AIAA SciTech
4-8 January 2016, San Diego, California, USA
54th AIAA Aerospace Sciences Meeting

AIAA 2016-1328

Investigation of Stabilization Methods for Multi-Dimensional Summation-by-parts Discretizations of the Euler Equations

Jared Crean[*], Kinshuk Panda[*], Anthony Ashley[*], and Jason E. Hicken[†]
Rensselaer Polytechnic Institute, Troy, New York, 12180

We present an extensible Julia-based solver for the Euler equations that uses a summation-by-parts (SBP) discretization on unstructured triangular grids. While SBP operators have been used for tensor-product discretizations for some time, they have only recently been extended to simplices. Here we investigate the accuracy and stability properties of simplex-based SBP discretizations of the Euler equations. Non-linear stabilization is a particular concern in this context, because SBP operators are nearly skew-symmetric. We consider an edge-based stabilization method, which has previously been used for advection-diffusion-reaction problems and the Oseen equations, and apply it to the Euler equations. Additionally, we discuss how the development of our software has been facilitated by the use of Julia, a new, fast, dynamic programming language designed for technical computing. By taking advantage of Julia's unique capabilities, code that is both efficient and generic can be written, enhancing the extensibility of the solver.

I. Introduction

High-order accurate discretizations have the potential to be more efficient than low-order methods in the numerical solution of partial differential equations [1, 2]. Among such methods, summation-by-parts (SBP) operators [3] are an attractive option for computational fluid dynamics (CFD), because they are high-order accurate and time stable for linear equations. However, classical SBP operators are one-dimensional finite-difference methods, so most of the previous work using high-order SBP operators has been limited to tensor-product discretizations on structured or multi-block grids [3–9].

Recently, Hicken, Del Rey Fernández, and Zingg [10] proposed a theoretical framework for multi-dimensional SBP operators that is suitable for discretizations on unstructured grids. The framework was illustrated by constructing SBP operators for triangular and tetrahedral elements. These simplex SBP operators produce discretizations that are similar to the mass-lumped spectral-element method [11, 12].

SBP-based discretizations, while time stable, may still suffer from aliasing errors that arise in the discretization of non-linear partial differential equations (PDEs). Stabilization methods to control these errors have been studied extensively over the last half century, particularly in the context of advection-dominated problems [13–17]. In this work, we focus on continuous SBP discretizations and draw upon stabilizations suitable for continuous Galerkin (CG) methods that are high-order accurate and locally conservative [18, 19].

Among the most popular stabilizations for finite-element methods is the streamwise-upwind Petrov-Galerkin (SUPG) method, which established the effectiveness and accuracy of artificial dissipation when consistency is maintained [14]. Although popular, it suffers from several limitations including non-physical coupling of velocity and pressure in the stabilization term and dual inconsistency [20].

[*]Graduate Student, Department of Mechanical, Aerospace, and Nuclear Engineering, Student Member AIAA
[†]Assistant Professor, Department of Mechanical, Aerospace, and Nuclear Engineering, Member AIAA

Instead, we investigate a stabilization based on penalizing jumps in the gradient, which has been shown to be effective, and in some cases optimal, in stabilizing advection-dominated problems [16]. The gradient-jump penalty of Burman and Hansbo was shown to bound the energy norm of the discrete solution of the convection-diffusion-reaction and Oseen equations [21, 22], and we adapt it here to the Euler equations.

In addition to studying stabilization of multi-dimensional SBP methods, one of our motivations for the present work is to develop a versatile multi-physics code for PDE-constrained optimization. PDE solvers used for optimization have several requirements beyond traditional analysis codes, most importantly the ability to calculate derivatives. To this end, we have implemented our solver in a new, fast, dynamic programming language called Julia, which enables the use of abstraction while retaining computational efficiency. We elaborate on the Julia implementation further below.

The remainder of the paper is organized as follows. The SBP discretization is discussed in Section II, and Section III describes edge stabilization. The use of abstractions in Julia and the computational benefits are detailed in Section IV. Numerical results for both steady and unsteady cases are given in Section V, and conclusions are provided in Section VI.

II. Summation-by-parts Discretization of the Euler Equations

Our discretizations use the multi-dimensional SBP simplex operators recently proposed in Ref. [10]. To keep the presentation self contained, the definition of a two-dimensional SBP first-derivative operator in the ξ direction is provided below for a generic reference element Ω^e. The definition for the derivative operator in the η direction is analogous and uses the same norm/mass matrix H. In the definition, we make use of the monomials $\mathcal{P}_k(\xi,\eta) \equiv \xi^i \eta^{j-i}$, where i, j, and k are related by $k = j(j+1)/2 + i + 1$, for all $j \in \{0, 1, \dots, p\}$ and $i \in \{0, 1, \dots, j\}$.

Definition 1. Two-dimensional summation-by-parts operator: *Consider an open and bounded domain $\Omega^e \subset \mathbb{R}^2$ with a piecewise-smooth boundary Γ^e. The matrix $\mathsf{D}_\xi \in \mathbb{R}^{n\times n}$ is a degree p SBP approximation to the first derivative $\frac{\partial}{\partial \xi}$ on the nodes $S = \{(\xi_i, \eta_i)\}_{i=1}^n$ if*

1. $\mathsf{D}_\xi \boldsymbol{p}_k = \boldsymbol{p}'_k, \qquad \forall\, k \in \{1, 2, \dots, (p+1)(p+2)/2\}$,
   where $\boldsymbol{p}_k \in \mathbb{R}^n$ and $\boldsymbol{p}'_k \in \mathbb{R}^n$ denote $\mathcal{P}_k$ and $\partial \mathcal{P}_k/\partial\xi$, respectively, evaluated at the nodes in S;
2. $\mathsf{D}_\xi = \mathsf{H}^{-1}\mathsf{S}_\xi$, *where H is symmetric positive-definite, and;*
3. $\mathsf{S}_\xi = \mathsf{Q}_\xi + \frac{1}{2}\mathsf{E}_\xi$, *where $\mathsf{Q}_\xi^T = -\mathsf{Q}_\xi$, $\mathsf{E}_\xi^T = \mathsf{E}_\xi$, and E_ξ satisfies*

$$\boldsymbol{p}_k^T \mathsf{E}_\xi \boldsymbol{p}_m = \oint_{\Gamma^e} \mathcal{P}_k \mathcal{P}_m n_\xi d\Gamma, \qquad \forall\, k, m \in \{1, 2, \dots, (\tau+1)(\tau+2)/2\},$$

   where $\tau \geq p$, and n_ξ is the ξ component of $\boldsymbol{n} = [n_\xi, n_\eta]^T$, the outward pointing unit normal on Γ^e.

The simplex-based SBP operators that we consider in this work were constructed with diagonal norm/mass matrices whose entries define a cubature rule with positive weights. Unlike most finite-difference methods, SBP discretizations approximate the weak form when integrated using their mass matrix. Unlike finite-element methods, SBP methods do not have unique shape functions. They only specify basis values and derivatives at the nodes (property 1 above) and require the matrix operators to obey properties 2 and 3.

