Efficient computation of Jacobian matrices for entropy stable summation-by-parts schemes

Jesse Chan, Christina G. Taylor

[a]*Department of Computational and Applied Mathematics, Rice University, 6100 Main St, Houston, TX, 77005*

Abstract

Entropy stable schemes replicate an entropy inequality at the semi-discrete level. These schemes rely on an algebraic summation-by-parts (SBP) structure and a technique referred to as flux differencing. We provide simple and efficient formulas for Jacobian matrices for the semi-discrete systems of ODEs produced by entropy stable discretizations. These formulas are derived based on the structure of flux differencing and derivatives of flux functions, which can be computed using automatic differentiation (AD). Numerical results demonstrate the efficiency and utility of these Jacobian formulas, which are then used in the context of two-derivative explicit time-stepping schemes and implicit time-stepping.

1. Introduction

This paper is concerned with the numerical discretization of systems of nonlinear conservation laws. In particular, we focus on the computation of Jacobian matrices for nonlinear residuals associated with entropy conservative and entropy stable semi-discretizations. Such matrices are useful in the context of implicit time-stepping schemes [1], as well as adjoint-based sensitivity computations and optimization [2, 3].

Entropy stable discretizations mimic a continuous dissipation of entropy for nonlinear conservation laws. Let Ω denote some domain with boundary $\partial\Omega$. Nonlinear conservation laws are expressed as a system of nonlinear partial differential equations (PDEs)

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{i=1}^{d} \frac{\partial \boldsymbol{f}_i(\boldsymbol{u})}{\partial x_i} = 0, \qquad S(\boldsymbol{u}) \text{ convex}, \qquad \boldsymbol{v}(\boldsymbol{u}) = \frac{\partial S}{\partial \boldsymbol{u}}, \tag{1}$$

where $\boldsymbol{u} \in \mathbb{R}^n$ are the conservative variables, $\boldsymbol{f}_i$ are nonlinear fluxes, and $\boldsymbol{v}(\boldsymbol{u})$ are the *entropy variables* with respect to the entropy $S(\boldsymbol{u})$. By multiplying (1) by the entropy variables, vanishing viscosity solutions [4] of many fluid systems [5, 6] can be shown to satisfy the following entropy inequality

$$\int_{\Omega} \frac{\partial S(\boldsymbol{u})}{\partial t} + \sum_{i=1}^{d} \int_{\partial\Omega} \left(\boldsymbol{v}^T \boldsymbol{f}_i(\boldsymbol{u}) - \psi_i(\boldsymbol{u})\right) n_i \leq 0, \tag{2}$$

where n_i denotes the ith component of the outward normal vector and $\psi_i(\boldsymbol{u})$ denotes the entropy potential in the ith coordinate. The entropy inequality (2) is a statement of stability for nonlinear conservation laws [7, 8].

High order entropy stable schemes (see for example [9, 10, 6, 11, 12, 13, 14]) reproduce this entropy inequality at the semi-discrete level. The resulting methods display significantly improved robustness while retaining high order accuracy [15, 16]. These schemes are based on entropy conservative finite volume fluxes [17], which are extended to high order discretizations through a procedure referred to as flux differencing.

Email address: `jesse.chan@rice.edu,cgt@rice.edu` (Jesse Chan, Christina G. Taylor)

These methods have mainly been tested in the context of explicit time-stepping. However, recent works have applied entropy stable methods to both the space-time and implicit settings [18, 19].

Both space-time and implicit time discretizations require the solution of a system of nonlinear equations. This can be done using Newton's method, which involves the Jacobian matrix of the nonlinear equations. While it is possible to compute the solution to the nonlinear system without explicitly computing the Jacobian matrix using Jacobian-free Newton-Krylov methods [20, 21], the Jacobian matrix is commonly used to construct preconditioners [1].

In this work, we present efficient formulas for Jacobian matrices of systems resulting from entropy stable formulations. We also show that computing the Jacobian matrix is not significantly more expensive than evaluating the residual of the nonlinear system. Finally, we apply the new Jacobian formulas to both explicit two-derivative and implicit time-stepping schemes.

1.1. *On notation*

The notation in this paper is motivated by notation in [11, 22]. Unless otherwise specified, vector and matrices are denoted using lower and upper case bold font, respectively. We denote spatially quantities related to the spatial discretization (e.g., operators for differentiation, interpolation, or quadrature) using a bold sans serif font. Finally, continuous functions with vector arguments are interpreted as applying the continuous function to each entry of the vector.

For example, if x denotes a vector of point locations, i.e., $(\mathsf{x})_i = \boldsymbol{x}_i$, then $u(\mathsf{x})$ is interpreted as the vector

$$(u(\mathsf{x}))_i = u(\boldsymbol{x}_i).$$

Similarly, if $\mathsf{u} = u(\mathsf{x})$, then $f(\mathsf{u})$ corresponds to the vector

$$(f(\mathsf{u}))_i = f(u(\boldsymbol{x}_i)).$$

Vector-valued functions are treated similarly. For example, given a vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ and a vector u with vector-valued entries $\mathsf{u}_i = \boldsymbol{u}_i \in \mathbb{R}^n$, $(\boldsymbol{f}(\mathsf{u}))_i = \boldsymbol{f}(\boldsymbol{u}_i)$.

2. Jacobian matrix formulas for entropy conservative schemes

For clarity of presentation, we consider first a scalar nonlinear conservation law in one spatial dimension

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0. \tag{3}$$

We assume periodic boundary conditions, which will simplify the presentation of the main results. Non-periodic boundaries are treated in Section 3.2.1. The generalization to systems of nonlinear conservation laws is postponed until Section 4.

Let $f_S(x, y)$ denote a bivariate scalar flux function which is symmetric and consistent. Suppose u is a vector of nodal values of the solution. Define the vector $\mathsf{r} = \mathsf{r}(\mathsf{u})$ approximating the flux derivative $\frac{\partial f(u)}{\partial x}$ as

$$\mathsf{r}(\mathsf{u}) = 2\left(\mathsf{Q} \circ \mathsf{F}\right)\mathbf{1}, \qquad \mathsf{F}_{ij} = f_S(\mathsf{u}_i, \mathsf{u}_j), \tag{4}$$

where Q is a discretization matrix to be specified later and $\circ$ denotes the matrix Hadamard product. The simplest entropy stable numerical schemes based on flux differencing discretize (3) via the system of ODEs

$$\mathsf{M}\frac{\mathrm{d}\mathsf{u}}{\mathrm{d}t} + \mathsf{r}(\mathsf{u}) = \mathbf{0}.$$

where M is a diagonal mass (norm) matrix with positive entries. If $f_S(x, y)$ is entropy conservative (in the sense of [17]) and Q is skew-symmetric, then the resulting scheme is also discretely entropy conservative. An entropy stable scheme can be constructed from an entropy conservative scheme by adding appropriate terms which dissipate entropy [6, 23, 19].

We are interested in computing the Jacobian matrix $\frac{\partial \mathsf{r}}{\partial \mathsf{u}}$. Let $\operatorname{diag}(\mathsf{x})$ denote the diagonal matrix with the vector x on the diagonal and let $\operatorname{diag}(\mathsf{A})$ denote the vector diagonal of A. We then have the following theorem:

Theorem 2.1. *Suppose that $\mathbf{Q} = \pm\mathbf{Q}^T$. Then, the Jacobian matrix of the entropy conservative scheme (4) can be expressed as either*

$$\frac{\partial \mathbf{r}}{\partial \mathbf{u}} = 2\left(\mathbf{Q} \circ \mathbf{F}_y\right) \pm \operatorname{diag}\left(\mathbf{1}^T\left(2\mathbf{Q} \circ \mathbf{F}_y\right)\right)$$
$$\frac{\partial \mathbf{r}}{\partial \mathbf{u}} = 2\left(\mathbf{Q} \circ \mathbf{F}_x^T\right) \pm \operatorname{diag}\left(\left(2\mathbf{Q} \circ \mathbf{F}_x\right)\mathbf{1}\right)$$

where the matrices $\mathbf{F}_x, \mathbf{F}_y$ are

$$(\mathbf{F}_x)_{ij} = \left.\frac{\partial f_S}{\partial x}\right|_{\mathbf{u}_i,\mathbf{u}_j}, \qquad (\mathbf{F}_y)_{ij} = \left.\frac{\partial f_S}{\partial y}\right|_{\mathbf{u}_i,\mathbf{u}_j}.$$

Proof. We will prove the first formula involving $\mathbf{F}_y$. The second formula follows via symmetry and similar steps. By the chain rule,

$$\left(\frac{\partial \mathbf{r}}{\partial \mathbf{u}}\right)_{ij} = \frac{\partial \mathbf{r}_i}{\partial \mathbf{u}_j} = \sum_k 2\mathbf{Q}_{ik}\frac{\partial}{\partial \mathbf{u}_j} f_S\left(\mathbf{u}_i, \mathbf{u}_k\right) = \sum_k 2\mathbf{Q}_{ik}\left(\left.\frac{\partial f_S}{\partial x}\right|_{\mathbf{u}_i,\mathbf{u}_k}\frac{\partial \mathbf{u}_i}{\partial \mathbf{u}_j} + \left.\frac{\partial f_S}{\partial y}\right|_{\mathbf{u}_i,\mathbf{u}_k}\frac{\partial \mathbf{u}_k}{\partial \mathbf{u}_j}\right)$$

If $i \neq j$, then $\frac{\partial \mathbf{u}_i}{\partial \mathbf{u}_j} = \delta_{ij} = 0$. Moreover, most terms in the sum over k vanish except for $k = j$. Since $\frac{\partial \mathbf{u}_k}{\partial \mathbf{u}_j} = 1$ for $k = j$, the formula reduces to

$$\frac{\partial \mathbf{r}_i}{\partial \mathbf{u}_j} = 2\mathbf{Q}_{ij}\left.\frac{\partial f_S}{\partial y}\right|_{\mathbf{u}_i,\mathbf{u}_j}.$$

When $i = j$, $\frac{\partial \mathbf{u}_i}{\partial \mathbf{u}_j} = \frac{\partial \mathbf{u}_i}{\partial \mathbf{u}_i} = 1$, and

$$\frac{\partial \mathbf{r}_i}{\partial \mathbf{u}_i} = \left(\sum_k 2\mathbf{Q}_{ik}\left.\frac{\partial f_S}{\partial x}\right|_{\mathbf{u}_i,\mathbf{u}_k}\right) + 2\mathbf{Q}_{ii}\left.\frac{\partial f_S}{\partial y}\right|_{\mathbf{u}_i,\mathbf{u}_i}.$$

The term $2\mathbf{Q}_{ii}\left.\frac{\partial f_S}{\partial y}\right|_{\mathbf{u}_i,\mathbf{u}_i}$ is the diagonal of the matrix $2\left(\mathbf{Q} \circ \mathbf{F}_y\right)$, and we can simplify the first summation term. By the symmetry of $f_S(x,y)$, we have that

$$\left.\frac{\partial f_S}{\partial y}\right|_{x,y} = \left.\frac{\partial f_S}{\partial x}\right|_{y,x}$$

Thus, by $\mathbf{Q} = \pm\mathbf{Q}^T$,

$$\sum_k 2\mathbf{Q}_{ik}\left.\frac{\partial f_S}{\partial x}\right|_{\mathbf{u}_i,\mathbf{u}_k} = \sum_k 2\mathbf{Q}_{ik}\left.\frac{\partial f_S}{\partial y}\right|_{\mathbf{u}_k,\mathbf{u}_i} = \left(\left(2\mathbf{Q} \circ \mathbf{F}_y^T\right)\mathbf{1}\right)_i = \left(\pm\mathbf{1}^T\left(2\mathbf{Q} \circ \mathbf{F}_y\right)\right)_i.$$

□

While we consider only symmetric and skew-symmetric matrices $\mathbf{Q}$ in this work, one can use this theorem to compute the Jacobian $\frac{\partial \mathbf{r}}{\partial \mathbf{u}}$ for arbitrary matrices $\mathbf{Q}$ since any real matrix can be decomposed into symmetric and skew parts

$$\mathbf{Q} = \frac{1}{2}\left(\mathbf{Q} + \mathbf{Q}^T\right) + \frac{1}{2}\left(\mathbf{Q} - \mathbf{Q}^T\right).$$

Two applications of Theorem 2.1 then provide a formula for the Jacobian of (4).

2.1. *Computing derivatives of bivariate flux functions*

The aforementioned proofs require partial derivatives of flux functions $f_S(u_L, u_R)$ with respect to at least one argument. This can be done by hand for simple fluxes. For example, for the Burgers' equation, the flux and its derivative are

$$f_S(u_L, u_R) = \frac{1}{6}\left(u_L^2 + u_L u_R + u_R^2\right), \qquad \frac{\partial f_S}{\partial u_R} = \frac{1}{6}\left(u_L + 2u_R\right).$$

However, this procedure can become cumbersome for complex or piecewise-defined flux functions such as the logarithmic mean [24, 25]. This can be avoided by using Automatic Differentiation (AD) [26]. AD is distinct from both symbolic differentiation and finite difference approximations in that it does not return an explicit expression, but constructs a separate function which evaluates the derivative accurately up to machine precision.

In this work, we utilize the Julia implementation of forward-mode automatic differentiation provided by `ForwardDiff.jl` [27]. The procedure is remarkably simple: given some flux function `f(x,y)`, `ForwardDiff.jl` returns the derivative with respect to either x or y as another function. For example, defining the function $\left.\frac{\partial f}{\partial y}\right|_{x,y}$ is a one-line operation:

```
dfdy(x,y) = ForwardDiff.derivative(y->f(x,y),y)
```

`ForwardDiff.jacobian` is the analogous routine for computing Jacobians of vector-valued flux functions. This simple API utilizes the flexible Julia type system [28].[1]

Automatic differentiation can be directly applied to $\mathbf{r}(\mathbf{u})$ to compute the Jacobian matrix. However, because AD scales with the number of inputs and outputs, the cost of applying AD directly to $\mathbf{r}(\mathbf{u})$ increases as the discretization resolution increases. In contrast, using the approach in this paper, AD is applied only to the flux function, which has a small fixed number of inputs and outputs which are independent of the discretization resolution. As a result, the cost of evaluating derivatives of the flux function is roughly the same as the cost of evaluating the flux function itself and entries of the Jacobian matrix can be computed for roughly the same cost as a single evaluation of the nonlinear term $\mathbf{r}(\mathbf{u})$. Moreover, when computing the Jacobian matrix, the formula in Theorem 2.1 makes it simpler to directly take advantage of sparsity in $\mathbf{Q}$ without having to perform graph coloring [29].

3. Examples of discretization matrices which appear in entropy conservative numerical schemes

In this section, we give some examples of matrices $\mathbf{Q}$ which appear in entropy stable numerical discretizations. We assume periodicity, which corresponds to a skew-symmetric structure for $\mathbf{Q}$. Non-periodic domains are treated later.

