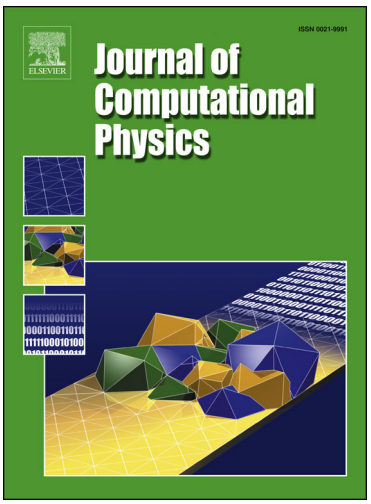
Journal Pre-proof



Efficient and error minimized coupling procedures for unstructured and moving meshes

Tomas Lundquist, Arnaud G. Malan, Jan Nordström

PII: S0021-9991(19)30863-0
DOI: https://doi.org/10.1016/j.jcp.2019.109158
Reference: YJCPH 109158

To appear in: *Journal of Computational Physics*

Received date: 21 June 2019
Revised date: 8 November 2019
Accepted date: 27 November 2019

Please cite this article as: T. Lundquist et al., Efficient and error minimized coupling procedures for unstructured and moving meshes, *J. Comput. Phys.* (2020), 109158, doi: https://doi.org/10.1016/j.jcp.2019.109158.

Highlights

- Automatic generation of interpolation operators for non-collocated couplings of meshes within a general summation-by-parts framework.
- A method-of-lines approach for moving meshes with a sliding interface.
- Issues of stability, conservation, minimal errors and computational efficiency are addressed.

Efficient and error minimized coupling procedures for unstructured and moving meshes

Tomas Lundquist[a], Arnaud G. Malan[b], Jan Nordström[c]

[a]*Industrial CFD Research Group, Department of Mechanical Engineering, University of Cape Town, South Africa (*tomas.lundquist@uct.ac.za*).*
[b]*Industrial CFD Research Group, Department of Mechanical Engineering, University of Cape Town, South Africa (*arnaud.malan@uct.ac.za*).*
[c]*Department of Mathematics, Computational Mathematics, Linköping University, Sweden (*jan.nordstrom@liu.se*).*

Abstract

We present a methodology for automatic generation and optimization of interpolation operators for the coupling of general non-collocated and/or moving numerical interfaces. The discrete equations are solved in a method-of-lines fashion by assuming volume preserving mesh motions. Interface interpolation errors are minimized effectively in a global least-squares sense, while satisfying strict stability conditions. The proposed automatic interface procedure is both more versatile and more accurate compared to previous techniques. We apply the new method to interfaces between hybrid meshes undergoing relative rigid body motion, demonstrating the stability, conservation and superior accuracy.

1. Introduction

Inner product preserving (IPP), also known as summation-by-parts (SBP) preserving, interpolation and projection operators have recently emerged as a robust and versatile tool in the design and analysis of interface couplings for non-collocated and hybrid meshes [1, 2, 3, 4, 5, 6, 7]. In [7], the present authors developed a unified theory for the stability and accuracy of IPP interpolation for both structured and unstructured meshes.

Despite these recent advances in IPP interface techniques, significant limitations still remain. Firstly, order reduction in accuracy is an issue for finely resolved calculations [7]. For diffusion dominated problems, this issue was

partly resolved in [8] by constructing two different sets of interpolation operators, but it is still unsolved for advection dominated problems. Secondly, some types of meshes remain more difficult to handle than others with the existing techniques. Examples of this include general unstructured meshes, but also the presence sliding motion between subdomains with structured meshes. The challenges are not only, or perhaps even primarily, of theoretical nature, but practical in the sense that interpolation operators must be generated for general mesh configurations. Finally, to the best of our knowledge, free-stream preservation and conservation has not yet been demonstrated on curved interfaces with an SBP approach.

In this paper we develop a comprehensive methodology for the generation and optimization of interpolation operators between stationary and sliding subdomains. The primary target is to automate the procedure as much as possible for general combinations of meshes and discretization methods. To this end, interpolation errors are minimized in a global least-squares sense, and stability can be ensured by satisfying IPP relations [1, 2, 3, 6, 7]. For interface meshes which are either stationary or undergoing rigid body motions, free-stream preservation and discrete conservation can also be achieved with the IPP automatic procedure.

The new global optimization procedure is highly versatile and can be applied to more or less arbitrary combinations of meshes and discretization methods of SBP type. However, the computational cost of optimization can be significant on a 3D mesh. As long as the mesh interface is stationary, this is not a serious problem since interpolation operators only have to be generated once. To reduce the computational cost for sliding interfaces, we also consider a modification to the new procedure by relaxing the IPP relations. Without compromising stability, this modification ensures that the algorithmic complexity is not more than linear with respect to the number of interface nodes, regardless of the number of space dimensions.

The paper is outlined as follows. In section 2 we propose a method-of-lines SBP approach for domains which move according to a volume preserving coordinate mapping. In section 3 we extend the SBP theory from [7] to moving and sliding mesh interfaces. The theory is presented for general meshes and SBP methods, and all derivations which are specific to either structured or unstructured methods are given in appendices. With the theory in place, we outline an optimization procedure for the automatic generation of interpolation operators for any given mesh interface in section 4. In section 5 we demonstrate with numerical examples the superior performance of the

new automatic methodology. Finally, in section 6, conclusions are drawn.

2. The continuous model problem

To keep the presentation short and to the point, we consider geometries in 2D, and make explicit references to 3D only where necessary. We start by considering the scalar advection model problem,

$$u_t + au_x + bu_y = 0, \quad (x, y) \in \Omega(t), \tag{1}$$

where a and b are constants, and where subscripts denote partial derivatives. To describe the time-dependent motion of Ω, we let the functions $x(t)$ and $y(t)$ denote a coordinate mapping following this motion. In particular, we let the coordinate velocity be described by the functions,

$$\frac{dx}{dt} = \dot{x}(t, x, y), \quad \frac{dy}{dt} = \dot{y}(t, x, y). \tag{2}$$

In the reference frame of Ω, the solution $u = u(t, x(t), y(t))$ has the total (substantive) time derivative

$$\frac{du}{dt} = u_t + \dot{x}u_x + \dot{y}u_y, \tag{3}$$

where u_t is the partial time derivative of u. Equation (1) thus becomes,

$$\frac{du}{dt} - \dot{x}u_x - \dot{y}u_y + au_x + bu_y = 0. \tag{4}$$

For any smooth test function ϕ, we can now derive, see Appendix A, the following integral identity from equation (4),

$$\frac{d}{dt}\left(\int_\Omega \phi u \, dV\right) = \int_\Omega \left(\frac{d\phi}{dt} + (a - \dot{x})\phi_x + (b - \dot{y})\phi_y\right) u \, dV - \oint_{\partial\Omega} \lambda\phi u \, dS, \tag{5}$$

where

$$\lambda = (a - \dot{x})n_x + (b - \dot{y})n_y, \tag{6}$$

and $(n_x, n_y)^T$ is the outward pointing normal to Ω. The function λ above is of fundamental importance since it controls the amount of boundary flux related to the quantity ϕu.

Two special cases of the general identity (5) are of particular interest. Firstly, an energy rate is obtained by inserting $\phi = u$ and using (4),

$$\frac{d}{dt}\left(\int_\Omega u^2 \, dV\right) = -\oint_{\partial\Omega} \lambda u^2 \, dS. \tag{7}$$

Similarly, for $\phi = 1$ a conservation law is obtained,

$$\frac{d}{dt}\left(\int_\Omega u \, dV\right) = -\oint_{\partial\Omega} \lambda u \, dS. \tag{8}$$

Equations (7) and (8) are fundamental properties of the problem that we will mimic in the discrete approximation of (4).

2.1. Parametric representation of λ

In the upcoming discrete analysis, a parametric representation of subdomain boundaries in space will be used. Thus, at a given point in time, let $s \in [0, 1]$ denote the normalized distance along some smooth segment of $\partial\Omega$, and consider the coordinates $x = x(s)$, $y = y(s)$ as functions of this parameter. The tangent is then given by $(x_s, y_s)^T$, and, if s is positively oriented towards the domain, the outward pointing unit normal $(n_x, n_y)^T$ is given by the explicit formulas,

$$n_x = J^{-1} y_s, \quad n_y = -J^{-1} x_s, \quad J = \sqrt{x_s^2 + y_s^2}. \tag{9}$$

Applying (9) to the definition of λ in (6) now yields

$$\lambda = J^{-1}((a - \dot{x}) y_s - (b - \dot{y}) x_s). \tag{10}$$

If instead the orientation of s is negative with respect to Ω, the signs in (10) are switched. The form (10) can now easily be discretized by application of an operator approximating $\partial/\partial s$.

2.2. Volume preserving motion

High order SBP discretizations of finite difference type for moving and deforming meshes were previously considered on fully discrete form in [9]. In this work we consider discretizations of method-of-lines type for simplicity. As we shall find, this requires a constant discrete L_2 norm, and hence we will

consider volume preserving coordinate mappings, i.e. such that the discrete mesh is not deforming. We thus postulate that,

$$\frac{d}{dt}\left(\int_{\Delta\Omega} dV\right) = 0, \tag{11}$$

for all subsets $\Delta\Omega$ of Ω. Satisfying this relation discretely will later allow for a method-of-lines approach to solve (4), without the need to address discrete geometric conservation. The volume preserving assumption simplifies the analysis by decoupling the spatial and temporal parts of discretization. However, the methodology presented in this paper can be extended to deforming meshes as well, for example in combination with the techniques used in [9].

3. SBP operators

As a general convention of notation, we shall use boldface letters to denote discrete parameters corresponding to $x, y, \dot{x}, \dot{y}, x_s, y_s, n_x, n_y, \lambda$ and J. The discrete values, which can be either exact or approximated, are inserted in diagonal matrices. Upper case letters will be used if all nodes in a subdomain are being considered, while lower case letters will denote the same quantity restricted to a subdomain boundary.

We extend the general SBP framework outlined in [7] to also include cases where the domain is moving. We thus seek a discrete approximation of (4) on a single computational domain of the form,

$$\frac{d\mathbf{U}}{dt} + \mathcal{P}^{-1}\mathcal{Q}\mathbf{U} = \mathcal{P}^{-1}E^T P(\boldsymbol{\lambda}\mathbf{u} - \widehat{\boldsymbol{\lambda}\mathbf{u}}), \tag{12}$$

where $\mathbf{U}$ is the vector of discrete solution values, and $\mathcal{P}$ is a diagonal matrix containing discrete integration weights. Depending on the type of method, these can either be control volumes or some type of higher order quadrature weights. Similarly, P contains integration weights along the domain boundary, while E is a so-called restriction operator from the computational domain to the boundary, i.e. $\mathbf{u} = E\mathbf{U}$ is the boundary restriction of the solution. Furthermore, $\widehat{\boldsymbol{\lambda}\mathbf{u}}$ is a numerical flux, where $\boldsymbol{\lambda} = (aI - \dot{\mathbf{x}})\mathbf{n}_x + (bI - \dot{\mathbf{y}})\mathbf{n}_y$ corresponds to the continuous function λ in (6). Building on the volume preserving coordinate mapping (11), we only consider the case where $\mathcal{P}$ is independent of time. This restriction can be motivated for both structured and unstructured operators, see Appendix B.