In the following sections, we illustrate how multi-dimensional SBP operators are used to discretization the Euler equations, and we highlight the close connection between SBP discretizations and the finite-element method.

A. Conservative Variable Formulation

The two-dimensional Euler equations are discretized in space using SBP operators on a simplex mesh. The equations in conservation form are

$$\frac{\partial \boldsymbol{q}}{\partial t} + \nabla \cdot \boldsymbol{F} = \boldsymbol{S}, \tag{1}$$

where $\boldsymbol{q} = [\rho, \rho u, \rho v, E]^T$ denotes the conservative variables, $\boldsymbol{S}$ is the source term, and the Euler fluxes are

$$\boldsymbol{F} = \begin{bmatrix} \rho u & \rho v \\ \rho u^2 + p & \rho uv \\ \rho uv & \rho v^2 + p \\ (E+p)u & (E+p)v \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_x & \boldsymbol{F}_{\boldsymbol{y}} \end{bmatrix}.$$

The calorically perfect ideal gas law is used to close the system. The semi-linear weak form of the Euler equations is derived below, in order to illustrate some particular details. Readers familiar with the derivation can proceed directly to Equation (3).

Consider an arbitrary element Ω with boundary Γ, and let $\boldsymbol{x} = (x(\xi,\eta), y(\xi,\eta))$ be a mapping from the reference element Ω^e to Ω, where (ξ, η) are the reference element coordinates[a]. Let $J = \left(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}\right)$ be the mapping Jacobian, and let $|J|$ denote its determinant. Transforming the fluxes in Equation (1) to reference space and multiplying the entire equation by $\frac{1}{|J|}$, we obtain

$$\frac{1}{|J|}\frac{\partial \boldsymbol{q}}{\partial t} + \nabla_{\boldsymbol{\xi}} \cdot \boldsymbol{F}_{\boldsymbol{\xi}} = \frac{\boldsymbol{S}}{|J|}, \tag{2}$$

where the transformed fluxes are

$$\boldsymbol{F}_\xi \equiv \frac{1}{|J|}\left(\boldsymbol{F}_x \frac{\partial \xi}{\partial x} + \boldsymbol{F}_y \frac{\partial \xi}{\partial y}\right), \qquad \text{and} \qquad \boldsymbol{F}_\eta \equiv \frac{1}{|J|}\left(\boldsymbol{F}_x \frac{\partial \eta}{\partial x} + \boldsymbol{F}_y \frac{\partial \eta}{\partial y}\right).$$

If the metric invariants are satisfied — they are for the affine transformations considered here — the $\frac{1}{|J|}$ factor can be moved inside the divergence operator to maintain conservative form in reference space [23].

Next, we introduce a weighting function $\boldsymbol{w} \in [\mathcal{V}]^4$, where the weighting space $\mathcal{V}$ is an appropriate Hilbert space. Multiplying element-wise by the vector $\boldsymbol{w}$, integrating over the reference element Ω^e, and applying integration by parts to the flux term, yields

$$\int_{\Omega^e} \boldsymbol{w} \frac{\partial \boldsymbol{q}}{\partial t} \frac{1}{|J|} \mathrm{d}\Omega^e = \int_{\Omega^e} \boldsymbol{w}\boldsymbol{S}\frac{1}{|J|}\mathrm{d}\Omega^e + \int_{\Omega^e} \nabla_{\boldsymbol{\xi}} \boldsymbol{w} \cdot [\boldsymbol{F}_\xi, \boldsymbol{F}_\eta]\, \mathrm{d}\Omega^e - \int_{\Gamma^e} \boldsymbol{w}\, [\boldsymbol{F}_\xi, \boldsymbol{F}_\eta]\, \mathbf{d\Gamma},$$

where $\mathbf{d\Gamma} = \boldsymbol{n}\, \mathrm{d}\Gamma$ and $\boldsymbol{n} = (n_\xi, n_\eta)$ is the outward unit vector normal to the boundary Γ^e.

On each element we introduce a finite dimensional approximation $\boldsymbol{q} \approx N_b(\boldsymbol{\xi})\hat{\boldsymbol{q}}_b \in \delta^h$, where δ^h is the trial space. In addition, the non-linear fluxes are projected onto the finite-dimensional space via

$$\int_{\Omega^e} \boldsymbol{w} \left[N_b(\boldsymbol{\xi}) \left(\hat{\boldsymbol{F}}_{\boldsymbol{\xi}}\right)_b - \boldsymbol{F}_{\boldsymbol{\xi}}(N_b(\boldsymbol{\xi})\hat{\boldsymbol{q}}_b) \right] \mathrm{d}\Omega^e = 0.$$

A similar projection is used to define $\hat{\boldsymbol{S}}_b$. Note that all quantities with hats are coefficients that are functions of time only, and all spatial dependence is contained in the shape functions $N_b(\boldsymbol{\xi})$. The subscript b indicates the basis function index, and repeated indices are summed.

Using the Bubnov-Galerkin approach, we can express $\boldsymbol{w} = N_a \hat{\boldsymbol{w}}_a \in \left[\mathcal{V}^h\right]^4$, and therefore $\delta^h = \mathcal{V}^h$. Requiring the weak-form to be satisfied for all choices of $\hat{\boldsymbol{w}}_a$, results in

$$\begin{aligned} \int_{\Omega^e} N_a N_b \frac{\partial \hat{\boldsymbol{q}}_b}{\partial t}\frac{1}{|J|}\mathrm{d}\Omega^e = \int_{\Omega^e} N_a N_b \hat{\boldsymbol{S}}_b \frac{1}{|J|}\mathrm{d}\Omega^e + \int_{\Omega^e} \nabla_{\boldsymbol{\xi}} N_a \cdot \left[N_b \left(\hat{\boldsymbol{F}}_\xi\right)_b, N_b \left(\hat{\boldsymbol{F}}_\eta\right)_b \right] \mathrm{d}\Omega^e \\ - \int_{\Gamma^e} N_a N_b \left[\left(\hat{\boldsymbol{G}}_\xi\right)_b, \left(\hat{\boldsymbol{G}}_\eta\right)_b \right] \mathbf{d\Gamma} \qquad \forall\ N_a \in \mathcal{V}^h. \end{aligned} \tag{3}$$

In order to impose the boundary conditions weakly, the fluxes in the boundary integrals have been replaced with numerical flux functions, specifically Roe flux functions, denoted $\hat{\boldsymbol{G}}_{\boldsymbol{\xi}}$; see, for example, [24].

[a]In this work, we consider only affine mappings between reference elements and physical elements.

