to calculate (see 3 for the case where calculate an objective function), and `q0_type` is a string that will automatically create an initial condition under `PdeBase.set_q0` (unless a specific initial condition is given to the solver as variable `q0`).

The initialization of `diffeq` will first trigger the initialization defined in `LinearConv`, which stores the wave speed `para` as an attribute, and then inherits the initialization from `PdeBase`. This similarly stores the initial condition setting as an attribute, then once again inherits the initialization of `DiffEqBase`. Aside from a few more parameters being redundantly stored, no further initialization occurs for this case. We notice however important differences for cases with varying parameters. Since these cases differ in treatment in several aspects of the code, they are described later in **??**.

`diffeq` is now a class instance with access to all the functions responsible for calculating the residual in any given physical element. These functions include various derivatives of the residual used for implicit time marching, and includes functions to calculate the contributions from the SATs at any particular interface. This is because `diffeq` inherited the functions from `PdeBaseCons` (specific to conservative PDEs in the form $q_t + E_x(q) = G(q,t)$) as well as its parent classes `PdeBase` and `DiffEqBase`, but also from `SatBaseCons` and `SatBase`. These functions however are not complete, as they require information about the spatial discretization. For this we must initialize the solver class.

```
from Source.Solvers.PdeSolver import PdeSolver
solver = PdeSolver(diffeq,
                   tm_method, dt, tf, t_init,
                   q0, n_q0,
                   p, spat_disc_type, nn, nelem, nen,
                   isperiodic, xmin, xmax,
                   bool_plot_sol = bool_plot_sol,
                   print_sol_norm = print_sol_norm)
```

The function `PdeSolver` simply returns an initialization of the `PdeSolverSbp` class with the relevant parameters. The first step of this initialization is to initialize an instance of the `MakeSbpOp` class under the attribute `self.sbp` within `solver`. This stores all information relevant for the spatial discretization on the reference element. Based on the degree `p`, and either the total number of nodes `nn` or number of elements `nelem` and number of nodes per element `nen`, the SBP operators are created and the remaining variables are assigned. Next the 1D mesh is created by an initialization of the `Disc/MakeMesh` class under `self.mesh` using the physical boundaries `xmin` and `xmax`, a flag `isperiodic` indicating periodicity, and the number of elements and nodal locations defined in the previous step. Because a simple linear mapping with equal size elements is employed, the mesh Jacobian is constant over the entire domain. This allows us to compute the mesh Jacobian, its inverse, and the determinant only once using the first element. Likewise, we then use these quantities to compute the physical element-wise operators only once by then calling `self.sbp.ref_2_phys_op`, as the physical operators are the same in each element. Note that this function as defined in `MakeSbpOp` has yet to

be generalized for higher dimensions, and would also need to be called individually for each element should a more general mesh mapping be employed. Finally, a Kroncker product is applied for cases where $q$ is a vector (i.e. systems).

The final step is to incorporate these quantities with the methods defined in the class instance `diffeq`. To do this, first `diffeq` is stored as an attribute within `solver` under the name `self.diffeq_in` (the reason for this name will become apparent soon). The mesh attributes and physical operators currently defined in `solver` are then passed to `solver.diffeq_in` to be stored as attributes there with the functions `solver.diffeq_in.set_mesh(...)` and `solver.diffeq_in.set_sbp_op(...)`. Without this step, as with any call to an instance of the `DiffEq` class, functions in `solver.diffeq_in` will return errors as the instance is missing the required attributes. `solver.diffeq_in` now contains all the relevant functions to calculate the residual in a given physical element, however, crucially, it does not have the ability to compute the global residual. Therefore, an instance of the class `Diffeq4SbpSolver` (defined in file `PdeSolverSbp.py`) is now initiated as an attribute within `solver` under the name `self.diffeq`. Not to be confused with `solver.diffeq_in`, the initialization of `solver.diffeq` takes functions originally defined in `solver.diffeq_in`, such as `solver.diffeq_in.dqdt` and `solver.diffeq_in.calc_sat`, which calculate the residual for an individual element interior and SAT contributions at an individual element interface, and combines them into a global function `solver.diffeq.dqdt` that calculates the residual over the entire domain. The initialization of `solver.diffeq` takes global functions defined in `PdeSolverSbp` as arguments (ex. `dqdt_sbp(...)` for the example above) and sets them as methods that can be called directly in `solver.diffeq`. In this way, the class instance `solver` now contains all the information required to solve the PDE.

The now semi-discrete system of ODEs (in time) can now be solved by simply calling

```
solver.solve()
```

For cases with non-varying parameters, this reverts to the function `solve_main` defined in `Solvers/OdeSolver`. For problems that do not require first solving for a steady initial condition, this first sets the initial condition using `q0 = solver.diffeq.set_q0()`, defined in `PdeBase`. For other problems, see **??**. The system of ODE's is then marched in time by

```
tm_class = TimeMarching(self.diffeq, self.tm_method, keep_all_ts,
                        bool_plot_sol = self.bool_plot_sol,
                        bool_calc_obj = self.bool_calc_obj,
                        print_sol_norm = self.print_sol_norm)
self.q_sol =  tm_class.solve(q0, self.dt, self.n_ts)
self.obj = tm_class.obj
self.obj_all_iter = tm_class.obj_all_iter
```

Upon initialization of the class instance `tm_class`, the relevant functions from `solver.diffeq`, which are passed as argument, are extracted and the time marching method is set according to the string `tm_method` originally passed to `solver`. The boolean `keep_all_ts`, hard coded in `OdeSolver.solve_t_final` (a subroutine of `OdeSolver.solve_main`), determines whether to return the solution vector at each time step or to simply return the final solution vector when calling `tm_class.solve(...)`. Similarly, the boolean flags `bool_plot_sol` and `print_sol_norm` that were originally passed to `solver` now determine whether or not to plot the solution and print the $L_2$ solution norm at each time step. `bool_calc_obj` controls the calculation of objective functions. Back when `diffeq` was initialized, the string `obj_name=None` was set, indicating there is no objective function to calculate, and setting `solver.bool_calc_obj=False`. Finally, the time marching method is set according to the string `solver.tm_method`. These methods are defined in parent classes inherited by the main `TimeMarching` class.

With all the required information to solve the residual and its derivatives stored in the passed class `solver.diffeq`, the problem is now marched in time by calling `tm_class.solve(...)` along with the initial condition, step size, and number of steps. The solution vector (at all time steps), and two empty variables corresponding to the non-existent objective function are finally passed back to `solver`.

# 3 Objective Functions

Upon initialization of `diffeq`, `obj_name` can be set to either a string or a tuple of strings, each labelling a particular objective function. The length of this tuple determines the parameter `diffeq.n_obj`. When `diffeq.n_obj` $> 0$, the initialization now calls the function `DiffEqBase.init4obj()`. Once again we assume the case of nonvarying parameters, and set `self.all_para_set = True`. For the alternative case, see **??**. If the objective has been previously evaluated and saved locally within a text file, the path to this file is now stored as an attribute. Note that these paths must be specified in the `__init__` functions of the problem-specific `DiffEq` files, otherwise an Attribute Error will be raised. In addition there must be functions defined in the problem-specific `DiffEq` file responsible for calculating each objective function, with a general function `calc_obj()` serving to call the individual functions and return a single vector with all desired objectives.

In the initialization of the solver class, the argument `bool_calc_obj` is passed as a boolean and stored as an attribute. This flag simply determines whether or not to evaluate the objective, and the default is set to `True` (unless `diffeq.n_obj` $= 0$). An additional method `solver.calc_obj` is also defined, which simply calls the main solve routine with an optional intial condition, but only returns the objective and standard deviation. This could be useful in post-analysis if only the objective is of interest. We stress the difference between the functions `diffeq.calc_obj` (and similarly `solver.diffeq.calc_obj`) and `solver.calc_obj`. The former defines the actual routine to calculate the objective function, wheras the latter calls the `solve()` routine.

When the time marching class is initiated, there is an option to manually overwrite the objective function by passing a new function as an argument. If using this option however, one must be careful that it take the three arguments (current solution vector, total number of time steps, time step size) and return `len{obj_name}` values. If this manual override option is not used, the objective function `self.fun_obj` is simply set as the default function from the given `diffeq` class instance, `solver.diffeq.calc_obj`. On each iteration of the time marching scheme, the function `self.fun_obj` is called within the `common(...)` routine of `solve(...)`, and the results are stored.

At the end of the time marching (return to `OdeSolver/solve_main()`), three quantities are returned. The first is the solution vector, the second is an array that contains the objective functions at each iteration, and the third is the sum of the objectives over all time steps. The second and third quantities are stored as `solver.obj_all_iter` and `solver.obj`, respectively. A running average of the objective functions is then performed, and quantities such as the standard deviation, `solver.std_obj`, are also stored.

# 4 Time Marching

# 5 Spatial Semi-Discretizations

We use the following notation. We take a mapping from reference coordinates $\xi_i$ to physical coordinates $x_i$ of an element $\kappa$. The Jacobian matrix $\dfrac{\partial x_i}{\partial \xi_j}$ defines the mapping. We define the determinant of this matrix as the "metric Jacobian"

$$\mathcal{J}_\kappa \equiv \left| \frac{\partial x_i}{\partial \xi_j} \right|$$

Note that this determinant of course depends on $\boldsymbol{x}$. The volume metrics

$$\text{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_\kappa$$

are defined at element nodes, whereas the surface metrics

$$\text{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l, b}$$

are defined on the 'mortar' facet nodes between the two elements. We denote by the superscript $(\xi_l, b)$ the quantities associated with the $b$ facet along the $\xi_l$ direction, or the facet between elements $\kappa$ and $\nu_{\xi_l, b}$. The matrix $H^\perp$ is a $d-1$ dimension tensor product of the other $H^{(1D)}$ matrices. For example, if considering a facet in the $\xi_1$ direction in 3D (so a 2D face) we use $H^\perp_{\xi_1} = H^{(1D)}_{\xi_2} \otimes H^{(1D)}_{\xi_3}$. Due to the SBP property we also have

$$Q_{\xi_l} + Q_{\xi_l}^T = E_{\xi,l} = E_{\xi_l,b} + E_{\xi_l,a} = \boldsymbol{t}_{\xi_l,b} H^\perp_{\xi_l} \boldsymbol{t}^T_{\xi_l,b} - \boldsymbol{t}_{\xi_l,a} H^\perp_{\xi_l} \boldsymbol{t}^T_{\xi_l,a}$$

where for example in 2D, $\boldsymbol{t}_{\xi,b} = \boldsymbol{t}^{(1D)}_b \otimes I_\eta$, $\boldsymbol{t}_{\xi,a} = \boldsymbol{t}^{(1D)}_a \otimes I_\eta$ are the extrapolation operators to the boundary facets along the $\xi$ direction. Note that the use of $H^{(1D)}$ is because the extrapolation operators extract the $\xi$ boundary components, but leave the $\eta$ components as a vector, and hence we keep the line integral along $\eta$.