The discrete operator in (12) satisfies an SBP property of the form,

$$\mathcal{Q} + \mathcal{Q}^T = E^T P \boldsymbol{\lambda} E. \tag{13}$$

Detailed examples of discrete operators satisfying this general form of SBP property are given in Appendix B. A discrete analogue of (5) can now be obtained by multiplying (12) with $\boldsymbol{\Phi}^T \mathcal{P}$ from the left, where $\boldsymbol{\Phi}$ is any discrete test function, and applying (13). We find, since $\mathcal{P}$ is time-independent,

$$\frac{d}{dt}(\boldsymbol{\Phi}^T \mathcal{P} \mathbf{U}) = \mathbf{U}^T \mathcal{P} \left(\frac{d}{dt}\boldsymbol{\Phi} + \mathcal{P}^{-1}\mathcal{Q}\boldsymbol{\Phi} \right) - \boldsymbol{\phi}^T P \widehat{\boldsymbol{\lambda}\mathbf{u}}, \quad \boldsymbol{\phi} = E\boldsymbol{\Phi}. \tag{14}$$

Since the focus in this paper is on interface couplings, we restrict the analysis to a subdomain boundary which forms an interface together with a neighbouring subdomain. The situation is illustrated in Figure 1, where

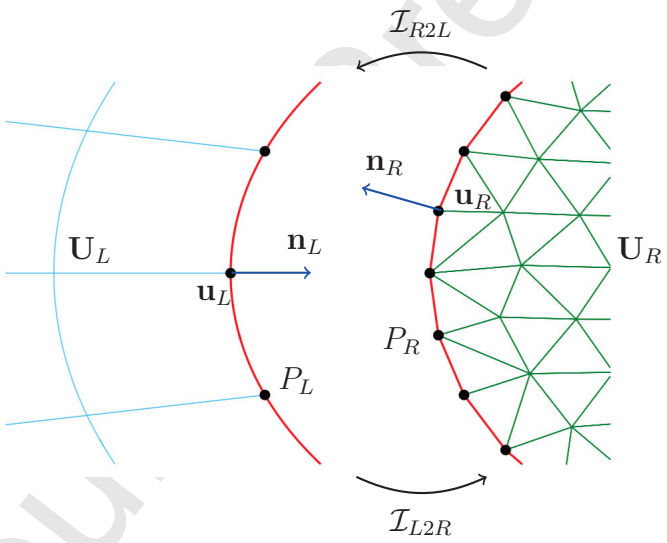


Figure 1: Illustration of interface coupling between two general meshes.

two general meshes are connected through a common interface. Using interpolation operators, we define $\widehat{\boldsymbol{\lambda}\boldsymbol{u}}$ in the following way on the two sides, respectively,

$$\begin{aligned}
\widehat{\boldsymbol{\lambda}\mathbf{u}}_L &= \frac{\boldsymbol{\lambda}_L \mathbf{u}_L + \mathcal{I}_{R2L}(-\boldsymbol{\lambda}_R \mathbf{u}_R)}{2} + \sigma \frac{P_L^{-1} \mathcal{I}_{L2R}^T P_R |\boldsymbol{\lambda}_R| (\mathcal{I}_{L2R}\mathbf{u}_L - \mathbf{u}_R)}{2} \\
\widehat{\boldsymbol{\lambda}\mathbf{u}}_R &= \frac{\boldsymbol{\lambda}_R \mathbf{u}_R + \boldsymbol{\lambda}_R \mathcal{I}_{L2R}\mathbf{u}_L}{2} + \sigma \frac{|\boldsymbol{\lambda}_R| (\mathbf{u}_R - \mathcal{I}_{L2R}\mathbf{u}_L)}{2},
\end{aligned} \tag{15}$$

where P_L and P_R denote the two boundary integration operators, and $\sigma \geq 0$ is a parameter controlling the strength of interface dissipation. Note that both of the dissipation terms in (15) include the same difference between the solution to the right of the interface with the corresponding interpolated value from the left. This is a design choice based on the coarseness of the mesh; since Ω_R is defined as the finer of the two meshes in Figure 1, the bandwidth of the product $P_L^{-1}\mathcal{I}_{L2R}^T P_R|\boldsymbol{\lambda}_R|\mathcal{I}_{L2R}$ in (15) is minimized. If instead the mesh on Ω_L is the more fine one, (15) can be modified accordingly. (A more symmetric looking form of (15) can be obtained by using a weighted sum of differences obtained from interpolating in both directions.)

Inserted into (14), the fluxes in (15) yield the terms,

$$\begin{aligned}\boldsymbol{\phi}_L^T P_L \widehat{\boldsymbol{\lambda}\mathbf{u}}_L &= \left[\boldsymbol{\phi}_L^T P_L(\boldsymbol{\lambda}_L \mathbf{u}_L) - \boldsymbol{\phi}_L^T P_L(\mathcal{I}_{R2L}\boldsymbol{\lambda}_R \mathbf{u}_R) + \sigma(\mathcal{I}_{L2R}\boldsymbol{\phi}_L)^T P_R|\boldsymbol{\lambda}_R|(\mathcal{I}_{L2R}\mathbf{u}_L - \mathbf{u}_R)\right]/2 \\ \boldsymbol{\phi}_R^T P_R \widehat{\boldsymbol{\lambda}\mathbf{u}}_R &= \left[\boldsymbol{\phi}_R^T P_R(\boldsymbol{\lambda}_R \mathbf{u}_R) + \boldsymbol{\phi}_R^T P_R(\boldsymbol{\lambda}_R \mathcal{I}_{L2R}\mathbf{u}_L) + \sigma\boldsymbol{\phi}_R^T P_R|\boldsymbol{\lambda}_R|(\mathbf{u}_R - \mathcal{I}_{L2R}\mathbf{u}_L)\right]/2.\end{aligned} \tag{16}$$

Remark 1. *Eq. (15) is a simplification of a more general form of fluxes considered in [7]. The latter was designed to include a broader class of IPP formulations, such as the so-called mortar and glue grid techniques [2, 6]. For the purpose of this work however, the form (15) is sufficient to facilitate couplings between arbitrary types of SBP operators.*

The IPP relations for the interpolation operators in (15) reads [1, 7],

$$P_L \mathcal{I}_{R2L} = \mathcal{I}_{L2R}^T P_R. \tag{17}$$

This property ensures that inner products induced by P_L and P_R are preserved in the following sense,

$$\boldsymbol{v}_L^T P_L(\mathcal{I}_{R2L}\boldsymbol{v}_R) = \boldsymbol{v}_R^T P_R(\mathcal{I}_{L2R}\boldsymbol{v}_L), \tag{18}$$

for general vectors $\boldsymbol{v}_L$ and $\boldsymbol{v}_R$.

3.1. Stability

In (14) we first consider the case $\boldsymbol{\Phi} = \mathbf{U}$, and use (12). This yields the following discrete version of (7),

$$\frac{d}{dt}(\mathbf{U}^T \mathcal{P} \mathbf{U}) = \mathbf{u}^T P \left(\boldsymbol{\lambda}\mathbf{u} - 2\widehat{\boldsymbol{\lambda}\mathbf{u}}\right), \tag{19}$$

valid for both the left and the right domain. When summing over the two domains, the interface treatment should yield a non-positive contribution for stability. From (16) we have,

$$\mathbf{u}_L^T P_L(\boldsymbol{\lambda}_L \mathbf{u}_L - 2\widehat{\boldsymbol{\lambda}\mathbf{u}}_L) = \mathbf{u}_L^T P_L \mathcal{I}_{R2L}(\boldsymbol{\lambda}_R \mathbf{u}_R) - \sigma(\mathcal{I}_{L2R}\mathbf{u}_L)^T P_R |\boldsymbol{\lambda}_R|(\mathcal{I}_{L2R}\mathbf{u}_L - \mathbf{u}_R)$$
$$\mathbf{u}_R^T P_R(\boldsymbol{\lambda}_R \mathbf{u}_R - 2\widehat{\boldsymbol{\lambda}\mathbf{u}}_R) = -\mathbf{u}_R^T P_R \boldsymbol{\lambda}_R \mathcal{I}_{L2R}\mathbf{u}_L - \sigma \mathbf{u}_R^T P_R |\boldsymbol{\lambda}_R|(\mathbf{u}_R - \mathcal{I}_{L2R}\mathbf{u}_L).$$

Since P_R is a diagonal matrix, it commutes with $\boldsymbol{\lambda}_R$. Summing over the two domains, we can thus write,

$$\begin{aligned}\frac{d}{dt}(\mathbf{U}_L^T \mathcal{P}_L \mathbf{U}_L + \mathbf{U}_R^T \mathcal{P}_R \mathbf{U}_R) = \mathbf{u}_L^T P_L \mathcal{I}_{R2L}(\boldsymbol{\lambda}_R \mathbf{u}_R) - (\boldsymbol{\lambda}_R \mathbf{u}_R)^T P_R \mathcal{I}_{L2R}\mathbf{u}_L \\ - \sigma(\mathcal{I}_{L2R}\mathbf{u}_L - \mathbf{u}_R)^T P_R |\boldsymbol{\lambda}_R|(\mathcal{I}_{L2R}\mathbf{u}_L - \mathbf{u}_R),\end{aligned}$$

and we have now proven the following result.

Proposition 1. *Assume that the operators in (15) at all times satisfy*

$$\mathbf{u}_L^T P_L \mathcal{I}_{R2L}(\boldsymbol{\lambda}_R \mathbf{u}_R) = (\boldsymbol{\lambda}_R \mathbf{u}_R)^T P_R \mathcal{I}_{L2R}\mathbf{u}_L. \tag{20}$$

Then stability follows, and the energy rate becomes,

$$\frac{d}{dt}(\mathbf{U}_L^T \mathcal{P}_L \mathbf{U}_L + \mathbf{U}_R^T \mathcal{P}_R \mathbf{U}_R) = -\sigma(\mathcal{I}_{L2R}\mathbf{u}_L - \mathbf{u}_R)^T P_R |\boldsymbol{\lambda}_R|(\mathcal{I}_{L2R}\mathbf{u}_L - \mathbf{u}_R),$$

which is non-positive for $\sigma \geq 0$.

Remark 2. *For IPP operators, the stability condition (20) is always satisfied due to (18).*

Remark 3. *By using an automatic procedure to generate new operators $\mathcal{I}_{L2R}$ and $\mathcal{I}_{R2L}$ in each discrete time step, (20) can be enforced even without the assumption of IPP. This requires that $\mathbf{u}_L$ and $\mathbf{u}_R$ are used as inputs to the procedure, and hence the resulting operators will depend on the discrete solution. This approach makes the interface treatment non-linear in a weak sense. We will not elaborate further on aspects of this non-linear type of treatment, but rather deal with that in the next paper.*

For stability, we shall later consider both the IPP and the non IPP approach as discussed in the two remarks above.

