Noname manuscript No.
(will be inserted by the editor)

Mortar-based entropy stable discontinuous Galerkin methods on non-conforming quadrilateral and hexahedral meshes

Jesse Chan, David C. Del Rey Fernandez,
Mario Bencomo

Abstract High order entropy stable discontinuous Galerkin (DG) methods for nonlinear conservation laws reproduce a discrete entropy inequality by combining entropy conservative finite volume fluxes with summation-by-parts (SBP) discretization matrices. On tensor product (quadrilateral and hexahedral) elements, SBP matrices are typically constructed by collocating at Lobatto quadrature points. Recent work has extended the construction of entropy stable DG schemes to collocation at more accurate Gauss quadrature points [1].

In this work, we extend entropy stable Gauss collocation schemes to non-conforming meshes. Entropy stable DG schemes require computing entropy conservative numerical fluxes between volume and surface quadrature nodes. On conforming tensor product meshes where volume and surface nodes are aligned, flux evaluations are required only between "lines" of nodes. However, on non-conforming meshes, volume and surface nodes are no longer aligned, resulting in a larger number of flux evaluations. We reduce this expense by introducing an entropy stable mortar-based treatment of non-conforming interfaces via a face-local correction term, and provide truncation error estimates for the resulting discretizations. Numerical experiments confirm the stability and accuracy of this approach, and we conclude by discussing an extension of this mortar treatment of non-conforming interfaces to general elements and arbitrary quadrature rules [2].

1. Intro
2. Gauss collocation schemes, conditions for entropy stability
3. Increased cost on non-conforming meshes
4. Mortar scheme and entropy stability
5. Analysis of accuracy of Global SBP operator. Gauss nodes = exact for degree N, while Lobatto = exact for degree $(N-1)$. Approach in Friedrichs + DCDR for hp meshes yields degree $(N-1)$ global operator.
6. Extensions
   (a) Curved meshes
   (b) p-nonconforming meshes
   (c) General elements and why you'd want to implement this.

Address(es) of author(s) should be given

7. Numerical experiments (2D vortex, 2D entropy conservation test, 3D convergence). Note that reduced rates of convergence are not observed for $N > 1$ on p-nonconforming meshes, or on hybrid meshes with mixed quadrature.

1 Introduction

Discretely entropy stability has emerged as a methodology for designing high order schemes for nonlinear conservation laws. Entropy stable discretizations ensure the satisfaction of a semi-discrete entropy inequality by combining specific finite volume numerical fluxes with summation-by-parts (SBP) discretization matrices. Compared to traditional high order methods, the resulting schemes demonstrate significantly improved robustness in the presence of under-resolved solution features such as shocks or turbulence while retaining high order accuracy.

Entropy stable discontinuous Galerkin (DG) methods were originally constructed for quadrilateral and hexahedral meshes based on nodal collocation at Lobatto quadrature points [3–5]. Entropy stable schemes were later extended to simplicial and more general elements using tailored volume and surface quadrature rules [6, 7]. More general quadrature rules were addressed in [1, 2, 8, 9], including entropy stable collocation schemes on quadrilateral and hexahedral elements based on more accurate Gauss quadrature rules and generalized SBP operators [1].

The work presented here focuses on geometrically non-conforming meshes. Such meshes may arise when applying domain decomposition techniques to a complex geometry (e.g. meshing sub-domains independently) [10] or performing local mesh refinement. Entropy stable Lobatto collocation schemes have been constructed on non-conforming meshes in [11] using SBP projection operators. In this work, we extend entropy stable Gauss collocation to non-conforming quadrilateral and hexahedral meshes. While it is straightforward to construct Gauss collocation schemes on non-conforming meshes, the treatment of non-conforming interfaces results in signficantly increased computational costs. We reduce such costs by adopting a mortar-based treatment of non-conforming interfaces. Moreover, while the cost of the proposed scheme is similar to that of [11] in 2D, the mortar-based approach is more computationally efficient for both Lobatto and Gauss collocation schemes on 3D non-conforming hexahedral meshes.

The paper is organized as follows: Section 2 briefly reviews the derivation of an entropy inequality for a system of nonlinear conservation laws. Section 3 introduces "hybridized" SBP operators as a unified way to treat both Lobatto and Gauss collocation schemes on conforming meshes of tensor product elements. Section 4 describes a naive extension to non-conforming meshes and illustrates why this formulation results in an increase in computational costs. Section 5 introduces a mortar-based formulation which addresses such costs, and characterizes the accuracy of the resulting formulations. Section **??** describes the extension to curved meshes, as well as more general elements and p non-conformity. We conclude with numerical validation of theoretical results in Section **??**.

2 Entropy stability for systems of nonlinear conservation laws

We are interested in the numerical approximation of solutions to systems of nonlinear conservation laws

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{i=1}^{d} \frac{\partial \boldsymbol{f}_i\left(\boldsymbol{u}\right)}{\partial x_i} = 0. \tag{1}$$

Here, $\boldsymbol{u}$ denotes the conservative variables and $\boldsymbol{f}_i(\boldsymbol{u})$ are nonlinear fluxes. We briefly review entropy stability for systems of conservation laws in d dimensions. We assume there exists a convex scalar entropy $S(\boldsymbol{u})$ associated with (1). We then define the entropy variables $\boldsymbol{v}(\boldsymbol{u})$ as the gradient of the entropy $S(\boldsymbol{u})$ with respect to the conservative variables

$$\boldsymbol{v} = \frac{\partial S(\boldsymbol{u})}{\partial \boldsymbol{u}}.$$

For $S(\boldsymbol{u})$ convex, $\boldsymbol{v}(\boldsymbol{u})$ defines an invertible mapping between the conservative and entropy variables, whose inverse (from entropy to conservative variables) we denote by $\boldsymbol{u}(\boldsymbol{v})$. Viscosity solutions to (1) satisfy an integrated form of the entropy inequality [12]

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

where F_i denotes the ith scalar entropy flux function, $\psi_i(\boldsymbol{u}) = \boldsymbol{v}^T\boldsymbol{f}_i(\boldsymbol{u}) - F_i(\boldsymbol{u})$ denotes the ith entropy potential, $\partial\Omega$ denotes the boundary of Ω and $\boldsymbol{n}_i$ denotes the ith component of the outward normal on $\partial\Omega$. This can be interpreted as implying that the time rate of change of entropy is bounded by the entropy flux through the boundary.

3 Entropy stable collocation DG methods and hybridized SBP operators

In this section, we summarize the work of [1] on the construction of entropy stable collocation DG methods based on generalized summation by parts (GSBP) operators. These constructions are applicable to collocation schemes based on either Lobatto and Gauss nodes.

3.1 Entropy conservative schemes in 1D

We begin by introducing collocation discretization matrices on the reference interval $[-1, 1]$. We assume the solution is collocated at $(N+1)$ quadrature points x_i with associated quadrature weights w_i, and consider primarily Lobatto or Gauss quadrature points. The collocation assumption is equivalent to approximating the solution using a degree N Lagrange basis $\ell_j(x)$ defined on the $(N+1)$ quadrature points.