## 5.1 Divergence Form

By divergence form, we mean the discretization uses the 'standard' straighforward discretization where the derivative operator acts directly on the flux. This is as opposed to the flux differencing Hadamard form. This does NOT refer to divergence form of the metrics. Skew-symmetric metrics terms are employed here as necessary to prove stability. We assume that $H$ is diagonal in order to prove energy stability.

$$\frac{d\boldsymbol{u}_\kappa}{dt} + \sum_{m=1}^{d} D_{x_m} \boldsymbol{\mathcal{F}}_m \left( \boldsymbol{u}_\kappa \right) = H^{-1} \sum_{l=1}^{d} \left[ \boldsymbol{t}_{\xi_l,b} H^\perp_{\xi_l} \left( \boldsymbol{f}_{\xi_l,b} - \boldsymbol{f}^\star_{\xi_l,b} \right) - \boldsymbol{t}_{\xi_l,a} H^\perp_{\xi_l} \left( \boldsymbol{f}_{\xi_l,a} - \boldsymbol{f}^\star_{\xi_l,a} \right) \right]$$

where

$$D_{x_m} = \frac{1}{2} \left[ \text{diag}\left( \mathcal{J}_\kappa \right) \right]^{-1} \sum_{l=1}^{d} \left[ D_{\xi_l} \text{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_\kappa + \text{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_\kappa D_{\xi_l} \right] \quad , \quad H = \text{diag}\left( \mathcal{J}_\kappa \right) H_\xi$$

$$\boldsymbol{f}_{\xi_l,a} = \sum_{m=1}^{d} \boldsymbol{t}^T_{\xi_l,a} \text{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_\kappa \boldsymbol{\mathcal{F}}_m \left( \boldsymbol{u}_\kappa \right) \quad , \quad \boldsymbol{f}_{\xi_l,b} = \sum_{m=1}^{d} \boldsymbol{t}^T_{\xi_l,b} \text{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_\kappa \boldsymbol{\mathcal{F}}_m \left( \boldsymbol{u}_\kappa \right)$$

and $\boldsymbol{f}^\star(\boldsymbol{u}_\kappa, \boldsymbol{u}_\nu)$ are some consistent numerical flux (and antisymmetric if we have boundary nodes or linear transformations). For example a local Lax-Friedrichs flux is given by

$$\begin{aligned}
\boldsymbol{f}^\star_{\xi_l,a} &= \sum_{m=1}^{d} \frac{1}{2}\left[\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a} \boldsymbol{t}^T_{\xi_l,b}\boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_{\nu_{\xi_l,a}}\right) + \boldsymbol{t}^T_{\xi_l,a}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\kappa}\boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_\kappa\right)\right] \\
&\quad - \frac{\sigma}{2}\left|\sum_{m=1}^{d}\max(\lambda_m)\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\right|\left(\boldsymbol{t}^T_{\xi_l,a}\boldsymbol{u}_\kappa - \boldsymbol{t}^T_{\xi_l,b}\boldsymbol{u}_{\nu_{\xi_l,a}}\right) \\
\boldsymbol{f}^\star_{\xi_l,b} &= \sum_{m=1}^{d} \frac{1}{2}\left[\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b} \boldsymbol{t}^T_{\xi_l,a}\boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_{\nu_{\xi_l,b}}\right) + \boldsymbol{t}^T_{\xi_l,b}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\kappa}\boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_\kappa\right)\right] \\
&\quad - \frac{\sigma}{2}\left|\sum_{m=1}^{d}\max(\lambda_m)\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\right|\left(\boldsymbol{t}^T_{\xi_l,a}\boldsymbol{u}_{\nu_{\xi_l,b}} - \boldsymbol{t}^T_{\xi_l,b}\boldsymbol{u}_\kappa\right)
\end{aligned}$$

where $\sigma = 1$ yields upwinding, and $\sigma = 0$ returns the nondissipative SAT. $\max(\lambda_m)$ is the maximum eigenvalue of the flux jacobian $\dfrac{\partial \boldsymbol{\mathcal{F}}}{\partial \boldsymbol{u}}$ computed at an intermediate state between the facets. It is useful to recognize that the SAT term can also be written in the following form,

$$\begin{aligned}
&\frac{1}{2}H^{-1}\sum_{l,m=1}^{d}\Bigg[E_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\kappa}\boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_\kappa) + \sigma\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\boldsymbol{f}^{\star,\text{diss}}_{\xi_l,b} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\boldsymbol{f}^{\star,\text{diss}}_{\xi_l,a}\right) \\
&\quad - \left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a}\boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_{\nu_{\xi_l,b}}) - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,b}\boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_{\nu_{\xi_l,a}})\right)\Bigg]
\end{aligned}$$

## 5.2 Numerical Surface Fluxes

Local Lax Friedrichs, Entropy Lax Friedrichs (SBP Notes / Crean), my attempt

## 5.3 Hadamard (Flux Differencing) Form

We can also construct schemes using a flux differencing Hadamard formulation.

$$\begin{aligned}
&\frac{d\boldsymbol{u}_\kappa}{dt} + 2\sum_{m=1}^{d}\left[D_{x_m}\circ F_{x_m}\right]\mathbf{1} = H^{-1}\sum_{l=1}^{d}\Bigg[E_{\xi_l}\circ\sum_{m=1}^{d}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\kappa}F_{x_m} \\
&\quad + \sum_{m=1}^{d}\boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,b}\circ F^{\xi_l,a}_{x_m} - \sum_{m=1}^{d}\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a}\circ F^{\xi_l,b}_{x_m}\Bigg]\mathbf{1} + \mathbf{diss}
\end{aligned}$$

where we use a symmetric and consistent volume flux $[F_{x_m}]_{ij} = \mathcal{F}^\star_{x_m}(u_i, u_j)$ and surface fluxes $[F^{\xi_l,a}_{x_m}]_{ij} = \mathcal{F}^\star_{x_m}(u^\kappa_i, u_j^{\nu_{\xi_l},a})$, $[F^{\xi_l,b}_{x_m}]_{ij} = \mathcal{F}^\star_{x_m}(u^\kappa_i, u_j^{\nu_{\xi_l},b})$. The metric terms can be approximated in different ways, but to ensure free-stream preservation, use optimized metrics for the volume and first SAT term, then pre-specified (ex. analytical) surface metrics for the last 2 SAT terms.

## 5.4 Two Point Conservative Fluxes

## 5.5 Dissipative Terms

Entropy-stable surface dissipation can be added using

$$\mathbf{diss} = -\frac{1}{2}H^{-1}\sum_{l=1}^{d}\Bigg[\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\Lambda(\boldsymbol{u}_\kappa,\boldsymbol{u}_{\nu_{\xi_l,b}})\left(\boldsymbol{t}^T_{\xi_l,b}\boldsymbol{w}_\kappa-\boldsymbol{t}^T_{\xi_l,a}\boldsymbol{w}_{\nu_{\xi_l,b}}\right) + \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\Lambda(\boldsymbol{u}_\kappa,\boldsymbol{u}_{\nu_{\xi_l,a}})\left(\boldsymbol{t}^T_{\xi_l,a}\boldsymbol{w}_\kappa-\boldsymbol{t}^T_{\xi_l,b}\boldsymbol{w}_{\nu_{\xi_l,a}}\right)\Bigg]$$

where $\Lambda$ can be any symmetric positive semi-definite matrix. For example a Lax-Friedrichs flux has

$$\Lambda_{\mathrm{LF}}(\boldsymbol{u}_\kappa,\boldsymbol{u}_\nu)=\left|\sum_{m=1}^{d}\max\left(\lambda_m\right)\frac{\partial\boldsymbol{\mathcal{U}}}{\partial\boldsymbol{\mathcal{W}}}\mathrm{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)_{\xi_l,\nu}\right|$$

where the matrix $\dfrac{\partial\boldsymbol{\mathcal{U}}}{\partial\boldsymbol{\mathcal{W}}}$ is block diagonal, each block corresponding to a $n_{eq}\times n_{eq}$ matrix at each node. The surface metric terms are of course a fully diagonal matrix. For the 2D Euler equations, one can choose to approximate this eigenvalue using [2]

$$\max^*\left(\lambda_m\right)\approx\left[\frac{1}{2}\left(\left(u_\kappa^2+v_\kappa^2\right)^2+\left(u_\nu^2+v_\nu^2\right)^2+a_\kappa^4+a_\nu^4\right)\right]^{1/4}\geq\max\left(\lambda_m\right)$$

This flux however tends to be overly-dissipative. Alternatively, a Roe flux is

$$\Lambda_{\mathrm{Roe}}(\boldsymbol{u}_\kappa,\boldsymbol{u}_\nu)=\sum_{m=1}^{d}Y_m\left|\Lambda_m\mathrm{sign}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)_{\xi_l,\nu}\right|Y_m^{-1}\frac{\partial\boldsymbol{\mathcal{U}}}{\partial\boldsymbol{\mathcal{W}}}\left|\mathrm{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)_{\xi_l,\nu}\right|$$

where For the Row flux we formulate the jacobian in terms of the entropy variables and therefore must multiply by $\dfrac{\partial U}{\partial W}$. This is made remarkably simpler by a scaling identity discovered by [6] and generalized by [1]

$$\frac{\partial\boldsymbol{\mathcal{U}}}{\partial\boldsymbol{\mathcal{W}}}=YY^T\quad\Rightarrow\quad\Lambda_{\mathrm{Roe}}(\boldsymbol{u}_\kappa,\boldsymbol{u}_\nu)=\sum_{m=1}^{d}Y_m\left|\Lambda_m\mathrm{sign}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)_{\xi_l,\nu}\right|Y^T\left|\mathrm{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)_{\xi_l,\nu}\right|$$

When a gas is callorically perfect (specific heat capacity $\gamma$ is constant throughout the fluid), the explicit flux jacobinas for the euler equations are. Otherwise one has to be more careful and use flux jabocian matrices described in [5].