3.2. Conservation and free-stream preservation

Next we consider $\mathbf{\Phi} = \mathbb{1}$ in (14), where $\mathbb{1}$ is a vector with the constant value 1 everywhere. This yields a discrete version of (8) given by,

$$\frac{d}{dt}\left(\mathbb{1}^T\mathcal{P}\mathbf{U}\right) = \mathbf{U}^T\mathcal{Q}\mathbb{1} - \mathbf{1}^T P\widehat{\boldsymbol{\lambda}\mathbf{u}} = -\mathbf{1}^T P\widehat{\boldsymbol{\lambda}\mathbf{u}}. \tag{21}$$

In the last step above we have assumed that the numerical scheme excluding boundary and interface conditions is free-stream preserving, i.e.

$$\mathcal{Q}\mathbb{1} = 0. \tag{22}$$

For a conservative interface treatment, we require that the remaining right hand side of (21) from the two domains add to zero, i.e.

$$\frac{d}{dt}\left(\mathbb{1}_L^T\mathcal{P}_L\mathbf{U}_L + \mathbb{1}_R^T\mathcal{P}_R\mathbf{U}_R\right) = -(\mathbf{1}_L^T P_L\widehat{\boldsymbol{\lambda}\mathbf{u}}_L + \mathbf{1}_R^T P_R\widehat{\boldsymbol{\lambda}\mathbf{u}}_R) = 0. \tag{23}$$

From (16) we get, using the IPP property (17),

$$\begin{aligned}
\mathbf{1}_L^T P_L\widehat{\boldsymbol{\lambda}\mathbf{u}}_L &= \left[(\boldsymbol{\lambda}_L\mathbf{1}_L)^T P_L\mathbf{u}_L - (\mathcal{I}_{L2R}\mathbf{1}_L)^T P_R(\boldsymbol{\lambda}_R\mathbf{u}_R) + \sigma(\mathcal{I}_{L2R}\mathbf{1}_L)^T P_R|\boldsymbol{\lambda}_R|(\mathcal{I}_{L2R}\mathbf{u}_L - \mathbf{u}_R)\right]/2\\
\mathbf{1}_R^T P_R\widehat{\boldsymbol{\lambda}\mathbf{u}}_R &= \left[\mathbf{1}_R^T P_R(\boldsymbol{\lambda}_R\mathbf{u}_R) + (\mathcal{I}_{R2L}\boldsymbol{\lambda}_R\mathbf{1}_R)^T P_L\mathbf{u}_L + \sigma\mathbf{1}_R^T P_R|\boldsymbol{\lambda}_R|(\mathbf{u}_R - \mathcal{I}_{L2R}\mathbf{u}_L)\right]/2.
\end{aligned}$$

Hence, (23) becomes,

$$\begin{aligned}
\mathbf{1}_L^T P_L\widehat{\boldsymbol{\lambda}\mathbf{u}}_L + \mathbf{1}_R^T P_R\widehat{\boldsymbol{\lambda}\mathbf{u}}_R = \big[&(\boldsymbol{\lambda}_L\mathbf{1}_L + \mathcal{I}_{R2L}\boldsymbol{\lambda}_R\mathbf{1}_R)^T P_L\mathbf{u}_L + (-\mathcal{I}_{L2R}\mathbf{1}_L + \mathbf{1}_R)^T P_R(\boldsymbol{\lambda}_R\mathbf{u}_R)\\
&+\sigma(\mathcal{I}_{L2R}\mathbf{1}_L - \mathbf{1}_R)^T P_R|\boldsymbol{\lambda}_R|(\mathcal{I}_{L2R}\mathbf{u}_L - \mathbf{u}_R)\big]/2.
\end{aligned}$$

For conservation to hold at the interface, we thus need,

$$\mathcal{I}_{L2R}\mathbf{1}_L = \mathbf{1}_R, \quad \mathcal{I}_{R2L}\boldsymbol{\lambda}_R\mathbf{1}_R = -\boldsymbol{\lambda}_L\mathbf{1}_L. \tag{24}$$

Note that if (24) holds, then inserting $\mathbf{u}_L = \mathbf{1}_L$ and $\mathbf{u}_R = \mathbf{1}_R$ into (15) yields $\widehat{\boldsymbol{\lambda}\mathbf{u}}_L = \boldsymbol{\lambda}_L\mathbf{u}_L$ and $\widehat{\boldsymbol{\lambda}\mathbf{u}}_R = \boldsymbol{\lambda}_R\mathbf{u}_R$. Together with (22), equation (24) thus ensures that the scheme (12) is free-stream preserving.

We have proven

Proposition 2. *The fluxes in (15) yield a conservative scheme (23) if both the IPP property (17) and the free-stream preserving properties (22) and (24) hold.*

Remark 4. *With a non-linear automatic procedure, it is possible to satisfy (23) even without the assumption of IPP, see also Remark 3 on the stability of such procedures. Directly from (16) and assuming free-stream preservation as before, the following condition is sufficient for satisfying (23),*

$$\mathbf{1}_L^T P_L(\boldsymbol{\lambda}_L \mathbf{u}_L) - \mathbf{1}_L^T P_L(\mathcal{I}_{R2L}\boldsymbol{\lambda}_R \mathbf{u}_R) + \mathbf{1}_R^T P_R(\boldsymbol{\lambda}_R \mathbf{u}_R) + \mathbf{1}_R^T P_R(\boldsymbol{\lambda}_R \mathcal{I}_{L2R}\mathbf{u}_L) = 0.$$

However, in order to avoid a possible abuse of terminology we abstain from referring to this non IPP (and thus non-linear) approach as being conservative, regardless of whether (23) holds or not.

The question still remains whether or not IPP interpolation operators can be constructed in such a way that the free-stream preserving conditions (24) are satisfied. The following Lemma provides a necessary condition.

Lemma 1. *For IPP (17) interpolation operators to satisfy the free-stream preserving property (24), it is necessary that the norms P_L and P_R satisfy,*

$$\mathbf{1}_L^T P_L \boldsymbol{\lambda}_L \mathbf{1}_L + \mathbf{1}_R^T P_R \boldsymbol{\lambda}_R \mathbf{1}_R = 0. \tag{25}$$

Proof. We first multiply the second condition in (24) with $\mathbf{1}_L^T P_L$ from the left,

$$\mathbf{1}_L^T P_L \mathcal{I}_{R2L} \boldsymbol{\lambda}_R \mathbf{1}_R = -\mathbf{1}_L^T P_L \boldsymbol{\lambda}_L \mathbf{1}_L.$$

Using the IPP property (17), this yields,

$$(\mathcal{I}_{L2R}\mathbf{1}_L)^T P_R \boldsymbol{\lambda}_R \mathbf{1}_R = -\mathbf{1}_L^T P_L \boldsymbol{\lambda}_L \mathbf{1}_L.$$

By inserting the first condition in (24) into the above, (25) follows. □

Note that (25) is a discrete version of the continuous relation (due to the opposite sign of λ on the two sides),

$$\oint_{\partial\Omega_L} \lambda \, dS + \oint_{\partial\Omega_R} \lambda \, dS = 0,$$

where only the interface contributions to these integrals are considered.

Even though a general proof is not yet available, (25) is according to our experience not only necessary but also sufficient to facilitate generation of IPP operators satisfying (24) with an automatic procedure. Thus, free-stream preservation reduces to the problem of defining SBP operators $\mathcal{P}_L^{-1}\mathcal{Q}_L$ and $\mathcal{P}_R^{-1}\mathcal{Q}_R$ such that (25) is satisfied. To address this issue, we shall first need to parametrize the interface as in section 2.1.

3.3. Parametrization

As discussed in section 2.1, it is often practical to consider a one-dimensional parametric representation of the interface. Let $\mathbf{J}$ contain discrete values of the the stretching factor J in (9). An integration operator over the discrete space corresponding to $s \in [0, 1]$ can then easily be defined as,

$$\hat{P} = \mathbf{J}^{-1} P. \tag{26}$$

For structured methods on SBP form, $\hat{P}$ in (26) is by construction always associated with an SBP operator in one dimension [7]. This is however not necessary in order to define an operator $\hat{P}$ in this way, e.g. starting with an unstructured method.

Once a pair of accurate interpolation operators $\hat{\mathcal{I}}_{L2R}$ and $\hat{\mathcal{I}}_{R2L}$ have been generated for the parametrized space, we can then apply the formulas,

$$\mathcal{I}_{L2R} = \mathbf{J}_R^{-1/2} \hat{\mathcal{I}}_{L2R} \mathbf{J}_L^{1/2}, \quad \mathcal{I}_{R2L} = \mathbf{J}_L^{-1/2} \hat{\mathcal{I}}_{R2L} \mathbf{J}_R^{1/2}, \tag{27}$$

to obtain the operators appearing in the scheme (15). It was shown in [7] that the IPP property (17) is preserved after applying (27). In other words, if $\hat{\mathcal{I}}_{L2R}$ and $\hat{\mathcal{I}}_{R2L}$ are IPP with respect to $\hat{P}_L$ and $\hat{P}_R$, then $\mathcal{I}_{L2R}$ and $\mathcal{I}_{R2L}$ are automatically IPP with respect to P_L and P_R. The different notation used in physical and parametrized space is illustrated in Figure 2.

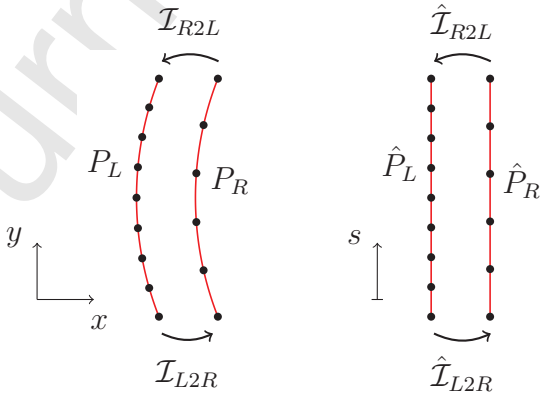


Figure 2: Integration and interpolation operators in physical and parametrized space.

Remark 5. *The procedure described above is a simplified version of a similar approach used in [7], where the parametrization was only considered on the sides of an interface where structured methods were employed. The advantage with this procedure is that (27) applies to interfaces between arbitrary meshes.*

Before moving on to formulate a strategy on how to generate the operators $\hat{\mathcal{I}}_{L2R}$ and $\hat{\mathcal{I}}_{R2L}$ on a general mesh, we briefly discuss how the parametrization can be utilized to satisfy condition (25) in Lemma 1. One way to ensure that (25) always holds is so guarantee that on both sides of the equality sign we have the same exact continuous integral of λ. To achieve this property using general classes of SBP operators however, it is necessary to restrict the types of admissible mesh motions to only include rigid body motions. Thus, let

$$\dot{x} = \bar{a} - \omega y, \quad \dot{y} = \bar{b} + \omega x, \tag{28}$$

describe a general combination of rotation and translation, where ω, $\bar{a}$ and $\bar{b}$ may depend on time but not on x or y. With this restriction, (10) becomes,

$$\lambda = J^{-1}\left[(a - \bar{a} + \omega y)y_s - (b - \bar{b} - \omega x)x_s\right],$$

which we can also write as

$$\lambda = J^{-1}\left[(a - \bar{a})y - (b - \bar{b})x + \omega(x^2 + y^2)/2\right]_s.$$

Let's now assume that the corresponding discrete variable can be written on either of the two forms,

$$\boldsymbol{\lambda} = \mathbf{J}^{-1}\text{Diag}\left[((a - \bar{a})I + \omega\mathbf{y})D_s(\mathbf{y}\mathbf{1}) - ((b - \bar{b})I - \omega\mathbf{x})D_s(\mathbf{x}\mathbf{1})\right] \tag{29}$$

$$\boldsymbol{\lambda} = \mathbf{J}^{-1}\text{Diag}\left[D_s((a - \bar{a})\mathbf{y} - (b - \bar{b})\mathbf{x} + \omega(\mathbf{x}^2 + \mathbf{y}^2)/2)\mathbf{1}\right], \tag{30}$$

where $D_s = \hat{P}^{-1}Q_s$ is an SBP operator in one dimension, approximating the s-derivative. Recall that $\mathbf{1}$ is a vector with the constant value 1 everywhere, while $\boldsymbol{\lambda}$, $\mathbf{J}$, $\mathbf{x}$ and $\mathbf{y}$ are diagonal matrices. In Appendix C we show that (29) applies for unstructured methods, and (30) for structured methods.