3.1. *Finite volume methods*

The spatial discretization for most second order finite volume schemes can be reformulated in terms of (4) [30]. Suppose that the 1D interval $[-1, 1]$ is decomposed into K non-overlapping elements of size h. An entropy conservative finite volume scheme is given as

$$\begin{aligned}
\frac{\mathrm{d}\mathbf{u}_1}{\mathrm{d}t} + \frac{f_S(\mathbf{u}_2, \mathbf{u}_1) - f_S(\mathbf{u}_1, \mathbf{u}_K)}{h} &= 0\\
\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \frac{f_S(\mathbf{u}_{i+1}, \mathbf{u}_i) - f_S(\mathbf{u}_i, \mathbf{u}_{i-1})}{h} &= 0, \qquad i = 2, \ldots, K-1,\\
\frac{\mathrm{d}\mathbf{u}_K}{\mathrm{d}t} + \frac{f_S(\mathbf{u}_1, \mathbf{u}_K) - f_S(\mathbf{u}_K, \mathbf{u}_{K-1})}{h} &= 0,
\end{aligned}$$

[1] In practice, derivative and Jacobian functions are initialized with information about the size and data type of the input to ensure type stability in Julia.

where $\mathbf{u}_i$ denotes the average value of the solution on each element and f_S is an entropy conservative flux. Let $\mathbf{M} = h\mathbf{I}$ and let $\mathbf{Q}$ be the periodic second-order central difference matrix

$$\mathbf{Q} = \frac{1}{2}\begin{bmatrix} 0 & 1 & & \dots & -1 \\ -1 & 0 & 1 & & \\ & -1 & 0 & 1 & \\ & & & \ddots & \\ 1 & & \dots & -1 & 0 \end{bmatrix}.$$

An entropy conservative finite volume scheme is then equivalent to

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + 2\left(\mathbf{Q} \circ \mathbf{F}\right)\mathbf{1} = \mathbf{0}, \qquad \mathbf{F}_{ij} = f_S(u_i, u_j)$$

where $\mathbf{u} = [\mathbf{u}_1, \dots, \mathbf{u}_K]^T$ is the vector of solution values.

3.2. Multi-block summation-by-parts finite differences and discontinuous Galerkin spectral element methods

We consider next a multi-element summation-by-parts (SBP) finite element discretization [31, 32]. Suppose again that a one-dimensional domain Ω is decomposed into K non-overlapping elements D^k of size h. Let $\mathbf{M}$ and $\mathbf{Q} \in \mathbb{R}^{N_p \times N_p}$ denote diagonal mass (norm) and nodal differentiation matrices such that $\mathbf{M}^{-1}\mathbf{Q}$ approximates the first derivative on a reference interval and is exact for polynomials up to degree N. The operators $\mathbf{M}, \mathbf{Q}$ satisfy an SBP property if

$$\mathbf{Q} + \mathbf{Q}^T = \mathbf{B}, \qquad \mathbf{B} = \begin{bmatrix} -1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}. \tag{5}$$

We note that nodal discontinuous Galerkin spectral element (DG-SEM) discretizations [33] also fall into a SBP framework [34] and are thus also included in this framework. Since the finite volume methods described in the previous section can be interpreted as DG methods with polynomial degree $p = 0$, they also fall into this framework.

These matrices can be used to construct entropy conservative high order discretizations. Let $J = h/2$ be the Jacobian of the mapping from the reference element $[-1, 1]$ to a physical interval of size h and let $\mathbf{F}^k_{ij} = f_S(\mathbf{u}_{i,k}, \mathbf{u}_{j,k})$ denote the matrix of flux interactions between different nodes on the element D^k. A local formulation on the element D^k is given by

$$J_k \mathbf{M} \frac{\mathrm{d}\mathbf{u}_k}{\mathrm{d}t} + 2\left(\mathbf{Q} \circ \mathbf{F}^k\right)\mathbf{1} + \mathbf{B}\left(\mathbf{f}^* - f(\mathbf{u}_k)\right) = \mathbf{0},$$
$$\mathbf{f}^* = \begin{bmatrix} f_S(\mathbf{u}^+_{1,k}, \mathbf{u}_{1,k}) \\ 0 \\ \vdots \\ 0 \\ f_S(\mathbf{u}^+_{N_p,k}, \mathbf{u}_{N_p,k}) \end{bmatrix}. \tag{6}$$

where $\mathbf{u}^+_{1,k}, \mathbf{u}^+_{N_p,k}$ denote the exterior values of $\mathbf{u}_{1,k}, \mathbf{u}_{N_p,k}$ on neighboring elements. Assuming that the elements are ordered from left to right in ascending order, for interior element indices $1 < k < K$, these are given by

$$\mathbf{u}^+_{1,k} = \mathbf{u}_{N_p,k-1}, \qquad \mathbf{u}^+_{N_p,k} = \mathbf{u}_{1,k+1}.$$

In other words, the first node on D^k is connected to the last node on the previous element, and the last node on D^k is connected to the first node on the next element.

For periodic boundary conditions this local formulation can be understood as inducing a global skew-symmetric matrix. To show this, we first use the SBP property to rewrite (6) in a skew-symmetric form [35]

$$J_k \mathbf{M}\frac{\mathrm{d}\mathbf{u}_k}{\mathrm{d}t} + \left(\left(\mathbf{Q}-\mathbf{Q}^T\right)\circ \mathbf{F}^k\right)\mathbf{1} + \mathbf{B}\mathbf{f}^* = \mathbf{0}.$$

We now define a global vector $\mathbf{u}_\Omega = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K]^T$. Let the global flux matrix be defined as

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{11} & \ldots & \mathbf{F}_{1K} \\ \vdots & \ddots & \vdots \\ \mathbf{F}_{K1} & \ldots & \mathbf{F}_{KK} \end{bmatrix}, \qquad (\mathbf{F}_{k_1,k_2})_{ij} = f_S(\mathbf{u}_{i,k_1}, \mathbf{u}_{j,k_2}).$$

The blocks of the matrix $\mathbf{F}$ capture flux interactions between solution values at different nodes and elements. The local formulations can now be concatenated into a single skew-symmetric matrix

$$\mathbf{M}_\Omega \frac{\mathrm{d}\mathbf{u}_\Omega}{\mathrm{d}t} + 2\left(\mathbf{Q}_\Omega \circ \mathbf{F}\right)\mathbf{1} = \mathbf{0}, \tag{7}$$

where $\mathbf{M}_\Omega$ is the block-diagonal matrix with blocks $J_k\mathbf{M}$, and

$$\mathbf{Q}_\Omega = \frac{1}{2}\begin{bmatrix} \mathbf{S} & \mathbf{B}_R & & -\mathbf{B}_L \\ -\mathbf{B}_L & \mathbf{S} & \mathbf{B}_R & \\ & -\mathbf{B}_L & \ddots & \mathbf{B}_R \\ \mathbf{B}_R & & -\mathbf{B}_L & \mathbf{S} \end{bmatrix}, \qquad \mathbf{S} = \left(\mathbf{Q}-\mathbf{Q}^T\right), \tag{8}$$

where the matrices $\mathbf{B}_L, \mathbf{B}_R$ are zeros except for a single entry

$$\mathbf{B}_L = \begin{bmatrix} & & 1 \\ & \iddots & \\ 0 & & \end{bmatrix}, \qquad \mathbf{B}_R = \mathbf{B}_L^T = \begin{bmatrix} & & 0 \\ & \iddots & \\ 1 & & \end{bmatrix} \tag{9}$$

The matrix $\mathbf{Q}_\Omega$ can be considered a high order generalization of the finite volume matrix (5). Similar "global SBP operator" approaches were used to construct simultaneous approximation (SBP-SAT) interface coupling terms in [11, 36, 22].

3.2.1. *Non-periodic boundary conditions*

For non-periodic domains, the structure of the global differentiation matrix $\mathbf{Q}_\Omega$ changes. For finite volume, DG, and multi-block SBP methods, boundary conditions are typically imposed by specifying appropriate "exterior" values $\mathbf{u}^+_{i,k}$ in flux expressions such as (6). The resulting formulation is a small modification of (7)

$$\mathbf{M}_\Omega \frac{\mathrm{d}\mathbf{u}_\Omega}{\mathrm{d}t} + 2\left(\mathbf{Q}_\Omega \circ \mathbf{F}\right)\mathbf{1} + \mathbf{B}_\Omega \mathbf{f}^*_\Omega = \mathbf{0},$$

where $\mathbf{Q}_\Omega$, $\mathbf{B}_\Omega$, and $\mathbf{f}^*_\Omega$ are now given by

$$\mathbf{Q}_\Omega = \frac{1}{2}\begin{bmatrix} \mathbf{S} & \mathbf{B}_R & & \\ -\mathbf{B}_L & \mathbf{S} & \mathbf{B}_R & \\ & -\mathbf{B}_L & \ddots & \mathbf{B}_R \\ & & -\mathbf{B}_L & \mathbf{S} \end{bmatrix}, \qquad \mathbf{B}_\Omega = \begin{bmatrix} -\mathbf{B}_L & & & \\ & \mathbf{0} & & \\ & & \ddots & \\ & & & \mathbf{B}_R \end{bmatrix}, \qquad \mathbf{f}^*_\Omega = \begin{bmatrix} f_S(\mathbf{u}_{1,1}, \mathbf{u}^+_{1,1}) \\ 0 \\ \vdots \\ f_S(\mathbf{u}_{N_p,K}, \mathbf{u}^+_{N_p,K}) \end{bmatrix}.$$

Here, $\mathbf{u}^+_{1,1}$ and $\mathbf{u}^+_{N_p,K}$ denote the exterior values at the left and right endpoints, respectively. Since $\mathbf{Q}_\Omega$ is still skew-symmetric, we can reuse the formulas from Theorem 2.1. The term $\mathbf{B}_\Omega\mathbf{f}^*_\Omega$ can be differentiated efficiently using AD, since $\mathbf{B}_\Omega$ is a sparse diagonal matrix and $\mathbf{B}_\Omega\mathbf{f}^*_\Omega$ is a vector whose few nonzero terms are straightforward scalings of flux evaluations.

4. Systems of conservation laws

In this section, we extend the Jacobian formulas of Theorem 2.1 from scalar nonlinear conservation laws to an $n \times n$ system of conservation laws. Let $\boldsymbol{f}_S(\boldsymbol{u}_L, \boldsymbol{u}_R) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ denote an entropy conservative flux function for a 1D system of conservation laws. We first formulate a system of ODEs by modifying the definition of the arrays and matrices in (8).

Let $\mathbf{u}_\Omega$ denote a vector of vectors

$$\mathbf{u}_\Omega = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_n \end{bmatrix}, \qquad \mathbf{u}_i = \begin{bmatrix} \mathbf{u}_{i,1} \\ \mathbf{u}_{i,2} \\ \vdots \\ \mathbf{u}_{i,K} \end{bmatrix}, \qquad \mathbf{u}_{i,k} = \begin{bmatrix} \mathbf{u}_{i,k,1} \\ \mathbf{u}_{i,k,2} \\ \vdots \\ \mathbf{u}_{i,k,N_p} \end{bmatrix} \tag{10}$$

Here, $\mathbf{u}_{\ell,k,j}$ denotes the jth degree of freedom for the ℓth component of the solution on the kth element (for $\ell = 1, \ldots, n$, $k = 1, \ldots, K$ and $j = 1, \ldots, N_p$). Let (k_1, j_1) and (k_2, j_2) be multi-indices which correspond to row and columns indices of a matrix, respectively. We define the block-diagonal flux matrix $\mathbf{F}$ consisting of n diagonal blocks $\mathbf{F}_i \in \mathbb{R}^{N_pK \times N_pK}$ as

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_1 & & \\ & \ddots & \\ & & \mathbf{F}_n \end{bmatrix}, \qquad (\mathbf{F}_\ell)_{(k_1,j_1),(k_2,j_2)} = \left(\boldsymbol{f}_S\left(\mathbf{u}_{:,k_1,j_1}, \mathbf{u}_{:,k_2,j_2}\right)\right)_\ell . \tag{11}$$

where $\mathbf{u}_{:,k,j}$ denotes the vector containing all solution components at the kth element and jth node, and each entry of the block $\mathbf{F}_\ell$ for $\ell = 1, \ldots, n$ corresponds to the ℓth component of the vector-valued flux evaluated at solution states $\mathbf{u}_{:,k_1,j_1}, \mathbf{u}_{:,k_2,j_2}$.

Let $\mathbf{M}_\Omega, \mathbf{Q}_\Omega$ denote the global mass and differentiation matrices in (8). Then, an entropy conservative scheme is given by

$$(\mathbf{I}_n \otimes \mathbf{M}_\Omega) \frac{\mathrm{d}\mathbf{u}_\Omega}{\mathrm{d}t} + 2\left((\mathbf{I}_n \otimes \mathbf{Q}_\Omega) \circ \mathbf{F}\right)\mathbf{1} = \mathbf{0}.$$

where $\mathbf{I}_n \in \mathbb{R}^n$ is the $n \times n$ identity matrix.

We now provide Jacobian matrix formulas for systems of nonlinear conservation laws. The proofs are straightforward extensions of the proof of Theorem 2.1 to the vector-valued case, and we omit them for conciseness. The right hand side function $\mathbf{r}(\mathbf{u})$ for systems can be rewritten as

$$\mathbf{r}(\mathbf{u}) = 2\left((\mathbf{I}_n \otimes \mathbf{Q}_\Omega) \circ \mathbf{F}\right)\mathbf{1} = 2 \begin{bmatrix} (\mathbf{Q}_\Omega \circ \mathbf{F}_1) \\ \vdots \\ (\mathbf{Q}_\Omega \circ \mathbf{F}_n) \end{bmatrix} \mathbf{1}.$$

Then, the Jacobian matrix is

$$\frac{\partial \mathbf{r}}{\partial \mathbf{u}} = \begin{bmatrix} \boldsymbol{\partial}\mathbf{F}_{1,\mathbf{u}_1} & \ldots & \boldsymbol{\partial}\mathbf{F}_{1,\mathbf{u}_n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\partial}\mathbf{F}_{n,\mathbf{u}_1} & \ldots & \boldsymbol{\partial}\mathbf{F}_{n,\mathbf{u}_n} \end{bmatrix} \tag{12}$$

where each Jacobian block $\boldsymbol{\partial}\mathbf{F}_{i,\mathbf{u}_j}$ is evaluated as in Theorem 2.1

$$\begin{aligned} \boldsymbol{\partial}\mathbf{F}_{i,\mathbf{u}_j} &= 2\left(\mathbf{Q}_\Omega \circ \mathbf{F}_{i,\mathbf{u}_j}\right) \pm \operatorname{diag}\left(\mathbf{1}^T\left(2\mathbf{Q}_\Omega \circ \mathbf{F}_{i,\mathbf{u}_j}\right)\right) \\ \boldsymbol{\partial}\mathbf{F}_{i,\mathbf{u}_j} &= 2\left(\mathbf{Q}_\Omega \circ \mathbf{F}^T_{\mathbf{u}_i,j}\right) \pm \operatorname{diag}\left(\left(2\mathbf{Q}_\Omega \circ \mathbf{F}_{\mathbf{u}_i,j}\right)\mathbf{1}\right). \end{aligned}$$

for $\mathbf{Q}_\Omega = \pm\mathbf{Q}_\Omega^T$. Here, the flux matrix $\mathbf{F}_{i,\mathbf{u}_j}$ is evaluated via one of two formulas

$$\left(\mathbf{F}_{i,\mathbf{u}_j}\right)_{(j_1,k_1),(j_2,k_2)} = \left.\frac{\partial\left(\boldsymbol{f}_S\right)_i}{\partial \boldsymbol{u}_{R,j}}\right|_{\mathbf{u}_{:,k_1,j_1},\mathbf{u}_{:,k_2,j_2}}$$
$$\left(\mathbf{F}_{\mathbf{u}_i,j}\right)_{(j_1,k_1),(j_2,k_2)} = \left.\frac{\partial\left(\boldsymbol{f}_S\right)_i}{\partial \boldsymbol{u}_{L,j}}\right|_{\mathbf{u}_{:,k_1,j_1},\mathbf{u}_{:,k_2,j_2}},$$

where $\frac{\partial(\boldsymbol{f}_S)_i}{\partial \boldsymbol{u}_{L,j}}, \frac{\partial(\boldsymbol{f}_S)_i}{\partial \boldsymbol{u}_{R,j}}$ denote the derivatives of the ith component of the flux $\boldsymbol{f}_S\left(\boldsymbol{u}_L,\boldsymbol{u}_R\right)$ with respect to the jth solution component of $\mathbf{u}_L,\mathbf{u}_R$. Thus, each entry of the block $\mathbf{F}_{i,\mathbf{u}_j}$ corresponds to an entry of the Jacobian (with respect to $\boldsymbol{u}_L$ or $\boldsymbol{u}_R$) of $\boldsymbol{f}_S(\boldsymbol{u}_L,\boldsymbol{u}_R)$ and an entry of the global differentiation matrix $\mathbf{Q}_\Omega$.