Only surface? Can it be done for Volume (entropy dissipative surface flux)?

## 5.6 GCL Constraints and Metric Term Optimization

For details see [3]. We consider a straightforward discretization ($\alpha=1$, $\beta=0$ in the general form of the next section). The more general formulations have slightly different GCL constraints, though they are equally as easy to obtain. Also note that dissipative terms do not affect the GCL conditions because they vanish for constant flows. The $d$ free stream preservation constraints, one for each physical dimension $m$, can be obtained by plugging in a constant

value solution and flux into the discretization. We find

$$\left[\text{diag}\left(\mathcal{J}_\kappa\right)\right]^{-1}\sum_{l=1}^{d} D_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \mathbf{1} = H^{-1}\sum_{l=1}^{d}\Bigg[E_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \mathbf{1} - \left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,b}\right)\mathbf{1}\Bigg]$$

A more convenient form is obtained by multiplying by $-H$, then using the SBP property $Q_{\xi_l} = -Q^T_{\xi_l} + E_{\xi_l}$ to cancel the first term on the right hand side, leaving

$$\sum_{l=1}^{d} Q^T_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \mathbf{1} = \sum_{l=1}^{d}\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,b}\right)\mathbf{1}$$

For now take $d = 3$. We can write this system in a global matrix form as

$$\begin{aligned} M\boldsymbol{a}_{\kappa,m} = \boldsymbol{c}_{\kappa,m} \quad , \quad M &= \begin{bmatrix} Q^T_{\xi_1} & Q^T_{\xi_2} & Q^T_{\xi_3}\end{bmatrix} \\ \boldsymbol{a}_{\kappa,m} &= \left[\left(\mathcal{J}\frac{\partial \xi_1}{\partial x_m}\right)_\kappa \quad \left(\mathcal{J}\frac{\partial \xi_2}{\partial x_m}\right)_\kappa \quad \left(\mathcal{J}\frac{\partial \xi_3}{\partial x_m}\right)_\kappa\right]^T \\ \boldsymbol{c}_{\kappa,m} &= \sum_{l=1}^{d}\left[\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\right] \end{aligned}$$

Note that $\boldsymbol{c}_{\kappa,m}$ includes only pre-specified metric terms (ex. analytical). We then find an optimal $\boldsymbol{a}_{\kappa,m}$ using a least squares error to the exact analytical metrics $\boldsymbol{a}^{\text{ex}}_{\kappa,m}$. The solution is given by

$$\boldsymbol{a}_{\kappa,m} = \boldsymbol{a}^{\text{ex}}_{\kappa,m} - M^{\dagger}\left(M\boldsymbol{a}^{\text{ex}}_{\kappa,m} - \boldsymbol{c}_{\kappa,m}\right)$$

where $M^{\dagger}$ is the Moore-Penrose pseudoinverse of $M$. An additional necessary condition is that $\mathbf{1}^T\boldsymbol{c}_{\kappa,m} = 0$, the discrete equivalent to ensuring the volume integral of the metric terms vanish, or equivalently using Gauss' Theorem, that the metric terms vanish along line integrals of the boundary. See the following subsection for details on ensuring that this holds.

### 5.6.1 Additional Considerations for Computing Metrics

In 2D with explicit computations, we approximate

$$\begin{aligned} \left(\mathcal{J}\frac{\partial \xi}{\partial x}\right)_\kappa = D_\nu \boldsymbol{y}_\kappa \quad , \quad \left(\mathcal{J}\frac{\partial \xi}{\partial y}\right)_\kappa = -D_\nu \boldsymbol{x}_\kappa \\ \left(\mathcal{J}\frac{\partial \nu}{\partial x}\right)_\kappa = -D_\xi \boldsymbol{y}_\kappa \quad , \quad \left(\mathcal{J}\frac{\partial \nu}{\partial y}\right)_\kappa = D_\xi \boldsymbol{x}_\kappa \end{aligned}$$

and therefore

$$\begin{aligned} \sum_{l=1}^{d} D_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x}\right)_\kappa \mathbf{1} = D_\xi D_\nu \boldsymbol{y}_\kappa - D_\nu D_\xi \boldsymbol{y}_\kappa = 0 \\ \sum_{l=1}^{d} D_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial y}\right)_\kappa \mathbf{1} = D_\xi D_\nu \boldsymbol{x}_\kappa - D_\nu D_\xi \boldsymbol{x}_\kappa = 0 \end{aligned}$$

since derivative matrices commute for tensor product formulations. The same holds for metrics computed using the Thomas Lombard or Vinokur and Yee method in 3D. These relations hold

in the code to accuracy $10^{-14}$. This error can be increased a few orders of magnitude when multiplied by the jacobian inverse. In these cases, the GCL constraints reduce to the SAT contribution.

$$
\begin{aligned}
0 = \sum_{l=1}^{d} \Bigg[ & \left(\boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \boldsymbol{t}^T_{\xi_l,b} - \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \boldsymbol{t}^T_{\xi_l,a}\right) \operatorname{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\kappa} \mathbf{1} \\
& - \left(\boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \operatorname{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b} \boldsymbol{t}^T_{\xi_l,a} - \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \operatorname{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a} \boldsymbol{t}^T_{\xi_l,b}\right) \mathbf{1} \Bigg] \\
= \sum_{l=1}^{d} \Bigg[ & \boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \left(\boldsymbol{t}^T_{\xi_l,b} \operatorname{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\kappa} - \operatorname{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b} \boldsymbol{t}^T_{\xi_l,a}\right) \\
& - \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \left(\boldsymbol{t}^T_{\xi_l,a} \operatorname{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\kappa} - \operatorname{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a} \boldsymbol{t}^T_{\xi_l,b}\right) \Bigg] \mathbf{1}
\end{aligned}
$$

This is satisfied only when the extrapolated volume metrics are equal to the mortar element interface metrics. Even if we use extrapolation to calculate the interface metrics however, unless the metrics are polynomials of degree $\leq p$ (meaning the mapping is degree $< p$ for 2D and $< \left\lfloor \frac{p}{2} \right\rfloor$ for 3D[1]), then since we use the average of the extrapolation on both sides, the resulting interface metric is not guaranteed to match the extrapolation of the volume metrics on either side. Therefore, the SAT contribution to free stream is in general not zero for non-polynomial or polynomial order $> p$ and $> \left\lfloor \frac{p}{2} \right\rfloor$ mappings. Hence we use the optimization procedure.

### 5.6.2 Ensuring Surface Integral Consistency

Recall the condition $\mathbf{1}^T \boldsymbol{c}_{\kappa,m} = 0$, here rewritten as

$$
\mathbf{1}^T \sum_{l=1}^{d} \left[ H^{\perp}_{\xi_l} \left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b} - H^{\perp}_{\xi_l} \left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a} \right] = 0
$$

This is clearly a discrete equivalent to the surface integral condition

$$
\oint_{\Gamma_\kappa} \left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,n} \cdot \hat{n} \, d\Gamma_\kappa = 0 = \int_{\Omega_\kappa} \sum_{l=1}^{d} \frac{\partial}{\partial \xi_l} \left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,n} d\Omega_\kappa
$$

where we have used the divergence theorem to again relate the surface metric condition to the volume GCL constraint, though this time for the interface metrics. This condition is necessary for the optimization procedure defined above so that the $\boldsymbol{c}_{\kappa,m}$ vector does not fall in the null space of $M$. Since $M$ is composed of the $Q_\xi$ matrices, it's null space is spanned by a single vector, the constant vector. Roughly speaking, we therefore want $\boldsymbol{c}_{\kappa,m}$ to be orthogonal to the constant vector (see [4] for details). Since we know this condition holds in the continuous setting, if the facet integration and metric terms are exact, this will also hold discretely. We have a few options.

1. Ensure the calculation and projection of the volume metrics is exact, so the mesh must be a polynomial of degree $\leq p$ , $\left\lfloor \frac{p}{2} \right\rfloor$ for 2D, 3D.

[1] When I speak of a mapping being degree $p$, it is a polynomial of maximum degree $p$ *in each direction*. The total degree may be higher.

2. Use the exact metrics on the interfaces, but then ensure that these can be integrated exactly, i.e. be polynomials of order $\leq 2p-1$ and $\leq 2p+1$ for LGL and LG, respectively. Therefore the mesh must be a polynomial of order $\leq 2p-1, p-1$ and $\leq 2p+1, p$.

3. Fit the mesh mapping to a polynomial of order $\leq 2p-1, p-1$ (2D, 3D) and $\leq 2p+1, p$ (2D, 3D) for LGL and LG, respectively.

4. Perform a preliminary optimization to force the interface metrics to satisfy this condition while being as close to the exact surface metrics as possible.

The fourth option, unique to ESSBP, is as follows. Find an optimal $\boldsymbol{c}_{\kappa,m}$ by minimizing the least squares error to the exact metrics $\boldsymbol{c}^{\text{ex}}_{\kappa,m}$ subject to the constraint $\mathbf{1}^T\boldsymbol{c}_{\kappa,m}=0$.