Depending on if we consider the whole subdomain boundary forming a closed curve or only a single smooth segment, one of the following periodic or non-periodic SBP formulas apply,

$$Q_s + Q_s^T = 0, \quad Q_s + Q_s^T = Diag\left([-1, 0, \ldots, 0, 1]\right).$$

The one-dimensional operator is also free-stream preserving, i.e.

$$Q_s \mathbf{1} = 0.$$

Starting from (29) and using (26), we now get,

$$\mathbf{1}^T P \boldsymbol{\lambda} \mathbf{1} = \left[(a-\bar{a})\mathbf{1} + \omega(\mathbf{y1})\right]^T Q_s(\mathbf{y1}) - \left[(b-\bar{b})\mathbf{1} - \omega(\mathbf{x1})\right]^T Q_s(\mathbf{x1}).$$

For a periodic Q_s, the SBP and free-stream preserving properties lead to a vanishing right-hand-side above, and in the non-periodic SBP case we get,

$$\mathbf{1}^T P \boldsymbol{\lambda} \mathbf{1} = \left[(a-\bar{a})y - (b-\bar{b})x + \omega(x^2+y^2)/2\right]_{s=0}^{1}.$$

Since the parameter s has the opposite sign on the two sides of an interface, (25) now follows. The same result can be shown in a similar fashion for (30). We have thus shown that both (29) and (30) automatically leads to (25).

4. Interpolation operator construction

The theoretical part of this paper is now complete, and the remaining issue is the practical construction of interpolation operators $\hat{\mathcal{I}}_{L2R}$ and $\hat{\mathcal{I}}_{R2L}$ for a given mesh. We will consider and compare two different approaches to achieve stability; either by satisfying the IPP relations (17), or alternatively by explicitly enforcing the identity (20). The differences between the two can summarized as follows. For the IPP approach, discrete conservation follows directly from free-stream preservation due to Lemma 2, and we recall that free-stream preservation is possible for any mesh undergoing relative rigid body motion. The non IPP approach (i.e. enforcing (20) explicitly) is on the other hand more computationally efficient, which will become an especially important consideration for moving 3D meshes. Due to the nonlinear nature of the non IPP treatment, we have chosen in this paper not to consider the issue of conservation for this case.

4.1. Notation

As a first step in the automatic generation procedure, the number and positions of non-zero interpolation coefficients are assigned. Let N_L and N_R be the number of interface nodes on the two sides. To avoid confusion, we will consistently use the symbol i to denote an index between the values 1

and N_R, and j for an index between 1 and N_L. With this convention, we denote the non-zero coefficients on each row of the two operators with,

$$(\hat{\mathcal{I}}_{L2R})_{ij} = \alpha_{ij}, \quad i = 1, \ldots, N_R, \quad j = j_s(i), \ldots, j_e(i)$$
$$(\hat{\mathcal{I}}_{R2L})_{ji} = \beta_{ji}, \quad j = 1, \ldots, N_L, \quad i = i_s(j), \ldots, i_e(j).$$

With $N_L = 5$, $N_R = 4$, a simple example of this could e.g. be given by,

$$\hat{\mathcal{I}}_{L2R} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & & & \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & & \\ & & \alpha_{33} & \alpha_{34} & \alpha_{35} \\ & & & \alpha_{44} & \alpha_{45} \end{pmatrix}, \quad \hat{\mathcal{I}}_{R2L} = \begin{pmatrix} \beta_{11} & \beta_{12} & & \\ \beta_{21} & \beta_{22} & & \\ & \beta_{32} & \beta_{33} & \\ & & \beta_{43} & \beta_{44} \\ & & \beta_{54} & \beta_{55} \end{pmatrix}.$$

In coefficient form, the IPP property (17) reads,

$$\beta_{ji} = (\hat{P}_R)_{ii}\alpha_{ij}/(\hat{P}_L)_{jj}. \tag{31}$$

Note that this requires the non-zero structure of $\hat{\mathcal{I}}_{R2L}$ to be the same as that of $\hat{\mathcal{I}}^T_{L2R}$, as in the example above.

To simplify the notation below, we also list all coefficients on each row of $\hat{\mathcal{I}}_{L2R}$ and $\hat{\mathcal{I}}_{R2L}$ in the separate vectors,

$$\boldsymbol{\alpha}_i = \begin{pmatrix} \alpha_{i,j_s(i)} \\ \vdots \\ \alpha_{i,j_e(i)} \end{pmatrix}, \quad \boldsymbol{\beta}_j = \begin{pmatrix} \beta_{j,i_s(j)} \\ \vdots \\ \beta_{j,i_e(j)} \end{pmatrix}, \tag{32}$$

for $i = 1, \ldots, N_R$ and $j = 1 \ldots, N_L$. In the example above, we now have

$$\boldsymbol{\alpha}_1 = \begin{pmatrix} \alpha_{11} \\ \alpha_{12} \end{pmatrix}, \quad \boldsymbol{\alpha}_2 = \begin{pmatrix} \alpha_{21} \\ \alpha_{22} \\ \alpha_{23} \end{pmatrix}, \quad \boldsymbol{\alpha}_3 = \begin{pmatrix} \alpha_{33} \\ \alpha_{34} \\ \alpha_{35} \end{pmatrix}, \quad \boldsymbol{\alpha}_4 = \begin{pmatrix} \alpha_{44} \\ \alpha_{45} \end{pmatrix},$$

and

$$\boldsymbol{\beta}_1 = \begin{pmatrix} \beta_{11} \\ \beta_{12} \end{pmatrix}, \quad \boldsymbol{\beta}_2 = \begin{pmatrix} \beta_{21} \\ \beta_{22} \end{pmatrix}, \quad \boldsymbol{\beta}_3 = \begin{pmatrix} \beta_{32} \\ \beta_{33} \end{pmatrix}, \quad \boldsymbol{\beta}_4 = \begin{pmatrix} \beta_{43} \\ \beta_{44} \end{pmatrix}, \quad \boldsymbol{\beta}_5 = \begin{pmatrix} \beta_{54} \\ \beta_{55} \end{pmatrix}.$$

Accounting for the scalings in (27), the free-stream preserving conditions in (24) become,

$$\mathbf{f}_i^T \boldsymbol{\alpha}_i = 1, \quad \mathbf{g}_j^T \boldsymbol{\beta}_j = (\boldsymbol{\lambda}_L)_j, \quad \boldsymbol{f}_i = \begin{pmatrix} f_{i,j_s(i)} \\ \vdots \\ f_{i,j_e(i)} \end{pmatrix}, \quad \boldsymbol{g}_j = \begin{pmatrix} g_{j,i_s(j)} \\ \vdots \\ g_{j,i_e(j)} \end{pmatrix}, \tag{33}$$

where

$$f_{i,j} = (\boldsymbol{J}_R^{-1/2})_i (\boldsymbol{J}_L^{1/2})_j, \quad g_{j,i} = (\boldsymbol{J}_L^{-1/2})_j (\boldsymbol{J}_R^{1/2})_i (\boldsymbol{\lambda}_R)_i.$$

Finally, we also define a pair of vectors containing all the non-zero coefficients in $\hat{\mathcal{I}}_{L2R}$ and $\hat{\mathcal{I}}_{R2L}$,

$$\boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \vdots \\ \boldsymbol{\alpha}_{N_R} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_{N_L} \end{pmatrix}. \tag{34}$$

With the necessary notation in place, we are now ready to consider the problem of optimizing the interpolation coefficients in the general case.

4.2. Optimization procedure 1: the IPP case

Consider using $\hat{\mathcal{I}}_{L2R}$ and $\hat{\mathcal{I}}_{R2L}$ to interpolate an arbitrary smooth function $\phi(s)$ of the interface parameter s. Let $\mathbf{s}_L$ and $\mathbf{s}_R$ be vectors containing the nodal values of s on the mesh interface. We define $\boldsymbol{\Phi}_L = \phi(\mathbf{s}_L)$, $\boldsymbol{\Phi}_R = \phi(\mathbf{s}_R)$, containing the exact values of ϕ. For each row in the two operators, we now apply Taylor's formula to write the interpolation errors as,

$$(\hat{\mathcal{I}}_{L2R}\boldsymbol{\Phi}_L - \boldsymbol{\Phi}_R)_i = \phi((\mathbf{s}_R)_i) \left(\sum_{j=j_s(i)}^{j_e(i)} \alpha_{ij} - 1 \right) + \sum_{p=1}^{\infty} \frac{\phi^{(p)}((\mathbf{s}_R)_i)}{p!} \left(\sum_{j=j_s(i)}^{j_e(i)} \alpha_{ij} ((\mathbf{s}_L)_j - (\mathbf{s}_R)_i)^p \right)$$

$$(\hat{\mathcal{I}}_{R2L}\boldsymbol{\Phi}_R - \boldsymbol{\Phi}_L)_j = \phi((\mathbf{s}_L)_j) \left(\sum_{i=i_s(j)}^{i_e(j)} \beta_{ji} - 1 \right) + \sum_{p=1}^{\infty} \frac{\phi^{(p)}((\mathbf{s}_L)_j)}{p!} \left(\sum_{i=i_s(j)}^{i_e(j)} \beta_{ji} ((\mathbf{s}_R)_i - (\mathbf{s}_L)_j)^p \right).$$

Instead of forcing leading order error terms to be zero to machine precision, a more stable procedure is obtained by minimizing all coefficients in the two expansions simultaneously in a least-squares sense. For some large enough cut-off value $\tilde{p}$, we thus consider the least-squares error function,

$$\begin{aligned} \mathcal{E} = &\sum_{i=1}^{N_R} \left[\left(\sum_{j=j_s(i)}^{j_e(i)} \alpha_{ij} - 1 \right)^2 + \sum_{p=1}^{\tilde{p}} \frac{1}{p!} \left(\sum_{j=j_s(i)}^{j_e(i)} \alpha_{ij} ((\mathbf{s}_L)_j - (\mathbf{s}_R)_i)^p \right)^2 \right] \\ &+ \sum_{j=1}^{N_L} \left[\left(\sum_{i=i_s(j)}^{i_e(j)} \beta_{ji} - 1 \right)^2 + \sum_{p=1}^{\tilde{p}} \frac{1}{p!} \left(\sum_{i=i_s(j)}^{i_e(j)} \beta_{ji} ((\mathbf{s}_R)_i - (\mathbf{s}_L)_j)^p \right)^2 \right]. \end{aligned} \tag{35}$$

Note that if $\tilde{p}$ is large, we are not aiming for a particular order of accuracy, but rather to minimize the total error for a given mesh. This specific choice of objective function could be interpreted as assuming all $\phi^{(p)}$ being of similar magnitude with respect to p. Other choices could be obtained by scaling the terms in (35) differently, e.g. by explicitly optimizing for a specific, non-unit, frequency component.

Using the notation introduced previously in (32), we can also write (35) as,

$$\mathcal{E} = \sum_{i=1}^{N_R} \|A_i \boldsymbol{\alpha}_i - \mathbf{a}_i\|^2 + \sum_{j=1}^{N_L} \|B_j \boldsymbol{\beta}_j - \mathbf{b}_j\|^2,$$

where

$$A_i = \begin{pmatrix} 1 & \dots & 1 \\ (\mathbf{s}_L)_{j_s(i)} - (\mathbf{s}_R)_i & \dots & (\mathbf{s}_L)_{j_e(i)} - (\mathbf{s}_R)_i \\ \vdots & & \vdots \\ \frac{1}{\tilde{p}!}((\mathbf{s}_L)_{j_s(i)} - (\mathbf{s}_R)_i)^{\tilde{p}} & \dots & \frac{1}{\tilde{p}!}((\mathbf{s}_L)_{j_e(i)} - (\mathbf{s}_R)_i)^{\tilde{p}} \end{pmatrix}, \quad \mathbf{a}_i = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

and,

$$B_j = \begin{pmatrix} 1 & \dots & 1 \\ (\mathbf{s}_R)_{i_s(j)} - (\mathbf{s}_L)_j & \dots & (\mathbf{s}_R)_{i_e(j)} - (\mathbf{s}_L)_j \\ \vdots & & \vdots \\ \frac{1}{\tilde{p}!}((\mathbf{s}_R)_{i_s(j)} - (\mathbf{s}_R)_j)^{\tilde{p}} & \dots & \frac{1}{\tilde{p}!}((\mathbf{s}_L)_{i_e(j)} - (\mathbf{s}_R)_j)^{\tilde{p}} \end{pmatrix}, \quad \mathbf{b}_j = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Recall that $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ contain the non-zero interpolation coefficients on each row of the two operators.