Remark 1. *The ordering in this paper is chosen for notational convenience. In practice, other orderings typically yield more efficient solution procedures. For example, ordering the degrees of freedom by variable fields (as in [11]) yields a Jacobian matrix with a more compact bandwidth, while also making it easier to use a block compressed storage format for sparse matrices and block-based preconditioners.*

5. Extension to entropy stable (dissipative) schemes

We now consider entropy stable schemes, which include entropy dissipation terms to produce a semi-discrete dissipation (rather than conservation) of entropy. These can correspond either to physical or artificial viscosity mechanisms [37, 23] or numerical interface dissipation [38]. Because Jacobian matrices for artificial viscosity mechanisms have been discussed in more detail in the time-implicit literature [39] we focus instead on numerical interface dissipation.

Let $\boldsymbol{d}_S\left(\boldsymbol{u}_L,\boldsymbol{u}_R\right)$ be an entropy dissipative anti-symmetric flux such that

$$\boldsymbol{d}_S\left(\boldsymbol{u}_L,\boldsymbol{u}_R\right) = -\boldsymbol{d}_S\left(\boldsymbol{u}_R,\boldsymbol{u}_L\right), \qquad \left(\boldsymbol{v}_L-\boldsymbol{v}_R\right)^T\boldsymbol{d}_S\left(\boldsymbol{u}_L,\boldsymbol{u}_R\right) \geq 0.$$

Note that the anti-symmetry of $\boldsymbol{d}_S$ implies that $\boldsymbol{d}_S(\boldsymbol{u},\boldsymbol{u}) = \mathbf{0}$. Fluxes which fall into this category include the Lax-Friedrichs flux

$$\boldsymbol{d}_S(\boldsymbol{u}_L,\boldsymbol{u}_R) = \frac{|\lambda|}{2}(\boldsymbol{u}_L-\boldsymbol{u}_R), \qquad \lambda = \text{estimate of maximum wavespeed},$$

as well as HLLC fluxes [6] and matrix dissipation fluxes [38].

5.1. Scalar dissipative fluxes

We will begin by considering scalar dissipative fluxes $d_S(u_L,u_R)$ and dissipation terms of the form

$$\mathbf{d}(\mathbf{u}) = \left(\mathbf{B}\circ\mathbf{D}\right)\mathbf{1}$$

where $\mathbf{B}$ is a symmetric non-negative matrix and the entries of $\mathbf{D}_{ij} = d_S(\mathbf{u}_i,\mathbf{u}_j)$ correspond to evaluations of the dissipative flux. For the high order DG-SBP discretizations of periodic domains described in (7), $\boldsymbol{B}_\Omega$ is the matrix

$$\mathbf{B} = \frac{1}{2}\begin{bmatrix} & \mathbf{B}_R & & \mathbf{B}_L \\ \mathbf{B}_L & & \mathbf{B}_R & \\ & \mathbf{B}_L & \ddots & \mathbf{B}_R \\ \mathbf{B}_R & & \mathbf{B}_L & \end{bmatrix} \tag{13}$$

where $\mathbf{B}_L,\mathbf{B}_R$ are defined as in (9).

To compute the Jacobian of this term, we can note that Theorem 2.1 assumes that the discretization matrix is skew-symmetric (or symmetric), while the flux matrix is symmetric. Here, the orders are reversed — the flux matrix $\boldsymbol{D}$ is now skew-symmetric, while the discretization matrix $\boldsymbol{B}_\Omega$ is symmetric. Thus, repeating the steps of the proof of Theorem 2.1, one can show that the Jacobians of the dissipative term can be computed using one of two formulas.

Theorem 5.1. *Let* $\mathbf{d}(\mathbf{u}) = (\mathbf{B} \circ \mathbf{D})\, \mathbf{1}$*, where* $\mathbf{B}$ *is a symmetric matrix,* $\mathbf{D}_{ij} = d_S(\mathbf{u}_i, \mathbf{u}_j)$*, and* d_S *is an anti-symmetric bivariate function. Then,*

$$\begin{aligned}\frac{\partial \mathbf{d}}{\partial \mathbf{u}} &= -(\mathbf{B} \circ \mathbf{D}_x^T) + \operatorname{diag}\left(\left(\mathbf{B} \circ \mathbf{D}_x^T\right) \mathbf{1}\right), \\ \frac{\partial \mathbf{d}}{\partial \mathbf{u}} &= (\mathbf{B} \circ \mathbf{D}_y) - \operatorname{diag}\left(\mathbf{1}^T (\mathbf{B} \circ \mathbf{D}_y)\right)\end{aligned} \tag{14}$$

where the matrices $\mathbf{D}_x, \mathbf{D}_y$ *are*

$$(\mathbf{D}_x)_{ij} = \left.\frac{\partial d_S}{\partial u_L}\right|_{\mathbf{u}_i, \mathbf{u}_j}, \qquad (\mathbf{D}_y)_{ij} = \left.\frac{\partial d_S}{\partial u_R}\right|_{\mathbf{u}_i, \mathbf{u}_j}.$$

Proof. We will prove the second formula in (14) involving $\mathbf{D}_y$ using the same approach as the proof of Theorem 2.1. The proof of the first formula results from the fact that $\mathbf{D}_y = -\mathbf{D}_x^T$ by the anti-symmetry of $d_S(u_L, u_R)$. Applying the chain rule yields

$$\frac{\partial \mathbf{d}_i}{\partial \mathbf{u}_j} = \sum_k \mathbf{B}_{ik} \left(\left.\frac{\partial d_S}{\partial x}\right|_{\mathbf{u}_i, \mathbf{u}_k} \frac{\partial \mathbf{u}_i}{\partial \mathbf{u}_j} + \left.\frac{\partial d_S}{\partial y}\right|_{\mathbf{u}_i, \mathbf{u}_k} \frac{\partial \mathbf{u}_k}{\partial \mathbf{u}_j} \right)$$

If $i \neq j$, then $\frac{\partial \mathbf{u}_i}{\partial \mathbf{u}_j} = 0$ and the sum reduces to the single term $k = j$

$$\frac{\partial \mathbf{d}_i}{\partial \mathbf{u}_j} = \mathbf{B}_{ij} \left.\frac{\partial d_S}{\partial y}\right|_{\mathbf{u}_i, \mathbf{u}_j}.$$

For $i = j$, $\frac{\partial \mathbf{u}_i}{\partial \mathbf{u}_j} = 1$. Using the symmetry of $\mathbf{B}$ and anti-symmetry of d_S yields

$$\frac{\partial \mathbf{d}_i}{\partial \mathbf{u}_i} = \left(\sum_k \mathbf{B}_{ik} \left.\frac{\partial d_S}{\partial x}\right|_{\mathbf{u}_i, \mathbf{u}_k} \right) + \mathbf{B}_{ii} \left.\frac{\partial d_S}{\partial y}\right|_{\mathbf{u}_i, \mathbf{u}_i} = \left(-\sum_k \mathbf{B}_{ik} \left.\frac{\partial d_S}{\partial y}\right|_{\mathbf{u}_k, \mathbf{u}_i} \right) + \mathbf{B}_{ii} \left.\frac{\partial d_S}{\partial y}\right|_{\mathbf{u}_i, \mathbf{u}_i}.$$

□

5.2. *Vector-valued dissipative fluxes*

For a vector-valued dissipative flux, the dissipative contribution is

$$\mathbf{d}(\mathbf{u}) = \begin{bmatrix} (\mathbf{B} \circ \mathbf{D}_1) \\ \vdots \\ (\mathbf{B} \circ \mathbf{D}_n) \end{bmatrix} \mathbf{1}$$

where each matrix block $(\mathbf{D}_\ell)_{(j_1,k_1),(j_2,k_2)} = \left(\boldsymbol{d}_S\left(\mathbf{u}_{:,k_1,j_1}, \mathbf{u}_{:,k_2,j_2}\right)\right)_\ell$ corresponds to the ith component of the dissipative flux, where $\mathbf{u}$ is ordered as in (10). Then, the Jacobian of $\mathbf{d}(\mathbf{u})$ yields the following block matrix

$$\frac{\partial \mathbf{d}}{\partial \mathbf{u}} = \begin{bmatrix} \boldsymbol{\partial}\mathbf{D}_{1,\mathbf{u}_1} & \dots & \boldsymbol{\partial}\mathbf{D}_{1,\mathbf{u}_n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\partial}\mathbf{D}_{n,\mathbf{u}_1} & \dots & \boldsymbol{\partial}\mathbf{D}_{n,\mathbf{u}_n} \end{bmatrix} \tag{15}$$

where each Jacobian block $\boldsymbol{\partial}\mathbf{D}_{i,\mathbf{u}_j}$ is evaluated as in (14) using one of two formulas

$$\begin{aligned}\boldsymbol{\partial}\mathbf{D}_{i,\mathbf{u}_j} &= \left(\mathbf{B} \circ \mathbf{D}_{i,\mathbf{u}_j}\right) - \operatorname{diag}\left(\mathbf{1}^T \left(\mathbf{B} \circ \mathbf{D}_{i,\mathbf{u}_j}\right)\right) \\ \boldsymbol{\partial}\mathbf{D}_{i,\mathbf{u}_j} &= -\left(\mathbf{B} \circ \mathbf{D}_{\mathbf{u}_i,j}\right) + \operatorname{diag}\left(\left(\mathbf{B} \circ \mathbf{D}_{\mathbf{u}_i,j}\right) \mathbf{1}\right),\end{aligned}$$

where the dissipative flux matrices $\mathbf{D}_{i,\mathbf{u}_j}, \mathbf{D}_{\mathbf{u}_i,j}$ are defined in terms of entries of the Jacobian of $\boldsymbol{d}_S$

$$\begin{aligned}\left(\mathbf{D}_{i,\mathbf{u}_j}\right)_{(j_1,k_1),(j_2,k_2)} &= \left.\frac{\partial\left(\boldsymbol{d}_S\right)_i}{\partial \boldsymbol{u}_{R,j}}\right|_{\mathbf{u}_{:,k_1,j_1},\mathbf{u}_{:,k_2,j_2}} \\ \left(\mathbf{D}_{\mathbf{u}_i,j}\right)_{(j_1,k_1),(j_2,k_2)} &= \left.\frac{\partial\left(\boldsymbol{d}_S\right)_i}{\partial \boldsymbol{u}_{L,j}}\right|_{\mathbf{u}_{:,k_1,j_1},\mathbf{u}_{:,k_2,j_2}},\end{aligned}$$

Remark 2. *If the derivative of $\boldsymbol{d}_S$ with respect to its second argument $\boldsymbol{u}_R$ is used to compute the dissipative flux matrices, then the structure of the dissipative Jacobian is identical to the structure of the entropy conservative Jacobian (12). Thus, given discretization matrices $\boldsymbol{Q}_\Omega, \boldsymbol{B}_\Omega$ and functions which evaluate derivatives of flux functions $\boldsymbol{f}_S, \boldsymbol{d}_S$ with respect to their second arguments, the same routine can be used to compute both the entropy conservative and dissipative Jacobians.*

6. Non-collocated schemes: hybridized SBP operators, entropy projection, over-integration

Most entropy stable schemes rely on "collocated" SBP operators (where the mass matrix $\boldsymbol{M}_\Omega$ is diagonal) constructed using nodal sets which include boundary nodes [6, 11]. However, in certain cases energy and entropy stable SBP schemes constructed using non-diagonal mass matrices [12, 30] and more general nodal sets [40, 41, 42, 36] achieve higher accuracy than SBP schemes built on nodal sets which include boundary nodes. We discuss how to extend Jacobian formulas to "modal" formulations for entropy conservative schemes (the extension to entropy stable schemes is similar).

6.1. "Modal" entropy conservative schemes

We now assume that the solution is represented using a "modal" expansion

$$u(\boldsymbol{x},t) \approx \sum_{i=1}^{N_p} \widehat{\mathbf{u}}_{k,i}(t)\phi_i(\boldsymbol{x}),$$

where $\widehat{\mathbf{u}}_{k,i}$ denotes the coefficients of the solution on an element D^k. We assume two sets of quadrature points: volume quadrature points and weights, $\{w_i, \mathbf{x}_{q,i}\}_{i=1}^{N_q}$, and surface quadrature points, $\{w_{f,i}, \mathbf{x}_{f,i}\}_{i=1}^{N_f}$. We assume both quadrature rules are exact for certain classes of integrands as detailed in [35, 30].

Evaluating $\mathbf{u}(\mathbf{x},t)$ at quadrature points requires multiplication by an interpolation matrix $\mathbf{V}$

$$\begin{aligned}\mathbf{V}_{ij} &= \phi_j(\mathbf{x}_i), \qquad i = 1,\dots,N_q, \qquad j = 1,\dots,N_p \\ \left(\mathbf{V}_f\right)_{ij} &= \phi_j(\mathbf{x}_{f,i}), \qquad i = 1,\dots,N_f, \qquad j = 1,\dots,N_p.\end{aligned}$$

We can similarly define mass and projection matrices $\mathbf{M}, \mathbf{P}$

$$\mathbf{M} = \mathbf{V}^T\mathbf{W}\mathbf{V}, \qquad \mathbf{P} = \mathbf{M}^{-1}\mathbf{V}^T\mathbf{W},$$

where $\mathbf{W}$ is a diagonal matrix whose entries are the quadrature weights w_i. We also define a face interpolation matrix

$$\mathbf{E} = \mathbf{V}_f\mathbf{P}$$

which evaluates the solution at face quadrature points given values at volume quadrature points. Finally, we define the matrix $\mathbf{V}_h$ as the mapping between local coefficients $\widehat{\mathbf{u}}_k$ and the combined vector of volume and surface quadrature points

$$\mathbf{V}_h = \begin{bmatrix}\mathbf{V}\\ \mathbf{V}_f\end{bmatrix}.$$

These matrices are involved in the application of hybridized SBP operators (originally referred to as decoupled SBP operators) [12, 43]. We present the main ideas in a 1D setting and refer the reader to [11, 12, 35] for details on multi-dimensional settings.