We now replace all operations on shape functions with the SBP matrix operators. Defining $\mathsf{J}^{-1} \equiv \operatorname{diag}(1/|J_1|, 1/|J_2|, \ldots, 1/|J_n|)$, we can replace the first integral on the right hand side of Equation (3) with $\mathsf{H}\mathsf{J}^{-1}\hat{\boldsymbol{S}}$, where H is the diagonal mass matrix. The second term is the volume integral contribution to the element stiffness matrix, which is approximated by $\mathsf{S}_{\boldsymbol{\xi}}^T = \frac{\partial N_a}{\partial \boldsymbol{\xi}} N_b$ [10]. Note that the element stiffness matrices in the parametric coordinate directions can be expressed as $\mathsf{S}_\xi^T = (\mathsf{H}\mathsf{D}_\xi)^T$; see Definition 1. As a result, a single matrix multiplication performs the action of integrating the stiffness contribution over an element. The final integral in Equation (3) can be expressed using the boundary integration matrices, E_ξ and E_η.

Applying these simplifications to the weak form gives

$$\mathsf{H}\mathsf{J}^{-1}\frac{\partial \hat{\boldsymbol{q}}}{\partial t} = \mathsf{H}\mathsf{J}^{-1}\hat{\boldsymbol{S}} + \mathsf{S}_\xi^T \hat{\boldsymbol{F}}_\xi + \mathsf{S}_\eta^T \hat{\boldsymbol{F}}_\eta - \mathsf{E}_\xi \hat{\boldsymbol{G}}_\xi - \mathsf{E}_\eta \hat{\boldsymbol{G}}_\eta. \tag{4}$$

The SBP weak form (4) can be rearranged into its conventional finite-difference strong form by multiplying from the left by the inverse norm matrix, H^{-1}:

$$\begin{aligned}\frac{\partial}{\partial t}\left(\mathsf{J}^{-1}\hat{\boldsymbol{q}}\right) &= \mathsf{J}^{-1}\hat{\boldsymbol{S}} + \mathsf{H}^{-1}\left[\mathsf{S}_\xi^T \hat{\boldsymbol{F}}_\xi + \mathsf{S}_\eta^T \hat{\boldsymbol{F}}_\eta - \mathsf{E}_\xi \hat{\boldsymbol{G}}_\xi - \mathsf{E}_\eta \hat{\boldsymbol{G}}_\eta\right] \\ &= \mathsf{J}^{-1}\hat{\boldsymbol{S}} - \mathsf{D}_\xi \hat{\boldsymbol{F}}_\xi - \mathsf{D}_\eta \hat{\boldsymbol{F}}_\eta - \mathsf{H}^{-1}\mathsf{E}_\xi\left[\hat{\boldsymbol{G}}_\xi - \hat{\boldsymbol{F}}_\xi\right] - \mathsf{H}^{-1}\mathsf{E}_\eta\left[\hat{\boldsymbol{G}}_\eta - \hat{\boldsymbol{F}}_\eta\right],\end{aligned}$$

where we have used properties 2 and 3 from Definition 1. Note that the two terms multiplied by H^{-1} on the right-hand side, the so-called simultaneous approximation terms, represent penalties with vanishing truncation errors that impose the boundary conditions weakly [25, 26].

B. Entropy Variable Formulation

The formulation above uses SBP operators to discretize the Euler equations based on the conservative variables. The resulting semi-discrete system is not entropy stable and may not be suitable for long-time simulations. On the other hand, Hughes *et al.* proved that a Galerkin discretization of the symmetrized Euler and Navier-Stokes equations in entropy variables would satisfy the discrete Clausis-Duhem inequality [27, 28]. Entropy stability has been shown to improve the robustness of high-order discretizations involving turbulence [29].

Motivated by the results in [29], we consider an SBP discretization of the symmetrized Euler equations in order to take advantage of the proven "energy" stability in the physically significant entropy norm. However, in this preliminary work we have not implemented a skew-symmetric discretization of the derivative, so our semi-discrete scheme is not provably entropy stable. Provable entropy stability will be the focus of future work.

We discretize the entropy variable formulation using SBP operators starting with Equation (2). Introducing a change of variables from $\boldsymbol{q}$ to $\boldsymbol{v}$, where $\boldsymbol{v}$ are the so-called entropy variables, defined in [28], we have

$$\frac{\partial \boldsymbol{q}}{\partial t} = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{v}}\frac{\partial \boldsymbol{v}}{\partial t} = \mathsf{A}_0\frac{\partial \boldsymbol{v}}{\partial t},$$

and

$$\nabla \cdot \boldsymbol{F} = \frac{\partial F_i}{\partial x_i} = \frac{\partial F_i}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}}{\partial x_i} = \mathsf{A}_i\frac{\partial \boldsymbol{q}}{\partial x_i} = \mathsf{A}_i^S\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{v}}\frac{\partial \boldsymbol{v}}{\partial x_i} = \mathsf{A}_i^S\mathsf{A}_0\frac{\partial \boldsymbol{v}}{\partial x_i} = \mathsf{A}_i^S\frac{\partial \boldsymbol{v}}{\partial x_i} = \nabla \cdot \boldsymbol{F}^S$$

where $\boldsymbol{F}^S$ are the Euler fluxes and the A_i^S are the flux Jacobians with respect to entropy variables. The definitions of these quantities in terms of entropy variables are given in the appendix of [28]. Thus the Euler equations become

$$\mathsf{A}_0\frac{\partial \boldsymbol{v}}{\partial t} + \nabla \cdot \boldsymbol{F}^S = \boldsymbol{S}. \tag{5}$$

The derivation of the SBP weak form follows in the same fashion presented in Subsection II.A. The result is

$$\mathsf{H}\mathsf{J}^{-1}\left(\mathsf{A}\frac{\partial \hat{\boldsymbol{v}}}{\partial t}\right) = \mathsf{H}\mathsf{J}^{-1}\hat{\boldsymbol{S}} + \mathsf{S}_\xi^T \hat{\boldsymbol{F}}_\xi^S + \mathsf{S}_\eta^T \hat{\boldsymbol{F}}_\eta^S - \mathsf{E}_\xi \hat{\boldsymbol{G}}_\xi^S - \mathsf{E}_\eta \hat{\boldsymbol{G}}_\eta^S, \tag{6}$$

where A is a block-diagonal matrix holding the A_0 matrices evaluated at the nodes.

III. Edge Stabilization

As discussed in the introduction, SBP operators do not provide the dissipation necessary to prevent aliasing errors. To address this, we consider the edge stabilization originally presented by Douglas and Dupont [30] (see also [21] and [16]). It is obtained by adding the term

$$J(\boldsymbol{w},\boldsymbol{q}) = -\frac{1}{2}\sum_{\nu}\int_{\Gamma_\nu} \gamma_{\text{stab}} h_\nu^2 \left[\nabla \boldsymbol{q}\right]\cdot\left[\nabla \boldsymbol{w}\right] \mathrm{d}\Gamma = -\frac{1}{2}\sum_{\nu}\int_{\Gamma_\nu} \gamma_{\text{stab}} h_\nu^2 \left[\mathbf{n}\cdot\nabla \boldsymbol{q}\right]\left[\mathbf{n}\cdot\nabla \boldsymbol{w}\right] \mathrm{d}\Gamma$$

to Equation (3), or the entropy-variable equivalent, where Γ_ν denotes an interior edge, h_ν is the nominal edge length, and γ_{stab} is a chosen constant. Square brackets denote the jump operator on the element boundary: $[u] = \lim_{\delta\to 0} u(\mathbf{x}+\delta\mathbf{n}) - u(\mathbf{x}-\delta\mathbf{n})$ for $\mathbf{x}\in\Gamma_\nu$. The jump-stabilization term acts at the interface between two elements, penalizing differences in the normal derivative.