After adding a regularization term to $\mathcal{E}$, as well as enforcing free-stream preservation (33) with a penalty term, we obtain the objective function,

$$\mathcal{F} = \sum_{i=1}^{N_R} \|\mathcal{M}_i \boldsymbol{\alpha}_i - \mathbf{m}_i\|^2 + \sum_{j=1}^{N_L} \|\mathcal{N}_j \boldsymbol{\beta}_j - \mathbf{n}_j\|^2, \tag{36}$$

where

$$\mathcal{M}_i = \begin{pmatrix} A_i \\ \gamma_1 I \\ \gamma_2 \mathbf{f}_i^T \end{pmatrix}, \quad \mathbf{m}_i = \begin{pmatrix} \mathbf{a}_i \\ \mathbf{0} \\ \gamma_2 \end{pmatrix}, \quad \mathcal{N}_j = \begin{pmatrix} B_j \\ \gamma_1 I \\ \gamma_2 \mathbf{g}_i^T \end{pmatrix}, \quad \hat{\mathbf{n}}_j = \begin{pmatrix} \mathbf{b}_j \\ \mathbf{0} \\ \gamma_2 (\boldsymbol{\lambda}_L)_j \end{pmatrix}.$$

The regularization parameter γ_1 has the dual purpose of making the optimization problem sufficiently well-conditioned, as well as to limit the stiffness of the resulting operators. With a proper tuning of γ_1, the largest magnitude of coefficients in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ should be approximately unity. Increasing γ_1 too far beyond this point could adversely affect the optimal value of (35). Thus, some care is needed for tuning this parameter, but we expect that this step would be possible to automate, given the need. Furthermore, as long as the necessary condition (25) for free-stream preservation is satisfied, a large enough γ_2 should enforce (33) to machine precision [10]. Strictly speaking, this requires the total set of equality constraints (31) and (24) to have full rank (see e.g. [10] for a proof of this). Even though we can not prove full rank in the general case, our experience indicates that it is highly unlikely to encounter that. If free-stream preservation is not deemed to be required for a given application, γ_2 can be set to be zero.

For the generation of IPP operators, we finally consider the regularized, constrained least-squares problem,

$$\begin{aligned} &\text{minimize} && \mathcal{F} \\ &\text{subject to} && (31), \end{aligned} \tag{37}$$

where $\mathcal{F}$ is defined in (36).

4.3. Optimization procedure 2: the non IPP case

For simulations with a stationary mesh, the optimization problem (37) only has to be solved once. In this case the computational cost is not a critical consideration. However, if the mesh is changing continuously in time, new interpolation operators must be generated at arbitrary points in time as a part of the time integration process. In this case, it is essential to limit the complexity of the optimization problem as much as possible.

Note that the objective function in (36) has the block diagonal form,

$$\mathcal{F} = \|\mathcal{M}\boldsymbol{\alpha} - \mathbf{m}\|^2 + \|\mathcal{N}\boldsymbol{\beta} - \mathbf{n}\|^2, \tag{38}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are defined in (34), and

$$\mathcal{M} = \begin{pmatrix} \mathcal{M}_1 & & \\ & \ddots & \\ & & \mathcal{M}_{N_R} \end{pmatrix}, \quad \mathbf{m} = \begin{pmatrix} \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_{N_R} \end{pmatrix}, \quad \mathcal{N} = \begin{pmatrix} \mathcal{N}_1 & & \\ & \ddots & \\ & & \mathcal{N}_{N_L} \end{pmatrix}, \quad \mathbf{n} = \begin{pmatrix} \mathbf{n}_1 \\ \vdots \\ \mathbf{n}_{N_L} \end{pmatrix}.$$

Again, recall that each block corresponds to a single row in $\hat{\mathcal{I}}_{L2R}$ and $\hat{\mathcal{I}}_{R2L}$ respectively.

The IPP condition (31) introduces dependencies between the rows in $\hat{\mathcal{I}}_{L2R}$ and the columns in $\hat{\mathcal{I}}_{R2L}$, and vice versa. Hence, after substituting (31) into (38), the block diagonal structure is lost. For a 1D interface, this is not a major concern since the resulting matrices still have finite bandwidth. However, by extension to interfaces in two or more space dimensions, even the finite bandwidth structure will be lost. As a consequence, for 3D calculations the cost of solving (37) is not proportional to the number of nodes in the interface mesh.

To reduce the cost of optimization for a 3D moving mesh, we consider a modification of the optimization procedure above based on relaxing the IPP property. We recall from Proposition 1 that, for stability, it is enough to satisfy (20). The IPP property is just one particular way of doing that. Without IPP, (20) can instead be enforced directly provided that the solution components $\mathbf{u}_L$ and $\mathbf{u}_R$ are used as a part of the procedure, see Remark 3. While it would also be possible to satisfy the conservation constraint (23) in the same way, see Remark 4, we are not sure if there are any benefits of doing so for an already non-linear interface treatment. Hence, we only consider the stability constraint (20) below.

After applying the scalings in (26) and (27), (20) becomes,

$$(\hat{P}_L \mathbf{J}_L^{1/2} \mathbf{u}_L)^T \hat{\mathcal{I}}_{R2L} (\mathbf{J}_R^{1/2} \boldsymbol{\lambda}_R \mathbf{u}_R) = (\hat{P}_R \mathbf{J}_R^{1/2} \boldsymbol{\lambda}_R \mathbf{u}_R)^T \hat{\mathcal{I}}_{L2R} (\mathbf{J}_L^{1/2} \mathbf{u}_L).$$

In terms of the interpolation coefficients, this yields,

$$\sum_{i=1}^{N_R} \sum_{j=1}^{N_L} c_{ij} \alpha_{ij} = \sum_{i=1}^{N_R} \sum_{j=1}^{N_L} d_{ji} \beta_{ji},$$

where

$$c_{ij} = (\hat{P}_L \mathbf{J}_L^{1/2} \mathbf{u}_L)_i (\mathbf{J}_R^{1/2} \boldsymbol{\lambda}_R \mathbf{u}_R)_j, \quad d_{ji} = (\hat{P}_R \mathbf{J}_R^{1/2} \boldsymbol{\lambda}_R \mathbf{u}_R)_j (\mathbf{J}_L^{1/2} \mathbf{u}_L)_i.$$

On vector form, this can be written as,

$$\mathbf{c}^T \boldsymbol{\alpha} = \mathbf{d}^T \boldsymbol{\beta}, \tag{39}$$

where the elements in $\mathbf{c}$ and $\mathbf{d}$ are listed in the same order as the corresponding elements in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, see (34).

By ensuring stability through (39), instead of (37) we can now solve the fully unconstrained problem,

$$\text{minimize} \quad \mathcal{F} + \gamma_3^2 \|\mathbf{c}^T\boldsymbol{\alpha} - \mathbf{d}^T\boldsymbol{\beta}\|_2^2, \tag{40}$$

where γ_3 is penalty parameter of sufficient magnitude to enforce (39) to machine precision. In our experience, the three user defined parameters in (40) (i.e. γ_1 and γ_2 and γ_3) only needs to be tuned once (at $t = 0$) for optimal performance (in terms of accuracy, conditioning and stiffness) during a simulation with a moving or sliding interface.

Consider again the block diagonal form of $\mathcal{F}$ in (38). Assuming that (40) is sufficiently well conditioned for a unique solution to exist, then this solution is given by the set of normal equations,

$$\left(\mathcal{A}^T\mathcal{A} + \gamma_3^2\mathbf{w}\mathbf{w}^T\right)\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} = \mathcal{A}^T\begin{pmatrix}\mathbf{m}\\ \mathbf{n}\end{pmatrix}, \tag{41}$$

where

$$\mathcal{A} = \begin{pmatrix}\mathcal{M} & \\ & \mathcal{N}\end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix}\mathbf{c}\\ -\mathbf{d}\end{pmatrix}.$$

Solving (41) now only requires the inversion of the block diagonal symmetric matrix $\mathcal{A}^T\mathcal{A}$. The Sherman-Morrison formula for a rank 1 updated system yields,

$$\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} = \left((\mathcal{A}^T\mathcal{A})^{-1} - \frac{\gamma_3^2(\mathcal{A}^T\mathcal{A})^{-1}(\mathbf{w}\mathbf{w}^T)(\mathcal{A}^T\mathcal{A})^{-1})}{1+\gamma_3^2\mathbf{w}^T(\mathcal{A}^T\mathcal{A})^{-1}\mathbf{w}}\right)\mathcal{A}^T\begin{pmatrix}\mathbf{m}\\ \mathbf{n}\end{pmatrix}. \tag{42}$$

Finally, (42) can be computed in a stable way by first QR-factorizing the blocks in $\mathcal{A}$, followed by one or a few levels of iterative refinement, see [11].

Remark 6. *Due to the block diagonal structure of the unconstrained optimization problem, the computational complexity associated with (42) is linear with respect to the number of blocks in $\mathcal{A}$, i.e. to the number of nodes on the mesh interface. Most importantly, this remains the case for extensions of the optimization procedure to any number of space dimensions.*

5. Numerical results

We first investigate the accuracy and computational cost of the new IPP (37) and non IPP (42) global optimization procedures.

5.1. Optimization: 1D interface

We start with the simple mesh configuration shown in Figure 3, where two Cartesian grids are connected to each other. The left and right grid spacings are kept at a ratio of 9 to 16, and we use $\hat{P}_L$ and $\hat{P}_R$ from a pair of SBP finite difference operators of fourth order accuracy in the interior, see e.g. [12].

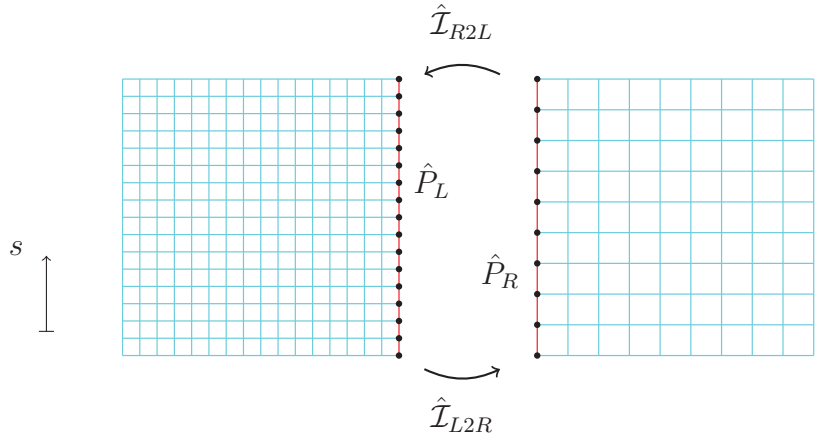


Figure 3: Interface between two structured Cartesian grids.