We define mass and integrated differentiation matrices $\boldsymbol{M}, \boldsymbol{Q}$

$$\boldsymbol{M}_{ij} = \int_{-1}^{1} \ell_i(x)\ell_j(x), \qquad \boldsymbol{Q}_{ij} = \int_{-1}^{1} \frac{\partial \ell_j}{\partial x}\ell_i.$$

We assume that all integrals are computed using the collocated quadrature rule, such that

$$\boldsymbol{M}_{ij} = \int_{-1}^{1} \ell_i(x)\ell_j(x) \approx \sum_{k=1}^{N+1} \ell_i(x_k)\ell_j(x_k) w_k = \delta_{ij} w_i$$

$$\boldsymbol{Q}_{ij} = \int_{-1}^{1} \frac{\partial \ell_j}{\partial x}\ell_i \approx \sum_{k=1}^{N+1} \ell_i(x_k) \left.\frac{\partial \ell_j}{\partial x}\right|_{x_k} w_k.$$

In other words, the mass matrix is diagonal with entries equal to the quadrature weights. Since the integrands of $\boldsymbol{M}$ are degree $2N$ polynomials, the collocation approximation of $\boldsymbol{M}$ is exact for Gauss quadrature, but not for Lobatto quadrature. For both Gauss and Lobatto quadrature, the matrix $\boldsymbol{Q}$ is exact under collocation quadrature.

We introduce the $2 \times (N+1)$ matrix $\boldsymbol{E}$ which interpolates values at collocation nodes to values at the endpoints $x = -1$ and $x = 1$. These two matrices are defined entrywise as

$$(\boldsymbol{E})_{1i} = \ell_i(-1), \qquad (\boldsymbol{E})_{2i} = \ell_i(1).$$

The mass and differentiation matrices satisfy a generalized summation by parts property [13]

$$\boldsymbol{Q} = \boldsymbol{E}^T\boldsymbol{B}\boldsymbol{E} - \boldsymbol{Q}^T, \qquad \boldsymbol{B} = \begin{bmatrix} -1 & \\ & 1 \end{bmatrix}. \tag{3}$$

The GSBP property holds for both Lobatto and Gauss nodes, and switching between these two nodal sets simply requires redefining the matrices $\boldsymbol{D}, \boldsymbol{E}$. For Gauss nodes, $\boldsymbol{E}$ is dense. For Lobatto nodes, since the collocation nodes include boundary points, the interpolation matrix $\boldsymbol{E}$ reduces to the matrix which extracts nodal values associated with the left and right endpoints

$$\boldsymbol{E} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 1 \end{bmatrix}.$$

It is possible to construct energy stable discretizations of linear hyperbolic systems using GSBP operators based on Gauss nodes [13]. However, for nonlinear conservation laws, the presence of the dense $\boldsymbol{E}$ matrix in the GSBP property (3) complicates the imposition of boundary conditions and computation of inter-element numerical fluxes [1, 2, 7].

We can avoid this issue by introducing "hybridized" (or decoupled) SBP operators [2, 14], which are defined as the block matrix $\boldsymbol{Q}_h$ [9]

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

Note that the hybridized SBP operator satisfies a block form of the SBP property

$$\boldsymbol{Q}_h + \boldsymbol{Q}_h^T = \begin{bmatrix} \boldsymbol{0} & \\ & \boldsymbol{B} \end{bmatrix}. \tag{4}$$

where $\boldsymbol{E}$ does not appear in the block boundary matrix on the right hand side. Additionally, it is straightforward to show that

$$\boldsymbol{Q}_h \boldsymbol{1} = \boldsymbol{0}$$

using the GSBP property and properties of $\boldsymbol{Q}$.

The aforementioned matrices can now be used to construct high order accurate entropy stable and entropy conservative schemes. We first introduce an entropy conservative numerical flux [15]. Let $\boldsymbol{u}_L, \boldsymbol{u}_R$ denote left and right states in the conservative variables. Then, an entropy conservative numerical flux is a vector-valued function $\boldsymbol{f}_S$ which satisfies the following three properties

$$\begin{aligned} \boldsymbol{f}_S\left(\boldsymbol{u}, \boldsymbol{u}\right) &= \boldsymbol{f}(\boldsymbol{u}), \qquad \text{(consistency)} \\ \boldsymbol{f}_S\left(\boldsymbol{u}_L, \boldsymbol{u}_R\right) &= \boldsymbol{f}_S(\boldsymbol{u}_R, \boldsymbol{u}_L), \qquad \text{(symmetry)} \\ \left(\boldsymbol{v}_L - \boldsymbol{v}_R\right)^T \boldsymbol{f}_S\left(\boldsymbol{u}_L, \boldsymbol{u}_R\right) &= \psi(\boldsymbol{u}_L) - \psi(\boldsymbol{u}_R), \qquad \text{(conservation)}. \end{aligned}$$

Here, $\boldsymbol{v}_L, \boldsymbol{v}_R$ are the entropy variables evaluated at $\boldsymbol{u}_L, \boldsymbol{u}_R$, and ψ is the entropy potential which appears in (2). Then, an entropy conservative scheme on the reference interval is given as

$$\begin{aligned} &\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{dt}} + \begin{bmatrix}\boldsymbol{I}\\ \boldsymbol{E}\end{bmatrix}^T \left(\boldsymbol{Q}_h \circ \boldsymbol{F}\right)\mathbf{1} + \boldsymbol{E}^T\boldsymbol{B}\left(\boldsymbol{f}^* - \boldsymbol{f}(\widetilde{\boldsymbol{u}}_f)\right) = 0 \qquad (5) \\ &\widetilde{\boldsymbol{u}} = \begin{bmatrix}\boldsymbol{u}_h\\ \widetilde{\boldsymbol{u}}_f\end{bmatrix}, \qquad \widetilde{\boldsymbol{u}}_f = \boldsymbol{u}\left(\boldsymbol{E}\boldsymbol{v}(\boldsymbol{u}_h)\right), \qquad \boldsymbol{f}^* = \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_f^+, \widetilde{\boldsymbol{u}}_f\right). \end{aligned}$$

Here, $\widetilde{\boldsymbol{u}}_f^+$ denotes some exterior state which can be used to enforce boundary conditions or coupling between different elements, and $\circ$ denotes the matrix Hadamard product.

Maybe add proof here and show how rest of proofs just rely on SBP and $\boldsymbol{Q}_h\mathbf{1} = \mathbf{0}$?. For proofs of high order accuracy and entropy conservation, we refer the reader to [1].

The matrix $\boldsymbol{F}$ is formed by computing $\boldsymbol{F}_{ij} = \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right)$ on the fly. Each entry requires evaluating the entropy conservative numerical flux $\boldsymbol{f}_S$, resulting in a large number of floating point operations [16]. However, due to properties of the Hadamard product in (5), flux evaluations are only necessary between pairs of nodes corresponding to non-zero entries of $\boldsymbol{Q}_h$. We will revisit this point later in the construction of entropy stable schemes on non-conforming meshes.