Our solver is intended for compressible flow problems, so we have adapted the original edge-stabilization method. In particular, we use the spectral radius of the normal flux Jacobian as a scaling term to ensure the stabilization is dimensionally consistent with the Euler equations, and we consider the jump in all solution variables across the element boundaries. The adapted form of the stabilization term used in the solver can be seen in Equation (7).

$$J(\boldsymbol{w},\boldsymbol{q}) = -\frac{1}{2}\sum_{\nu}\int_{\Gamma_\nu} (|\boldsymbol{u}\cdot\mathbf{n}| \ + a)\gamma_{\text{stab}} h_\nu^2 \left[\mathbf{n}\cdot\nabla \boldsymbol{q}\right]\left[\mathbf{n}\cdot\nabla \boldsymbol{w}\right] \mathrm{d}\Gamma, \tag{7}$$

where a is the local speed of sound. We will explore the effects of γ_{stab} in Section V.

To implement edge stabilization within an SBP discretization, we use the following approximation to the normal-derivative jump operator.

$$[\boldsymbol{n}\cdot\nabla] \approx \mathsf{D}_\nu \equiv \left(\mathsf{B}_{\xi,L,\nu}\mathsf{L}_\nu\mathsf{D}_{\xi,L} + \mathsf{B}_{\eta,L,\nu}\mathsf{L}_\nu\mathsf{D}_{\eta,L}\right) + \left(\mathsf{B}_{\xi,R,\nu}\mathsf{R}_\nu\mathsf{D}_{\xi,R} + \mathsf{B}_{\eta,R,\nu}\mathsf{R}_\nu\mathsf{D}_{\eta,R}\right).$$

Here, $\mathsf{D}_{\xi,L}$ and $\mathsf{D}_{\eta,L}$ denote the SBP derivative operators on the nominal left side of the interface, and $\mathsf{D}_{\xi,R}$ and $\mathsf{D}_{\eta,R}$ denote the corresponding operators on the right side of the interface. The operators L_ν and R_ν are binary matrices that select the boundary nodes on Γ_ν for the left and right elements, respectively; L_ν and R_ν are $n_\nu\times n$ rectangular matrices, where n_ν is the number of nodes on Γ_ν. The matrices $\mathsf{B}_{\xi,L,\nu}$, $\mathsf{B}_{\eta,L,\nu}$, $\mathsf{B}_{\xi,R,\nu}$, and $\mathsf{B}_{\eta,L,\nu}$ contain geometric information that accounts for the transformation from reference to physical space. They are $n_\nu\times n_\nu$ diagonal matrices given by

$$\begin{aligned}
\mathsf{B}_{\xi,L,\nu} &= \operatorname{diag}\left[(\nabla\xi\cdot\nabla\xi)n_\xi + (\nabla\xi\cdot\nabla\eta)n_\eta\right]_L, &\qquad \mathsf{B}_{\eta,L,\nu} &= \operatorname{diag}\left[(\nabla\xi\cdot\nabla\eta)n_\xi + (\nabla\eta\cdot\nabla\eta)n_\eta\right]_L,\\
\mathsf{B}_{\xi,R,\nu} &= \operatorname{diag}\left[(\nabla\xi\cdot\nabla\xi)n_\xi + (\nabla\xi\cdot\nabla\eta)n_\eta\right]_R, &\qquad \mathsf{B}_{\eta,R,\nu} &= \operatorname{diag}\left[(\nabla\xi\cdot\nabla\eta)n_\xi + (\nabla\eta\cdot\nabla\eta)n_\eta\right]_R,
\end{aligned}$$

where (n_ξ, n_η) is the normal to Γ_ν in reference space.

Using the edge mass matrix M_ν to approximate integration over Γ_ν, the SBP form of edge stabilization is

$$J^h(w_i, q_j) = -\frac{\gamma}{2}\sum_{\nu} h_\nu^2 \left(\mathsf{D}_\nu w_i\right)^T \mathsf{M}_\nu \Upsilon \mathsf{D}_\nu q_j,$$

where $w_i, q_j\in\mathbb{R}^n$ for $i,j = 1:4$ are components of the $\boldsymbol{w}$ and $\boldsymbol{q}$ vectors at the nodes, and $\Upsilon = \operatorname{diag}(|\boldsymbol{u}\cdot\mathbf{n}|+a)$ contains the fastest acoustic wave speeds at each node on Γ_ν. For SBP discretizations, the test function w_i is a binary unit vector taking the value of 1 for one node and zero otherwise. Thus, by considering each node in turn, the above form becomes the vector

$$J^h(\mathsf{I}, q_j) = -\frac{\gamma}{2}\sum_{\nu} h_\nu^2 \mathsf{D}_\nu^T \mathsf{M}_\nu \Upsilon \mathsf{D}_\nu q_j,$$

where I is the $n \times n$ identity.

If $|\boldsymbol{u} \cdot \mathbf{n}| + a$ is constant over Γ_ν, then edge stabilization does not interfere with entropy stability, because the operator is negative semi-definite. Proving entropy stability in the case of a spatially varying $|\boldsymbol{u} \cdot \mathbf{n}| + a$ is also possible, but requires interpolation to cubature nodes on Γ_ν; see, for example, [31].

IV. Solver Abstraction and Implementation

A. Implementation in Julia

Having described the SBP discretization, we now pause to discuss our implementation and solution of the resulting non-linear algebraic equations using the programming language Julia. One of the main features of Julia is a generic, type optional, programming style where type information can be specified by the user or inferred by the compiler. When a function is called with a particular set of values as arguments, the Just In Time (JIT) compiler uses the types of the values to compile a version of the function specialized to those types, as if the programmer had specified the datatype of every variable. This gives Julia the efficiency of statically typed languages like C (without vectorization), but with the flexibility to call functions with different argument types [32], and permits the use of abstraction in a natural and readable syntax, even in low-level numerical routines.

The evaluation of the Euler equations in the form of Equation (4) is implemented as a system of non-linear algebraic equations. Using Julia's parametric type system, the entire calculation is parameterized on the types of the input variables $\hat{\boldsymbol{q}}$, the type of the result $\frac{\partial \hat{\boldsymbol{q}}}{\partial t}$, and the type of the mesh Jacobian terms $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}$. As a result, it is possible to differentiate with respect to either the solution variables or the mesh variables.