We compare the accuracy of operators obtained with the new automatic algorithms to the IPP so-called glue grid method of [2]. For the case $N_L = 33$, $N_R = 19$, the non-zero pattern of the operator $\hat{\mathcal{I}}_{L2R}$ obtained with this method is given by,

$$\hat{\mathcal{I}}_{L2R} = \left(\begin{array}{c} \text{×××××××××××} \\ \text{×××××××××××} \\ \text{×××××××××××} \\ \text{×××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \\ \text{×××××××××××××} \\ \text{××××××××××××} \end{array} \right).$$

The non-zero pattern of $\hat{\mathcal{I}}_{R2L}$ follows from (31) since the method is IPP. Even though a smaller operator bandwidth than this would be feasible, for the sake

of comparison we apply the new automatic procedure using the same non-zero pattern. In Figure 4 we show the convergence results in L_2 norm for

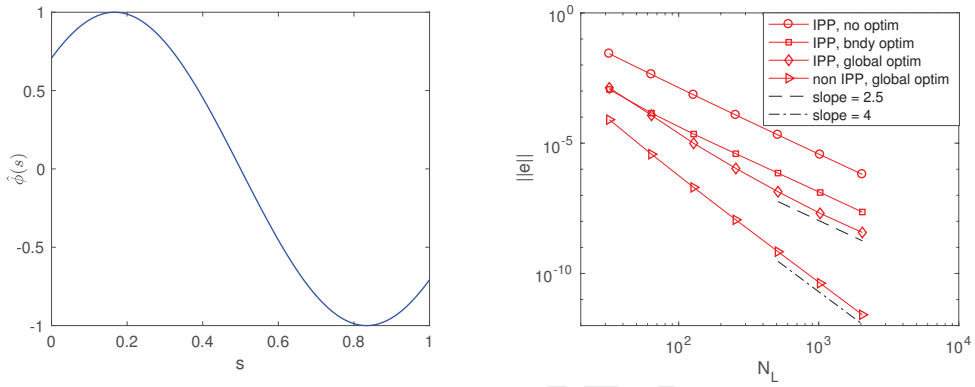


Figure 4: L_2 convergence for a continuous test function.

the test function $\phi(s) = sin(\frac{3\pi}{2}(s+\frac{1}{6}))$. We have used four different methods to construct the interpolation operators with the given non-zero patterns. Firstly we have used the basic glue grid technique which does not employ any least-squares error optimization. Secondly, we have kept the same rows in the interior from the glue grid operator, but optimized the boundary rows as in [1, 7]. Finally, we have applied the global optimization algorithms (37) and (42), where as we recall the former is IPP and the latter is not.

Even though in the asymptotic limit the convergence rate drops from 3 to 2.5 in all of the three IPP cases, as proven in [7], we find that optimizing the boundary coefficients reduces error levels by more than an order of magnitude compared to no optimization. In addition, the global optimization (37) results in almost a further order of magnitude improvement. Even more accurate is the non IPP global optimization approach, showing fourth order asymptotic convergence. We conclude from this that the approach of optimizing all coefficients simultaneously is clearly a superior way of reducing interpolation errors compared to the previous IPP techniques. Even though the non IPP algorithm was developed with the specific goal of reducing cost of 3D calculations, it is also clearly the most accurate one.

5.2. Optimization: 2D interface

In three dimensions, the requirements of computing power and memory become significantly different for the two global optimization approaches. Recall that the non IPP procedure of section 4.3 has a linear complexity with respect to the number of interface nodes. This is true for any number of dimensions, but the same can not be expected for the IPP procedure. To demonstrate the difference in computational cost for the two approaches, we consider the 2D Cartesian grid interface as illustrated in Figure 5. To save computational cost, we use a more limited non-zero pattern than previously. In one coordinate direction, we use a non-zero pattern to $\hat{\mathcal{I}}_{L2R}$ of the form,

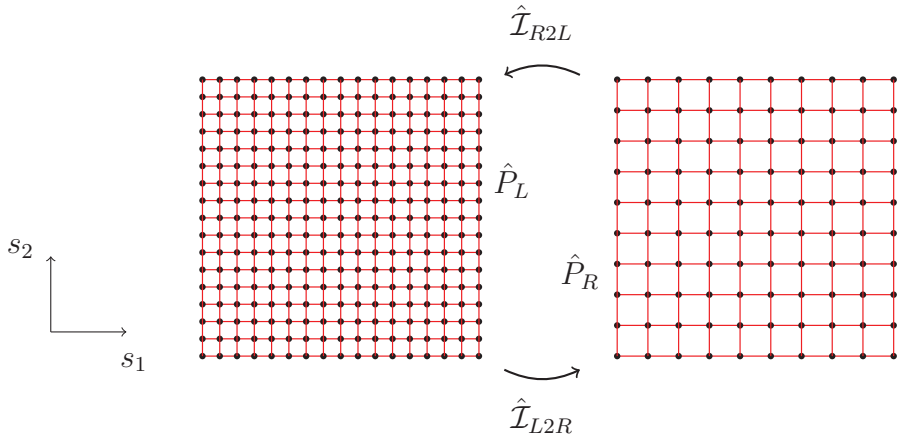
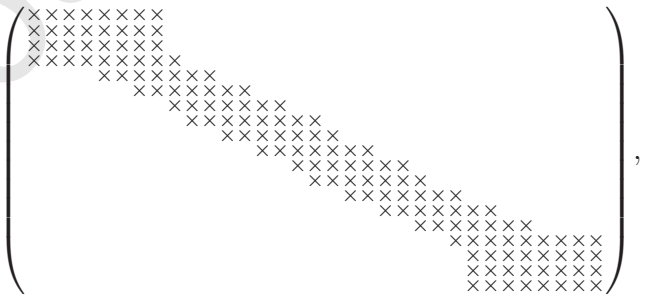


Figure 5: 2D interface between structured Cartesian grids.



and this structure is extended to 2D in a tensor product fashion. As before, the pattern of $\hat{\mathcal{I}}_{R2L}$ is given by the transpose of this. Recall that all coeffi-

cients in the 2D interpolation operators are considered as free parameters in the optimization procedures.

To assess the computational efficiency, we have compared the amount of fill-in required to factorize the least-squares normal equations with the two approaches, and the results are shown to the right in Figure 6. Here we have used the multifrontal *SuiteSparseQR* [13] package to compute the QR factorizations. The non IPP algorithm scales quadratically with the number of nodes in one coordinate direction, as expected. The IPP algorithm instead scales approximately with $log(N_L)N_L^2$. Even though this is lower than the complexity of N_L^3 associated with explicit time stepping on a $3D$ domain, the large proportionality factor associated with the IPP optimization procedure means that this cost can probably not be ignored if the mesh is non stationary. To the left in Figure 6 we also compare the L_2 errors obtained in the $2D$ interface case. Just as for 1D interfaces, the non IPP approach appears to be the more accurate one. For the grid parameters considered here, the IPP case has not yet reached the asymptotic order of convergence.

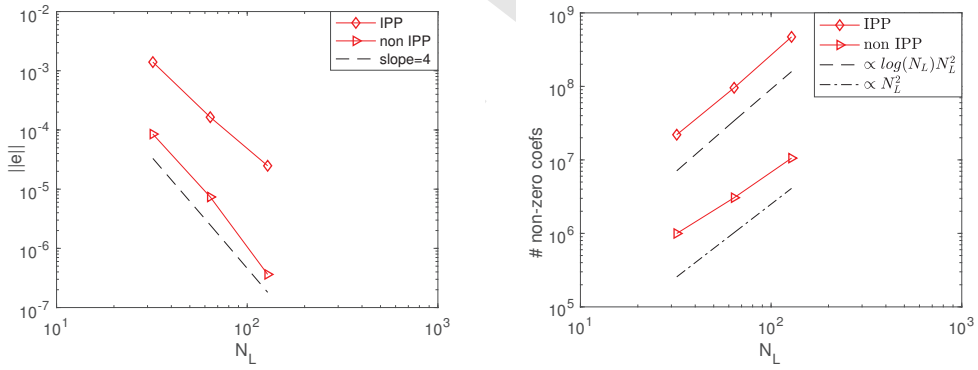


Figure 6: Left: convergence for a 2D interface using global optimization approaches. Right: number of non-zero coefficients in the normal equation factorization.

5.3. A sliding grid calculation

To demonstrate the performance of the automatic interpolation procedure for sliding grid calculations, we consider the time-dependent block geometry shown in Figure 7. An interior circular domain is here surrounded by four rotating curvilinear domains. The circular domain is discretized with a second

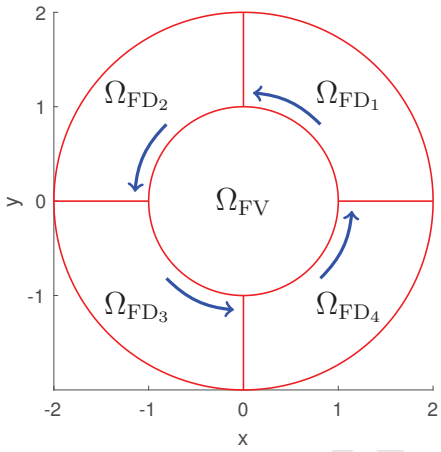


Figure 7: A sliding interface model involving four rotating curvilinear subdomains.

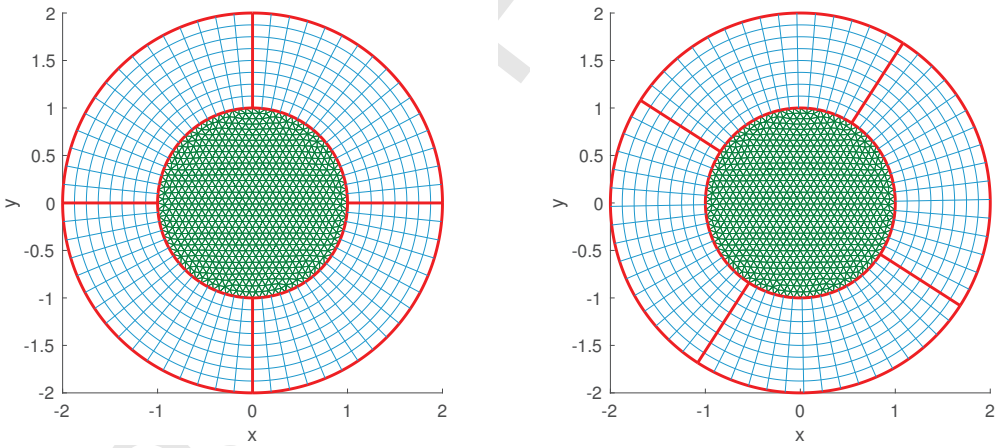


Figure 8: Mesh realization of the sliding block configuration in Figure 7. Left: $t = 0$, right: $t = 1$.

order accurate nodal finite volume method [14, 15], and the curvilinear domains using SBP finite difference operators with sixth order accuracy in the interior [12]. By dividing the outer domain into four blocks, we demonstrate the capability to deal with multi-block interfaces. One mesh realization is shown in Figure 8 at $t = 0$ and $t = 1$. The number of nodes on the interface

is here $N_R = 176$ on the unstructured side, and $N_1 = N_2 = N_3 = N_4 = 17$ on the structured side.