3.2 Entropy conservation formulations in higher dimensions

The formulation (2) can be extended to higher dimensions through a tensor product construction. For simplicity, we illustrate this in 2D (the extension to 3D is straightforward). Let $\boldsymbol{M}_{1\mathrm{D}}, \boldsymbol{Q}_{1\mathrm{D}}$ denote one-dimensional SBP norm (mass) and differentiation matrices, and let $\boldsymbol{E}_{1\mathrm{D}}$ denote the 1D face interpolation matrix. We define multi-dimensional mass and differentiation matrices in terms of Kronecker products

$$\widehat{\boldsymbol{Q}}_1 = \boldsymbol{Q}_{1\mathrm{D}} \otimes \boldsymbol{M}_{1\mathrm{D}}, \qquad \widehat{\boldsymbol{Q}}_2 = \boldsymbol{M}_{1\mathrm{D}} \otimes \boldsymbol{Q}_{1\mathrm{D}}, \qquad \widehat{\boldsymbol{M}} = \boldsymbol{M}_{1\mathrm{D}} \otimes \boldsymbol{M}_{1\mathrm{D}}.$$

We also construct 2D face interpolation matrices from Kronecker products. Let $\boldsymbol{E}_{1\mathrm{D}}$ denote the one-dimensional face interpolation matrix, and let $\boldsymbol{B}_{1\mathrm{D}}$ denote the 1D boundary matrix in (3). For a specific ordering of the face points, the two-dimensional face interpolation matrix $\boldsymbol{E}$ and reference boundary matrices $\widehat{\boldsymbol{B}}_1, \widehat{\boldsymbol{B}}_2$ are given by

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{E}_{1\mathrm{D}} \otimes \boldsymbol{I}_2 \\ \boldsymbol{I}_2 \otimes \boldsymbol{E}_{1\mathrm{D}} \end{bmatrix}, \qquad \widehat{\boldsymbol{B}}_1 = \begin{bmatrix} \boldsymbol{B}_{1\mathrm{D}} \otimes \boldsymbol{M}_{1\mathrm{D}} & \\ & \boldsymbol{0} \end{bmatrix}, \qquad \widehat{\boldsymbol{B}}_2 = \begin{bmatrix} \boldsymbol{0} & \\ & \boldsymbol{M}_{1\mathrm{D}} \otimes \boldsymbol{B}_{1\mathrm{D}} \end{bmatrix}, \tag{6}$$

where $\boldsymbol{I}_2$ is the 2×2 identity matrix. Recall that $\boldsymbol{M}_{1\mathrm{D}}$ is the diagonal matrix of quadrature weights, so $\widehat{\boldsymbol{B}}_i$ are diagonal.

The 2D differentiation and interpolation matrices $\widehat{\boldsymbol{Q}}_i, \boldsymbol{E}$ can now be used to construct 2D hybridized SBP operators. Let $\widehat{\boldsymbol{Q}}_{i,h}$ denote the hybridized SBP operator for the ith coordinate on the reference element, where $\widehat{\boldsymbol{Q}}_{i,h}$ is defined as

$$\widehat{\boldsymbol{Q}}_{i,h} = \frac{1}{2} \begin{bmatrix} \widehat{\boldsymbol{Q}}_i - \widehat{\boldsymbol{Q}}_i^T & \boldsymbol{E}^T \widehat{\boldsymbol{B}}_i \\ -\widehat{\boldsymbol{B}}_i \boldsymbol{E} & \widehat{\boldsymbol{B}}_i \end{bmatrix}.$$

These matrices satisfy the multi-dimensional SBP property

$$\widehat{\boldsymbol{Q}}_{i,h} + \widehat{\boldsymbol{Q}}_{i,h}^T = \begin{bmatrix} \boldsymbol{0} & \\ & \widehat{\boldsymbol{B}}_i \end{bmatrix}. \tag{7}$$

The construction of 3D differentiation and interpolation matrices proceeds similarly. Then, a d-dimensional entropy conservative formulation on the reference element $[-1,1]^d$ is given as

$$\widehat{\boldsymbol{M}} \frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{dt}} + \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{E} \end{bmatrix}^T \sum_{i=1}^{d} \left(\widehat{\boldsymbol{Q}}_{i,h} \circ \boldsymbol{F}_i \right) \boldsymbol{1} + \boldsymbol{E}^T \widehat{\boldsymbol{B}}_i \left(\boldsymbol{f}_i^* - \boldsymbol{f}_i(\widetilde{\boldsymbol{u}}_f) \right) = 0 \tag{8}$$

$$\left(\boldsymbol{F}_i \right)_{jk} = \boldsymbol{f}_S^i \left(\widetilde{\boldsymbol{u}}_j, \widetilde{\boldsymbol{u}}_k \right) \tag{9}$$

where $\boldsymbol{f}_S^i$ is the numerical flux for the ith coordinate direction, and $\widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{u}}_f$ are defined as in (5).

Finally, we can extend this formulation to the multi-element case. Suppose the domain Ω is decomposed into non-overlapping elements D^k which are images of the reference mapping, such that D^k is a differentiable mapping of the reference element $\widehat{D} = [-1,1]^d$. Let $\boldsymbol{x}_i$ denote the ith physical coordinate on D^k and let $\widehat{\boldsymbol{x}}_j$ denote the jth reference coordinate. The mapping between reference and physical element induces scaled geometric terms

$$G_{ij} = J \frac{\partial \widehat{\boldsymbol{x}}_j}{\partial \boldsymbol{x}_i},$$

where J denotes the determinant of the Jacobian of the physical-to-reference mapping. We refer to J as the Jacobian from here onwards. These geometric terms also relate the normal vectors on reference and physical elements. Let $\widehat{\boldsymbol{n}} \widehat{J}_f$ denote the outward normal vector on $\widehat{D}$, scaled by the reference face Jacobians. Then, the (scaled) physical normal vectors on D^k are related to $\widehat{\boldsymbol{n}} \widehat{J}_f$ through

$$\boldsymbol{n} J_f = G \widehat{\boldsymbol{n}} \widehat{J}_f.$$

We assume that the mesh is watertight, such that the scaled normal vectors are equal and opposite across shared faces between two elements.

We can now assemble physical mass and differentiation matrices on D^k from reference mass and differentiation matrices using the chain rule. Let $\widehat{\boldsymbol{Q}}_{j,h}$ denote the jth reference differentiation matrix on $\widehat{D}$. We can define the physical differentiation matrices $\boldsymbol{Q}_{i,h}$ on D^k via the skew-symmetric splitting

$$\boldsymbol{Q}_{i,h} = \frac{1}{2}\sum_{j=1}^{d} \operatorname{diag}\left(\boldsymbol{g}_{ij}\right)\widehat{\boldsymbol{Q}}_{j,h} + \widehat{\boldsymbol{Q}}_{j,h}\operatorname{diag}\left(\boldsymbol{g}_{ij}\right),$$

where $\boldsymbol{g}_{ij}$ denotes the vector of values of $G_{ij} = J\frac{\partial \widehat{\boldsymbol{x}}_j}{\partial \boldsymbol{x}_i}$ at both volume and surface points. We also assume that the geometric terms $\boldsymbol{g}_{ij}$ are constructed such that they satisfy a discrete geometric conservation law (GCL)

$$\sum_{j=1}^{d} \widehat{\boldsymbol{Q}}_{j,h}\boldsymbol{g}_{ij} = \boldsymbol{0}.$$

The discrete GCL is satisfied automatically for isoparametric mappings in 2D, and there exist several techniques to enforce the satisfaction of a discrete GCL on various three-dimensional domains [7, 8, 17–20].

Because the reference matrices $\widehat{\boldsymbol{Q}}_{j,h}$ satisfy the SBP property (7), one can show [8] that the physical matrices $\boldsymbol{Q}_{i,h}$ satisfy a physical SBP property.

Lemma 1

$$\boldsymbol{Q}_{i,h} + \boldsymbol{Q}_{i,h}^T = \begin{bmatrix} \boldsymbol{0} & \\ & \boldsymbol{B}_i \end{bmatrix}, \qquad \boldsymbol{B}_i = \operatorname{diag}\left(\boldsymbol{n}_i \circ \boldsymbol{w}_f\right),$$

where $\boldsymbol{n}_i$ is a vector whose entries consist of values of the ith component of the scaled normals $n_i J_f$ at face points, and $\boldsymbol{w}_f$ is the vector of face quadrature weights w_i.

