MORTAR-BASED ENTROPY STABLE DISCONTINUOUS GALERKIN METHODS

JESSE CHAN, DAVID C. DEL REY FERNANDEZ

1. Introduction. Boilerplate introduction on high order + stability

2. Mortar methods for hybrid and non-conforming meshes. This work is motivated by complications which arise when designing entropy stable couplings between elements which do not share the same boundary nodes. This can arise, for example, for hybrid and non-conforming meshes.

Hybrid meshes:. entropy stable DG methods on hybrid meshes were introduced in [1] using a skew-symmetric formulation. The resulting methods are stable for more arbitrary choices of surface quadrature, in particular when an SBP property may not hold.

Non-conforming meshes. On conforming meshes, it is most efficient to utilize both Gauss quadrature for volume integrals and Gauss quadrature for face or surface integrals. For solutions represented in terms of their values at tensor product volume Gauss nodes, extrapolation to face Gauss nodes can be done in an efficient line-by-line manner using one-dimensional interpolation matrices.

For non-conforming meshes, it can be advantageous to use composite Gauss quadratures on non-conforming interfaces. Reference JK and LCW's paper on full-side vs split side mortars. However, interpolating the solution at volume Gauss nodes to split-side Gauss nodes is no longer a one-dimensional operation.

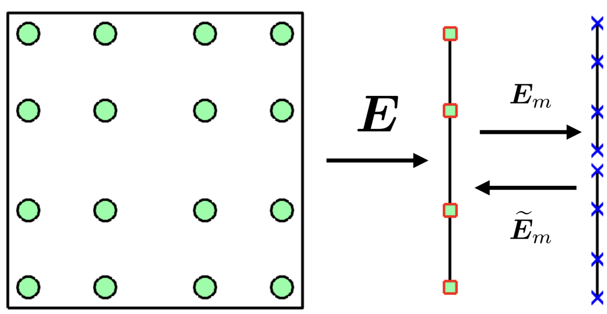

Fig. 1: Illustration of mortar operators. The matrix $\boldsymbol{E}$ maps from volume quadrature points to surface quadrature points, $\boldsymbol{E}_m$ maps from surface to mortar surface points, and $\widetilde{\boldsymbol{E}}_m$ maps from mortar surface points to surface points.

3. Mortar formulation. Let $\boldsymbol{V}_q$ and $\boldsymbol{V}_f$ denote interpolation matrices which evaluate at volume and surface quadrature points, respectively, and let $\boldsymbol{W}, \boldsymbol{W}_f$ denote diagonal matrices whose entries consist of volume and surface quadrature weights

add defs of matrices

Let $\boldsymbol{E} = \boldsymbol{V}_f \boldsymbol{P}_q$ denote the polynomial mapping which extrapolates values from volume quadrature points to values at surface quadrature points. We also define $\boldsymbol{B}_i$ as the diagonal matrix containing products of quadrature weights and the ith component of the outward normal vector $\widehat{\boldsymbol{n}}_i$

$$\boldsymbol{B}_i = \boldsymbol{W}_f \operatorname{diag}\left(\widehat{\boldsymbol{n}}_i\right).$$

An entropy stable skew-symmetric formulation can be given on the reference element $\widehat{D}$ as follows:

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

We incorporate mortars by modifying this formulation. Let $\boldsymbol{V}_m$ denote the matrix which maps volume quadrature points to values at surface *mortar* points. We also need to introduce interpolation operators $\boldsymbol{T}_f, \boldsymbol{T}_m$ for surface trace spaces. Here, $\boldsymbol{T}_f$ maps from polynomials on the surface $\partial\widehat{D}$ to surface quadrature points, while $\boldsymbol{T}_m$ maps from polynomials on $\partial\widehat{D}$ to mortar quadrature points. We can then define surface and mortar mass and projection matrices

$$\begin{aligned}\boldsymbol{M}_m &= \boldsymbol{T}_m^T\boldsymbol{W}_m\boldsymbol{T}_m, \qquad & \boldsymbol{P}_m &= \boldsymbol{M}_m^{-1}\boldsymbol{T}_m^T\boldsymbol{W}_m\\ \boldsymbol{M}_f &= \boldsymbol{T}_f^T\boldsymbol{W}_f\boldsymbol{T}_f, \qquad & \boldsymbol{P}_f &= \boldsymbol{M}_m^{-1}\boldsymbol{T}_f^T\boldsymbol{W}_f.\end{aligned}$$