One use of the parameterization is to calculate the Jacobian for Newton's method, which we use to solve steady problems. An abstract form of algorithmic differentiation (AD) is implemented, taking in a seed vector whose elements can be of any datatype. Because of Julia's type inference capability and the JIT compiler, when the evaluation function is called with complex-perturbed solution variables, every function is compiled with datatypes of all the variables known to the compiler. If a different AD datatype is used, such as hyper-dual numbers [33], all the numerical routines are recompiled, generating efficient machine code to operate on dual numbers. The result is a highly efficient mechanism for algorithmic differentiation, with the full range of compiler optimization available and no source code duplication.

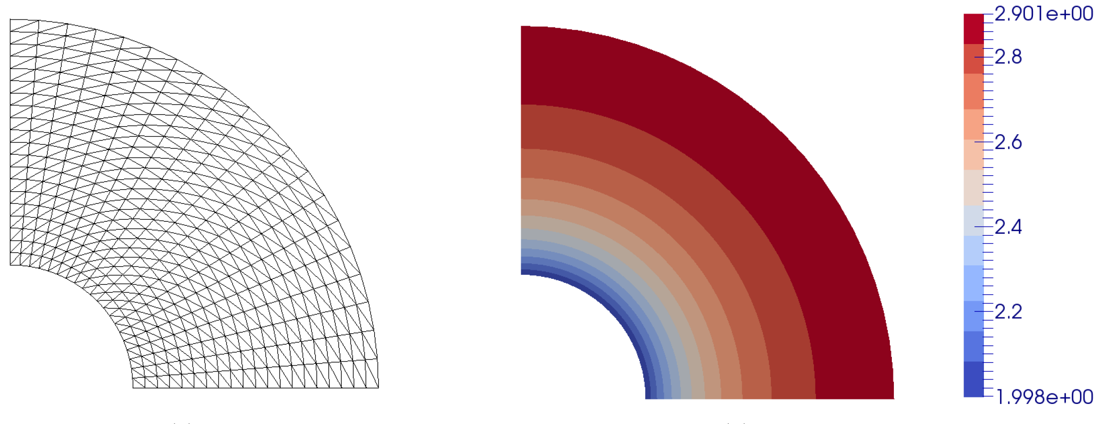
One of the benefits of the parameterization is its extensibility. For example, in order to implement entropy variables, an additional type parameter is introduced. This parameter enables the use of Julia's multiple dispatch system to call the correct methods for the variables being used. For example, methods to calculate pressure and the Euler flux at a node are defined for both variables, and the compiler chooses between them based on the type parameter during method dispatch at compile time. Thus, the implementation of entropy variables required only defining the node-based operations and the transformation matrices described in Section II.B. This ease of extensibility for low-level routines demonstrates the utility of the multiple dispatch paradigm in designing numerical software.

B. Algebraic Equation Solvers

Steady flow problems are solved using Newton's method. To solve the linear systems that arise in Newton's method, a sparse multi-frontal LU factorization from UMFPACK is employed [34]. For unsteady problems, we use the classical 4th-order explicit Runge-Kutta method.

In order to calculate the Jacobian for Newton's method with as few (complex) residual evaluations as possible, mesh coloring and an element-based data structure are used. In the element-based data arrays, the solution and residual values are stored for every node on each element, duplicating the values for nodes that are shared between elements. This enables perturbation of the $\hat{\boldsymbol{q}}$ variables on one element without affecting neighboring elements.

For the unstabilized equations a distance-0 coloring (i.e. all elements are the same color) is sufficient to ensure there is a one-to-one mapping from residual perturbations to perturbations in $\hat{\boldsymbol{q}}$. Each degree of freedom on an element must be perturbed independently, therefore the Jacobian can be calculated in $m = 4n$



(a) Mesh (b) Contours of ρ

Figure 1: Steady isentropic vortex solution

residual evaluations, where m is the number of degrees of freedom on an element.

For edge stabilization, where perturbations to a degree of freedom affect all neighboring elements that share an edge, a distance-2 graph coloring is necessary, where graph vertices correspond to mesh elements and graph edges connect elements that share a mesh edge. We use Algorithm 3.1 from Ref. [35] to perform this coloring, which has an upper bound of $\Delta^2 + 1$ colors, where Δ is the maximum number of neighboring elements. Considering only edge neighbors for triangular elements, the maximum number of colors is 10, and the maximum number of residual evaluations is $10m$. The graph-based Parallel Unstructured Mesh Interface [36] was used for all meshing, and it facilitated the implementation of the coloring algorithm by allowing querying of the topological adjacencies of mesh elements.

V. Results

A. Isentropic Vortex

An isentropic vortex problem was used to verify the asymptotic convergence rates of the discretizations. The vortex problem consists of a quarter circle domain with an inner radius $r_{\text{in}} = 1$ and outer radius $r_{\text{out}} = 3$. An example of the expected computational solution can be seen in Figure 1(b), in which density is plotted. Here, the problem was solved with degree $p = 2$ elements, with 20 elements along each boundary edge; see Figure 1(a) for the corresponding mesh.

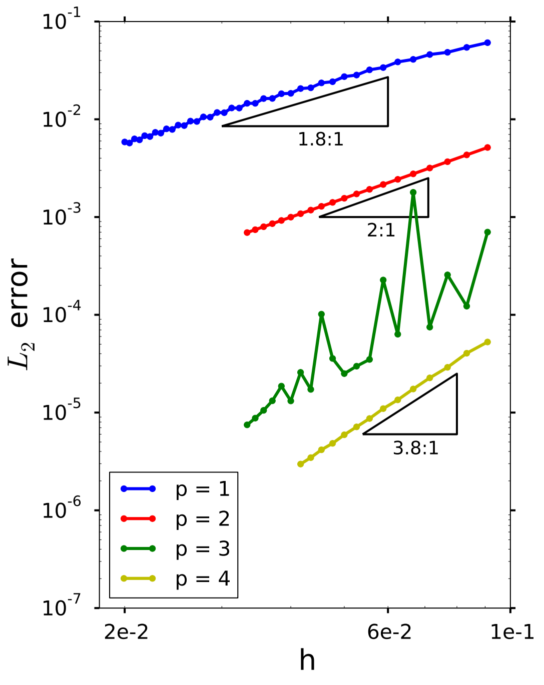
The analytical solution of this problem is known to be

$$\rho(r) = \rho_{in}\left[1 + \frac{\gamma - 1}{2} M_{\text{in}}^2 \left(1 - \frac{r_{\text{in}}^2}{r^2}\right)\right]^{\frac{1}{\gamma-1}}.$$