Then, an entropy conservative scheme is given by

$$\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{dt}} + \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{E} \end{bmatrix}^T \sum_{i=1}^{d} \left(\boldsymbol{Q}_{i,h} \circ \boldsymbol{F}_i\right)\boldsymbol{1} + \boldsymbol{E}^T\boldsymbol{B}_i\left(\boldsymbol{f}_i^* - \boldsymbol{f}_i(\widetilde{\boldsymbol{u}}_f)\right) = 0 \tag{10}$$

where $\boldsymbol{F}_i$ is defined as in (8). Here, the exterior state $\widetilde{\boldsymbol{u}}_f$ used to evaluate the numerical flux $\boldsymbol{f}_i^*$ corresponds either to interface values on a neighboring element or an exterior state used to enforce boundary conditions.

All entropy conservative schemes described here can be made entropy stable by introducing entropy dissipation through mechanisms such as physical or artificial viscosity [21, 22]. We introduce dissipation by incorporating a penalization term into the interface flux [23]. For example, Lax-Friedrichs dissipation can be added by modifying the interface flux $\boldsymbol{f}_i^*$

$$\boldsymbol{f}_i^* \Longrightarrow \boldsymbol{f}_i^* - \frac{\lambda}{2}[\![\widetilde{\boldsymbol{u}}_f]\!], \qquad [\![\widetilde{\boldsymbol{u}}_f]\!] = \widetilde{\boldsymbol{u}}_f^+ - \widetilde{\boldsymbol{u}}_f,$$

where λ is an estimate of the maximum wave speed [2, 6].

4 Non-conforming meshes

Section 3 describes the construction of entropy stable schemes on geometrically conforming meshes, where each element shares at most one neighbor across a face. We extend the construction of stable schemes to meshes containing geometric non-conformity, where an element can share a face with two or more neighboring elements. To ensure stability, the coupling conditions imposed at this non-conforming face must be handled appropriately. For DG discretizations, this is most naturally achieved by combining composite quadrature rules on non-conforming faces with appropriate evaluations of volume terms [19]. However, for entropy stable schemes, the naive use of composite quadrature at non-conforming interfaces can significantly increase the computational cost.

For quad and hex elements under tensor product volume quadrature, the differentiation matrices are Kronecker products of 1D differentiation matrices and diagonal mass matrices, which result in sparse operators. As a result, flux evaluations are only required between "lines" of volume nodes [1, 4]. Flux evaluations also follow the sparsity pattern of the matrix $\boldsymbol{E}$, which maps from volume nodes to surface nodes. For Lobatto or Gauss collocation methods on conforming quadrilateral and hexahedral meshes, $\boldsymbol{E}$ is sparse if the surface quadrature nodes are aligned with volume quadrature nodes [1]. In such cases, flux evaluations are required only between lines of volume nodes and adjacent surface nodes, as shown in Figure 1a.

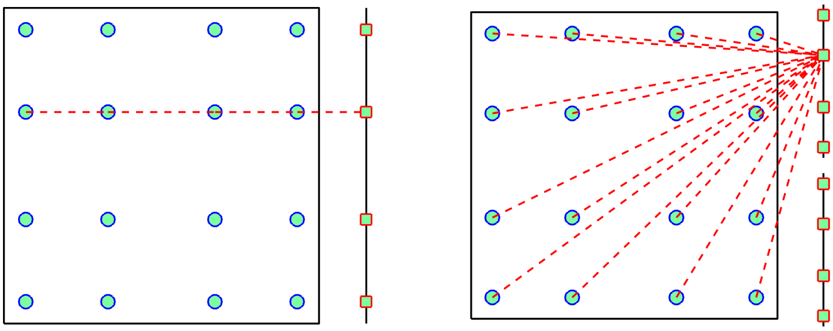
Composite quadrature rules on non-conforming meshes, however, are not aligned with volume nodes. Suppose that D^k is a quadrilateral element with two neighbors across each face, such that every face is non-conforming and utilizes a composite quadrature rule. The entropy conservative formulation (10) can be extended to the non-conforming case by redefining the interpolation matrix $\boldsymbol{E}$ as the matrix which interpolates from volume nodes to composite surface nodes. However, this version of $\boldsymbol{E}$ is fully dense, and evaluating the formulation (10) requires flux evaluations between each surface node and *all* volume nodes, as illustrated in Figure 1b. This greatly increases computational costs, especially at high orders of approximation.

5 An entropy stable mortar formulation

The goal of this work is to reduce computational costs for Gauss collocation schemes in the presence of non-conforming interfaces. This can be done by treating composite quadrature nodes as a layer of "mortar" nodes which are coupled directly to surface nodes, but not directly to the volume nodes, as illustrated in Figure 2. This results in modifications of the matrices involved in the entropy stable formulation (10). These modifications preserve both high order accuracy and entropy stablility, and yield an implementation which is identical to that of (10) except for a face-local correction to the numerical flux.

We introduce new matrices on the reference element $\widehat{D}$. Let $\widehat{\boldsymbol{x}}$ denote volume collocation points, and let $\widehat{\boldsymbol{x}}_f$ denote face (surface) points on the boundary $\partial\widehat{D}$. To simplify notation, we assume from this point onwards that the nodes are ordered face-by-face. In 2D, this implies that $\widehat{\boldsymbol{x}}_f$ is

$$\widehat{\boldsymbol{x}}_f = \left[\widehat{\boldsymbol{x}}_{f,1}\ \widehat{\boldsymbol{x}}_{f,2}\ \widehat{\boldsymbol{x}}_{f,3}\ \widehat{\boldsymbol{x}}_{f,4}\right]^T, \tag{11}$$



(a) Flux evaluations required for aligned surface nodes

(b) Flux evaluations required for non-aligned surface nodes

Fig. 1: Nodes between which flux evaluations are required for Gauss nodes. Aligned surface nodes (conforming interfaces) require evaluations between each surface node and a line of volume nodes, while non-aligned surface nodes (non-conforming interfaces) require flux evaluations between a surface node and *all* volume nodes.

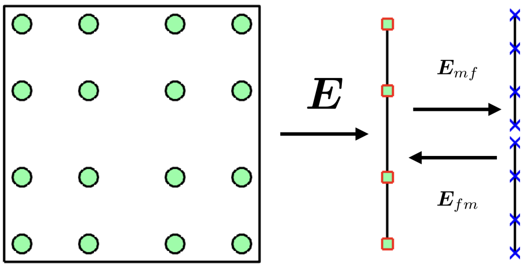


Fig. 2: Illustration of mortar operators for a Gauss collocation scheme. The matrix $\boldsymbol{E}$ maps from volume quadrature points to surface quadrature points, $\boldsymbol{E}_m$ maps from surface to mortar surface points, and $\widetilde{\boldsymbol{E}}_m$ maps from mortar surface points to surface points.

where $\widehat{\boldsymbol{x}}_{f,j}$ denotes the vector of 1D nodal positions on the jth face of the reference quadrilateral $[-1,1]^2$. Recall that the matrix $\boldsymbol{E}$ as defined in (6) interpolates from volume collocation points $\widehat{\boldsymbol{x}}$ to surface points $\widehat{\boldsymbol{x}}_f$; the ordering (11) simply corresponds to a permutation of the interpolation matrix $\boldsymbol{E}$.