Note that we have used the mortar mass matrix $\boldsymbol{M}_m$ in the above definition of the face projection. This is only necessary if the surface and mortar quadratures do not exactly integrate degree $2N$ polynomials. If both the surface and mortar quadratures are exact for polynomials of degree $2N$, then $\boldsymbol{M}_m = \boldsymbol{M}_f$, and no distinction is necessary between the two mass matrices.

Remark. *In order to show stability, we require that both $\boldsymbol{P}_f$ and $\boldsymbol{P}_m$ are defined using the same mass matrix. We have used the mortar mass matrix $\boldsymbol{M}_m$ here; however, we could also use the surface mass matrix $\boldsymbol{M}_f$ in both $\boldsymbol{P}_f$ and $\boldsymbol{P}_m$. We use the mortar mass matrix in this work, as the mortar quadrature is generally more accurate than the surface quadrature for our use cases (e.g. hybrid and non-conforming meshes).*

We can now define operators which map between surface and mortar quadrature points. Let $\boldsymbol{E}_m$ denote the map from surface to mortar points, and let $\widetilde{\boldsymbol{E}}_m$ denote the map from mortar to surface points. Both operators are defined through an L^2 projection to the trace space and interpolation to appropriate points

$$\boldsymbol{E}_m = \boldsymbol{T}_m\boldsymbol{P}_f, \qquad \widetilde{\boldsymbol{E}}_m = \boldsymbol{T}_f\boldsymbol{P}_m.$$

We also define a mortar boundary matrix

$$\widetilde{\boldsymbol{B}}_i = \boldsymbol{W}_m \operatorname{diag}\left(\widehat{\boldsymbol{n}}_i\right).$$

Then, a mortar-based formulation can be given as follows:

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

Here, we have appended an extra row and column to the decoupled SBP matrix, and the inter-element numerical flux $\boldsymbol{f}^*$ is now computed in terms of the *mortar* nodes.

We first note that

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

where we have used that $\boldsymbol{W}_m$ is diagonal and that the outward normal $\widehat{\boldsymbol{n}}_i$ on the reference element is constant over each face. This implies skew-symmetry of the decoupled SBP operator and entropy stability of the overall system.

Finish

4. 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 (1) such that the only modification from the implementations in [2] is a pre-processing step on mortar faces.

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

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

We can similarly reformulate the remaining mortar contributions within (1). 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 (1) can be expanded as follows

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

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

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

Then, (1) can be rewritten as

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

If the surface and mortar nodes are identical, then $\boldsymbol{E}_m = \widetilde{\boldsymbol{E}}_m = \boldsymbol{I}$ and $\left(\widetilde{\boldsymbol{E}}_m \circ \boldsymbol{F}_S^{sm}\right)\boldsymbol{1} = \left(\boldsymbol{E}_m \circ \boldsymbol{F}_S^{ms}\right)\boldsymbol{1}$. The correction terms within $\widetilde{\boldsymbol{f}}^*$ cancel out, and we recover the original scheme from [2].

Note that in (2), 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.

5. 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 GLL by giving equivalence w/skew form.

6. Numerical experiments.

6.1. Gauss Lobatto quadrature.

6.2. Gauss quadrature.

REFERENCES

[1] Jesse Chan. Skew-symmetric entropy stable discontinuous Galerkin formulations. 2018. In preparation.

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