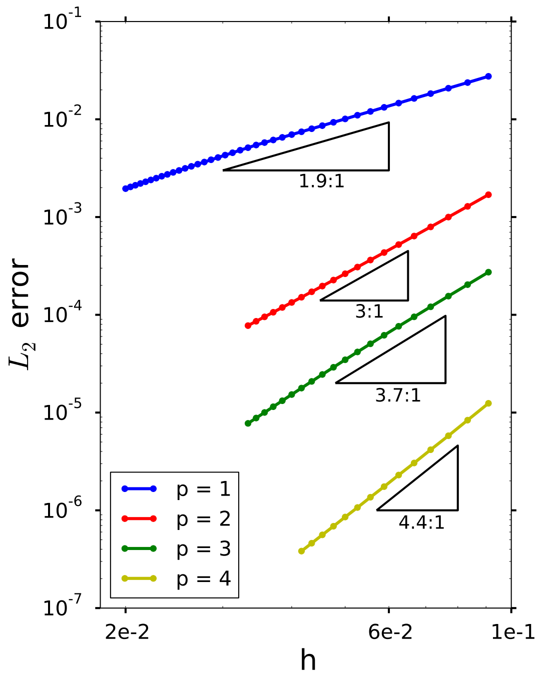
We have set the inner-radius density to $\rho_{in} = 2$ and inner-radius Mach number to $M_{\text{in}} = 0.95$. The specific heat ratio $\gamma = 1.4$ is used for all problems. All other flow variables can be calculated using the isentropic relations. Note that the analytical solution is imposed (weakly) along all boundaries.

To determine the convergence rates of the discretizations, the steady vortex problem was solved on a sequence of meshes, using the analytic solution as the initial guess. Newton's method was run until an absolute residual tolerance of 10^{-12} was achieved, or the residual was reduced by ten orders relative to the residual for a uniform flow. The SBP approximation to the continuous L_2 norm, $\sqrt{u^T \mathsf{HJ} u}$, is used for all error-norm calculations.

Figure 2 shows the convergence rates for the unstabilized and edge-stabilized discretizations. As in [10] for the linear advection equation, the $p = 2$ and $p = 4$ discretizations show suboptimal convergence rates of



(a) No stabilization



(b) Edge stabilization

Figure 2: Effect of mesh refinement upon L_2 error

Degree	Convergence: No Stabilization	Convergence: Edge Stabilization	γ_{stab}
$p = 1$	1.8	1.9	0.01
$p = 2$	2.0	3.0	0.9
$p = 3$	-	3.7	7.35
$p = 4$	3.8	4.4	25.0

Table 1: Convergence rates observed in the steady-vortex problem

order p. The odd-order discretizations show oscillatory convergence, especially the $p = 3$ discretization. In contrast, the edge-stabilized discretizations exhibit well-behaved convergence.

For edge-stabilization, values of the stabilization parameter that produce optimal $p+1$ convergence rates were easily found for the $p = 1$ and $p = 2$ elements. A range of values was found to produce optimal convergence rates, so the value corresponding to the minimal error norm is reported here. For the higher order elements, a Golden-Section optimization was performed to find the parameter value that maximized convergence rate. Despite this optimization, the $p = 3$ and $p = 4$ elements exhibit sub-optimal convergence rates, although the $p = 2$ and $p = 4$ convergence rates are improved relative to the unstabilized case. The convergence rates are estimated using a least-squares analysis of the five finest meshes and are listed in Table 1.

B. Unsteady Vortex

We now investigate the conservative- and entropy-variable formulations on an unsteady problem with and without edge stabilization. Specifically, we consider the unsteady isentropic vortex problem; see, for example, [37]. The analytical solution is known to be [38]

$$\begin{aligned} u &= 1 - \frac{\epsilon y}{2\pi} \exp\left(\frac{f(x,y,t)}{2}\right), \qquad & v &= \frac{\epsilon((x-x_0)-t)}{2\pi} \exp\left(\frac{f(x,y,t)}{2}\right) \\ \rho &= \left(1 - \frac{\epsilon^2(\gamma-1)M^2}{8\pi^2} \exp\left(f(x,y,t)\right)\right)^{\frac{1}{1-\gamma}}, \qquad & p &= \frac{\rho^\gamma}{\gamma M^2}, \end{aligned} \tag{8}$$

where $f(x,y,t) = 1 - (((x-x_0)-t)^2 + y^2)$, the Mach number is set to $M = 0.5$, ϵ, the vortex strength, is set to 1, and x_0, the x coordinate of the center of the vortex at $t = 0$, is 5. The y coordinate of the vortex's center is zero. We solve the problem on a rectangular domain $x \in [0, 20]$ and $y \in [-5, 5]$. The analytical solution is imposed for both the initial condition and boundary conditions. The simulation was run to a maximum time of 2 seconds.

A mesh defined by 50 elements along each edge was used for the $p = 1$ discretization, corresponding to 10,404 degrees-of-freedom. For the higher order elements, meshes with the closest number of degrees-of-freedom possible were used, subject to the constraint that the number of elements along each edge is equal; see Table 2.

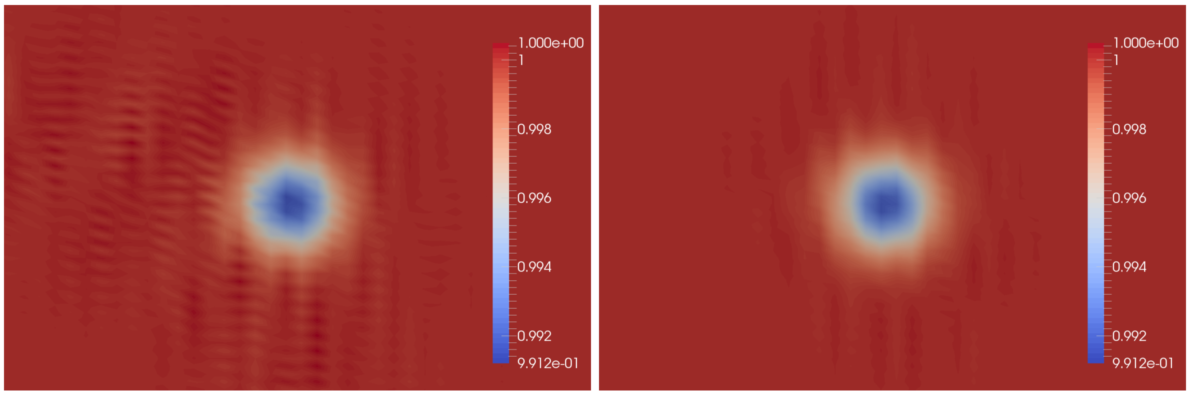
The maximum stable CFL number was determined to be 0.0001 for the $p = 4$ stabilized discretization and was used for all runs. The use of edge stabilization significantly reduced the maximum stable CFL number compared to the unstabilized case. In calculating the CFL number, the minimum distance between nodes for each degree element was used as the mesh spacing.

For the $p = 1$ discretization, the same stabilization constant was used as for the steady vortex problem. For high-order elements, $\gamma_{\text{stab}} = 0.5$ proved to be sufficient to maintain stability. For reasons of computational economy, higher values of γ_{stab} were not studied. Table 2 lists the value of the edge stabilization parameter γ_{stab} used for all degree operators.