We now introduce a second set of N_m mortar points $\widehat{\boldsymbol{x}}_m$ on a face of the reference element. We assume these points also correspond to quadrature nodes with corresponding weights $\boldsymbol{w}_m$. For example, mortar nodes can be composite Gauss or Lobatto quadrature nodes (for h non-conforming interfaces), higher degree Gauss or Lobatto nodes (for p non-conforming interfaces), or identical to the surface nodes $\widehat{\boldsymbol{x}}_f$ (non-mortar interface). For simplicity of notation, we assume that each face has the same set of mortar nodes; however, it is straightforward to extend this to the case when the mortar nodes vary face-by-face.

We define the interpolation matrix $\boldsymbol{E}_{mf}$ as the operator which maps values at surface nodes to values at mortar nodes. In 2D, since the face of a quadrilateral is a 1D line, we can define $\boldsymbol{E}_{mf}$ as the block matrix acting on the surface nodes on the four faces

$$\boldsymbol{E}_{mf} = \boldsymbol{I}_{4\times 4} \otimes \boldsymbol{E}_m,$$

where $\boldsymbol{I}_{4\times 4}$ is the 4-by-4 identity matrix and $\boldsymbol{E}_m$ is the mortar interpolation matrix over the reference face (interval) $[-1, 1]$. The matrix $\boldsymbol{E}_m$ is defined in terms of $\widehat{x}_{m,i}$, the mortar nodes mapped to the reference interval $[-1, 1]$,

$$\boldsymbol{E}_{mf} = \ell_j(\widehat{x}_{m,i}), \qquad 1 \leq i \leq N_m, \quad 1 \leq j \leq (N+1).$$

We can define an analogous matrix $\boldsymbol{E}_{fm}$ which maps from the mortar nodes back to the surface nodes. We first define the face mass matrix as the block diagonal matrix whose blocks are 1D diagonal mass matrices

$$\boldsymbol{M}_f = \boldsymbol{I}_{4\times 4} \otimes \boldsymbol{M}_{1\mathrm{D}}.$$

These matrices are defined in 2D for simplicity, but are straightforward to extend to 3D. The matrix $\boldsymbol{E}_{fm}$ can now be defined through a quadrature-based L^2 projection

$$\boldsymbol{E}_{fm} = \boldsymbol{M}_f^{-1} \boldsymbol{E}_{mf}^T \boldsymbol{M}_m, \qquad \boldsymbol{M}_m = \boldsymbol{I}_{4\times 4} \otimes \operatorname{diag}\left(\boldsymbol{w}_m\right).$$

Finally, we introduce diagonal boundary matrices on surface and mortar nodes

$$\widehat{\boldsymbol{B}}_{i,f} = \operatorname{diag}\left(\widehat{\boldsymbol{n}}_{i,f}\right) \boldsymbol{M}_f \qquad \widehat{\boldsymbol{B}}_{i,m} = \operatorname{diag}\left(\widehat{\boldsymbol{n}}_{i,m}\right) \boldsymbol{M}_m, \tag{12}$$

where $\widehat{\boldsymbol{n}}_{i,f}, \widehat{\boldsymbol{n}}_{i,m}$ are vectors containing components of the scaled reference normals at face and mortar points, respectively.

Note that $\boldsymbol{E}_{fm}$ exactly recovers polynomials of a certain degree on the reference face, where the degree of the polynomial is related to the accuracy of the surface and mortar quadratures.

Lemma 2 *Suppose the face (surface) quadrature is exact for polynomials of degree $N + M_f$ and mortar quadratures are exact for degree $N + M_m$ polynomials. Then, $\boldsymbol{E}_{fm}$ exactly recovers polynomials of degree* $\min(N, M_f, M_m)$.

Proof Let $u(\boldsymbol{x})$ be a polynomial of degree $\min(N, M_f, M_m)$ or less, and let $\boldsymbol{u}_f$ be its values on the face (surface) nodes. Then, $\boldsymbol{u}_m = \boldsymbol{E}_{mf}\boldsymbol{u}_f$ are the interpolated values of the polynomial on the mortar nodes. Applying $\boldsymbol{E}_{fm}$ yields

$$\boldsymbol{E}_{fm}\boldsymbol{u}_m = \boldsymbol{M}_f^{-1}\boldsymbol{E}_{mf}^T\boldsymbol{M}_m\boldsymbol{E}_{mf}\boldsymbol{u}_f.$$

The entries of $\boldsymbol{E}_{mf}^T\boldsymbol{M}_m\boldsymbol{E}_{mf}$ are integrals of products of Lagrange basis functions with $u(\boldsymbol{x})$, where all integrals are approximated using mortar quadrature.

Since u is degree $\min(N, M_f, M_m)$ and each Lagrange basis function is degree N, the integrand is computed exactly under the mortar quadrature. Moreover, since u is degree $\min(N, M_f, M_m)$, it is also computed exactly using the face (surface) quadrature. Thus, $\boldsymbol{E}_{mf}^T\boldsymbol{M}_m\boldsymbol{E}_{mf} = \boldsymbol{M}_f\boldsymbol{u}_f$ by exactness of the face and mortar quadratures, and $\boldsymbol{E}_{fm}\boldsymbol{u}_m = \boldsymbol{u}_f$. □

5.1 Mortar-based hybridized SBP operators

The matrices $\boldsymbol{E}_{mf}, \boldsymbol{E}_{fm}, \widehat{\boldsymbol{B}}_{i,f}, \widehat{\boldsymbol{B}}_{i,m}$ can now be used to construct SBP operators which involve mortar nodes. We define the mortar-based hybridized SBP operator on the reference element $\widehat{\boldsymbol{Q}}_{i,m}$ to be

$$\widehat{\boldsymbol{Q}}_{i,m} = \frac{1}{2}\begin{bmatrix} \widehat{\boldsymbol{Q}}_i - \widehat{\boldsymbol{Q}}_i^T & \boldsymbol{E}^T\widehat{\boldsymbol{B}}_{i,f} & \\ -\widehat{\boldsymbol{B}}_{i,f}\boldsymbol{E} & & \widehat{\boldsymbol{B}}_{i,f}\boldsymbol{E}_{fm} \\ & -\widehat{\boldsymbol{B}}_{i,m}\boldsymbol{E}_{mf} & \widehat{\boldsymbol{B}}_{i,m} \end{bmatrix}. \tag{13}$$

Since it is not clear at first glance how the hybridized operator $\widehat{\boldsymbol{Q}}_{i,h}$ or its mortar-based variant $\widehat{\boldsymbol{Q}}_{i,m}$ can be used to perform differentiation, we review intuitive explanations of each operator. Let $\boldsymbol{u}$ denote basis coefficients for some function $u(\boldsymbol{x})$, and let f, g denote two functions. Suppose $\boldsymbol{u}$ satisfies the following matrix system involving the hybridized operator $\widehat{\boldsymbol{Q}}_{i,h}$

$$\widehat{\boldsymbol{M}}\boldsymbol{u} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{E} \end{bmatrix}^T \operatorname{diag}(\boldsymbol{f})\,\widehat{\boldsymbol{Q}}_{i,h}\operatorname{diag}(\boldsymbol{g}), \qquad \boldsymbol{f} = \begin{bmatrix} f(\widehat{\boldsymbol{x}}) \\ f(\widehat{\boldsymbol{x}}_f) \end{bmatrix}, \qquad \boldsymbol{g} = \begin{bmatrix} g(\widehat{\boldsymbol{x}}) \\ g(\widehat{\boldsymbol{x}}_f) \end{bmatrix}. \tag{14}$$

It was shown in [2, 9] that this corresponds to a high order accurate approximation of $f\frac{\partial g}{\partial x_i}$. Let $\boldsymbol{v}$ denote basis coefficients for some arbitrary degree N polynomial test function. Multiplying (14) by $\boldsymbol{v}^T$, expanding out the different blocks of $\boldsymbol{Q}_{i,h}$, and using generalized summation by parts yields

$$\boldsymbol{M}\boldsymbol{u} = \operatorname{diag}(\boldsymbol{f})\,\boldsymbol{Q}_i\boldsymbol{g} + \frac{1}{2}\left(\operatorname{diag}(f(\widehat{\boldsymbol{x}}))\,\boldsymbol{E}^T + \boldsymbol{E}^T f(\widehat{\boldsymbol{x}}_f)\right)\boldsymbol{B}_i\left(g(\widehat{\boldsymbol{x}}_f) - \boldsymbol{E}g(\widehat{\boldsymbol{x}})\right).$$

The latter expression can be interpreted as a boundary correction term. Since $\boldsymbol{E}$ is a high order accurate boundary interpolation operator, this correction term vanishes if g is a degree N polynomial.