Given some modal weak differentiation matrix $\widehat{\mathbf{Q}}$ which acts on the basis coefficients $\widehat{\mathbf{u}}_k$, we define a nodal differentiation matrix $\mathbf{Q} = \mathbf{P}^T\widehat{\mathbf{Q}}\mathbf{P}$. Then we can define a hybridized SBP operator as

$$\mathbf{Q}_h = \frac{1}{2}\begin{bmatrix} \mathbf{Q} - \mathbf{Q}^T & \mathbf{E}^T\mathbf{B} \\ -\mathbf{B}\mathbf{E} & \mathbf{B} \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} -1 & \\ & 1 \end{bmatrix}.$$

The operator $\mathbf{Q}_h$ can be used to approximate coefficients of the derivative in the basis $\phi_i(\mathbf{x})$. Let $f(u)$ denote some function of $u(\mathbf{x})$, and let $\widehat{\mathbf{u}}$ denote the basis coefficients of $u(\mathbf{x})$. Then,

$$\frac{\partial f(u)}{\partial x} \approx \sum_j \widehat{\mathbf{f}}_j \phi_j(\mathbf{x}), \qquad \widehat{\mathbf{f}} = \mathbf{M}^{-1}\mathbf{V}_h^T\mathbf{Q}_h f\left(\mathbf{V}_h\widehat{\mathbf{u}}\right)$$

We now construct global matrices for the multi-element (periodic) case. We begin by concatenating the local coefficients $\widehat{\mathbf{u}}_{k,i}$ into a global coefficient vector $\widehat{\mathbf{u}}_\Omega$. We also introduce boundary matrices $\mathbf{B}, \mathbf{B}_L$, and $\mathbf{B}_R$ which enforce coupling between different elements and are defined as

$$\mathbf{B}_L = \begin{bmatrix} & 1 \\ 0 & \end{bmatrix}, \qquad \mathbf{B}_R = \begin{bmatrix} & 0 \\ 1 & \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} -1 & \\ & 1 \end{bmatrix}.$$

In the multi-dimensional case, the entries of $\mathbf{B}_L, \mathbf{B}_R$ correspond instead to outward normals scaled by surface quadrature weights and surface Jacobians (e.g., the area ratios between reference and physical surface elements) [11, 12].

We can also adapt $\mathbf{Q}_h$ to construct a globally skew-symmetric differentiation matrix (see also [36]). Define the matrix $\mathbf{S} = \mathbf{Q} - \mathbf{Q}^T$ and define $\mathbf{Q}_\Omega$ as the global block matrix

$$\mathbf{Q}_\Omega = \frac{1}{2}\left[\begin{array}{cc|cc|cc|cc}
\mathbf{S} & \mathbf{E}^T\mathbf{B} & & & & & & \\
-\mathbf{B}\mathbf{E} & & & \mathbf{B}_R & & & & -\mathbf{B}_L \\ \hline
& & \mathbf{S} & \mathbf{E}^T\mathbf{B} & & & & \\
& -\mathbf{B}_L & -\mathbf{B}\mathbf{E} & & & \mathbf{B}_R & & \\ \hline
& & & & \ddots & \ddots & & \\
& & & -\mathbf{B}_L & \ddots & \ddots & \ddots & \mathbf{B}_R \\ \hline
& & & & & \ddots & \mathbf{S} & \mathbf{E}^T\mathbf{B} \\
\mathbf{B}_R & & & & & -\mathbf{B}_L & -\mathbf{B}\mathbf{E} &
\end{array}\right],$$

We abuse notation and redefine $\mathbf{V}, \mathbf{E}, \mathbf{P}$ and $\mathbf{V}_h$ as *global* interpolation, projection, and extrapolation matrices

$$\begin{aligned} \mathbf{V} &\longrightarrow \mathbf{I}_K \otimes \mathbf{V}, & \mathbf{P} &\longrightarrow \mathbf{I}_K \otimes \mathbf{P} \\ \mathbf{E} &\longrightarrow \mathbf{I}_K \otimes \mathbf{E}, & \mathbf{V}_h &\longrightarrow \mathbf{I}_K \otimes \mathbf{V}_h \end{aligned}$$

Finally, we assume that the global solution $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^n$ is vector-valued, and order the solution coefficients as in Section 4.

It was shown in [13, 12] that when either the mass matrix is non-diagonal or the nodal set does not contain appropriate boundary points, it is necessary to perform an entropy projection (or extrapolation [43]) step to ensure discrete entropy stability. Let $\boldsymbol{v}(\boldsymbol{u})$ denote the entropy variables as a function of the conservative variables, and let $\boldsymbol{u}(\boldsymbol{v})$ denote the inverse mapping. We define the entropy projected variables $\widetilde{\mathbf{u}}_\Omega$ as

$$\widetilde{\mathbf{u}}_\Omega = \boldsymbol{u}\left(\mathbf{V}_h\mathbf{P}\boldsymbol{v}\left(\mathbf{V}\widehat{\mathbf{u}}_\Omega\right)\right). \tag{16}$$

Let $\mathbf{F}$ again denote the block-diagonal flux matrix in (11). We evaluate each flux block $\mathbf{F}_\ell$ using the entropy projected variables

$$\left(\mathbf{F}_\ell\right)_{(k_1,j_1),(k_2,j_2)} = \left(\boldsymbol{f}_S\left(\widetilde{\mathbf{u}}_{:,k_1,j_1}, \widetilde{\mathbf{u}}_{:,k_2,j_2}\right)\right)_\ell. \tag{17}$$

Then, an entropy conservative method is given by

$$\left(\mathbf{I}_n \otimes \mathbf{M}_\Omega\right) \frac{\mathrm{d}\mathbf{u}_\Omega}{\mathrm{d}t} + 2\left(\mathbf{I}_n \otimes \mathbf{V}_h\right)^T \left(\left(\mathbf{1}_n \mathbf{1}_n^T \otimes \mathbf{Q}_\Omega\right) \circ \mathbf{F}\right) \mathbf{1} = \mathbf{0}.$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix and $\mathbf{1}_n$ denotes the length n vector of all ones.

6.2. Jacobian matrices for modal entropy stable schemes

We redefine the nonlinear term as

$$\mathbf{r}(\widehat{\mathbf{u}}) = 2\left(\mathbf{I}_n \otimes \mathbf{V}_h\right)^T \left(\left(\mathbf{1}_n \mathbf{1}_n^T \otimes \mathbf{Q}_\Omega\right) \circ \mathbf{F}\right) \mathbf{1}.$$

where the flux matrix $\mathbf{F}$ is computed using the entropy projected conservative variables (16) via (17). Let $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}$ and $\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}}$ denote Jacobians of the conservative variables with respect to the entropy variables and vice versa. These have been explicitly derived for several equations (for example, the Jacobians for the compressible Navier-Stokes equations are given in [5]).

We can compute the Jacobian of $\mathbf{r}(\widehat{\mathbf{u}})$ via the chain rule. We assume a scalar equation $n = 1$ for simplicity, and motivate our approach by considering an entry $i \neq j$ of the Jacobian

$$\left(\frac{\partial \mathbf{r}}{\partial \widehat{\mathbf{u}}_\Omega}\right)_{ij} = 2\mathbf{V}_h^T \frac{\partial}{\partial \left(\widehat{\mathbf{u}}_\Omega\right)_j} \left(\left(\mathbf{Q}_\Omega \circ \mathbf{F}\right) \mathbf{1}\right)_i.$$

We focus on the latter term $\frac{\partial}{\partial \widehat{\mathbf{u}}_\Omega} \left(\mathbf{Q}_\Omega \circ \mathbf{F}\right) \mathbf{1}$

$$\left(\frac{\partial}{\partial \widehat{\mathbf{u}}_\Omega} \left(\mathbf{Q}_\Omega \circ \mathbf{F}\right) \mathbf{1}\right)_{ij} = \frac{\partial}{\partial \widehat{\mathbf{u}}_{\Omega,j}} \sum_k \left(\mathbf{Q}_\Omega\right)_{ik} \boldsymbol{f}_S\left(\widetilde{\mathbf{u}}_i, \widetilde{\mathbf{u}}_k\right) = \sum_k \left(\mathbf{Q}_\Omega\right)_{ik} \left.\frac{\partial \boldsymbol{f}_S}{\partial y}\right|_{\widetilde{\mathbf{u}}_i, \widetilde{\mathbf{u}}_k} \frac{\partial \widetilde{\mathbf{u}}_i}{\partial \widehat{\mathbf{u}}_{\Omega,j}}$$

We observe that the term $\frac{\partial \widetilde{\mathbf{u}}_i}{\partial \widehat{\mathbf{u}}_{\Omega,j}}$ does not disappear as it did in the proof of Theorem 2.1. We thus treat the Jacobian matrix in two parts. First, we define the "unassembled" Jacobian matrix $\frac{\partial \widetilde{\mathbf{r}}}{\partial \widetilde{\mathbf{u}}}$ as

$$\left(\frac{\partial \widetilde{\mathbf{r}}}{\partial \widetilde{\mathbf{u}}}\right)_{ij} = \left(\mathbf{Q}_\Omega\right)_{ij} \left.\frac{\partial \boldsymbol{f}_S}{\partial y}\right|_{\widetilde{\mathbf{u}}_i, \widetilde{\mathbf{u}}_j} = \left(\mathbf{Q}_\Omega \circ \mathbf{F}_y\right)_{ij}. \tag{18}$$

The construction of $\frac{\partial \widetilde{\mathbf{r}}}{\partial \widetilde{\mathbf{u}}}$ for systems $(n > 1)$ is carried out using the procedure described in Section 4. Let $\widetilde{\mathbf{v}}$ denote the projected entropy variables evaluated at volume quadrature points

$$\widetilde{\mathbf{v}} = \mathbf{V}_h \mathbf{P} \boldsymbol{v}\left(\mathbf{V} \widehat{\mathbf{u}}_\Omega\right).$$

The vector $\frac{\partial \widetilde{\mathbf{u}}}{\partial \widehat{\mathbf{u}}_\Omega}$ can be further expanded as

$$\frac{\partial \widetilde{\mathbf{u}}}{\partial \widehat{\mathbf{u}}_\Omega} = \left.\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\right|_{\widetilde{\mathbf{v}}} \mathbf{V}_h \mathbf{P} \left.\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}}\right|_{\mathbf{V}\widehat{\mathbf{u}}_\Omega} \boldsymbol{V}.$$

where the Jacobian matrices for the maps between conservative and entropy variables are block diagonal matrices given by

$$\left.\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\right|_{\widetilde{\mathbf{v}}} = \begin{bmatrix} \left.\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\right|_{\widetilde{\mathbf{v}}_1} & & \\ & \ddots & \\ & & \left.\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\right|_{\widetilde{\mathbf{v}}_K} \end{bmatrix}, \qquad \left.\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\right|_{\widetilde{\mathbf{u}}_k} = \begin{bmatrix} \left.\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\right|_{\widetilde{\mathbf{u}}_{1,k}} & & \\ & \ddots & \\ & & \left.\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\right|_{\widetilde{\mathbf{u}}_{N_p,k}} \end{bmatrix}, \tag{19}$$

where the local block $\left.\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\right|_{\widetilde{\mathbf{u}}_{j,k}}$ is the Jacobian matrix $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}$ evaluated at the jth nodal solution value $\widetilde{\mathbf{u}}_{j,k}$ on the kth element.

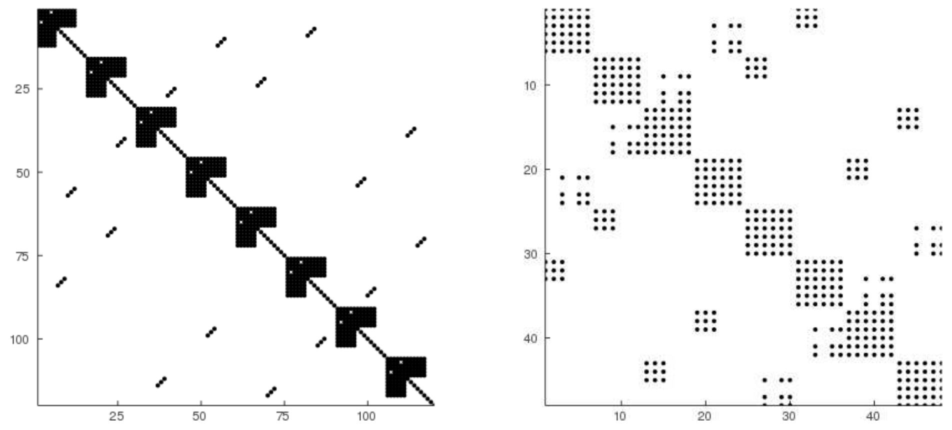

(a) "Unassembled" Jacobian matrix (18) (b) "Assembled" Jacobian matrix (20)

Figure 1: Spy plots of assembled and unassembled Jacobian matrices for Burgers' equation for $N = 2$ on a 2×2 uniform triangular mesh of $[-1, 1]^2$.

Let N_p, N_q, and N_f denote the number of total basis functions, quadrature points, and face quadrature points respectively, and define $N_{\text{total}} = N_q + N_f$. The structure and dimensions of matrices involved in constructing the "assembled" Jacobian matrix are illustrated as follows:

$$\frac{\partial \mathbf{r}}{\partial \widehat{\mathbf{u}}_\Omega} = \underbrace{\boxed{\mathbf{V}_h^T}}_{(N_p \times N_{\text{total}})} \underbrace{\boxed{\mathbf{Q}_\Omega \circ \frac{\partial \widetilde{\mathbf{r}}}{\partial \widetilde{\mathbf{u}}}}}_{(N_{\text{total}} \times N_{\text{total}})} \underbrace{\boxed{\left.\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}\right|_{\widetilde{\mathbf{v}}}}}_{(N_{\text{total}} \times N_{\text{total}})} \underbrace{\boxed{\mathbf{V}_h \mathbf{P}}}_{(N_{\text{total}} \times N_q)} \underbrace{\boxed{\left.\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}}\right|_{\mathbf{V}\widehat{\mathbf{u}}_\Omega}}}_{(N_q \times N_q)} \underbrace{\boxed{\mathbf{V}}}_{(N_q \times N_p)} \tag{20}$$

"Unassembled" and "assembled" Jacobian matrices (18) and (20) for an entropy conservative discretization of a 2D Burgers' equation [12] are shown in Figure 1. We note that the structure of these matrices becomes simplified under common assumptions for entropy stable discretizations. The most common assumptions are either "collocated volume nodes" or "collocated volume and surface nodes" [44]. When volume nodes are collocated, the solution is represented using a nodal Lagrange basis constructed using $N_q = N_p$ volume quadrature nodes.[2] When surface nodes are collocated, the surface quadrature points are also a subset of the volume quadrature nodes [34, 6].