Degree	CFL	γ_{stab}	DOFs
$p = 1$	0.0001	0.01	10,404
$p = 2$	0.0001	0.5	10,560
$p = 3$	0.0001	0.5	10,804
$p = 4$	0.0001	0.5	10,644

Table 2: CFL and edge stabilization parameter values used for the unsteady vortex problem. The $p = 4$ stabilized elements dictated the maximum stable timestep used for all discretizations.

Figure 3 shows the solution field at the final time. Density is plotted for the $p = 2$ discretization with and without stabilization. Qualitatively, one can see the effects of adding stabilization; Figure 3(a) demonstrates



(a) $p = 2$, no stabilization (b) $p = 2$, stabilized

Figure 3: Density at final time of the unsteady vortex problem

slight oscillations around the vortex that indicate instability, whereas Figure 3(b) shows significantly fewer oscillations.

Because the exact solution is isentropic, it is desirable for a numerical method to retain this property. The results in Figure 4 indicate that both the conservative and entropy formulations increase entropy over time. The unstabilized formulations generate similar amounts of entropy, but for $p = 1$ through $p = 3$ the stabilized entropy variable formulation generates less entropy than the stabilized conservative formulation. This suggests the entropy formulation is effective in reducing non-physical entropy generated by the stabilization.

While increasing entropy is expected for the unstabilized conservative case, Hughes *et al.* proved that the finite-element discretization of the Euler equations in entropy variables will inherit the continuous equation's property of conserving entropy [28]. As noted in Section II.B, a skew-symmetric discretization is necessary to realize provable entropy stability and will be the focus of future work.

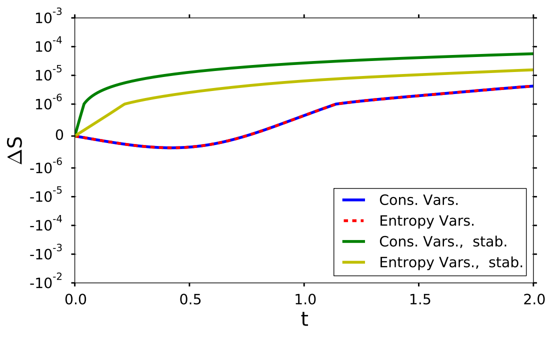
VI. Conclusion

High-order simplex SBP methods show promise as a means to efficiently solve PDEs numerically on unstructured grids; however, they require stabilization for non-linear problems. To this end, we have investigated edge stabilization for simplex SBP operators, examining both the convergence rate and entropy stability of the spatial discretizations. Retaining high-order convergence in the stabilized scheme is necessary in order to realize the computational efficiency of high-order methods. Additionally, entropy stability is important for long-time simulations, such as those used for unsteady turbulence simulations.

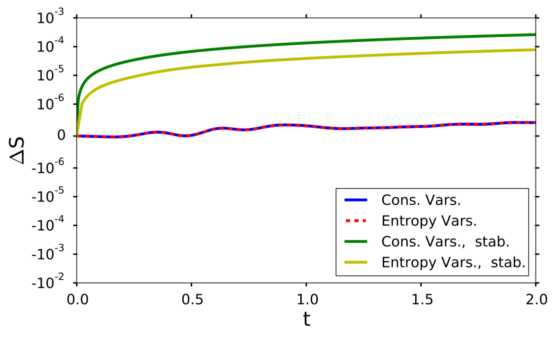
We have shown that edge stabilization is effective in stabilizing a simplex-based SBP discretization of the Euler equations for all orders of simplex SBP elements, although it significantly increases the cost of using algorithmic differentiation to compute the Jacobian and, more significantly, imposes a restriction on the CFL number for unsteady problems using explicit time marching. For unsteady problems, entropy variables increase the solution accuracy of the stabilized scheme with respect to entropy, but does not render the discretization entropy stable without a skew-symmetric derivative approximation. Further investigation is required into a skew-symmetric SBP formulation for entropy stability.

Acknowledgements

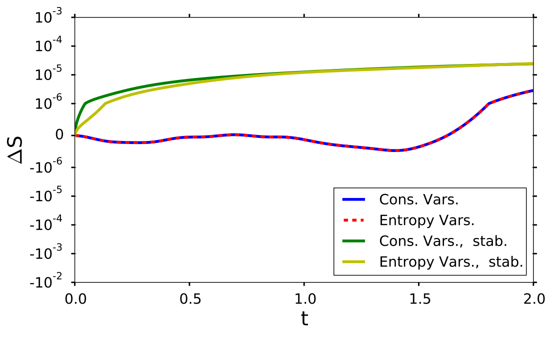
A. Ashley and J. Hicken gratefully acknowledge the support of the Air Force Office of Scientific Research Award FA9550-15-1-0242 under Dr. Jean-Luc Cambier.



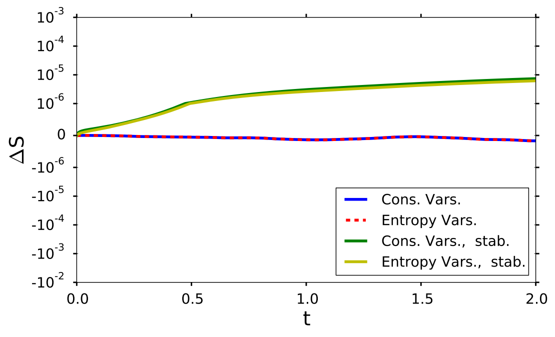
(a) $p = 1$



(b) $p = 2$



(c) $p = 3$



(d) $p = 4$

Figure 4: Change in entropy over solution time

References

[1]Kreiss, H.-O. and Oliger, J., "Comparison of accurate methods for the integration of hyperbolic equations," *Tellus*, Vol. 24, No. 3, 1972, pp. 199–215.

[2]Swartz, B. and Wendroff, B., "The Relative Efficiency of Finite Difference and Finite Element Methods. I: Hyperbolic Problems and Splines," *SIAM Journal on Numerical Analysis*, Vol. 11, No. 5, 1974, pp. pp. 979–993.

[3]Kreiss, H.-O. and Scherer, G., "Finite element and finite difference methods for hyperbolic partial differential equations," *Mathematical Aspects of Finite Elements in Partial Differential Equations*, edited by C. de Boor, Mathematics Research Center, the University of Wisconsin, Academic Press, 1974.

[4]Strand, B., "Summation by Parts for Finite Difference Approximations for d/dx," *Journal of Computational Physics*, Vol. 110, No. 1, 1994, pp. 47 – 67.

[5]Hicken, J. E. and Zingg, D. W., "A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms," *AIAA Journal*, Vol. 46, No. 11, Nov. 2008, pp. 2773–2786.

[6]Nordström, J., Gong, J., van der Weide, E., and Svärd, M., "A stable and conservative high order multi-block method for the compressible Navier–Stokes equations," *Journal of Computational Physics*, Vol. 228, No. 24, 2009, pp. 9020–9035.