For each physical direction $m$, define a global vector $\boldsymbol{b}^{i_x,i_y,i_z}_{l,m} = \left(\mathcal{J}\dfrac{\partial \xi_l}{\partial x_m}\right)^{i_x,i_y,i_z}_{\xi_l}$, where $l$ determines the direction of the interface, and the indices $i_x$, $i_y$, $i_z$ denote the interface (similar to labelling elements, though $i_s$ range from 1 to $N_{e_s}$ if $s\neq l$, and range to $N_{e_s}+1$ instead if $s=l$). The element-wise constraints $\mathbf{1}^T\boldsymbol{c}_{\kappa,m}=0$ can now be rewritten in a global manner as $N_{e_x}\times N_{e_y}\times N_{e_z}$ individual constraints

$$\mathbf{1}^T\left[H_x^\perp\left(\boldsymbol{b}^{i_x+1,i_y,i_z}_{x,m}-\boldsymbol{b}^{i_x,i_y,i_z}_{x,m}\right)+H_y^\perp\left(\boldsymbol{b}^{i_x,i_y+1,i_z}_{y,m}-\boldsymbol{b}^{i_x,i_y,i_z}_{y,m}\right)+H_z^\perp\left(\boldsymbol{b}^{i_x,i_y,i_z+1}_{z,m}-\boldsymbol{b}^{i_x,i_y,i_z}_{z,m}\right)\right]=0$$

We want to find $\boldsymbol{b}^{i_x,i_y,i_z}_{l,m}$ as close as possible to the target values $\tilde{\boldsymbol{b}}^{i_x,i_y,i_z}_{l,m}$. For each $m$, define the Lagrangian

$$L_m=\sum_{j=1}^{N_n}\sum_{i_x,i_y,i_z=1}^{N_e}\sum_{l=3}^{d}\frac{1}{2}\left(\tilde{b}^{i_x,i_y,i_z}_{l,m,j}-b^{i_x,i_y,i_z}_{l,m,j}\right)^2-\sum_{i_x,i_y,i_z=1}^{N_e}\lambda_{i_x,i_y,i_z}\sum_{j=1}^{N_n}\Big[H^\perp_{x,j}\left(b^{i_x+1,i_y,i_z}_{x,m,j}-b^{i_x,i_y,i_z}_{x,m,j}\right)+H^\perp_{y,j}\left(b^{i_x,i_y+1,i_z}_{y,m,j}-b^{i_x,i_y,i_z}_{y,m,j}\right)+H^\perp_{z,j}\left(b^{i_x,i_y,i_z+1}_{z,m,j}-b^{i_x,i_y,i_z}_{z,m,j}\right)\Big]$$

We now follow the KKT conditions by looking for a saddle point of the Lagrangian,

$$0=\frac{\partial L_m}{\partial b^{i_x,i_y,i_z}_{l,m,j}}=b^{i_x,i_y,i_z}_{l,m,j}-\tilde{b}^{i_x,i_y,i_z}_{l,m,j}-\left(\lambda_{i_l-1,i_{s_1},i_{s_2}}-\lambda_{i_l,i_{s_1},i_{s_2}}\right)H^\perp_{l,j}$$

for the interior facets. The boundary facets $i_x=1$ and $i_x=N_e+1$ have

$$\begin{aligned}
0&=\frac{\partial L_m}{\partial b^{1,i_y,i_z}_{x,m,j}}=b^{1,i_y,i_z}_{x,m,j}-\tilde{b}^{1,i_y,i_z}_{x,m,j}+\lambda_{1,i_y,i_z}H^\perp_{x,j}\\
0&=\frac{\partial L_m}{\partial b^{1,i_y,i_z}_{y,m,j}}=b^{1,i_y,i_z}_{y,m,j}-\tilde{b}^{1,i_y,i_z}_{y,m,j}-\left(\lambda_{1,i_y+1,i_z}-\lambda_{1,i_y,i_z}\right)H^\perp_{y,j}\\
0&=\frac{\partial L_m}{\partial b^{N_e+1,i_y,i_z}_{x,m,j}}=b^{N_e+1,i_y,i_z}_{x,m,j}-\tilde{b}^{N_e+1,i_y,i_z}_{x,m,j}-\lambda_{N_e,i_y,i_z}H^\perp_{x,j}
\end{aligned}$$

ans similarly for the $y$ and $z$ facets. From this I conclude

$$\begin{aligned}
b^{i_x,i_y,i_z}_{l,m,j}&=\tilde{b}^{i_x,i_y,i_z}_{l,m,j}+\left(\lambda_{i_l-1,i_{s_1},i_{s_2}}-\lambda_{i_l,i_{s_1},i_{s_2}}\right)H^\perp_{l,j}\\
b^{1,i_y,i_z}_{l,m,j}&=\tilde{b}^{1,i_y,i_z}_{l,m,j}-\lambda_{1,i_y,i_z}H^\perp_{x,j}\\
b^{N_e+1,i_y,i_z}_{l,m,j}&=\tilde{b}^{N_e+1,i_y,i_z}_{l,m,j}+\lambda_{N_e,i_y,i_z}H^\perp_{x,j}
\end{aligned}$$

where

$$0 = \frac{\partial L_m}{\partial \lambda_i} = \sum_{j=1}^{N_n} \left[ H^{\perp}_{x,j} \left( b^{i_x+1,i_y,i_z}_{x,m,j} - b^{i_x,i_y,i_z}_{x,m,j} \right) + H^{\perp}_{y,j} \left( b^{i_x,i_y+1,i_z}_{y,m,j} - b^{i_x,i_y,i_z}_{y,m,j} \right) + H^{\perp}_{z,j} \left( b^{i_x,i_y,i_z+1}_{z,m,j} - b^{i_x,i_y,i_z}_{z,m,j} \right) \right]$$

For simplicity in the following sections, we artificially set

$$\lambda_{0,i_y,i_z} = \lambda_{i_x,0,i_z} = \lambda_{i_x,i_y,0} = \lambda_{N_e+1,i_y,i_z} = \lambda_{i_x,N_e+1,i_z} = \lambda_{i_x,i_y,N_e+1} = 0$$

This recovers the boundary terms from the above general form without any additional work. Plugging these in, I find

$$\begin{aligned}
0 &= \sum_{j=1}^{N_n} \left[ H^{\perp}_{x,j} \left( b^{i_x+1,i_y,i_z}_{x,m,j} - b^{i_x,i_y,i_z}_{x,m,j} \right) + H^{\perp}_{y,j} \left( b^{i_x,i_y+1,i_z}_{y,m,j} - b^{i_x,i_y,i_z}_{y,m,j} \right) + H^{\perp}_{z,j} \left( b^{i_x,i_y,i_z+1}_{z,m,j} - b^{i_x,i_y,i_z}_{z,m,j} \right) \right] \\
&= \sum_{j=1}^{N_n} \Big[ H^{\perp}_{x,j} \left( \tilde{b}^{i_x+1,i_y,i_z}_{x,m,j} - \tilde{b}^{i_x,i_y,i_z}_{x,m,j} \right) + H^{\perp}_{y,j} \left( \tilde{b}^{i_x,i_y+1,i_z}_{y,m,j} - \tilde{b}^{i_x,i_y,i_z}_{y,m,j} \right) + H^{\perp}_{z,j} \left( \tilde{b}^{i_x,i_y,i_z+1}_{z,m,j} - \tilde{b}^{i_x,i_y,i_z}_{z,m,j} \right) \\
&\quad + \left( H^{\perp}_{x,j} \right)^2 \left( -\lambda_{i_x-1,i_y,i_z} + 2\lambda_{i_x,i_y,i_z} - \lambda_{i_x+1,i_y,i_z} \right) + \left( H^{\perp}_{y,j} \right)^2 \left( -\lambda_{i_x,i_y-1,i_z} + 2\lambda_{i_x,i_y,i_z} - \lambda_{i_x,i_y+1,i_z} \right) \\
&\quad + \left( H^{\perp}_{z,j} \right)^2 \left( -\lambda_{i_x,i_y,i_z-1} + 2\lambda_{i_x,i_y,i_z} - \lambda_{i_x,i_y,i_z+1} \right) \Big]
\end{aligned}$$

Note that in the case that the $H^{\perp}$ matrices are identical in each element (as in ESSBP), we can simplify the above to

$$\begin{aligned}
0 = \sum_{j=1}^{N_n} H^{\perp}_j \Big[ & \left( \tilde{b}^{i_x+1,i_y,i_z}_{x,m,j} - \tilde{b}^{i_x,i_y,i_z}_{x,m,j} \right) + \left( \tilde{b}^{i_x,i_y+1,i_z}_{y,m,j} - \tilde{b}^{i_x,i_y,i_z}_{y,m,j} \right) + \left( \tilde{b}^{i_x,i_y,i_z+1}_{z,m,j} - \tilde{b}^{i_x,i_y,i_z}_{z,m,j} \right) \\
& + \left( H^{\perp}_j \right)^2 \left( -\lambda_{i_x-1,i_y,i_z} - \lambda_{i_x,i_y-1,i_z} - \lambda_{i_x,i_y,i_z-1} + 6\lambda_{i_x,i_y,i_z} - \lambda_{i_x+1,i_y,i_z} - \lambda_{i_x,i_y+1,i_z} - \lambda_{i_x,i_y,i_z+1} \right) \Big]
\end{aligned}$$

In order to solve for $\boldsymbol{\lambda}$ we can rewrite it in the following way,

$$A^{i_x,i_y,i_z}_{j_x,j_y,j_z} \lambda_{j_x,j_y,j_z} = \sum_{j=1}^{N_n} \sum_{l=1}^{d} H^{\perp}_{l,j} B^{i_x,i_y,i_z}_{l,j_x,j_y,j_z} \tilde{b}^{j_x,j_y,j_z}_{l,m,j}$$

where

$$B^{i_x,i_y,i_z}_{l,j_x,j_y,j_z} = \delta^{i_l}_{j_l} - \delta^{i_l+1}_{j_l} \quad , \quad H^{\perp} B \ : \ \mathbb{R}^{d \times N_n} \otimes \mathbb{R}^{(N_{ex}+1)N_{ey}N_{ez} + N_{ex}(N_{ey}+1)N_{ez} + N_{ex}N_{ey}(N_{ez}+1)} \rightarrow \mathbb{R}^{N_{ex} \times N_{ey} \times N_{ez}}$$

$$A^{i_x,i_y,i_z}_{j_x,j_y,j_z} = \sum_{l=1}^{d} \left( \sum_{j=1}^{N_n} \left( H^{\perp}_{l,j} \right)^2 \right) \left( -\delta^{i_l-1}_{j_l} + 2\delta^{i}_{j} - \delta^{i_l+1}_{j_l} \right)$$

where because of our introduced notation, whenever the $i_l$ or $j_l$ components become 0 and $N_e + 1$, the corresponding $\delta$ are zero. This can be flattened to a simple matrix-vector linear system, from which the $\boldsymbol{\lambda}$ can be obtained, in turn giving the optimum surface metrics $\boldsymbol{b}$.

# 6 Stability and Equivalence Proofs

I begin by showing the equivalence of the divergence formulation and the Hadamard formulation. I then prove energy stability for the linear convection equation and entropy stability for a general Hadamard formulation, assuming entropy-stable volume fluxes.