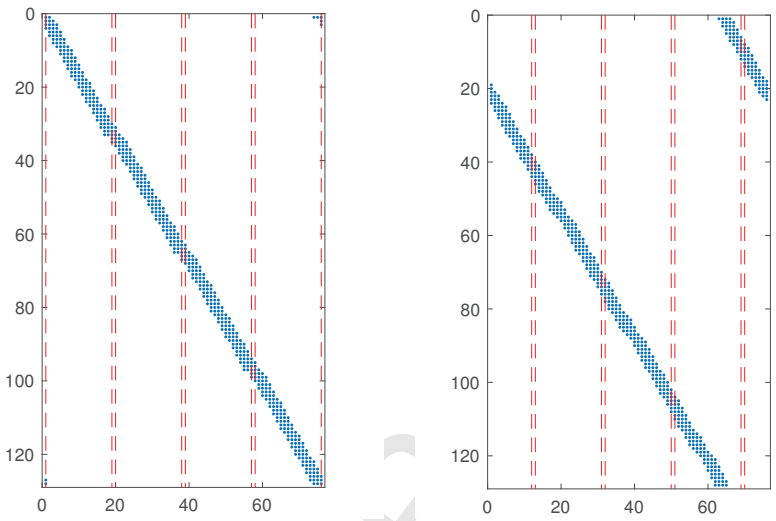


Figure 9: Non-zero structure of $\mathcal{I}_{L2R}$ for the meshes in Figure 8.

We let the finite difference blocks rotate counterclockwise around the origin as illustrated in Figure 8, with a speed of 1 radian per time unit. A curvilinear transformation from the unit square is given by,

$$x = \xi cos(\varphi_0 + t + \eta), \quad y = \xi sin(\varphi_0 + t + \eta),$$

where φ_0 is given by 0, $\pi/2$, π and $3\pi/2$ for the four curvilinear blocks, respectively. This transformation satisfies (28), allowing for conservative interpolation schemes, see Appendix C. We define $\Omega_R = \Omega_{\text{FV}}$, and $\Omega_L = \Omega_{\text{FD}_1} \cup \Omega_{\text{FD}_2} \cup \Omega_{\text{FD}_3} \cup \Omega_{\text{FD}_4}$. The interface integration operators are then defined by,

$$P_R = P_{FV}, \quad P_L = \begin{pmatrix} P_{FD1} & & & \\ & P_{FD2} & & \\ & & P_{FD3} & \\ & & & P_{FD4} \end{pmatrix},$$

and we define the continuous interface parameter as $s = \pm\varphi/2\pi$, where φ is the angle in radians along the interface.

Next, we fill each row in the interpolation operators $\hat{\mathcal{I}}_{L2R}$ and $\hat{\mathcal{I}}_{R2L}$ with a suitable number of non-zero coefficients. In general, this number can be chosen based on experience for a given type of mesh, where the goal is to make interpolation errors small compared to other discretization errors in the scheme. Figure 9 shows the resulting non-zero pattern for the operator $\hat{\mathcal{I}}_{R2L}$ at $\tau = 0$ and $\tau = 1$ respectively, and the non-zero pattern of $\hat{\mathcal{I}}_{L2R}$ is again given by the transpose. On the $y-$axis are the interface nodes on the stationary unstructured domain, and on the $x-$axis the nodes on the moving, curvilinear side of the interface. The block boundaries in Figure 8 are highlighted with vertical dotted lines in Figure 9. As a part of the time integration process, we use either the IPP algorithm (37) or the non IPP (42) algorithm to generate the interpolation coefficients.

5.4. Accuracy

Consider an exact solution to (1) with $a = b = 1$, given by a Gaussian pulse that propagates through the domain,

$$u = e^{(x+\sqrt{2}-t)^2+(y+\sqrt{2}-t)^2}.$$

Since the interpolation operators can be generated at arbitrary points in time, there is no lower restriction on the time step size. We thus choose the time step sizes to be small enough so that spatial errors are dominating. The initial solution as well as absolute error levels for different times using the non IPP algorithm (42) are plotted in Figure 10. Each of the curvilinear blocks are discretized with 73×155 grid points. The finite volume domain has 1024 interface points along the whole circumference, which gives a 16 to 9 ratio in the number of nodes at the interface. Disturbances introduced at the interface are small compared to the cumulative error from propagating the pulse across the unstructured domain.

Convergence as a function of the number of interface nodes is shown in Figure 11, also comparing with the IPP algorithm (37). The small impact on solution accuracy from the interface already observed in Figure 10 is further confirmed by the second order convergence observed in both maximum and L_2 norms. In this case, there is a relatively small difference between the two approaches.

5.5. Free-stream preservation

For a large enough value of the parameter γ_2, both the IPP (37) and the non IPP (42) version of the automatic interface algorithm yields a free-

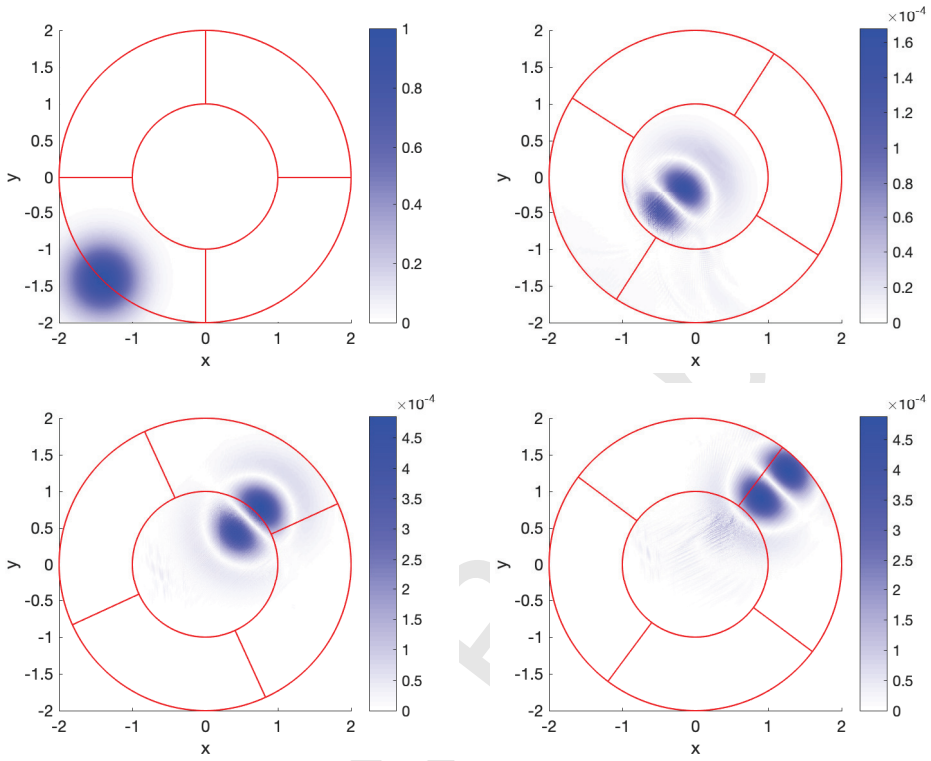


Figure 10: Initial solution and absolute errors at $t = 1$, $t = 2$ and $t = 2.5$.

stream preserving scheme. For the IPP version, discrete conservation follows as direct consequence of this due to Lemma 2, whereas we do not consider the property of conservation for the non IPP version. In Figure 12 we demonstrate for the IPP approach that free-stream preservation and hence conservation holds to machine precision. To the left, we also compare this result to the corresponding one with $\gamma_2 = 0$ in (37), i.e. without explicitly enforcing free-stream preservation.

6. Conclusions

An automatic methodology was developed for the generation and optimization of interpolation operators within the SBP framework of numerical methods. Based on solving a linear least-squares optimization problem, the new algorithms allow for completely general meshes while at the same time

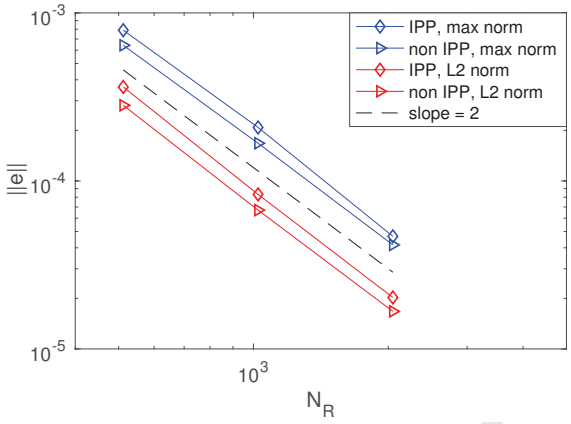


Figure 11: Convergence at $t = 1$ as a function of interface nodes on the unstructured side.

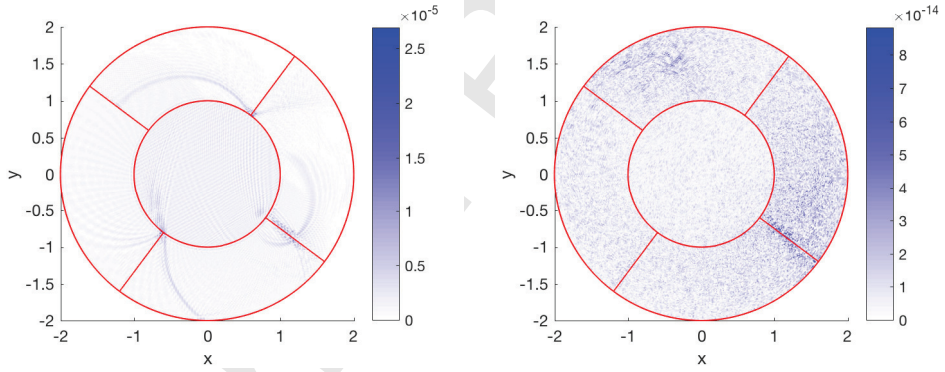


Figure 12: Free-stream solution at $t = 2.5$ for a non-conservative (left figure) and a conservative (right figure) interface treatment.

being highly effective at minimizing interpolation errors. To achieve stability, we have considered both an IPP and a non IPP approach. Both of these demonstrate superior accuracy compared to previous techniques. In particular, for 3D meshes the non IPP approach is both cheaper and more accurate than the IPP one, which makes it ideal for calculations with moving or sliding interfaces. Stability and free-stream preservation is achieved by both methods, but discrete conservation is unclear for the non IPP approach, due to the nonlinear nature of the coupling.

Acknowledgments

This work is based on the research supported in part by the National Research Foundation of South Africa (Grant numbers: 89916). In addition, part of the research leading to this work was supported by the AEROGUST project (funded by the European Commission under grant agreement number 636053). The partners in AEROGUST are: University of Bristol, INRIA, NLR, DLR, University of Cape Town, NUMECA, Optimad Engineering S.r.l., University of Liverpool, Airbus Defence and Space, Dassault Aviation, Piaggio Aerospace and Valeol.

References

[1] K. Mattsson, M. H. Carpenter, Stable and accurate interpolation operators for high-order multiblock finite difference methods, SIAM Journal on Scientific Computing 32 (2010) 2298–2320.

[2] J. E. Kozdon, L. C. Wilcox, Stable coupling of nonconforming, high-order finite difference methods, SIAM Journal on Scientific Computing (2016) A923–A952.

[3] A. Nissen, K. Kormann, M. Grandin, K. Virta, Stable difference methods for block-oriented adaptive grids, Journal of Scientific Computing 65 (2015) 486–511.

[4] M. H. Carpenter, M. Parsani, T. C. Fisher, E. J. Nielsen, Entropy stable staggered grid spectral collocation for the Burgers' and compressible Navier-Stokes equations, Technical report NASA TM 218990 (2015).

[5] M. Parsani, M. H. Carpenter, T. C. Fisher, E. J. Nielsen, Entropy stable staggered grid discontinuous spectral collocation methods of any order for the compressible Navier-Stokes equations, SIAM Journal on Scientific Computing 38 (2016) A3129–A3162.