If only volume nodes are collocated [36], then $\mathbf{V} = \mathbf{I}$. If both volume and surface nodes are collocated, the system reduces to the simplified system described in Section 4 using the fact that $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}} = \left(\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}}\right)^{-1}$.

7. Numerical experiments

In this section, we verify our theoretical results and compare the computational efficiency of the formulas derived in this paper with other methods for computing the Jacobian. Additional numerical experiments are included within Supplementary Material.

7.1. Verification of Jacobian formulas

[2]This definition refers to discretizations which utilize an explicit basis. For entropy stable SBP discretizations, volume nodes are typically collocated by construction. This is possible because nodal degrees of freedom for SBP discretizations do not necessarily correspond to a nodal basis.

We begin by verifying the correctness of Theorem 2.1 and its extension to systems of nonlinear conservation laws. We do so by comparing these formulas to Jacobians computed directly using automatic differentiation for $\mathbf{r}(\mathbf{u}) = (\mathbf{Q} \circ \mathbf{F})\mathbf{1}$, where $\mathbf{Q}$ is a randomly generated symmetric or skew-symmetric matrix and $\mathbf{F}$ is the flux matrix defined in (4) for scalar fluxes and (11) for systems.

We compute Jacobians for three different fluxes. The first is the entropy conservative flux for Burgers' equation

$$f_S(u_L, u_R) = \frac{1}{6}\left(u_L^2 + u_L u_R + u_R^2\right).$$

The second set of fluxes are entropy conservative fluxes for the two-dimensional shallow water equations with solution fields h, hu, hv corresponding to water height and x, y momentum [45, 46]. Let the average be defined as $\{\{u\}\} = \frac{u_L + u_R}{2}$. The two-dimensional shallow water fluxes $\boldsymbol{f}_{i,S}$ for each coordinate direction $i = 1, \ldots, d$ are given by

$$\boldsymbol{f}_{1,S} = \begin{bmatrix} \{\{hu\}\} \\ \{\{hu\}\}\{\{u\}\} + \frac{g}{2} h_L h_R \\ \{\{hu\}\}\{\{v\}\} \end{bmatrix}, \qquad \boldsymbol{f}_{2,S} = \begin{bmatrix} \{\{hv\}\} \\ \{\{hv\}\}\{\{u\}\} \\ \{\{hv\}\}\{\{v\}\} + \frac{g}{2} h_L h_R \end{bmatrix}.$$

The third set of fluxes are kinetic energy preserving and entropy conservative fluxes for the 3D compressible Euler equations [47]. The solution fields are $\rho, \rho u, \rho v, \rho w, E$, corresponding to density, $x/y/z$ momentum, and total energy. Let $\{\{\cdot\}\}^{\log}$ denote the logarithmic mean

$$\{\{u\}\}^{\log} = \frac{u_R - u_L}{\log(u_R) - \log(u_L)},$$

which we compute using the numerically stable expansion of [25] with $\gamma = 1.4$. The fluxes $\boldsymbol{f}_{i,S}$ for each coordinate direction $i = 1, \ldots, d$ are then given by

$$\boldsymbol{f}_{1,S} = \begin{pmatrix} \{\{\rho\}\}^{\log}\{\{u\}\} \\ \{\{\rho\}\}^{\log}\{\{u\}\}^2 + p_{\text{avg}} \\ \{\{\rho\}\}^{\log}\{\{u\}\}\{\{v\}\} \\ \{\{\rho\}\}^{\log}\{\{u\}\}\{\{w\}\} \\ (E_{\text{avg}} + p_{\text{avg}})\{\{u\}\} \end{pmatrix}, \qquad \boldsymbol{f}_{2,S} = \begin{pmatrix} \{\{\rho\}\}^{\log}\{\{v\}\} \\ \{\{\rho\}\}^{\log}\{\{u\}\}\{\{v\}\} \\ \{\{\rho\}\}^{\log}\{\{v\}\}^2 + p_{\text{avg}} \\ \{\{\rho\}\}^{\log}\{\{v\}\}\{\{w\}\} \\ (E_{\text{avg}} + p_{\text{avg}})\{\{v\}\} \end{pmatrix},$$
$$\boldsymbol{f}_{3,S} = \begin{pmatrix} \{\{\rho\}\}^{\log}\{\{w\}\} \\ \{\{\rho\}\}^{\log}\{\{u\}\}\{\{w\}\} \\ \{\{\rho\}\}^{\log}\{\{v\}\}\{\{w\}\} \\ \{\{\rho\}\}^{\log}\{\{w\}\}^2 + p_{\text{avg}} \\ (E_{\text{avg}} + p_{\text{avg}})\{\{u_3\}\} \end{pmatrix},$$

where the auxiliary quantities are defined as

$$\beta = \frac{\rho}{2p}, \qquad p_{\text{avg}} = \frac{\{\{\rho\}\}}{2\{\{\beta\}\}}, \qquad E_{\text{avg}} = \frac{\{\{\rho\}\}^{\log}}{2(\gamma-1)\{\{\beta\}\}^{\log}} + \frac{1}{2}\{\{\rho\}\}^{\log} u_{\text{avg}}^2,$$
$$u_{\text{avg}}^2 = u_L u_R + v_L v_R + w_L w_R.$$

We also verify Theorem 5.1 for each system by computing the Jacobian matrix for the dissipative Lax-Friedrichs flux

$$\boldsymbol{d}_S(\boldsymbol{u}_L, \boldsymbol{u}_R) = \frac{\lambda_{\max}}{2}(\boldsymbol{u}_L - \boldsymbol{u}_R)$$

where $\lambda_{\max}$ is an estimate of the maximum 1D wavespeed between $\boldsymbol{u}_L, \boldsymbol{u}_R$ along some unit vector $\boldsymbol{n}$ (e.g., the outward normal). In all cases, we take $\lambda_{\max} = \max(\lambda(\boldsymbol{u}_L), \lambda(\boldsymbol{u}_R))$, where $\lambda(\boldsymbol{u})$ is an upper bound on the wavespeed. For both shallow water and Euler, $\lambda(\boldsymbol{u}) = |u_n| + c$, where u_n is the normal component of velocity and c is the speed of sound. For shallow water, $c = \sqrt{gh}$, while for Euler, $c = \sqrt{\gamma p/\rho}$.

Table 1 shows differences between Jacobian matrices computed using automatic differentiation and using formulas in Theorems 2.1 and 5.1. Discretization matrices of size 25×25 and corresponding solution vectors were generated randomly from a normal distribution. For the shallow water and Euler equations, positive solution values (e.g., water height for shallow water, density and pressure for Euler) were generated from a uniform distribution over $(0, 1)$. For Jacobians of dissipative fluxes, the normal vector is taken to be a random unit vector. In all cases, the difference close to machine precision.

Burgers'	Shallow water	Euler	LF (Burgers)	LF (SWE)	LF (Euler)
1.56616230e-15	9.17858305e-13	2.62285783e-14	2.03313333e-14	3.05403043e-12	5.04444613e-14

Table 1: Computed differences between Jacobian matrices of $\mathbf{r}(\mathbf{u})$ (measured in the Frobenius norm) when computed using AD and formulas from Theorems 2.1 and 5.1. LF refers to the "Lax-Friedrichs" flux.

A Julia code which reproduces these results is included in the Supplementary Materials.

7.2. Comparisons of computational cost

We first compare the cost of computing the Jacobian matrix using the formulas in this paper to other approaches. All computations are performed on a 2019 Macbook Pro with a 2.3 GHz 8-Core Intel Core i9 processor using Julia version 1.4 and all timings are computed using the `BenchmarkTools.jl` package [48].

The cost of forward-mode automatic differentiation is known to be minimal for functions with low-dimensional inputs and outputs [26]. To give a sense for the efficiency of AD in Julia, we compare the cost of evaluating a flux function $f_S(u_L, u_R)$ to the cost of computing its derivative using `ForwardDiff.jl` for 10000 random values of u_L, u_R. The evaluation of the entropy conservative Burgers' flux takes 7.087 microseconds, while the derivative takes 7.063 microseconds to evaluate. The logarithmic mean takes 129.254 microseconds to evaluate, while its derivative takes 161.322 microseconds to evaluate. The cost of computing Jacobians using `ForwardDiff.jl` scales similarly.

Next, we compare the cost of computing both the full Jacobian and a Jacobian-vector product using the formulas in Theorem 2.1 and competing approaches. Let $f_S(u_L, u_R)$ denote the scalar flux Burgers' flux, and define

$$\mathbf{r}(\mathbf{u}) = (\mathbf{Q} \circ \mathbf{F})\,\mathbf{1}, \qquad \mathbf{F}_{ij} = f_S(\mathbf{u}_i, \mathbf{u}_j), \tag{21}$$

where $\mathbf{Q} \in \mathbb{R}^{N,N}$ is a dense randomly generated skew-symmetric matrix. We compute the Jacobian matrix using the formula from Theorem 2.1 (referred to as "Formula from Theorem 2.1" in Table 2), with $\frac{\partial f_S}{\partial u_R}$ computed using both the analytical formula and automatic differentiation, which are tagged as "(analytic)" and "(AD)" in Table 2. We also compute the full Jacobian matrix directly using `ForwardDiff.jl` (referred to as "Automatic differentiation" in Table 2). We also compute the Jacobian matrix using the `FiniteDiff.jl` toolkit within the `DifferentialEquations.jl` framework [49], which computes the Jacobian matrix efficiently using cached in-place function evaluations and finite difference approximations (referred to as "finite differences" in Table 2). Finally, we provide timings for evaluating $\mathbf{r}(\mathbf{u})$ for reference. Implementations of both $\mathbf{r}(\mathbf{u})$ and its Jacobian are optimized for performance in Julia.[3] We have included code to compute timings in the Supplementary Materials.

We observe that the cost of evaluating the full Jacobian matrix using the formula of Theorem 2.1 is 1-2 orders of magnitude less expensive than automatic differentiation or finite differences applied directly to the nonlinear term $\mathbf{r}(\mathbf{u})$. These results highlight the fact that Theorem 2.1 allows one to take advantage of the Hadamard product and symmetry/skew-symmetry, which is difficult to do when directly applying automatic differentiation.

[3]In our implementations of the evaluation of $\mathbf{r}(\mathbf{u})$ and the Jacobian $\frac{\partial \mathbf{r}}{\partial \mathbf{u}}$ (computed using Theorem 2.1), we pre-allocate all output vectors and matrices for efficiency. For the implementation of $\frac{\partial \mathbf{r}}{\partial \mathbf{u}}$, we compute the sum $(\mathbf{Q} \circ \mathbf{F})\,\mathbf{1}$ by looping over rows of $\boldsymbol{Q}$ and accumulating contributions from $\mathbf{Q} \circ \mathbf{F}$ column-by-column. We access entries of $\mathbf{Q}^T$ to take advantage of the column-major storage of matrices in Julia.

	N = 10	N = 25	N = 50
Automatic differentiation	3.160	26.386	166.689
Finite differences	1.536	17.397	129.510
Formula from Theorem 2.1 (analytic)	.125	.628	2.357
Formula from Theorem 2.1 (AD)	.128	.628	2.530
Evaluation of $\mathbf{r}(\mathbf{u})$ (for reference)	.129	.623	2.517

Table 2: Timings for the computation of $\mathbf{r}(\mathbf{u})$ in (21) and various methods of computing the full Jacobian $\frac{d\mathbf{r}}{d\mathbf{u}}$ using the scalar Burgers' flux $f_S(u_L, u_R) = (u_L^2 + u_L u_R + u_R^2)/6$ (times in microseconds).

Because the number of flux evaluations required to evaluate $\mathbf{r}(\mathbf{u})$ is comparable to the number of AD function evaluations required to evaluate the Jacobian matrix, the cost of evaluating the full Jacobian matrix is proportional to the cost of directly evaluating $\mathbf{r}(\mathbf{u})$. Here, the constant of proportionality is roughly equal to the ratio of the cost of evaluating the flux function derivative (or Jacobian) and the cost of directly evaluating the flux function. This ratio of this cost will vary depending on the specific flux and the implementation. For the entropy conservative fluxes for the two-dimensional compressible Euler equations [47], the cost of evaluating the flux Jacobian matrix is only 1.625 times more expensive than directly evaluating the flux (both the Jacobian matrix and the flux were evaluated only for a single coordinate direction).

Finally, we note that if the Jacobian matrix is not explicitly required, Jacobian-vector products can be evaluated in a matrix-free fashion using either forward mode AD [26] or finite difference approximations [20] at much lower computational cost. The formulas in Theorem 2.1 can still be applied in a matrix-free fashion, but it is unclear if there are computational advantages over AD for computing Jacobian-vector products.

8. Conclusion

In this work, we derive efficient formulas for Jacobian matrices resulting from entropy conservative and entropy stable schemes based on flux differencing and summation-by-parts operators. These formulas are given in terms of summation-by-parts matrices and derivatives of flux functions, the latter of which can be computed efficiently using automatic differentiation. The computation of Jacobians using these formulas is significantly faster than directly computing Jacobian matrices using automatic differentiation, especially for dense operators. Future work will investigate the application of such formulas towards preconditioners and sensitivity analysis.

9. Acknowledgments

The authors gratefully acknowledge support from the National Science Foundation under award DMS-CAREER-1943186. Christina Taylor also acknowledges support from the Ken Kennedy Institute 2019-2020 BP Graduate Fellowship.

References

[1] P-O Persson and Jaime Peraire. Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 30(6):2709–2733, 2008.

[2] Stefan Ulbrich. A sensitivity and adjoint calculus for discontinuous solutions of hyperbolic conservation laws with source terms. *SIAM journal on control and optimization*, 41(3):740–797, 2002.

[3] Max D Gunzburger. *Perspectives in flow control and optimization*, volume 5. Siam, 2003.

[4] Stanislav N Kružkov. First order quasilinear equations in several independent variables. *Mathematics of the USSR-Sbornik*, 10(2):217, 1970.

[5] Thomas JR Hughes, LP Franca, and M Mallet. A new finite element formulation for computational fluid dynamics: I. Symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics. *Computer Methods in Applied Mechanics and Engineering*, 54(2):223–234, 1986.

[6] Tianheng Chen and Chi-Wang Shu. Entropy stable high order discontinuous Galerkin methods with suitable quadrature rules for hyperbolic conservation laws. *Journal of Computational Physics*, 345:427–461, 2017.

[7] Michael S Mock. Systems of conservation laws of mixed type. *Journal of Differential equations*, 37(1):70–88, 1980.

[8] Amiram Harten. On the symmetric form of systems of conservation laws with entropy. *Journal of computational physics*, 49(1):151–164, 1983.

[9] 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, 2014.

[10] Gregor J Gassner, Andrew R Winters, and David A Kopriva. Split form nodal discontinuous Galerkin schemes with summation-by-parts property for the compressible Euler equations. *Journal of Computational Physics*, 327:39–66, 2016.

[11] Jared Crean, Jason E Hicken, David C Del Rey Fernández, David W Zingg, and Mark H Carpenter. Entropy-stable summation-by-parts discretization of the Euler equations on general curved elements. *Journal of Computational Physics*, 356:410–438, 2018.