## 6.1 Equivalence of Divergence and Hadamard Forms

We will begin with a very general treatment of metrics in a non-dissipative Hadamard form,

$$\begin{aligned}
\frac{d\boldsymbol{u}_\kappa}{dt} + 2\sum_{m=1}^{d}\left[D_{x_m}\circ F_{x_m}\right]\mathbf{1} = H^{-1}\sum_{l,m=1}^{d}\Bigg[ & E_{\xi_l}\left[\alpha\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)_\kappa + (1-\alpha)\beta\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_\kappa\right]\circ F_{x_m} \\
& + (1-\alpha)(1-\beta)\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,b} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,a}\right)\circ F_{x_m} \\
& - \gamma\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a}\circ F^{\xi_l,b}_{x_m} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,b}\circ F^{\xi_l,a}_{x_m}\right) \\
& - (1-\gamma)\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\boldsymbol{t}^T_{\xi_l,a}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\nu_{\xi_l,b}}\circ F^{\xi_l,b}_{x_m} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\boldsymbol{t}^T_{\xi_l,b}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\nu_{\xi_l,a}}\circ F^{\xi_l,a}_{x_m}\right)\Bigg]\mathbf{1}
\end{aligned}$$

where normally $\alpha = 1$, $\beta = 1$, and $\gamma = 1$. In [7], they use $\alpha = 2$ (the $\beta$ and $\gamma$ parameters are irrelevant if boundary nodes are included or if the volume metrics are extrapolated exactly to the boundaries). We will then show that using a central volume flux composed of arithmetic means $\mathcal{F}^{\star}_{x_m}(u_i,u_j) = \frac{1}{2}(f_i+f_j)$, we recover a nondissipative divergence formulation.

$$\begin{aligned}
\frac{d\boldsymbol{u}_\kappa}{dt} + \sum_{m=1}^{d}D_{x_m}\boldsymbol{f}^\kappa = \frac{1}{2}H^{-1}\sum_{l,m=1}^{d}\Bigg[ & E_{\xi_l}\left[\alpha\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)_\kappa + (1-\alpha)\beta\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_\kappa\right]\boldsymbol{f}_\kappa \\
& + (1-\alpha)(1-\beta)\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,b} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,a}\right)\boldsymbol{f}_\kappa \\
& - \gamma\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a}\boldsymbol{f}_{\nu_{\xi_l,b}} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,b}\boldsymbol{f}_{\nu_{\xi_l,a}}\right) \\
& - (1-\gamma)\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\boldsymbol{t}^T_{\xi_l,a}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\nu_{\xi_l,b}}\boldsymbol{f}_{\nu_{\xi_l,b}} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\boldsymbol{t}^T_{\xi_l,b}\text{diag}\left(\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right)^{\text{ex}}_{\nu_{\xi_l,a}}\boldsymbol{f}_{\nu_{\xi_l,a}}\right)\Bigg]
\end{aligned}$$

This choice of arithmetic flux makes our lives easier, but note that the same steps can be followed using any numerical flux constructed through arithmetic means to recover other split-form discretizations. We begin with the volume term.

$$\begin{aligned}
2\sum_{m=1}^{d}\left[D_{x_m}\circ F_{x_m}\right]\mathbf{1} &= 2\sum_{m=1}^{d}\sum_{j=1}^{N}[D_{x_m}]_{ij}[F_{x_m}]_{ij} = \sum_{m=1}^{d}\sum_{j=1}^{N}[D_{x_m}]_{ij}(f^\kappa_i + f^\kappa_j) \\
&= \sum_{m=1}^{d}\left(f^\kappa_i\sum_{j=1}^{N}[D_{x_m}]_{ij} + \sum_{j=1}^{N}[D_{x_m}]_{ij}f^\kappa_j\right) \\
&= \text{FS}^{\text{vol}} + \sum_{m=1}^{d}D_{x_m}\boldsymbol{f}^\kappa \quad , \quad \text{FS}^{\text{vol}} = \text{diag}(\boldsymbol{f}^\kappa)\sum_{m=1}^{d}D_{x_m}\mathbf{1}
\end{aligned}$$

where we recognize the volume term $\mathrm{FS}^{\mathrm{vol}}$ from the free-stream preservation (or GCL) condition in the $m^{\mathrm{th}}$ direction,

$$\begin{aligned}
\sum_{j=1}^{N}[D_{x_m}]_{ij} &= \frac{1}{2}\sum_{j=1}^{N}\left[\frac{1}{\mathcal{J}_\kappa}\right]_i\sum_{l=1}^{d}\left[[D_{x_l}]_{ij}\left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_j + \left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_i[D_{\xi_l}]_{ij}\right] \\
&= \frac{1}{2}\left[\frac{1}{\mathcal{J}_\kappa}\right]_i\sum_{l=1}^{d}\left[\sum_{j=1}^{N}[D_{\xi_l}]_{ij}\left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_j + \left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_i\underbrace{\sum_{j=1}^{N}[D_{\xi_l}]_{ij}}_{0}\right] \\
&= \frac{1}{2}\left[\mathrm{diag}\left(\mathcal{J}_\kappa\right)\right]^{-1}\sum_{l=1}^{d}D_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \mathbf{1}
\end{aligned}$$

We will find that this free-stream term will cancel with contributions from the SAT, assuming the metrics are constructed in a way that respects free-stream conservation. We now consider the first term from the SAT.

$$\begin{aligned}
\left[E_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \circ F_{x_m}\right]\mathbf{1} &= \sum_{j=1}^{N}[E_{\xi_l}]_{ij}\left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_j[F_{x_m}]_{ij} \\
&= \frac{1}{2}\sum_{j=1}^{N}[E_{\xi_l}]_{ij}\left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_j(f_i + f_j) \\
&= \frac{1}{2}E_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \boldsymbol{f}_\kappa + \frac{1}{2}\mathrm{diag}(\boldsymbol{f}_\kappa)E_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \mathbf{1}
\end{aligned}$$

The other volume SAT terms follow similarly. Let's consider now the first coupling term,

$$\begin{aligned}
\left[\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a} \circ F^{\xi_l,b}_{x_m}\right]\mathbf{1} &= \frac{1}{2}\sum_{j=1}^{N}\left[\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a}\right]_{ij}\left(f_{\kappa,i} + f_{\nu_{\xi_l,b},j}\right) \\
&= \frac{1}{2}\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a}\boldsymbol{f}_{\nu_{\xi_l,b}} + \frac{1}{2}\mathrm{diag}(\boldsymbol{f}_\kappa)\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a}\mathbf{1}
\end{aligned}$$

The remaining terms are similar. Grouping the results together and denoting

$$\begin{aligned}
\mathrm{FS}^{\mathrm{surf}} = \frac{1}{2}H^{-1}\mathrm{diag}\left(\boldsymbol{f}_\kappa\right)\sum_{l,m=1}^{d}\Bigg[&E_{\xi_l}\left[\alpha\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + (1-\alpha)\beta\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_\kappa\right] \\
&+ (1-\alpha)(1-\beta)\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,b} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,a}\right) \\
&- \gamma\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,b}\right) \\
&- (1-\gamma)\left(\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\boldsymbol{t}^T_{\xi_l,a}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\nu_{\xi_l,b}} - \boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\boldsymbol{t}^T_{\xi_l,b}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)^{\mathrm{ex}}_{\nu_{\xi_l,a}}\right)\Bigg]\mathbf{1}
\end{aligned}$$

then assuming diagonal $H$, we recover the desired divergence form as long as $\mathrm{FS}^{\mathrm{vol}} = \mathrm{FS}^{\mathrm{surf}}$. This condition comes from the free stream preservation (GCL) constraints, one for each physical

dimension $m$.

$$
\begin{aligned}
\left[\mathrm{diag}\left(\mathcal{J}_\kappa\right)\right]^{-1}\sum_{l=1}^{d} D_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \mathbf{1} = \frac{1}{2}H^{-1}\sum_{l=1}^{d}\Bigg[ & E_{\xi_l}\Bigg[\alpha\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + (1-\alpha)\beta\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa^{\mathrm{ex}}\Bigg] \\
& + (1-\alpha)(1-\beta)\left(\boldsymbol{t}_{\xi_l,b}H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}^{\mathrm{ex}}\boldsymbol{t}_{\xi_l,b}^{T} - \boldsymbol{t}_{\xi_l,a}H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}^{\mathrm{ex}}\boldsymbol{t}_{\xi_l,a}^{T}\right) \\
& - \gamma\left(\boldsymbol{t}_{\xi_l,b}H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}^{\mathrm{ex}}\boldsymbol{t}_{\xi_l,a}^{T} - \boldsymbol{t}_{\xi_l,a}H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}^{\mathrm{ex}}\boldsymbol{t}_{\xi_l,b}^{T}\right) \\
& - (1-\gamma)\left(\boldsymbol{t}_{\xi_l,b}H_{\xi_l}^{\perp}\boldsymbol{t}_{\xi_l,a}^{T}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\nu_{\xi_l,b}}^{\mathrm{ex}} - \boldsymbol{t}_{\xi_l,a}H_{\xi_l}^{\perp}\boldsymbol{t}_{\xi_l,b}^{T}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\nu_{\xi_l,a}}^{\mathrm{ex}}\right)\Bigg]\mathbf{1}
\end{aligned}
$$

Once again recognize that choosing $\alpha = \beta = \gamma = 1$ recovers the standard free stream preservation constraint. One must be careful therefore to modify the optimization procedure for the metric terms appropriately.