The mortar-based hybridized SBP operator can be interpreted similarly. Recall that $\boldsymbol{E}_{mf}$ interpolates from face nodes to mortar nodes. Thus, the matrix $\boldsymbol{E}_{mf}\boldsymbol{E}$ interpolates from volume nodes to mortar nodes. We can use this matrix to replicate (14) for the mortar-based operator $\widehat{\boldsymbol{Q}}_{i,m}$. Let $\boldsymbol{u}$ solve the following system

$$\widehat{\boldsymbol{M}}\boldsymbol{u} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{E} \\ \boldsymbol{E}_{mf}\boldsymbol{E} \end{bmatrix}^T \operatorname{diag}(\boldsymbol{f})\,\widehat{\boldsymbol{Q}}_{i,m}\operatorname{diag}(\boldsymbol{g}), \qquad \boldsymbol{f} = \begin{bmatrix} f(\widehat{\boldsymbol{x}}) \\ f(\widehat{\boldsymbol{x}}_f) \\ f(\widehat{\boldsymbol{x}}_m) \end{bmatrix}, \quad \boldsymbol{g} = \begin{bmatrix} g(\widehat{\boldsymbol{x}}) \\ g(\widehat{\boldsymbol{x}}_f) \\ g(\widehat{\boldsymbol{x}}_m) \end{bmatrix}. \tag{15}$$

Expanding out terms yields a similar expression involving multiple correction terms

$$\begin{aligned} \boldsymbol{M}\boldsymbol{u} = {} & \operatorname{diag}(\boldsymbol{f})\,\boldsymbol{Q}_i\boldsymbol{g} \\ & + \frac{1}{2}\operatorname{diag}(f(\widehat{\boldsymbol{x}}))\,\boldsymbol{E}^T\boldsymbol{B}_{i,f}\left(g(\widehat{\boldsymbol{x}}_f) - \boldsymbol{E}g(\widehat{\boldsymbol{x}})\right) \\ & + \frac{1}{2}\boldsymbol{E}^T\operatorname{diag}\left(f(\widehat{\boldsymbol{x}}_f)\right)\boldsymbol{B}_{i,f}\left(\boldsymbol{E}_{fm}g(\widehat{\boldsymbol{x}}_m) - \boldsymbol{E}g(\widehat{\boldsymbol{x}})\right) \\ & + \frac{1}{2}\boldsymbol{E}^T\boldsymbol{E}_{mf}^T\operatorname{diag}(f(\widehat{\boldsymbol{x}}_m))\,\boldsymbol{B}_{i,m}\left(g(\widehat{\boldsymbol{x}}_m) - \boldsymbol{E}_{mf}g(\widehat{\boldsymbol{x}}_f)\right) \end{aligned} \tag{16}$$

We can show that these correction terms vanish for polynomial functions g.

Lemma 3 *Suppose that the face (surface) quadrature is exact for polynomials of degree $N+M_f$ and mortar quadratures are exact for degree $N+M_m$ polynomials. Let $f=1$ and g be a degree $\min(N, M_f, M_m)$ polynomial. Then, the equations (15) and (16) are exact.*

Proof Following [2], if $f=1$ and $g(\boldsymbol{x})$ is a degree N polynomial, then (16) is exact if the correction terms vanish. The first and third correction terms are zero by the fact that $\boldsymbol{E}, \boldsymbol{E}_{mf}$ are degree N interpolation operators, and the second correction term vanishes by Lemma 2. □

Finally, we note that the mortar-based hybridized SBP operators satisfy a summation by parts property.

Lemma 4 *Let $\boldsymbol{Q}_{i,m}$ be defined as in (13). Then,*

$$\boldsymbol{Q}_{i,m} + \boldsymbol{Q}_{i,m}^T = \begin{bmatrix} \boldsymbol{0} & & \\ & \boldsymbol{0} & \\ & & \boldsymbol{B}_{i,m} \end{bmatrix}, \qquad \boldsymbol{Q}_{i,m}\boldsymbol{1} = \boldsymbol{0}.$$

Proof The first property holds if

$$\widehat{\boldsymbol{B}}_{i,f}\boldsymbol{E}_{fm} = \left(\widehat{\boldsymbol{B}}_{i,m}\boldsymbol{E}_{mf}\right)^T = \boldsymbol{E}_{mf}^T\widehat{\boldsymbol{B}}_{i,m}, \tag{17}$$

where we have used that $\widehat{\boldsymbol{B}}_{i,m}$ is diagonal. Recall that $\boldsymbol{E}_{fm} = \boldsymbol{M}_f^{-1}\boldsymbol{E}_{mf}^T\boldsymbol{M}_m$. Then, by the definition of $\boldsymbol{B}_{i,m}, \boldsymbol{B}_{i,h}$ in (12), we have that

$$\widehat{\boldsymbol{B}}_{i,f}\boldsymbol{E}_{fm} = \operatorname{diag}\left(\widehat{\boldsymbol{n}}_{i,f}\right)\boldsymbol{M}_f\boldsymbol{M}_f^{-1}\boldsymbol{E}_{mf}^T\boldsymbol{M}_m = \operatorname{diag}\left(\widehat{\boldsymbol{n}}_{i,f}\right)\boldsymbol{E}_{mf}^T\boldsymbol{M}_m.$$

where we have used that $\boldsymbol{M}_m$ is diagonal. Then, since the scaled outward normals $\widehat{n}_i$ are constant over each face of the reference element and $\boldsymbol{E}_{mf}, \boldsymbol{M}_m$ are block matrices (with each block corresponding to a face), we have that

$$\boldsymbol{E}_{mf}^T\boldsymbol{M}_m\operatorname{diag}\left(\widehat{\boldsymbol{n}}_{i,m}\right) = \boldsymbol{E}_{mf}^T\boldsymbol{B}_{i,m}.$$