[12] Jesse Chan. On discretely entropy conservative and entropy stable discontinuous Galerkin methods. *Journal of Computational Physics*, 362:346 – 374, 2018.

[13] Matteo Parsani, Mark H Carpenter, Travis C Fisher, and Eric J Nielsen. Entropy Stable Staggered Grid Discontinuous Spectral Collocation Methods of any Order for the Compressible Navier–Stokes Equations. *SIAM Journal on Scientific Computing*, 38(5):A3129–A3162, 2016.

[14] David C Del Rey Fernández, Jared Crean, Mark H Carpenter, and Jason E Hicken. Staggered-grid entropy-stable multidimensional summation-by-parts discretizations on curvilinear coordinates. *Journal of Computational Physics*, 392:161–186, 2019.

[15] Andrew R Winters, Rodrigo C Moura, Gianmarco Mengaldo, Gregor J Gassner, Stefanie Walch, Joaquim Peiro, and Spencer J Sherwin. A comparative study on polynomial dealiasing and split form discontinuous Galerkin schemes for under-resolved turbulence computations. *Journal of Computational Physics*, 372:1–21, 2018.

[16] Diego Rojas, Radouan Boukharfane, Lisandro Dalcin, David C Fernandez, Hendrik Ranocha, David E Keyes, and Matteo Parsani. On the robustness and performance of entropy stable discontinuous collocation methods for the compressible Navier-Stokes equations. *arXiv preprint arXiv:1911.10966*, 2019.

[17] Eitan Tadmor. The numerical viscosity of entropy stable schemes for systems of conservation laws. I. *Mathematics of Computation*, 49(179):91–103, 1987.

[18] Lucas Friedrich, Gero Schnücke, Andrew R Winters, David C Del Rey Fernández, Gregor J Gassner, and Mark H Carpenter. Entropy stable space–time discontinuous Galerkin schemes with summation-by-parts property for hyperbolic conservation laws. *Journal of Scientific Computing*, 80(1):175–222, 2019.

[19] Jason E Hicken. Entropy-stable, high-order summation-by-parts discretizations without interface penalties. *Journal of Scientific Computing*, 82(2):50, 2020.

[20] Dana A Knoll and David E Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[21] Philipp Birken, Gregor J Gassner, and Lea M Versbach. Subcell finite volume multigrid preconditioning for high-order discontinuous Galerkin methods. *International Journal of Computational Fluid Dynamics*, pages 1–9, 2019.

[22] David C Fernandez, Mark H Carpenter, Lisandro Dalcin, Stefano Zampini, and Matteo Parsani. Entropy Stable h/p-Nonconforming Discretization with the Summation-by-Parts Property for the Compressible Euler and Navier-Stokes Equations. *arXiv preprint arXiv:1910.02110*, 2019.

[23] Johnathon Upperman and Nail K Yamaleev. Entropy stable artificial dissipation based on Brenner regularization of the Navier-Stokes equations. *Journal of Computational Physics*, 393:74–91, 2019.

[24] Farzad Ismail and Philip L Roe. Affordable, entropy-consistent Euler flux functions II: Entropy production at shocks. *Journal of Computational Physics*, 228(15):5410–5436, 2009.

[25] Andrew R Winters, Christof Czernik, Moritz B Schily, and Gregor J Gassner. Entropy stable numerical approximations for the isothermal and polytropic Euler equations. *BIT Numerical Mathematics*, pages 1–34, 2019.

[26] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*, volume 105. SIAM, 2008.

[27] J. Revels, M. Lubin, and T. Papamarkou. Forward-Mode Automatic Differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016.

[28] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[29] Thomas F Coleman and Arun Verma. The efficient computation of sparse Jacobian matrices using automatic differentiation. *SIAM Journal on Scientific Computing*, 19(4):1210–1233, 1998.

[30] Jesse Chan. Entropy stable reduced order modeling of nonlinear conservation laws. *arXiv preprint arXiv:1909.09103*, 2019.

[31] H-O Kreiss and Godela Scherer. Finite element and finite difference methods for hyperbolic partial differential equations. In *Mathematical aspects of finite elements in partial differential equations*, pages 195–212. Elsevier, 1974.

[32] Mark H Carpenter, Jan Nordström, and David Gottlieb. A stable and conservative interface treatment of arbitrary spatial accuracy. *Journal of Computational Physics*, 148(2):341–365, 1999.

[33] David A Kopriva. *Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers*. Springer Science & Business Media, 2009.

[34] Gregor J Gassner. A skew-symmetric discontinuous Galerkin spectral element discretization and its relation to SBP-SAT finite difference methods. *SIAM Journal on Scientific Computing*, 35(3):A1233–A1253, 2013.

[35] Jesse Chan. Skew-Symmetric Entropy Stable Modal Discontinuous Galerkin Formulations. *Journal of Scientific Computing*, 81(1):459–485, Oct 2019.

[36] Jesse Chan, David C Del Rey Fernández, and Mark H Carpenter. Efficient entropy stable Gauss collocation methods. *SIAM Journal on Scientific Computing*, 41(5):A2938–A2966, 2019.

[37] Gregor J Gassner, Andrew R Winters, Florian J Hindenlang, and David A Kopriva. The BR1 scheme is stable for the compressible Navier–Stokes equations. *Journal of Scientific Computing*, pages 1–47, 2017.

[38] Andrew R Winters, Dominik Derigs, Gregor J Gassner, and Stefanie Walch. A uniquely defined entropy stable matrix dissipation operator for high Mach number ideal MHD and compressible Euler simulations. *Journal of Computational Physics*, 332:274–289, 2017.

[39] Per-Olof Persson and Jaime Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. *AIAA*, 112, 2006.

[40] David C Del Rey Fernández, Jason E Hicken, and David W Zingg. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations. *Computers & Fluids*, 95:171–196, 2014.

[41] Hendrik Ranocha. Generalised summation-by-parts operators and variable coefficients. *Journal of Computational Physics*, 362:20 – 48, 2018.

[42] Jared Crean, Jason E Hicken, David C Del Rey Fernández, David W Zingg, and Mark H Carpenter. High-Order, Entropy-Stable Discretizations of the Euler Equations for Complex Geometries. In *23rd AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, 2017.

[43] Tianheng Chen and Chi-Wang Shu. Review of entropy stable discontinuous Galerkin methods for systems of conservation laws on unstructured simplex meshes, 2019. Accessed July 25, 2019.

[44] Siavosh Shadpey and David W Zingg. Energy-and Entropy-Stable Multidimensional Summation-by-Parts Discretizations on Non-Conforming Grids. In *AIAA Aviation 2019 Forum*, page 3204, 2019.

[45] Ulrik S Fjordholm, Siddhartha Mishra, and Eitan Tadmor. Well-balanced and energy stable schemes for the shallow water equations with discontinuous topography. *Journal of Computational Physics*, 230(14):5587–5609, 2011.

[46] Niklas Wintermeyer, Andrew R Winters, Gregor J Gassner, and David A Kopriva. An entropy stable nodal discontinuous Galerkin method for the two dimensional shallow water equations on unstructured curvilinear meshes with discontinuous bathymetry. *Journal of Computational Physics*, 340:200–242, 2017.

[47] Praveen Chandrashekar. Kinetic energy preserving and entropy stable finite volume schemes for compressible Euler and Navier-Stokes equations. *Communications in Computational Physics*, 14(5):1252–1286, 2013.

[48] Jiahao Chen and Jarrett Revels. Robust benchmarking in noisy environments. *arXiv e-prints*, Aug 2016.

[49] Christopher Rackauckas and Qing Nie. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, 5(1), 2017.

[50] Jesse Chan and Lucas C Wilcox. Discretely entropy stable weight-adjusted discontinuous Galerkin methods on curvilinear meshes. *Journal of Computational Physics*, 378:366 – 393, 2019.

[51] Jason E Hicken, David C Del Rey Fernández, and David W Zingg. Multidimensional summation-by-parts operators: general theory and application to simplex elements. *SIAM Journal on Scientific Computing*, 38(4):A1935–A1958, 2016.

[52] PD Thomas and CK Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA journal*, 17(10):1030–1037, 1979.

[53] Robert PK Chan and Angela YJ Tsai. On explicit two-derivative Runge-Kutta methods. *Numerical Algorithms*, 53(2-3):171–194, 2010.

[54] Andrew J Christlieb, Sigal Gottlieb, Zachary Grant, and David C Seal. Explicit strong stability preserving multistage two-derivative time-stepping schemes. *Journal of Scientific Computing*, 68(3):914–942, 2016.

[55] T Warburton and Jan S Hesthaven. On the constants in *hp*-finite element trace inverse inequalities. *Computer methods in applied mechanics and engineering*, 192(25):2765–2773, 2003.

[56] Jesse Chan, Zheng Wang, Axel Modave, Jean-Francois Remacle, and T Warburton. GPU-accelerated discontinuous Galerkin methods on hybrid meshes. *Journal of Computational Physics*, 318:142–168, 2016.

[57] H Xiao and Zydrunas Gimbutas. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Comput. Math. Appl.*, 59:663–676, 2010.

[58] Timothy J Barth. Numerical methods for gasdynamic systems on unstructured meshes. In *An introduction to recent developments in theory and numerics for conservation laws*, pages 195–285. Springer, 1999.

[59] Hendrik Ranocha. Comparison of some entropy conservative numerical fluxes for the Euler equations. *Journal of Scientific Computing*, 76(1):216–242, 2018.

Appendix A. Higher-dimensional domains and curved elements

The generalization to higher dimensional domains and curved geometric mappings is straightforward, but notationally more complicated. The construction of skew-symmetric SBP matrices on curved meshes follows from approaches detailed in [9, 11, 12, 50, 36, 35, 19], which are summarized here.

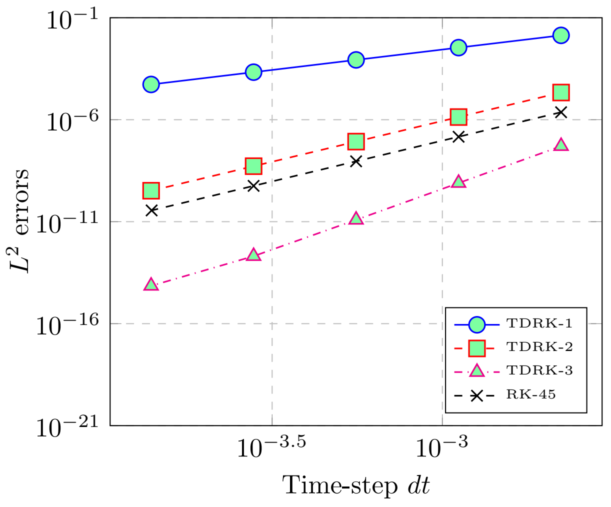
Let $\boldsymbol{x}, \widehat{\boldsymbol{x}} \in \mathbb{R}^d$ denote d-dimensional physical and reference coordinates, respectively. Let $\widehat{\mathbf{Q}}_j$ denote the reference SBP operator corresponding to differentiation with respect to $\widehat{x}_j$ which satisfies the SBP property (5). This operator can be constructed any number of ways: using the tensor product of 1D SBP operators [9], multi-dimensional SBP operators [51], or hybridized SBP operators [12, 35]. Consider now a curved element D^k which is the image of a reference element under some differentiable mapping such that for $\boldsymbol{x} \in D^k$, $\boldsymbol{x} = \boldsymbol{\Phi}(\widehat{\boldsymbol{x}})$. Then, derivatives with respect to physical coordinates can be computed via the chain rule $\frac{\partial u}{\partial x_i} = \sum_{j=1}^d \frac{\partial u}{\partial \widehat{x}_j}\frac{\partial \widehat{x}_j}{\partial x_i}$. For geometric terms which satisfy a discrete geometric conservation law (GCL) [52, 11, 50], we can further manipulate the chain rule to show that

$$\frac{\partial u}{\partial x_i} = \frac{1}{2}\sum_{j=1}^d \left(\frac{\partial u}{\partial \widehat{x}_j}\frac{\partial \widehat{x}_j}{\partial x_i} + \frac{\partial}{\partial \widehat{x}_j}\left(u\frac{\partial \widehat{x}_j}{\partial x_i}\right)\right).$$

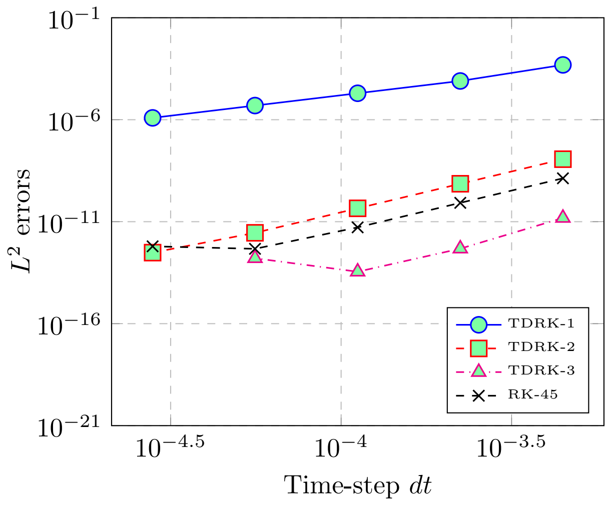
We will construct a physical SBP operator by mimicking this form of the chain rule. Let J denote the determinant of the Jacobian of $\boldsymbol{\Phi}$. Define the scaled geometric terms $\boldsymbol{g}_{ij} = J\frac{\partial \widehat{x}_j}{\partial x_i}$, and let $\mathbf{g}_{ij}$ denote the vector containing values of $\boldsymbol{g}_{ij}$ evaluated at nodal points. Define the physical SBP operator $\mathbf{Q}_i$ as

$$\mathbf{Q}_i = \frac{1}{2}\sum_{j=1}^d \left(\operatorname{diag}\left(\mathbf{g}_{ij}\right)\widehat{\mathbf{Q}}_j + \widehat{\mathbf{Q}}_j \operatorname{diag}\left(\mathbf{g}_{ij}\right)\right).$$

Then, one can show (using relationships between geometric terms $\boldsymbol{g}_{ij}$ and reference/physical normals) that $\mathbf{Q}_i$ satisfies a physical SBP property $\mathbf{Q}_i + \mathbf{Q}_i^T = \mathbf{B}_i$, where $\mathbf{B}_i$ is a diagonal matrix whose entries consist of values (at face nodes) of the ith component of the outward normal scaled by the surface Jacobian and surface quadrature weights [11, 50]. Given connectivity maps between face nodes on different elements, the physical SBP operators $\mathbf{Q}_i$ and $\mathbf{B}_i$ can then be used to construct global SBP operators analogous to (8) in two and three dimensions.



(a) Burgers' equation



(b) Shallow water equations

Figure B.2: L^2 errors for manufactured solutions of the Burgers and shallow water equations for three TDRK schemes under various time-step sizes. Errors for RK-45 are also included for reference.

Appendix B. Two-derivative time-stepping methods

Consider a general system of ODEs

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{f}(\mathbf{u}).$$

Two-derivative explicit time-stepping methods are constructed based on the assumption that second derivatives of $\mathbf{u}$ in time are available [53, 54]. The resulting schemes can achieve higher order accuracy with fewer stages and function evaluations compared to standard Runge-Kutta methods.