## 6.2 Weak Form

Here we prove that the Hadamard formulation can be recast in an equivalent weak form - convenient for DG formulations. We begin with a basic Hadamard formulation,

$$
\begin{aligned}
\frac{d\boldsymbol{u}_\kappa}{dt} + 2\sum_{m=1}^{d}\left[D_{x_m}\circ F_{x_m}\right]\mathbf{1} = H^{-1}\sum_{l=1}^{d}\Bigg[ & E_{\xi_l}\circ\sum_{m=1}^{d}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa F_{x_m} \\
& + \sum_{m=1}^{d}\boldsymbol{t}_{\xi_l,a}H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\boldsymbol{t}_{\xi_l,b}^{T}\circ F_{x_m}^{s}(\boldsymbol{u}_\kappa, \boldsymbol{u}_{\nu_{\xi_l,a}}) \\
& - \sum_{m=1}^{d}\boldsymbol{t}_{\xi_l,b}H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\boldsymbol{t}_{\xi_l,a}^{T}\circ F_{x_m}^{s}(\boldsymbol{u}_\kappa, \boldsymbol{u}_{\nu_{\xi_l,b}})\Bigg]\mathbf{1}
\end{aligned}
$$

We take the inner product with a text function through contracting by $\boldsymbol{v}_\kappa^T H$. We obtain

$$
\begin{aligned}
\frac{d\boldsymbol{u}_\kappa}{dt} + 2\sum_{m=1}^{d}\left[D_{x_m}\circ F_{x_m}\right]\mathbf{1} = H^{-1}\sum_{l=1}^{d}\Bigg[ & E_{\xi_l}\circ\sum_{m=1}^{d}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa F_{x_m} \\
& + \sum_{m=1}^{d}\boldsymbol{t}_{\xi_l,a}H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\boldsymbol{t}_{\xi_l,b}^{T}\circ F_{x_m}^{s}(\boldsymbol{u}_\kappa, \boldsymbol{u}_{\nu_{\xi_l,a}}) \\
& - \sum_{m=1}^{d}\boldsymbol{t}_{\xi_l,b}H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\boldsymbol{t}_{\xi_l,a}^{T}\circ F_{x_m}^{s}(\boldsymbol{u}_\kappa, \boldsymbol{u}_{\nu_{\xi_l,b}})\Bigg]\mathbf{1}
\end{aligned}
$$

## 6.3 Energy Stability - Linear Convection

We use the divergence form as it is easier to work with for energy stability. We also make the assumption that $H$ is diagonal so that it can commute with the metric jacobian and metric terms, which will be crucial in the following steps. Take the following upwinding discretization of the linear convection equation,

$$
\frac{d\boldsymbol{u}_\kappa}{dt} + \sum_{m=1}^{d} D_{x_m}\left(a_m\boldsymbol{u}_\kappa\right) = H^{-1}\sum_{l=1}^{d}\left[\boldsymbol{t}_{\xi_l,b}H_{\xi_l}^{\perp}\left(\boldsymbol{f}_{\xi_l,b} - \boldsymbol{f}_{\xi_l,b}^{\star}\right) - \boldsymbol{t}_{\xi_l,a}H_{\xi_l}^{\perp}\left(\boldsymbol{f}_{\xi_l,a} - \boldsymbol{f}_{\xi_l,a}^{\star}\right)\right]
$$

where

$$D_{x_m} = \frac{1}{2}\left[\text{diag}\left(\mathcal{J}_\kappa\right)\right]^{-1}\sum_{l=1}^{d}\left[D_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa D_{\xi_l}\right] \quad , \quad H = \text{diag}\left(\mathcal{J}_\kappa\right) H_\xi$$

$$\boldsymbol{f}_{\xi_l,a} = \sum_{m=1}^{d} \boldsymbol{t}_{\xi_l,a}^T \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa (a_m \boldsymbol{u}_\kappa) \quad , \quad \boldsymbol{f}_{\xi_l,b} = \sum_{m=1}^{d} \boldsymbol{t}_{\xi_l,b}^T \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa (a_m \boldsymbol{u}_\kappa)$$

$$\begin{aligned}
\boldsymbol{f}^\star_{\xi_l,a} = &\sum_{m=1}^{d} a_m \frac{1}{2}\left[\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a} \boldsymbol{t}_{\xi_l,b}^T \boldsymbol{u}_{\nu_{\xi_l,a}} + \boldsymbol{t}_{\xi_l,a}^T \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \boldsymbol{u}_\kappa\right] \\
&- \frac{\sigma}{2}\left|\sum_{m=1}^{d} a_m \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\right|\left(\boldsymbol{t}_{\xi_l,a}^T \boldsymbol{u}_\kappa - \boldsymbol{t}_{\xi_l,b}^T \boldsymbol{u}_{\nu_{\xi_l,a}}\right) \\
\boldsymbol{f}^\star_{\xi_l,b} = &\sum_{m=1}^{d} a_m \frac{1}{2}\left[\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b} \boldsymbol{t}_{\xi_l,a}^T \boldsymbol{u}_{\nu_{\xi_l,b}} + \boldsymbol{t}_{\xi_l,b}^T \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \boldsymbol{u}_\kappa\right] \\
&- \frac{\sigma}{2}\left|\sum_{m=1}^{d} a_m \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\right|\left(\boldsymbol{t}_{\xi_l,a}^T \boldsymbol{u}_{\nu_{\xi_l,b}} - \boldsymbol{t}_{\xi_l,b}^T \boldsymbol{u}_\kappa\right)
\end{aligned}$$

We now prove stability using the energy method. Temporarily ignoring the SATs and considering only the volume terms, we find

$$\begin{aligned}
\frac{d}{dt}\left|\left|\boldsymbol{u}_\kappa\right|\right|_H^2 &= \boldsymbol{u}_\kappa^T H \frac{d\boldsymbol{u}_\kappa}{dt} + \frac{d\boldsymbol{u}_\kappa^T}{dt} H \boldsymbol{u}_\kappa \\
&= -\sum_{m=1}^{d} \boldsymbol{u}_\kappa^T H_\xi \frac{1}{2}\sum_{l=1}^{d}\left[D_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa D_{\xi_l}\right](a_m \boldsymbol{u}_\kappa) \\
&\quad - \sum_{m=1}^{d}\left(a_m \boldsymbol{u}_\kappa^T\right)\frac{1}{2}\sum_{l=1}^{d}\left[D_{\xi_l}^T\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa D_{\xi_l}^T\right] H_\xi \boldsymbol{u}_\kappa \\
&= -\frac{1}{2}\sum_{m=1}^{d} a_m \boldsymbol{u}_\kappa^T \sum_{l=1}^{d}\left[Q_{\xi_l}\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa Q_{\xi_l}\right]\boldsymbol{u}_\kappa \\
&\quad - \frac{1}{2}\sum_{m=1}^{d} a_m \boldsymbol{u}_\kappa^T \sum_{l=1}^{d}\left[Q_{\xi_l}^T\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa Q_{\xi_l}^T\right]\boldsymbol{u}_\kappa \\
&= -\frac{1}{2}\boldsymbol{u}_\kappa^T \sum_{l,m=1}^{d} a_m \left[\left(Q_{\xi_l} + Q_{\xi_l}^T\right)\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \left(Q_{\xi_l} + Q_{\xi_l}^T\right)\right]\boldsymbol{u}_\kappa
\end{aligned}$$

where we used the fact that $H$ is diagonal to commute with the metric terms and the metric jacobian. Now consider the terms that we get from the SAT. It is straightforward to see that the above exactly cancels the contribution from the $E_{\xi_l}$ term in the nondissipative SAT, leaving the coupling terms to cancel from either side of the interface. I will now show this explicitly. For simplicity we first consider the nondissipative case $\sigma = 0$, and then show the additional terms are dissipative.

$$\begin{aligned}
\frac{d}{dt}\left|\left|\text{SAT}\right|\right|_H^2 &= \boldsymbol{u}_\kappa^T H \text{SAT} + \text{SAT}^T H \boldsymbol{u}_\kappa \\
&= \boldsymbol{u}_\kappa^T \sum_{l=1}^{d}\left[\boldsymbol{t}_{\xi_l,b} H_{\xi_l}^\perp\left(\boldsymbol{f}_{\xi_l,b} - \boldsymbol{f}^\star_{\xi_l,b}\right) - \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^\perp\left(\boldsymbol{f}_{\xi_l,a} - \boldsymbol{f}^\star_{\xi_l,a}\right)\right] \\
&\quad + \sum_{l=1}^{d}\left[\boldsymbol{t}_{\xi_l,b} H_{\xi_l}^\perp\left(\boldsymbol{f}_{\xi_l,b} - \boldsymbol{f}^\star_{\xi_l,b}\right) - \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^\perp\left(\boldsymbol{f}_{\xi_l,a} - \boldsymbol{f}^\star_{\xi_l,a}\right)\right]^T \boldsymbol{u}_\kappa
\end{aligned}$$

Consider the term in square brackets (both lines are identical, simply the transpose of each other). Substituting in the nondissipative numerical flux, we get

$$
\begin{aligned}
&\sum_{l,m=1}^{d} a_m \Bigg[ \boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \left( \boldsymbol{t}^T_{\xi_l,b} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\kappa} \boldsymbol{u}_{\kappa} - \frac{1}{2} \left[ \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,b} \boldsymbol{t}^T_{\xi_l,a} \boldsymbol{u}_{\nu_{\xi_l,b}} + \boldsymbol{t}^T_{\xi_l,b} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\kappa} \boldsymbol{u}_{\kappa} \right] \right) \\
&\qquad - \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \left( \boldsymbol{t}^T_{\xi_l,a} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\kappa} \boldsymbol{u}_{\kappa} - \frac{1}{2} \left[ \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,a} \boldsymbol{t}^T_{\xi_l,b} \boldsymbol{u}_{\nu_{\xi_l,a}} + \boldsymbol{t}^T_{\xi_l,a} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\kappa} \boldsymbol{u}_{\kappa} \right] \right) \Bigg] \\
&= \sum_{l,m=1}^{d} a_m \frac{1}{2} \Bigg[ \left( \boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \boldsymbol{t}^T_{\xi_l,b} - \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \boldsymbol{t}^T_{\xi_l,a} \right) \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\kappa} \boldsymbol{u}_{\kappa} \\
&\qquad\qquad - \boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,b} \boldsymbol{t}^T_{\xi_l,a} \boldsymbol{u}_{\nu_{\xi_l,b}} + \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,a} \boldsymbol{t}^T_{\xi_l,b} \boldsymbol{u}_{\nu_{\xi_l,a}} \Bigg] \\
&= \sum_{l,m=1}^{d} a_m \frac{1}{2} \Bigg[ \left( Q_{\xi_l} + Q^T_{\xi_l} \right) \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\kappa} \boldsymbol{u}_{\kappa} \\
&\qquad\qquad - \boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,b} \boldsymbol{t}^T_{\xi_l,a} \boldsymbol{u}_{\nu_{\xi_l,b}} + \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,a} \boldsymbol{t}^T_{\xi_l,b} \boldsymbol{u}_{\nu_{\xi_l,a}} \Bigg]
\end{aligned}
$$