□

We can now construct an entropy stable formulation on the reference element using the mortar-based hybridized SBP operator

$$\widehat{\boldsymbol{M}}\frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{d}t} + \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{E} \\ \boldsymbol{E}_{mf}\boldsymbol{E} \end{bmatrix}^T \sum_{i=1}^d \left(\widehat{\boldsymbol{Q}}_{i,m}\circ\boldsymbol{F}_i\right)\boldsymbol{1} + \boldsymbol{E}_{mf}^T\widehat{\boldsymbol{B}}_{i,m}\left(\boldsymbol{f}_i^* - \boldsymbol{f}_i(\widetilde{\boldsymbol{u}}_m)\right) = 0 \tag{18}$$

$$\widetilde{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}_h \\ \widetilde{\boldsymbol{u}}_f \\ \widetilde{\boldsymbol{u}}_m \end{bmatrix}, \qquad \widetilde{\boldsymbol{u}}_m = \boldsymbol{u}\left(\boldsymbol{E}_{mf}\boldsymbol{E}\boldsymbol{v}(\boldsymbol{u}_h)\right), \qquad \boldsymbol{f}^* = \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_m^+, \widetilde{\boldsymbol{u}}_m\right), \tag{19}$$

where $\left(\boldsymbol{F}_i\right)_{jk} = \boldsymbol{f}_S^i\left(\widetilde{\boldsymbol{u}}_j, \widetilde{\boldsymbol{u}}_k\right)$ for all indices j,k of $\widetilde{\boldsymbol{u}}$. This can be extended to mapped elements

6 A mortar-based implementation

While the formulation presented above is convenient for analysis, it is computationally expensive to implement. However, using properties of the mortar matrices $\boldsymbol{E}_m, \widetilde{\boldsymbol{E}}_m$, we can rewrite the above formulation in a way which reflects a traditional mortar-based finite element implementation. In other words, we wish to implement (**??**) such that the only modification from the implementations in [2] is an intermediate step on mortar faces.

We first note that, since $\boldsymbol{E}_m$ maps from surface nodes to mortar nodes, $\boldsymbol{V}_m = \boldsymbol{E}_m\boldsymbol{E}$. Thus, we can rewrite the numerical flux contribution as

$$\begin{aligned}\boldsymbol{V}_m^T\widetilde{\boldsymbol{B}}_i\boldsymbol{f}_i^* &= \boldsymbol{E}^T\boldsymbol{E}_m^T\boldsymbol{M}_m\operatorname{diag}\left(\widehat{\boldsymbol{n}}_i\right)\boldsymbol{f}_i^* = \boldsymbol{E}^T\boldsymbol{P}_f^T\boldsymbol{T}_m^T\boldsymbol{M}_m\operatorname{diag}\left(\widehat{\boldsymbol{n}}_i\right)\boldsymbol{f}_i^*\\ &= \boldsymbol{E}^T\boldsymbol{W}_f\operatorname{diag}\left(\widehat{\boldsymbol{n}}_i\right)\boldsymbol{T}_m\boldsymbol{M}_m^{-1}\boldsymbol{T}_m^T\boldsymbol{M}_m\boldsymbol{f}_i^* = \boldsymbol{E}^T\boldsymbol{B}_i\widetilde{\boldsymbol{E}}_m\boldsymbol{f}_i^*.\end{aligned}$$

We can similarly reformulate the remaining mortar contributions within (**??**). We decompose $\boldsymbol{F}_S$ into interactions between volume nodes, surface nodes, and mortar nodes

$$\boldsymbol{F}_S = \begin{bmatrix}\boldsymbol{F}_S^{vv} & \boldsymbol{F}_S^{vs} & \boldsymbol{F}_S^{vm}\\ \boldsymbol{F}_S^{sv} & \boldsymbol{F}_S^{ss} & \boldsymbol{F}_S^{sm}\\ \boldsymbol{F}_S^{mv} & \boldsymbol{F}_S^{ms} & \boldsymbol{F}_S^{mm}\end{bmatrix}$$

Then, the on-element contributions to (**??**) can be expanded as follows

$$\begin{aligned}&\begin{bmatrix}\boldsymbol{V}_q\\ \boldsymbol{E}\\ \boldsymbol{V}_m\end{bmatrix}^T\left(\begin{bmatrix}\boldsymbol{Q}_i-\boldsymbol{Q}_i^T & \boldsymbol{E}^T\boldsymbol{B}_i & \\ -\boldsymbol{B}_i\boldsymbol{E} & & \boldsymbol{B}_i\widetilde{\boldsymbol{E}}_m\\ & -\widetilde{\boldsymbol{B}}_i\boldsymbol{E}_m & \end{bmatrix}\circ\boldsymbol{F}_S\right)\boldsymbol{1}\\ &=\begin{bmatrix}\boldsymbol{V}_q\\ \boldsymbol{E}\end{bmatrix}^T\left(\begin{bmatrix}\boldsymbol{Q}_i-\boldsymbol{Q}_i^T & \boldsymbol{E}^T\boldsymbol{B}_i\\ -\boldsymbol{B}_i\boldsymbol{E} & \end{bmatrix}\circ\boldsymbol{F}_S\right)\boldsymbol{1}\\ &+\boldsymbol{E}^T\left(\boldsymbol{B}_i\widetilde{\boldsymbol{E}}_m\circ\boldsymbol{F}_S^{sm}\right)\boldsymbol{1}-\boldsymbol{V}_m^T\left(\widetilde{\boldsymbol{B}}_i\boldsymbol{E}_m\circ\boldsymbol{F}_S^{ms}\right)\boldsymbol{1}\end{aligned}$$

Since multiplication by diagonal matrices $\boldsymbol{B}_i, \widetilde{\boldsymbol{B}}_i$ is associative under the Hadamard product, the latter terms can be rewritten as

$$\begin{aligned}\boldsymbol{E}^T\left(\boldsymbol{B}_i\widetilde{\boldsymbol{E}}_m\circ\boldsymbol{F}_S^{sm}\right)\boldsymbol{1} &= \boldsymbol{E}^T\boldsymbol{B}_i\left(\widetilde{\boldsymbol{E}}_m\circ\boldsymbol{F}_S^{sm}\right)\boldsymbol{1}\\ \boldsymbol{V}_m^T\left(\widetilde{\boldsymbol{B}}_i\boldsymbol{E}_m\circ\boldsymbol{F}_S^{ms}\right)\boldsymbol{1} &= \boldsymbol{V}_m^T\widetilde{\boldsymbol{B}}_i\left(\boldsymbol{E}_m\circ\boldsymbol{F}_S^{ms}\right)\boldsymbol{1} = \boldsymbol{E}^T\boldsymbol{B}_i\widetilde{\boldsymbol{E}}_m\left(\boldsymbol{E}_m\circ\boldsymbol{F}_S^{ms}\right)\boldsymbol{1}\end{aligned}$$

Then, (**??**) can be rewritten as

$$\begin{aligned}\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}_N}{\mathrm{dt}}+\begin{bmatrix}\boldsymbol{V}_q\\ \boldsymbol{E}\end{bmatrix}^T\left(\begin{bmatrix}\boldsymbol{Q}_i-\boldsymbol{Q}_i^T & \boldsymbol{E}^T\boldsymbol{B}_i\\ -\boldsymbol{B}_i\boldsymbol{E} & \end{bmatrix}\circ\boldsymbol{F}_S\right)\boldsymbol{1}+\boldsymbol{E}^T\boldsymbol{B}_i\widetilde{\boldsymbol{f}}_i^* &= 0\\ \widetilde{\boldsymbol{f}}_i^* = \widetilde{\boldsymbol{E}}_m\boldsymbol{f}_i^*+\left(\widetilde{\boldsymbol{E}}_m\circ\boldsymbol{F}_S^{sm}\right)\boldsymbol{1}-\widetilde{\boldsymbol{E}}_m\left(\boldsymbol{E}_m\circ\boldsymbol{F}_S^{ms}\right)\boldsymbol{1}&\end{aligned} \tag{20}$$

If the surface and mortar nodes are identical, then

$$\boldsymbol{E}_m = \widetilde{\boldsymbol{E}}_m = \boldsymbol{I}, \qquad \left(\widetilde{\boldsymbol{E}}_m\circ\boldsymbol{F}_S^{sm}\right)\boldsymbol{1} = \left(\boldsymbol{E}_m\circ\boldsymbol{F}_S^{ms}\right)\boldsymbol{1}$$

and the correction terms within $\widetilde{\boldsymbol{f}}^*$ cancel out, and we recover the original scheme from [2].