Let $\mathbf{g}(\mathbf{u})$ denote the second derivative of $\mathbf{u}$ in time

$$\mathbf{g}(\mathbf{u}) = \frac{\mathrm{d}^2\mathbf{u}}{\mathrm{d}t^2} = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{f}(\mathbf{u}) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\mathbf{f}(\mathbf{u}),$$

where we have used the chain rule in the final step. The simplest two-derivative Runge-Kutta method is the one-stage second order scheme [53]

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \Delta t\mathbf{f}(\mathbf{u}^k) + \frac{\Delta t^2}{2}\mathbf{g}(\mathbf{u}^k),$$

where $\mathbf{u}^k$ denotes the solution at the kth time-step. We examine the one-stage, two-stage, and three-stage two-derivative Runge Kutta given in [53], which we refer to as TDRK-1, TDRK-2, TDRK-3.[4] These schemes are second, fourth, and fifth order accurate, respectively. We also provide reference results using a low-storage 4th order 5-stage Runge-Kutta method (RK-45).

We examine the performance of two-derivative time-stepping methods for the one-dimensional Burgers' and shallow water equations using an entropy conservative and entropy stable spectral (Lobatto) collocation method of degree $N = 40$ on a single periodic domain $[-1, 1]$. For the entropy stable scheme, we apply a local Lax-Friedrichs penalty at the boundaries to produce entropy dissipation. We compute L^2 errors for a

[4]Five different three-stage schemes are presented in [53]. We use the scheme corresponding to free parameter $c_3 = 2/3$, which the authors report as the best performing three-stage scheme.

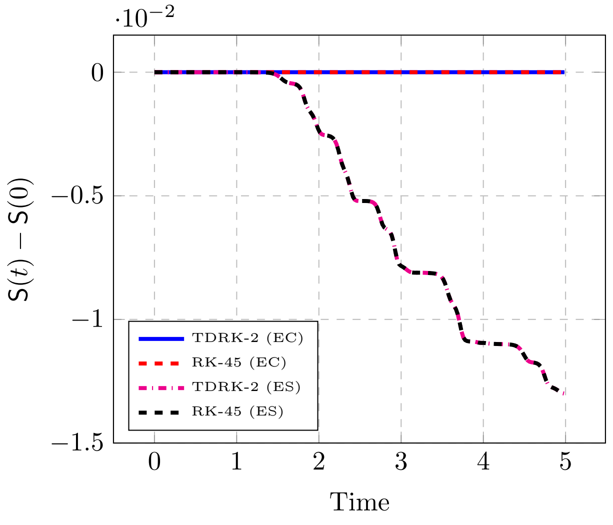
dt/dt_0	1/2	1/4	1/8	1/16
TDRK-1	1.997	1.999	2.000	2.000
TDRK-2	3.999	4.000	4.000	4.000
TDRK-3	6.006	5.993	*5.916*	*4.842*

(a) Burgers equation

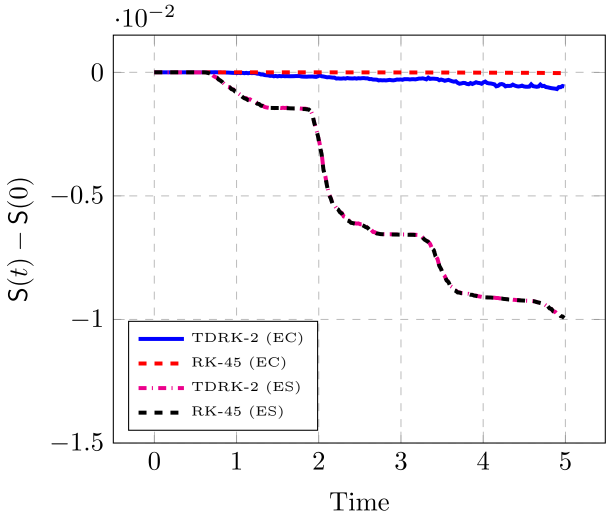
dt/dt_0	1/2	1/4	1/8	1/16
TDRK-1	2.620	2.003	2.001	2.001
TDRK-2	4.002	4.001	4.001	4.000
TDRK-3	4.998	*3.757*	*-2.193*	

(b) Shallow water equation

Table B.3: Computed rates of convergence with respect to dt for different TDRK schemes (dt_0 denotes the initial time-step). Italicized numbers denote rates which are likely affected by numerical roundoff.



(a) Burgers' equation



(b) Shallow water equations

Figure B.3: Evolution of entropy over time for TDRK-2 and RK-45 schemes using both entropy conservative (EC) and entropy stable (ES) spectral collocation formulations of the Burgers' and shallow water equations.

manufactured solution where all solution components have the form $\sin(kt)\sin(\pi x)$ with $k = 100$. TDRK methods also require derivatives of source terms $f(x,t)$ associated with manufactured solutions, which we compute analytically.

Figure B.2 plots L^2 errors (computed using a higher accuracy Gaussian quadrature rule at final time $T = 5$) against the time-step size, while Table B.3 shows computed rates of convergence for each TDRK scheme. We observe that all except one TDRK scheme achieves the expected rate of convergence up until the point at which errors are affected by numerical roundoff. The outlier is the TDRK-3 scheme, which converges at the expected rate of $O(dt^5)$ for the shallow water equations, but achieves a higher $O(dt^6)$ rate of convergence for the Burgers' equation. We also observe that the 4th order RK-45 scheme is slightly more accurate than the 4th order TDRK-2 scheme. As noted in [53], the 2-stage TDRK-2 scheme requires only one evaluation of $\mathbf{f}(\mathbf{u})$ and two evaluations of $\mathbf{g}(\mathbf{u})$. However, when $\mathbf{g}(\mathbf{u})$ is computed using a Jacobian-vector product, this corresponds to two evaluations of $\mathbf{f}(\mathbf{u})$ and two Jacobian-vector products. Because evaluating Jacobian-vector products are at least as expensive as evaluating $\mathbf{f}(\mathbf{u})$, it is not clear that the TDRK-2 scheme would be more efficient than either RK-45 or the standard 4-stage 4th order Runge-Kutta method in practice.

Finally, we plot the integrated entropy $\mathsf{S}(t) = \int_\Omega S(\boldsymbol{u})\,\mathrm{d}x$ over time in Figure B.3. For Burgers' equation, we do not observe significant differences in the entropy dissipation for TDRK-2 and RK-45 schemes. For the entropy conservative formulation of the shallow water equations, the TDRK-2 scheme produces slightly more entropy dissipation than RK-45; however, both two schemes produce similar entropy dissipation for the entropy stable formulation.

Appendix C. Time-implicit discretizations on triangular meshes

Jacobian matrices also appear in time-implicit discretizations of nonlinear ODEs. Consider the implicit midpoint rule

$$\mathbf{u}^{k+1} = \mathbf{u}^k - \Delta t \mathbf{r}\left(\frac{\mathbf{u}^{k+1} + \mathbf{u}^k}{2}\right).$$

This can be rewritten in the following form where $\mathbf{u}^{k+1/2} = \frac{\mathbf{u}^{k+1}+\mathbf{u}^k}{2}$

$$\mathbf{u}^{k+1/2} = \mathbf{u}^k - \frac{\Delta t}{2}\mathbf{r}\left(\mathbf{u}^{k+1/2}\right)$$
$$\mathbf{u}^{k+1} = 2\mathbf{u}^{k+1/2} - \mathbf{u}^k.$$

Solving for $\mathbf{u}^{k+1/2}$ is a nonlinear equation and can be done via Newton's method

$$\mathbf{u}^{k+1/2,\ell+1} = \mathbf{u}^{k+1/2,\ell} - \left(\mathbf{I} + \frac{\Delta t}{2}\left.\frac{\partial \mathbf{r}}{\partial \mathbf{u}}\right|_{\mathbf{u}^{k+1/2,\ell}}\right)^{-1}\left(\mathbf{u}^{k+1/2,\ell} + \frac{\Delta t}{2}\mathbf{r}\left(\mathbf{u}^{k+1/2,\ell}\right) - \mathbf{u}^k\right).$$

where ℓ denotes the Newton iteration index.

All linear systems are solved using Julia's sparse direct solver. We utilize a relative tolerance of $1e-11$ for the Newton iteration, and determine the time-step Δt using the following estimate

$$\Delta t = \text{CFL} \times \frac{h_{\min}}{C_N}, \qquad C_N = \frac{(N+1)(N+2)}{2},$$

where CFL is the CFL constant, $h_{\min}$ is the size of the smallest element in the mesh, and C_N is the N-dependent trace constant for a degree N polynomial space on the reference triangle [55].[5]

Appendix C.0.1. 2D Burgers' equation

We consider energy conservative and energy stable discretizations of 2D Burgers' equation

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = 0$$

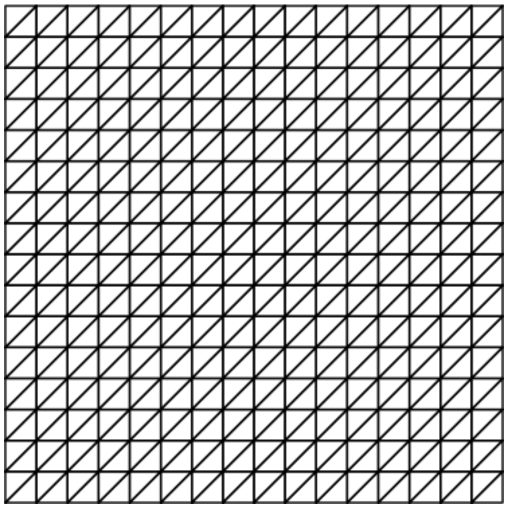
with periodic boundary conditions on the domain $[-1,1]^2$. For the initial condition $u(\boldsymbol{x},0) = -\sin(\pi x)$, the solution forms a shock around $T = 1/2$.

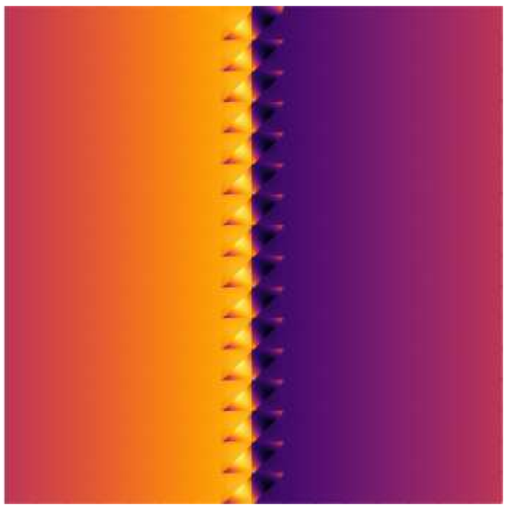
We discretize the Burgers' equation using an energy conservative (or stable) scheme in space [12, 35] and an implicit midpoint discretization in time. The spatial discretization utilizes a degree N polynomial space, degree $2N$ volume quadrature, and an $(N+1)$-point Gauss quadrature for faces. An energy stable scheme is constructed by adding a local Lax-Friedrichs penalization term, $-\frac{\lambda}{2}[\![u]\!]$, to the energy conservative flux contribution, where $\lambda = \max\left(|u^+|, |u|\right)$ is the maximum wavespeed at an interface. We utilize both uniform and "squeezed" triangular meshes with very small elements (see Figure C.4). This mesh is constructed by taking the x-coordinate x_i of vertices in a uniform triangular mesh (constructed by bisecting each element in a uniform mesh quadrilateral mesh) and transforming them via $x_i - .3\sin(\pi x_i)$ to produce a new mesh.

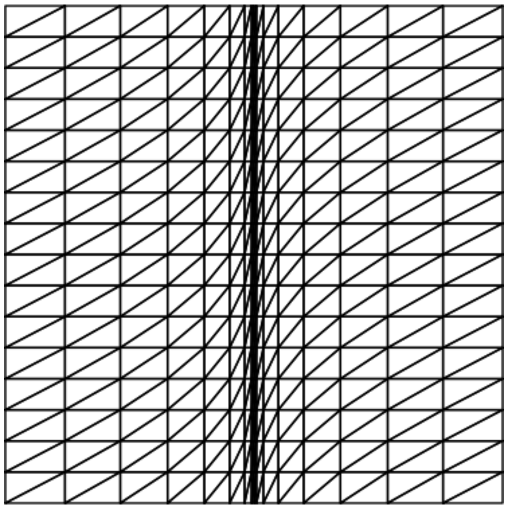
Since the implicit midpoint rule is a symplectic integrator, we expect energy to be conserved up to machine precision for an energy conservative scheme. We set the initial condition randomly, remove the Lax-Friedrichs penalization, and run until time $T = 1$ using a CFL of 10 on both uniform and squeezed 8×8 meshes with $N = 2$. For the uniform mesh, the total change in energy was $-2.665e-15$. The squeezed mesh behaved similarly, with a total change in energy of $3.553e-15$.

Next, we add local Lax-Friedrichs dissipation and run with the initial condition $-\sin(\pi x)$ until time $T = 1$ using a CFL of 250. Figure C.4 shows solutions for both cases. In each case, oscillations appear in a

[5]Trace constants C_N for other element types are derived in [56].


(a) Uniform mesh


(b) Solution on a uniform mesh


(c) Anisotropic mesh

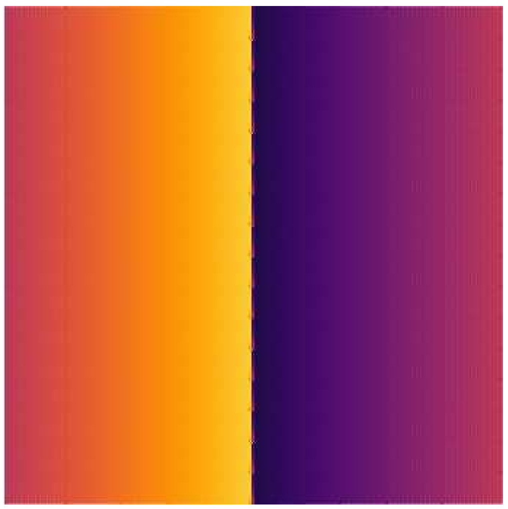

(d) Solution on an anisotropic mesh

Figure C.4: $N = 2$ solutions of Burgers' equation at $T = 1$ on uniform and anisotropic "squeezed" 16×16 meshes.

one-element vicinity around the shock. For both meshes, the Newton iteration converges in between 4 and 7 iterations. We note that for an entropy conservative scheme with a randomly generated initial condition, increasing the CFL further resulted in non-convergence of the Newton iteration. However, either switching to the initial condition $u(x, y, 0) = -\sin(\pi x)$ or adding local Lax-Friedrichs dissipation avoids stalling of the Newton iteration.