Therefore by adding the transpose, we have the entire contribution from the SAT,

$$
\begin{aligned}
\frac{d}{dt} ||\mathrm{SAT}||^2_H = \frac{1}{2} \boldsymbol{u}^T_{\boldsymbol{\kappa}} \sum_{l,m=1}^{d} a_m &\Bigg[ \left( Q_{\xi_l} + Q^T_{\xi_l} \right) \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\kappa} + \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\kappa} \left( Q_{\xi_l} + Q^T_{\xi_l} \right) \Bigg] \boldsymbol{u}_{\kappa} \\
- \sum_{l,m=1}^{d} a_m &\Bigg[ \boldsymbol{u}^T_{\kappa} \boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,b} \boldsymbol{t}^T_{\xi_l,a} \boldsymbol{u}_{\nu_{\xi_l,b}} - \boldsymbol{u}^T_{\kappa} \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,a} \boldsymbol{t}^T_{\xi_l,b} \boldsymbol{u}_{\nu_{\xi_l,a}} \Bigg]
\end{aligned}
$$

The first term cancels exactly with the volume contribution, and the second term cancels with the contributions from either side of an element interface. Finally we show the contribution from the dissipative terms.

$$
\begin{aligned}
\frac{d}{dt} ||\mathrm{SAT}^{\kappa}_{\mathrm{diss}}||^2_H &= \boldsymbol{u}^T_{\kappa} \sum_{l=1}^{d} \left[ -\boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \boldsymbol{f}^{\star}_{\xi_l,b,\mathrm{diss}} + \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \boldsymbol{f}^{\star}_{\xi_l,a,\mathrm{diss}} \right] \\
&\quad + \sum_{l=1}^{d} \left[ -\boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \boldsymbol{f}^{\star}_{\xi_l,b,\mathrm{diss}} + \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \boldsymbol{f}^{\star}_{\xi_l,a,\mathrm{diss}} \right]^T \boldsymbol{u}_{\kappa} \\
&= \frac{\sigma}{2} \sum_{l=1}^{d} \Bigg[ \boldsymbol{u}^T_{\kappa} \boldsymbol{t}_{\xi_l,b} H^{\perp}_{\xi_l} \left| \sum_{m=1}^{d} a_m \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,b} \right| \left( \boldsymbol{t}^T_{\xi_l,a} \boldsymbol{u}_{\nu_{\xi_l,b}} - \boldsymbol{t}^T_{\xi_l,b} \boldsymbol{u}_{\kappa} \right) \\
&\qquad - \boldsymbol{u}^T_{\kappa} \boldsymbol{t}_{\xi_l,a} H^{\perp}_{\xi_l} \left| \sum_{m=1}^{d} a_m \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,a} \right| \left( \boldsymbol{t}^T_{\xi_l,a} \boldsymbol{u}_{\kappa} - \boldsymbol{t}^T_{\xi_l,b} \boldsymbol{u}_{\nu_{\xi_l,a}} \right) \\
&\qquad + \left( \boldsymbol{u}^T_{\nu_{\xi_l,b}} \boldsymbol{t}_{\xi_l,a} - \boldsymbol{u}^T_{\kappa} \boldsymbol{t}_{\xi_l,b} \right) H^{\perp}_{\xi_l} \left| \sum_{m=1}^{d} a_m \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,b} \right| \boldsymbol{t}^T_{\xi_l,b} \boldsymbol{u}_{\kappa} \\
&\qquad - \left( \boldsymbol{u}^T_{\kappa} \boldsymbol{t}_{\xi_l,a} - \boldsymbol{u}^T_{\nu_{\xi_l,a}} \boldsymbol{t}_{\xi_l,b} \right) H^{\perp}_{\xi_l} \left| \sum_{m=1}^{d} a_m \mathrm{diag}\left( \mathcal{J} \frac{\partial \xi_l}{\partial x_m} \right)_{\xi_l,a} \right| \boldsymbol{t}^T_{\xi_l,a} \boldsymbol{u}^T_{\kappa} \Bigg]
\end{aligned}
$$

where again we have used the fact that a diagonal $H$ commutes with the surface metric terms. Consider now only the contribution of the SAT corresponding to the $\xi, b$ facet,

$$\frac{d}{dt}\left|\left|\mathrm{SAT}^{\kappa}_{\xi,b,\mathrm{diss}}\right|\right|^2_H = \frac{\sigma}{2}\Bigg[\boldsymbol{u}^T_{\kappa}\boldsymbol{t}_{\xi,b}H^{\perp}_{\xi}\left|\sum_{m=1}^{d} a_m \mathrm{diag}\left(\mathcal{J}\frac{\partial \xi}{\partial x_m}\right)_{\xi,b}\right|\left(\boldsymbol{t}^T_{\xi,a}\boldsymbol{u}_{\nu_{\xi,b}} - \boldsymbol{t}^T_{\xi,b}\boldsymbol{u}_{\kappa}\right) + \left(\boldsymbol{u}^T_{\nu_{\xi,b}}\boldsymbol{t}_{\xi,a} - \boldsymbol{u}^T_{\kappa}\boldsymbol{t}_{\xi,b}\right)H^{\perp}_{\xi}\left|\sum_{m=1}^{d} a_m \mathrm{diag}\left(\mathcal{J}\frac{\partial \xi}{\partial x_m}\right)_{\xi,b}\right|\boldsymbol{t}^T_{\xi,b}\boldsymbol{u}_{\kappa}\Bigg]$$

On the other side of this interface, we have a similar contribution to the $\nu_{\xi,b}$ element,

$$\frac{d}{dt}\left|\left|\mathrm{SAT}^{\nu_{\xi,b}}_{\xi,a,\mathrm{diss}}\right|\right|^2_H = \frac{\sigma}{2}\Bigg[-\boldsymbol{u}^T_{\nu_{\xi,b}}\boldsymbol{t}_{\xi,a}H^{\perp}_{\xi}\left|\sum_{m=1}^{d} a_m \mathrm{diag}\left(\mathcal{J}\frac{\partial \xi}{\partial x_m}\right)_{\xi,a=\kappa,\xi,b}\right|\left(\boldsymbol{t}^T_{\xi,a}\boldsymbol{u}_{\nu_{\xi,b}} - \boldsymbol{t}^T_{\xi,b}\boldsymbol{u}_{\kappa}\right) + \left(\boldsymbol{u}^T_{\nu_{\xi,b}}\boldsymbol{t}_{\xi,a} - \boldsymbol{u}^T_{\kappa}\boldsymbol{t}_{\xi,b}\right)H^{\perp}_{\xi}\left|\sum_{m=1}^{d} a_m \mathrm{diag}\left(\mathcal{J}\frac{\partial \xi}{\partial x_m}\right)_{\xi,a=\kappa,\xi,b}\right|\boldsymbol{t}^T_{\xi,a}\boldsymbol{u}_{\nu_{\xi,b}}\Bigg]$$

Adding these together, we find

$$\frac{d}{dt}\left|\left|\mathrm{SAT}_{\xi,b,\mathrm{diss}}\right|\right|^2_H = -\frac{\sigma}{2}\Bigg[\left(\boldsymbol{t}^T_{\xi,a}\boldsymbol{u}_{\nu_{\xi,b}} - \boldsymbol{t}^T_{\xi,b}\boldsymbol{u}_{\kappa}\right)^T H^{\perp}_{\xi}\left|\sum_{m=1}^{d} a_m \mathrm{diag}\left(\mathcal{J}\frac{\partial \xi}{\partial x_m}\right)_{\xi,b}\right|\left(\boldsymbol{t}^T_{\xi,a}\boldsymbol{u}_{\nu_{\xi,b}} - \boldsymbol{t}^T_{\xi,b}\boldsymbol{u}_{\kappa}\right) + \left(\boldsymbol{u}^T_{\nu_{\xi,b}}\boldsymbol{t}_{\xi,a} - \boldsymbol{u}^T_{\kappa}\boldsymbol{t}_{\xi,b}\right)^T H^{\perp}_{\xi}\left|\sum_{m=1}^{d} a_m \mathrm{diag}\left(\mathcal{J}\frac{\partial \xi}{\partial x_m}\right)_{\xi,b}\right|\left(\boldsymbol{u}^T_{\nu_{\xi,b}}\boldsymbol{t}_{\xi,a} - \boldsymbol{u}^T_{\kappa}\boldsymbol{t}_{\xi,b}\right)\Bigg]$$

which is clearly negative semidefinite, and hence dissipative.

## 6.4 Entropy Stability - Hadamard Form

We start with the following discretization,

$$\frac{d\boldsymbol{u}_{\kappa}}{dt} + 2\sum_{m=1}^{d}\left[D_{x_m}\circ F_{x_m}\right]\mathbf{1} = H^{-1}\sum_{l=1}^{d}\Bigg[E_{\xi_l}\circ\sum_{m=1}^{d}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\kappa}F_{x_m} + \sum_{m=1}^{d}\boldsymbol{t}_{\xi_l,a}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\boldsymbol{t}^T_{\xi_l,b}\circ F^s_{x_m}(\boldsymbol{u}_{\kappa},\boldsymbol{u}_{\nu_{\xi_l,a}}) - \sum_{m=1}^{d}\boldsymbol{t}_{\xi_l,b}H^{\perp}_{\xi_l}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\boldsymbol{t}^T_{\xi_l,a}\circ F^s_{x_m}(\boldsymbol{u}_{\kappa},\boldsymbol{u}_{\nu_{\xi_l,b}})\Bigg]\mathbf{1}$$

and assume that an entropy-consistent two-point flux is used, i.e.