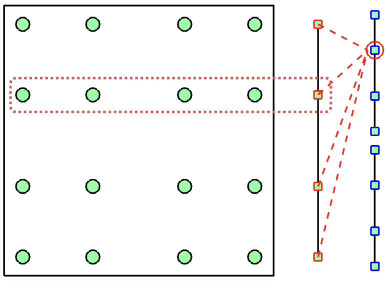


Fig. 3: Blah

Note that in (20), the only interactions between surface and mortar nodes occurs within the computation of $\widetilde{\boldsymbol{f}}_i^*$, which can be done solely on mortar interfaces. The computational structure of

1. Compute volume contributions to the RHS
2. Communicate surface contributions to mortar interfaces and compute $\widetilde{\boldsymbol{f}}_i^*$.
3. Communicate $\widetilde{\boldsymbol{f}}_i^*$ back to each element.

7 Error estimates

Weak derivative vs strong derivative error estimates. SBP yields that the skew-symmetric form is equivalent to weak form.

Line DG approach can work, but it's flipped: we increase quadrature strength in the direction *orthogonal* to derivative instead. May improve accuracy over Lobatto by giving equivalence w/skew form.

8 Numerical experiments

Figure **??** shows preliminary results for the 2D compressible Euler equations by the PI on a manually constructed non-conforming quadrilateral mesh, with volume, surface, and mortar quadratures constructed from one-dimensional Lobatto or Gauss quadrature rules. The PI has verified that this preliminary implementation of the mortar-based formulation is discretely entropy stable. A preliminary accuracy analysis for the skew-symmetric formulation (**??**) also suggests that Lobatto quadrature should achieve an $O(h^N)$ rate of convergence and that Gauss quadrature should achieve an $O(h^{N+1})$ rate of convergence. Both predicted rates are confirmed in Figure **??**.

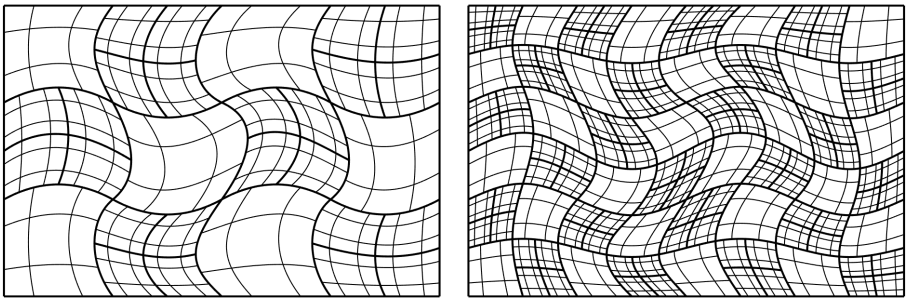
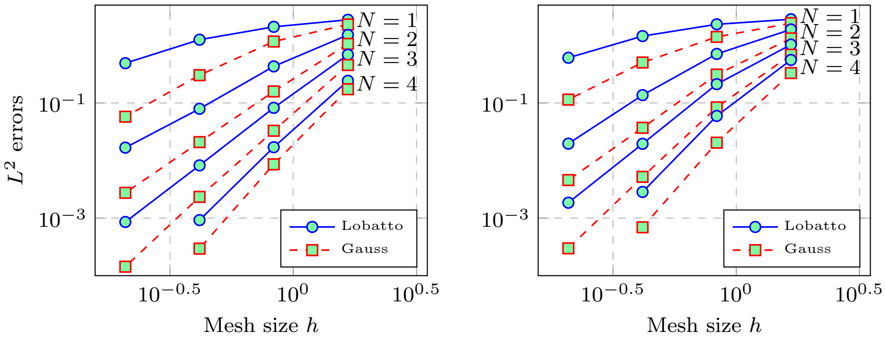


Fig. 4: Two $N = 3$ meshes used in the refinement and entropy conservation studies.



(a) Affine mesh

(b) Curved mesh

Fig. 5: Blah

8.1 Gauss Lobatto quadrature

8.2 Gauss quadrature

References

1. Jesse Chan, David C Fernandez, and Mark H Carpenter. Efficient entropy stable Gauss collocation methods. *arXiv preprint arXiv:1809.01178*, 2018.
2. Jesse Chan. On discretely entropy conservative and entropy stable discontinuous Galerkin methods. *Journal of Computational Physics*, 362:346 – 374, 2018.
3. Travis C Fisher and Mark H Carpenter. High-order entropy stable finite difference schemes for nonlinear conservation laws: Finite domains. *Journal of Computational Physics*, 252:518–557, 2013.
4. 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.
5. 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.

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. 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.
8. 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.
9. Jesse Chan. Skew-Symmetric Entropy Stable Modal Discontinuous Galerkin Formulations. *Journal of Scientific Computing*, 81(1):459–485, Oct 2019.
10. Christine Bernardi, Yvon Maday, and Anthony T Patera. Domain decomposition by the mortar element method. In *Asymptotic and numerical methods for partial differential equations with critical parameters*, pages 269–286. Springer, 1993.
11. Lucas Friedrich, Andrew R Winters, David C Del Rey Fernández, Gregor J Gassner, Matteo Parsani, and Mark H Carpenter. An entropy stable h/p non-conforming discontinuous Galerkin method with the summation-by-parts property. *Journal of Scientific Computing*, pages 1–37.
12. Constantine M Dafermos. *Hyperbolic conservation laws in continuum physics*. Springer, 2005.
13. David C Del Rey Fernández, Pieter D Boom, and David W Zingg. A generalized framework for nodal first derivative summation-by-parts operators. *Journal of Computational Physics*, 266:214–239, 2014.
14. 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.
15. Eitan Tadmor. The numerical viscosity of entropy stable schemes for systems of conservation laws. I. *Mathematics of Computation*, 49(179):91–103, 1987.
16. Niklas Wintermeyer, Andrew R Winters, Gregor J Gassner, and Timothy Warburton. An entropy stable discontinuous Galerkin method for the shallow water equations on curvilinear meshes with wet/dry fronts accelerated by GPUs. *Journal of Computational Physics*, 375:447–480, 2018.
17. PD Thomas and CK Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA journal*, 17(10):1030–1037, 1979.
18. David A Kopriva. Metric identities and the discontinuous spectral element method on curvilinear meshes. *Journal of Scientific Computing*, 26(3):301–327, 2006.
19. Jeremy E Kozdon and Lucas C Wilcox. An energy stable approach for discretizing hyperbolic equations with nonconforming discontinuous Galerkin methods. *Journal of Scientific Computing*, 76(3):1742–1784, 2018.
20. David A Kopriva, Florian J Hindenlang, Thomas Bolemann, and Gregor J Gassner. Free-Stream Preservation for Curved Geometrically Non-conforming Discontinuous Galerkin Spectral Elements. *Journal of Scientific Computing*, 79(3):1389–1408, 2019.

21. Eitan Tadmor and Weigang Zhong. Entropy stable approximations of Navier–Stokes equations with no artificial numerical viscosity. *Journal of Hyperbolic Differential Equations*, 3(03):529–559, 2006.
22. 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.
23. 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.