Appendix C.0.2. 2D compressible Euler equations

Finally, we consider a time-implicit discretization of the 2D compressible Euler equations

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x_1} + \frac{\partial(\rho v)}{\partial x_2} &= 0,\\
\frac{\partial \rho u}{\partial t} + \frac{\partial\left(\rho u^2 + p\right)}{\partial x_1} + \frac{\partial(\rho u v)}{\partial x_2} &= 0,\\
\frac{\partial \rho v}{\partial t} + \frac{\partial(\rho u v)}{\partial x_1} + \frac{\partial\left(\rho v^2 + p\right)}{\partial x_2} &= 0,\\
\frac{\partial E}{\partial t} + \frac{\partial(u(E+p))}{\partial x_1} + \frac{\partial(v(E+p))}{\partial x_2} &= 0,
\end{aligned}$$

Here, $\gamma = 1.4$, and $p = (\gamma - 1)\rho e$ is the pressure, where $\rho e = E - \frac{1}{2}\rho(u^2 + v^2)$ is the specific internal energy. We construct a scheme which is stable with respect to the unique entropy for the compressible Navier-Stokes equations [5]

$$S(\boldsymbol{u}) = -\frac{\rho s}{\gamma - 1}, \qquad \boldsymbol{u} = [\rho, \rho u, \rho v, E]^T,$$

where $s = \log\left(\frac{p}{\rho^\gamma}\right)$ denotes the specific entropy. Mappings between conservative variables $\boldsymbol{u}$ and entropy variables $\boldsymbol{v} = \{v_1, v_2, v_3, v_4\}$ in two dimensions are given by

$$\begin{aligned}
&v_1 = \frac{\rho e(\gamma + 1 - s) - E}{\rho e}, \qquad v_2 = \frac{\rho u}{\rho e}, \qquad v_3 = \frac{\rho v}{\rho e}, \qquad v_4 = -\frac{\rho}{\rho e}\\
&\rho = -(\rho e)v_4, \qquad \rho u = (\rho e)v_2, \qquad \rho v = (\rho e)v_3, \qquad E = (\rho e)\left(1 - \frac{v_2^2 + v_3^2}{2v_4}\right),
\end{aligned}$$

where ρe and s can be expressed in terms of the entropy variables as

$$\rho e = \left(\frac{(\gamma - 1)}{(-v_4)^\gamma}\right)^{1/(\gamma-1)} e^{\frac{-s}{\gamma-1}}, \qquad s = \gamma - v_1 + \frac{v_2^2 + v_3^2}{2v_4}.$$

We utilize the entropy conservative and kinetic energy preserving finite volume fluxes derived in [47], and apply entropy dissipation by adding a local Lax-Friedrichs penalization term, $-\frac{\lambda}{2}[\![\boldsymbol{u}]\!]$ [6, 50]. We compute at each point on an interface the local wavespeed $a = |\boldsymbol{u} \cdot \boldsymbol{n}| + c$, where $c = \sqrt{\gamma \rho / p}$ is the sound speed and $\boldsymbol{u} \cdot \boldsymbol{n}$ is the normal velocity. The local Lax-Friedrichs parameter is then computed via $\lambda = \sqrt{\frac{(a^+)^2 + a^2}{2}}$, where a^+, a are computed using the interior and exterior solution states, respectively.

We employ an entropy stable modal DG formulation from [50] on triangles using total degree N polynomials. The surface quadrature is constructed using 1D $(N + 1)$ point Gauss quadrature rules on each face and we use a volume quadrature [57] which is exact for degree $2N$ polynomials. Since this is a non-collocated formulation, we need the change of variables matrices $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}}, \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}}$ to evaluate (19). These matrices can be computed using automatic differentiation or using the explicit formulas [58]

$$
\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{v}} = \begin{bmatrix} \rho & \rho u & \rho v & E \\ \rho u & \rho u^2 + p & \rho u v & \rho u H \\ \rho v & \rho u v & \rho v^2 + p & \rho v H \\ E & \rho u H & \rho v H & \rho H^2 - c^2 \frac{p}{\gamma - 1} \end{bmatrix},
$$

$$
\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}} = -\frac{1}{\rho e v_4} \begin{bmatrix} \gamma + k^2 & k v_2 & k v_3 & (k+1) v_4 \\ k v_2 & v_2^2 - v_4 & v_2 v_3 & v_2 v_4 \\ k v_3 & v_2 v_3 & v_3^2 - v_4 & v_3 v_4 \\ (k+1) v_4 & v_2 v_4 & v_3 v_4 & v_4^2 \end{bmatrix}
$$

where c is the sound speed, $H = c^2/(\gamma - 1) + \frac{1}{2}(u^2 + v^2)$ is the enthalpy, and $k = \frac{1}{2}(v_2^2 + v_3^2)/v_4$.

There exist several choices for entropy conservative fluxes [24, 59, 47]. We utilize the the entropy conservative numerical fluxes given by Chandrashekar in [47]

$$
\begin{aligned}
f^1_{1,S}(\boldsymbol{u}_L, \boldsymbol{u}_R) &= \{\{\rho\}\}^{\log} \{\{u\}\}, & f^1_{2,S}(\boldsymbol{u}_L, \boldsymbol{u}_R) &= \{\{\rho\}\}^{\log} \{\{v\}\}, \\
f^2_{1,S}(\boldsymbol{u}_L, \boldsymbol{u}_R) &= f^1_{1,S} \{\{u\}\} + p_{\text{avg}}, & f^2_{2,S}(\boldsymbol{u}_L, \boldsymbol{u}_R) &= f^1_{2,S} \{\{u\}\}, \\
f^3_{1,S}(\boldsymbol{u}_L, \boldsymbol{u}_R) &= f^2_{2,S}, & f^3_{2,S}(\boldsymbol{u}_L, \boldsymbol{u}_R) &= f^1_{2,S} \{\{v\}\} + p_{\text{avg}}, \\
f^4_{1,S}(\boldsymbol{u}_L, \boldsymbol{u}_R) &= (E_{\text{avg}} + p_{\text{avg}}) \{\{u\}\}, & f^4_{2,S}(\boldsymbol{u}_L, \boldsymbol{u}_R) &= (E_{\text{avg}} + p_{\text{avg}}) \{\{v\}\},
\end{aligned}
$$

where the quantities $p_{\text{avg}}, E_{\text{avg}}, \|\boldsymbol{u}\|^2_{\text{avg}}$ are defined as

$$
p_{\text{avg}} = \frac{\{\{\rho\}\}}{2\{\{\beta\}\}}, \qquad E_{\text{avg}} = \frac{\{\{\rho\}\}^{\log}}{2\{\{\beta\}\}^{\log}(\gamma - 1)} + \frac{\|\boldsymbol{u}\|^2_{\text{avg}}}{2}, \qquad \beta = \frac{\rho}{2p},
$$
$$
\|\boldsymbol{u}\|^2_{\text{avg}} = 2(\{\{u\}\}^2 + \{\{v\}\}^2) - \left(\{\{u^2\}\} + \{\{v^2\}\}\right) = u^+ u + v^+ v,
$$

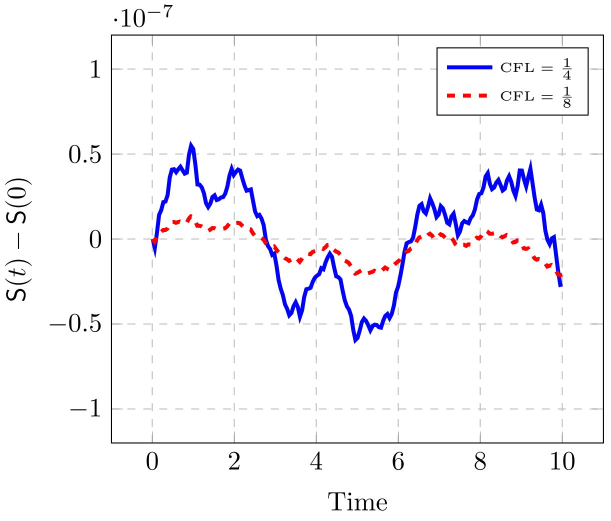
where $\{\{u\}\} = \frac{1}{2}(u^+ + u)$, where u^+, u denotes the exterior and interior states across the interface of an element D^k.

Let $\mathsf{S}(t) = \int_\Omega S(\boldsymbol{u}(\boldsymbol{x}, t))$ denote the total entropy in the domain Ω, where the integral is approximated using the same quadrature rule used to construct the DG mass matrix over each element. We begin by checking the change in entropy $\mathsf{S}(t) - \mathsf{S}(0)$ for an entropy conservative formulation. We utilize a discontinuous initial condition

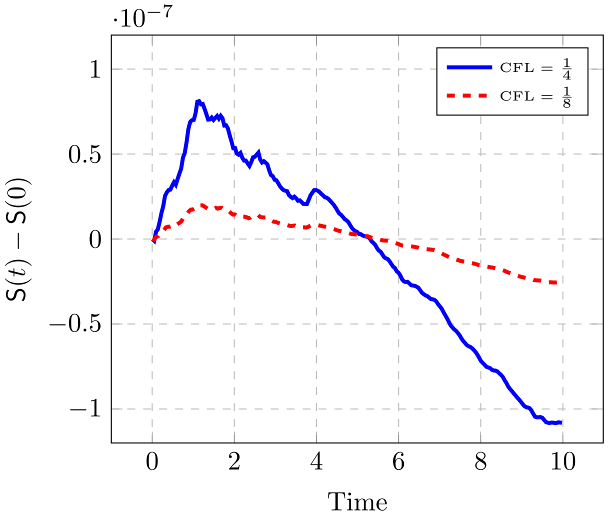
$$
\rho = \begin{cases} 1.1 & -.5 \leq x, y \leq .5 \\ 1 & \text{otherwise} \end{cases}, \qquad u, v = 0, \qquad E = \rho^\gamma.
$$

A triangular mesh is constructed by bisecting each element in a uniform mesh of 8×8 quadrilaterals, and the solution is evolved until final time $T = 10$. Figure C.5 shows the results for $N = 2$ and $N = 3$ for CFL $= \frac{1}{4}$ and CFL $= \frac{1}{8}$. We observe that halving the CFL reduces the change in entropy by a factor of 4, which corresponds to the second order time accuracy of the implicit midpoint rule. We also check the entropy dissipation for different CFL numbers in Figure C.6. Both $N = 2$ and $N = 3$ display similar results, with dissipation decreasing as the CFL increases. We also note that the number of Newton iterations remains relatively constant for different time-step sizes: for CFL $= .1$, Newton converged in $3 - 4$ iterations, for CFL $= 1$, Newton converged in $4 - 5$ iterations, and for CFL $= 10$, Newton converged in $5 - 6$ iterations. We also tried CFL $= 100$ over a longer time period, for which Newton also converged in $5 - 6$ iterations. However, for initial conditions with sufficiently large variations, Newton did not converge for CFL $= 100$.

Finally, we examine the behavior of the implicit midpoint method with respect to variations in element size. We use the isentropic vortex analytic solution (centered at $x = 0, y = 5$) on the domain $[-5, 5] \times [0, 20]$. We compute the L^2 error at time $T = 5$ for a uniform and "squeezed" anisotropic triangular mesh, both of which are constructed by bisecting each element of a uniform 24×16 quadrilateral mesh. Both cases use a degree $N = 3$ approximation and time-step of $dt = .1$, and Figure C.7 shows both DG solutions with the

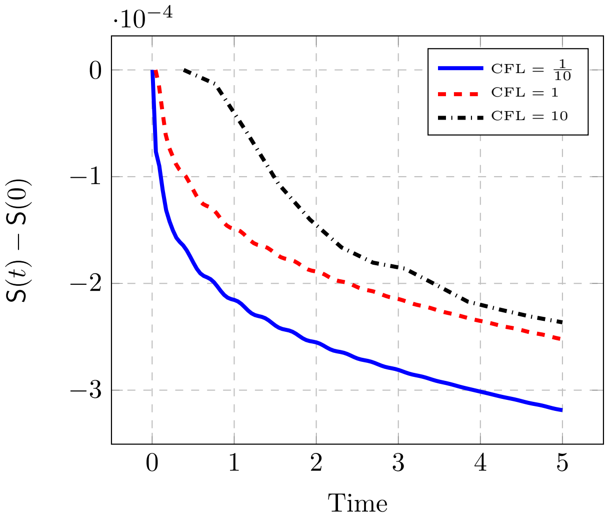


(a) $N = 2$

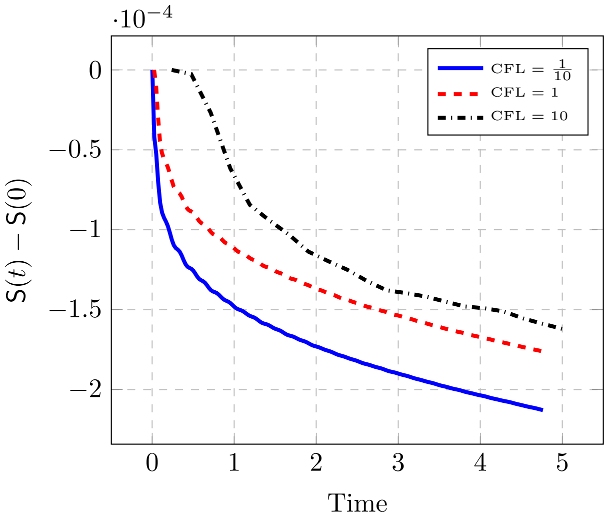


(b) $N = 3$

Figure C.5: Change in entropy over time for entropy conservative formulations of the compressible Euler equations and the implicit midpoint method.

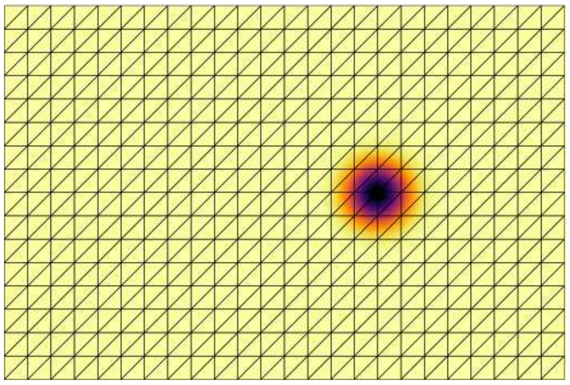


(a) $N = 2$



(b) $N = 3$

Figure C.6: Change in entropy over time for entropy stable formulations of the compressible Euler equations and the implicit midpoint method.


(a) Uniform mesh

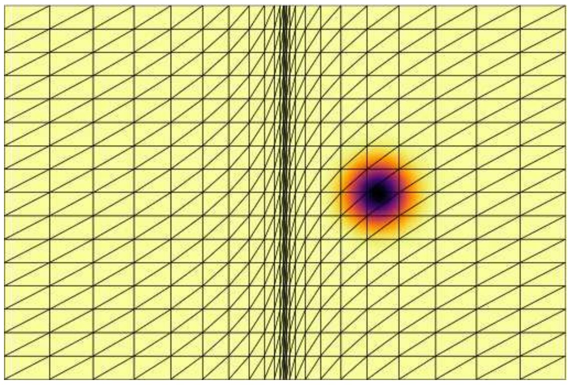

(b) Anisotropic mesh

Figure C.7: Isentropic vortex solutions at time $T = 5$ for $N = 3$ and $dt = .1$ on both uniform and "squeezed" anisotropic triangular meshes.

mesh overlaid. The L^2 errors for the isentropic vortex are 0.0901 and 0.0935 on the uniform and "squeezed" meshes, respectively, suggesting that implicit entropy stable formulations robustly handle settings where the maximum stable time-step for explicit methods is restricted by minimum element size.