$$[\![\boldsymbol{w}]\!]^T\boldsymbol{\mathcal{F}}^{\star} = [\![\psi]\!]$$

Recall the following definitions from entropy analysis,

$$\mathcal{S} \equiv \mathcal{U}_i\mathcal{W}_i - \varphi(\boldsymbol{\mathcal{W}}) \quad , \quad \mathcal{G}^{(k)} \equiv \mathcal{F}^{(k)}_i\mathcal{W}_i - \psi^{(k)}(\boldsymbol{\mathcal{W}}) \quad , \quad \frac{\partial \mathcal{S}}{\partial \mathcal{U}_i}A^{(k)}_{ij} \equiv \frac{\partial \mathcal{S}}{\partial \mathcal{U}_i}\frac{\partial \mathcal{F}^{(k)}_i}{\partial \mathcal{U}_j} = \frac{\partial \mathcal{G}^{(k)}}{\partial \mathcal{U}_j}$$

where we define entropy variables $\boldsymbol{\mathcal{W}}$, entropy potential $\varphi$, and the entropy potential fluxes $\psi^{(k)}$ as

$$\mathcal{W}_i = \frac{\partial \mathcal{S}}{\partial \mathcal{U}_i} \quad , \quad \frac{\partial \varphi}{\partial \mathcal{W}_i} = \mathcal{U}_i \quad , \quad \frac{\partial \psi^{(k)}}{\partial \mathcal{W}_i} = \mathcal{F}^{(k)}_i.$$

In order to prove entropy stability, we take the inner product of our discretization with respect to entropy variables. Throughout we will assume a diagonal $H$ so that it commutes with metric terms. We begin with the volume term,

$$\begin{aligned}\boldsymbol{w}^T H 2 \sum_{m=1}^{d} \left[D_{x_m} \circ F_{x_m}\right] \mathbf{1} &= \boldsymbol{w}^T \sum_{l.m=1}^{d} \left[\left(Q_{\xi_l} \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa Q_{\xi_l}\right) \circ F_{x_m}\right] \mathbf{1} \\ &= \sum_{l.m=1}^{d}\sum_{j=1}^{N} \boldsymbol{w}_i \left( [Q_{\xi_l}]_{ij} \left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_j + \left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_i [Q_{\xi_l}]_{ij} \right) [F_{x_m}]_{ij} \\ &= \sum_{l.m=1}^{d}\sum_{j=1}^{N} \left( [Q_{\xi_l}]_{ij} \left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_j + \left[\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa\right]_i [Q_{\xi_l}]_{ij} \right) \left(\boldsymbol{w}_i \boldsymbol{\mathcal{F}}^\star(u_i^\kappa, u_j^\kappa)\right)\end{aligned}$$

## 6.5 Conservation

We begin with a divergence form discretization,

$$\begin{aligned}\frac{d\boldsymbol{u}_\kappa}{dt} = &- \sum_{m=1}^{d} \frac{1}{2} \left[\text{diag}\left(\mathcal{J}_\kappa\right)\right]^{-1} \sum_{l=1}^{d} \left[ D_{\xi_l} \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa D_{\xi_l} \right] \boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_\kappa\right) \\ &+ \frac{1}{2} H^{-1} \sum_{l,m=1}^{d} \Bigg[ E_{\xi_l} \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa \boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_\kappa) + \boldsymbol{t}_{\xi_l,b} H_{\xi_l}^{\perp} \boldsymbol{f}_{\xi_l,b}^{\star,\text{diss}} - \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^{\perp} \boldsymbol{f}_{\xi_l,a}^{\star,\text{diss}} \\ &- \left( \boldsymbol{t}_{\xi_l,b} H_{\xi_l}^{\perp} \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b} \boldsymbol{t}_{\xi_l,a}^T \boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_{\nu_{\xi_l,b}}) - \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^{\perp} \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a} \boldsymbol{t}_{\xi_l,b}^T \boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_{\nu_{\xi_l,a}}) \right) \Bigg]\end{aligned}$$

To show conservation we contract on the left with $\mathbf{1}H$. Considering first only the volume term, we have

$$\begin{aligned}&\frac{1}{2} \sum_{l,m=1}^{d} \mathbf{1}^T \left[ Q_{\xi_l} \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa Q_{\xi_l} \right] \boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_\kappa\right) \\ =&\frac{1}{2} \sum_{l,m=1}^{d} \mathbf{1}^T \left[ \left(E_{\xi_l} - Q_{\xi_l}^T\right) \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa + \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa Q_{\xi_l} \right] \boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_\kappa\right)\end{aligned}$$

The first term cancels with the surface integral from the SAT. The second term cancels because $\mathbf{1}^T Q_{\xi_l}^T = 0$. The dissipative terms can also be assumed to be conservative across interfaces. Consider for example the local Lax-Friedrichs flux on the left interface,

$$-\mathbf{1}^T \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^{\perp} \boldsymbol{f}_{\xi_l,a}^{\star,\text{diss}} = -\sigma \mathbf{1}^T \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^{\perp} \left| \sum_{m=1}^{d} \max(\lambda_m) \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a} \right| \left( \boldsymbol{t}_{\xi_l,a}^T \boldsymbol{u}_\kappa - \boldsymbol{t}_{\xi_l,b}^T \boldsymbol{u}_{\nu_{\xi_l,a}} \right)$$

while the right side contribution from the left element is

$$\mathbf{1}^T \boldsymbol{t}_{\xi_l,b} H_{\xi_l}^{\perp} \boldsymbol{f}_{\kappa}^{\star,\text{diss}} = \sigma \mathbf{1}^T \boldsymbol{t}_{\xi_l,b} H_{\xi_l}^{\perp} \left| \sum_{m=1}^{d} \max(\lambda_m) \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a} \right| \left( \boldsymbol{t}_{\xi_l,a}^T \boldsymbol{u}_\kappa - \boldsymbol{t}_{\xi_l,b}^T \boldsymbol{u}_{\nu_{\xi_l,a}} \right)$$

These clearly cancel, as will be the case for any numerical flux function that is unique at the interfaces. This leaves us with

$$\begin{aligned}\mathbf{1}^T H \frac{d\boldsymbol{u}_\kappa}{dt} = -\frac{1}{2} \sum_{l,m=1}^{d} \mathbf{1}^T \Bigg[ &\text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa Q_{\xi_l} \boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_\kappa\right) \\ &+ \boldsymbol{t}_{\xi_l,b} H_{\xi_l}^{\perp} \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b} \boldsymbol{t}_{\xi_l,a}^T \boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_{\nu_{\xi_l,b}}) - \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^{\perp} \text{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a} \boldsymbol{t}_{\xi_l,b}^T \boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_{\nu_{\xi_l,a}}) \Bigg]\end{aligned}$$

We now take advantage of the fact that the above is a scalar to rewrite it as

$$\mathbf{1}^T H \frac{d\boldsymbol{u}_\kappa}{dt} = -\frac{1}{2}\sum_{l,m=1}^{d}\Bigg[\boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_\kappa\right)^T Q_{\xi_l}^T \mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa$$
$$+\boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_{\nu_{\xi_l,b}})^T \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\boldsymbol{t}_{\xi_l,b}^T - \boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_{\nu_{\xi_l,a}})^T \boldsymbol{t}_{\xi_l,b} H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\boldsymbol{t}_{\xi_l,a}^T\Bigg]\mathbf{1}$$

where we again used the fact that $H$ is diagonal. The final step is to consider the telescoping property. That is, we consider contributions from neighbouring elements. The element to the left (in the $\xi_l$ direction) will contribute terms

$$\boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_\kappa)^T \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}^{\kappa}\boldsymbol{t}_{\xi_l,b}^T$$

along its right boundary, whereas the element to the right will contribute terms

$$-\boldsymbol{\mathcal{F}}_m(\boldsymbol{u}_\kappa)^T \boldsymbol{t}_{\xi_l,b} H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}^{\kappa}\boldsymbol{t}_{\xi_l,a}^T$$

along its left boundary. We can combine this with the volume contribution from the $\kappa$ element to obtain

$$-\frac{1}{2}\boldsymbol{\mathcal{F}}_m\left(\boldsymbol{u}_\kappa\right)^T\sum_{l,m=1}^{d}\Bigg[Q_{\xi_l}^T\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_\kappa$$
$$-\left(\boldsymbol{t}_{\xi_l,b} H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,a}\boldsymbol{t}_{\xi_l,a}^T - \boldsymbol{t}_{\xi_l,a} H_{\xi_l}^{\perp}\mathrm{diag}\left(\mathcal{J}\frac{\partial \xi_l}{\partial x_m}\right)_{\xi_l,b}\boldsymbol{t}_{\xi_l,b}^T\right)\Bigg]\mathbf{1}$$

which is exactly the condition we used for free stream preservation. We conclude therefore that assuming proper treatment of boundary interfaces, if free stream is preserved then the discretization is conservative.

# References


[1] T. J. Barth. Numerical methods for gasdynamic systems on unstructured meshes. In D. Kroner, M. Ohlberger, and C. Rohde, editors, *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, volume 5 of *Lecture Notes in Computational Science and Engineering*, pages 195–285. Springer Berlin Heidelberg, Freiburg/Littenweiler, 1999.

[2] Mark H. Carpenter, Travis C. Fisher, Eric J. Nielsen, and Steven H. Frankel. Entropy Stable Spectral Collocation Schemes for the Navier–Stokes Equations: Discontinuous Interfaces. *SIAM Journal on Scientific Computing*, 36(5):B835–B867, January 2014.

[3] David C. Del Rey Fernández, Pieter D. Boom, Mark H. Carpenter, and David W. Zingg. Extension of Tensor-Product Generalized and Dense-Norm Summation-by-Parts Operators to Curvilinear Coordinates. *Journal of Scientific Computing*, 80(3):1957–1996, September 2019.

[4] David C. Del Rey Fernández, M. H. Carpenter, L. Dalcin, L. Fredrich, D. Rojas, A. R. Winters, G. J. Gassner, S. Zampini, and M. Parsani. Entropy Stable p-Nonconforming Discretizations with the Summation-by-Parts Property for the Compressible Euler equations. *arXiv:1909.12536 [math-ph, physics:physics]*, September 2019. arXiv: 1909.12536.

[5] Travis C. Fisher. *High Order L2 Stable Multi-Domain Finite Difference Method for Compressible Flows*. Ph.D. thesis, Purdue University, August 2012.

[6] Marshal L Merriam. An Entropy-Based Approach to Nonlinear Stability. NASA Technical Memorandum TM-101086, Ames Research Cente, Moffett Field, California, March 1989.

[7] Irving Reyna Nolasco, Lisandro Dalcin, David C. Del Rey Fernández, Stefano Zampini, and Matteo Parsani. Optimized geometrical metrics satisfying free-stream preservation. *Computers & Fluids*, 207:104555, July 2020.