[7]Mattsson, K. and Carpenter, M. H., "Stable and accurate interpolation operators for high-order multiblock finite difference methods," *SIAM Journal on Scientific Computing*, Vol. 32, No. 4, 2010, pp. 2298–2320.

[8]Svärd, M. and Nordström, J., "Review of summation-by-parts schemes for initial-boundary-value-problems," *Journal of Computational Physics*, Vol. 268, No. 1, 2014, pp. 17–38.

[9]Del Rey Fernández, D. C., Hicken, J. E., and Zingg, D. W., "Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations," *Computers and Fluids*, 2014.

[10]Hicken, J. E., Del Rey Fernández, D. C., and Zingg, D. W., "Multidimensional summation-by-parts operators: general theory and application to simplex elements," *SIAM Journal on Scientific Computing*, September 2015, Submitted (in revision).

[11]Cohen, G., Joly, P., Roberts, J. E., and Tordjman, N., "Higher Order Triangular Finite Elements with Mass Lumping for the Wave Equation," *SIAM Journal on Numerical Analysis*, Vol. 38, No. 6, 2001, pp. pp. 2047–2078.

[12]Giraldo, F. X. and Taylor, M. A., "A diagonal-mass-matrix triangular-spectral-element method based on cubature points," *Journal of Engineering Mathematics*, Vol. 56, No. 3, 2006, pp. 307–322.

[13]Von Neumann, J. and Richtmyer, R. D., "A Method for the Numerical Calculation of Hydrodynamic Shocks," *Journal of Applied Physics*, Vol. 21, No. 3, 1950.

[14]Brooks, A. N. and Hughes, T. J. R., "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 32, No. 1–3, 1982, pp. 199 – 259.

[15]Rice, J. G. and Schnipke, R. J., "A monotone streamline upwind finite element method for convection-dominated flows," *Computer Methods in Applied Mechanics and Engineering*, Vol. 48, No. 3, 1985, pp. 313 – 327.

[16]Burman, E., Fernández, M. A., and Hansbo, P., "Continuous Interior Penalty Finite Element Method for Oseen's Equations," *SIAM Journal on Numerical Analysis*, Vol. 44, No. 3, 2006, pp. pp. 1248–1274.

[17]Hughes, T. J. R. and Brooks, A., "A multidimensional upwind scheme with no crosswind diffusion," *Finite element methods for convection dominated flows*, Vol. 34, 1979, pp. 19–35.

[18]Hughes, T. J. R., Engel, G., Mazzei, L., and Larson, M. G., "The Continuous Galerkin Method Is Locally Conservative," *Journal of Computational Physics*, Vol. 163, No. 2, 2000, pp. 467–488.

[19]Venkatakrishnan, V., Allmaras, S., Kamenetskii, D., and Johnson, F., chap. Higher Order Schemes for the Compressible Navier-Stokes Equations, Fluid Dynamics and Co-located Conferences, American Institute of Aeronautics and Astronautics, Jun 2003.

[20]Collis, S. S. and Heinkenschloss, M., "Analysis of the streamline upwind/Petrov Galerkin method applied to the solution of optimal control problems," Tech. Rep. TR02-01, Houston, Texas, 2002.

[21]Burman, E. and Hansbo, P., "Edge stabilization for Galerkin approximations of convection–diffusion–reaction problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 193, No. 15–16, 2004, pp. 1437–1453, Recent Advances in Stabilized and Multiscale Finite Element Methods.

[22]Burman, E., Fernández, M. A., and Hansbo, P., "Edge stabilization for the incompressible Navier-Stokes equations: a continuous interior penalty method," 2004, CMCS-ARTICLE-2004-006.

[23]Pulliam, T. H., "Efficient Solution Methods for the Navier-Stokes Equations," Lecture Notes for the Von Karman Institute for Fluid Dynamics Lecture Series.

[24]Laney, C., *Computational Gas Dynamics*, Cambridge University Press, 1998, pp. 71–105.

[25]Carpenter, M. H., Gottlieb, D., and Abarbanel, S., "Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes," *Journal of Computational Physics*, Vol. 111, No. 2, 1994, pp. 220–236.

[26]Carpenter, M. H., Nordström, J., and Gottlieb, D., "Revisiting and Extending Interface Penalties for Multi-domain Summation-by-Parts Operators," *Journal of Scientific Computing*, Vol. 45, No. 1-3, 2010, pp. 118–150.

[27]Harten, A., "On the symmetric form of systems of conservation laws with entropy," *Journal of Computational Physics*, Vol. 49, No. 1, 1983, pp. 151–164.

[28]Hughes, T. J. R., Franca, L. P., and Mallet, M., "A New Finite Element Formulation for Computational Fluid Dynamics: I. Symmetric Forms of hte Compressible Euler and Navier-Stokes Equations and the Second Law of Theromdynamics," *Computer Methods in Applied Mechanics and Engineering*, 1985.

[29]Diosady, L. T. and Murman, S. M., chap. Higher-Order Methods for Compressible Turbulent Flows Using Entropy Variables, AIAA SciTech, American Institute of Aeronautics and Astronautics, Jan 2015.

[30]Douglas, J. and Dupont, T., "Interior Penalty Procedures for Elliptic and Parabolic Galerkin Methods," *Computing Methods in Applied Sciences*, edited by R. Glowinski and J. Lions, Vol. 58 of *Lecture Notes in Physics*, Springer Berlin Heidelberg, 1976, pp. 207–216.

[31]Hicken, J. E., Del Rey Fernández, D. C., and Zingg, D. W., "Simultaneous Approximation Terms for Multidimensional Summation-by-parts Operators," *Aviation 2016 (under review)*, Dallas, Texas, June 2016.

[32]Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B., "Julia: A Fresh Approach to Numerical Computing," November 2014.

[33]Fike, J. A., Jongsma, S., Alonso, J. J., and van der Weide, E., "Optimization with Gradient and Hessian Information Calculated Using Hyper-Dual Numbers," AIAA, 2011.

[34]Davis, T. A., "An Unsymmetric-Pattern Multifrontal Method," *ACM Transactions on Mathematical Software*, Vol. 30, No. 2, 2004.

[35]Gebremedhin, A. H., Manne, F., and Pothen, A., "What Color Is Your Jacobian? Graph Coloring for Computing Derivatives," *SIAM Review*, Vol. 47, No. 4, 2005.

[36]Seol, S., Smith, C. W., Ibanez, D. A., and Shephard, M. S., "A Parallel Unstructured Mesh Infrastructure," *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, Nov 2012, pp. 1124–1132.

[37]Mattsson, K., Svärd, M., Carpenter, M., and Nordström, J., "High Order Accurate Computations for Unsteady Aerodynamics," *Computers and Fluids*, Vol. 36, No. 3, 2007, pp. 636–649.

[38]Erlebacher, G., Hussaini, M. Y., and Shu, C.-W., "Interaction of a shock with a longitudinal vortex," *Journal of Fluid Mechanics*, Vol. 337, Apr 1997, pp. 129–153.