[6] L. Friedrich, A. R. Winters, D. C. Del Rey Fernández, G. J. Gassner, M. Parsani, M. H. Carpenter, An entropy stable h / p non-conforming discontinuous galerkin method with the summation-by-parts property, Journal of Scientific Computing 77 (2018) 689–725.

[7] T. Lundquist, A. Malan, J. Nordström, A hybrid framework for coupling arbitrary summation-by-parts schemes on general meshes, Journal of Computational Physics 362 (2018) 49–68.

[8] M. Almquist, S. Wang, J. Werpers, Order-preserving interpolation for summation-by-parts operators at nonconforming grid interfaces, SIAM Journal on Scientific Computing 41 (2019) A1201–A1227.

[9] S. Nikkar, J. Nordström, Fully discrete energy stable high order finite difference methods for hyperbolic problems in deforming domains, Journal of Computational Physics 291 (2015) 82 – 98.

[10] C. V. Loan, On the method of weighting for equality-constrained least-squares problems, SIAM Journal on Numerical Analysis 22 (1985) 851–864.

[11] N. J. Higham, Iterative refinement enhances the stability of QR factorization methods for solving linear equations, BIT Numerical Mathematics 31 (1991) 447–468.

[12] B. Gustafsson, High Order Difference Methods for Time Dependent PDE, Springer-Verlag, 2008.

[13] T. A. Davis, Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization, ACM Trans. Math. Softw. 38 (2011) 8:1–8:22.

[14] J. Nordström, K. Forsberg, C. Adamsson, P. Eliasson, Finite volume methods, unstructured meshes and strict stability for hyperbolic problems, Applied Numerical Mathematics 45 (2003) 453–473.

[15] M. Svärd, J. Nordström, Stability of finite volume approximations for the Laplacian operator on quadrilateral and triangular grids, Applied Numerical Mathematics 51 (2004) 101–125.

[16] G. Leal, Advanced Transport Phenomena: Fluid Mechanics and Convective Transport Processes, Cambridge University Press, 2007.

[17] B. Sjögreen, H. Yee, M. Vinokur, On high order finite-difference metric discretizations satisfying GCL on moving and deforming grids, Journal of Computational Physics 265 (2014) 211 – 220.

Appendix A. Derivation of continuous integral identity

To see that (5) holds, we multiply (1) with a smooth function ϕ, integrate over Ω,

$$\int_\Omega \phi u_t \, dV + \int_\Omega \phi(au_x + bu_y) \, dV = 0, \tag{A.1}$$

and apply the product rule to the first term,

$$\int_\Omega \phi u_t \, dV = \int_\Omega (\phi u)_t \, dV - \int_\Omega \phi_t u \, dV. \tag{A.2}$$

The first term on the right hand side of (A.2) can be written out using Reynold's transport theorem (see e.g. [16]),

$$\int_\Omega (\phi u)_t \, dV = \frac{d}{dt}\left(\int_\Omega \phi u \, dV\right) - \oint_{\partial\Omega} \phi u(\dot{x}n_x + \dot{y}n_y) \, dS. \tag{A.3}$$

The second term in (A.1) can be written, using integration-by-parts,

$$\int_\Omega \phi(au_x + bu_y) \, dV = \oint_{\partial\Omega} \phi u(an_x + bn_y) \, dS - \int_\Omega (a\phi_x + b\phi_y)u \, dV. \tag{A.4}$$

After inserting equations (A.2), (A.3) and (A.4) into (A.1) and rearranging the terms, we get the form (5).

Appendix B. SBP operators

Appendix B.1. General unstructured form

We first assume that SBP operators in space $\mathcal{D}_x = \mathcal{P}^{-1}\mathcal{Q}_x$ and $\mathcal{D}_y = \mathcal{P}^{-1}\mathcal{Q}_y$ are available, such that

$$\mathcal{Q}_x + \mathcal{Q}_x^T = E^T P \boldsymbol{n}_x E, \quad \mathcal{Q}_y + \mathcal{Q}_y^T = E^T P \boldsymbol{n}_y E. \tag{B.1}$$

For example, if the finite volume method is used, then $\mathcal{P}$ contains the control volumes, and is hence time-independent for coordinate transformations where the control volumes are unchanged. In general, both unstructured and structured curvilinear operators can be expressed in this way, see [7]. As before, in (B.1) we only consider a single smooth segment of the boundary forming an interface with a neighbouring subdomain.

Next, consider the case $\phi = u = 1$ in (5),

$$\frac{d}{dt}\left(\int_\Omega dV\right) = -\oint_{\partial\Omega} \lambda \, dS = -\int_\Omega ((a-\dot{x})_x + (b-\dot{y})_y) \, dV = \int_\Omega (\dot{x}_x + \dot{y}_y) \, dV,$$

where in the second step we have applied the divergence theorem. Note that $\dot{x}_x$ and $\dot{y}_y$ denote partial derivatives of the coordinate velocity functions in (2). The same relation above of course also applies to smaller subsets of Ω. For a volume preserving transformation, we thus have

$$\dot{x}_x + \dot{y}_y = 0. \tag{B.2}$$

In the equation (4), we can now write,

$$\dot{x}u_x + \dot{y}u_y = (\dot{x}u)_x + (\dot{y}u)_y. \tag{B.3}$$

Using the spatial operators already introduced, we define the following split form discrete approximation of the terms in (4),

$$\mathcal{P}^{-1}\mathcal{Q} = -\frac{1}{2}\left(\dot{\mathbf{X}}\mathcal{D}_x + \mathcal{D}_x\dot{\mathbf{X}} + \dot{\mathbf{Y}}\mathcal{D}_y + \mathcal{D}_y\dot{\mathbf{Y}}\right) + a\mathcal{D}_x + b\mathcal{D}_y. \tag{B.4}$$

This yields, using (B.1),

$$\mathcal{Q} + \mathcal{Q}^T = -(E^T P \mathbf{n}_x E\dot{\mathbf{X}} + E^T P \mathbf{n}_y E\dot{\mathbf{Y}}) + aE^T P \mathbf{n}_x E + bE^T P \mathbf{n}_x E. \tag{B.5}$$

Since each row in E is unique and only contains a single non-zero entry with the value 1, we have $EE^T = I$, and $E^T E$ is a diagonal matrix with the value 1 inserted in the positions corresponding to the boundary [7]. Hence $E^T E$ commutes with the diagonal matrix $\dot{\mathbf{X}}$. Using this, we can write,

$$\begin{aligned} E^T P \mathbf{n}_x E\dot{\mathbf{X}} &= E^T P \mathbf{n}_x (EE^T) E\dot{\mathbf{X}} = E^T P \mathbf{n}_x E(E^T E)\dot{\mathbf{X}} \\ &= E^T P \mathbf{n}_x E\dot{\mathbf{X}}(E^T E) = E^T P \mathbf{n}_x \dot{\mathbf{x}} E, \end{aligned} \tag{B.6}$$

where $\dot{\mathbf{x}} = E\dot{\mathbf{X}}E^T$ is simply the restriction of $\dot{\mathbf{X}}$ to the boundary. After repeating the same procedure for the second term in (B.5), the SBP property (13) follows.

Appendix B.2. Structured operators

Even though the operator in (B.4) can be based on any form of SBP operators in space, in order to achieve free-stream preservation for structured methods on curvilinear domains, we will need to be more careful when defining $\mathcal{Q}$. Consider a bijective curvilinear transformation from $\Omega(t)$ to the unit square $(\xi, \eta) \in \hat{\Omega} = [0, 1]^2$. We can then write $u = u(t, \xi(t, x, y), \eta(t, x, y))$, and the substantive derivative du/dt can now bee seen as the time derivative expressed in the reference space $\hat{\Omega}$. The chain rule thus yields,

$$u_t = \frac{du}{dt} + \xi_t u_\xi + \eta_t u_\eta.$$

In terms of the notation used in (4) and throughout the rest of this paper, comparing this form of the chain rule to (3) now yields,

$$\dot{x}u_x + \dot{y}u_y = \frac{du}{dt} - u_t = -(\xi_t u_\xi + \eta_t u_\eta). \tag{B.7}$$

For a volume preserving transformation, geometric conservation (see e.g. eq. (9) in [17]) states,

$$(\mathcal{J}\xi_t)_\xi + (\mathcal{J}\eta_t)_\eta = 0,$$

where $\mathcal{J}$ is the Jacobian determinant of the transformation, which is constant in time. We can thus equivalently write (B.7) as,

$$\dot{x}u_x + \dot{y}u_y = -\mathcal{J}^{-1}\mathcal{J}(\xi_t u_\xi + \eta_t u_\eta) = -\mathcal{J}^{-1}((\mathcal{J}\xi_t u)_\xi + (\mathcal{J}\eta_t u)_\eta). \tag{B.8}$$

For future reference, the following metric relations are routinely utilized to compute the terms in (B.8) numerically.

$$\mathcal{J}\xi_t = -\dot{x}y_\eta + \dot{y}x_\eta \quad \mathcal{J}\eta_t = \dot{x}y_\xi - \dot{y}x_\xi. \tag{B.9}$$

To approximate (B.8) discretely, we start out from 1D SBP operators $D_\xi = P_\xi^{-1}Q_\xi$ and $D_\eta = P_\eta^{-1}Q_\eta$, and use them to define corresponding two-dimensional operators $\mathcal{D}_\xi = \hat{\mathcal{P}}^{-1}\mathcal{Q}_\xi$ and $\mathcal{D}_\eta = \hat{\mathcal{P}}^{-1}\mathcal{Q}_\eta$ in a tensor product fashion,

$$\hat{\mathcal{P}} = P_\xi \otimes P_\eta, \quad \mathcal{Q}_\xi = Q_\xi \otimes P_\eta, \quad \mathcal{D}_\eta = P_\xi \otimes Q_\eta.$$

Any type of one-dimensional SBP operators can be extended to two or more dimension in this way. Using a split between the two forms of $\dot{x}u_x + \dot{y}u_y$ in (B.8), we approximate the problem (4) using the operator,

$$\mathcal{P}^{-1}\mathcal{Q} = \frac{1}{2}\boldsymbol{\mathcal{J}}^{-1}((\boldsymbol{\mathcal{J}\xi}_t)\mathcal{D}_\xi + \mathcal{D}_\xi(\boldsymbol{\mathcal{J}\xi}_t) + (\boldsymbol{\mathcal{J}\eta}_t)\mathcal{D}_\eta + \mathcal{D}_\eta(\boldsymbol{\mathcal{J}\eta}_t)) + a\mathcal{D}_x + b\mathcal{D}_y, \tag{B.10}$$

together with $\mathcal{P} = \boldsymbol{\mathcal{J}}\hat{\mathcal{P}}$. Note that $\mathcal{P}$ defined in this way is trivially time-independent for a volume preserving mesh motion, due to the constant Jacobian determinant. The spatial operator $a\mathcal{D}_x + b\mathcal{D}_y = \mathcal{P}^{-1}(a\mathcal{Q}_x + b\mathcal{Q}_y)$ is given by,

$$a\mathcal{D}_x + b\mathcal{D}_y = \frac{1}{2}\boldsymbol{\mathcal{J}}^{-1}(\mathcal{D}_\xi\hat{\mathbf{A}} + \hat{\mathbf{A}}\mathcal{D}_\xi + \mathcal{D}_\eta\hat{\mathbf{B}} + \hat{\mathbf{B}}\mathcal{D}_\eta),$$

where

$$\hat{\mathbf{A}} = a\mathbf{Y}_\eta - b\mathbf{X}_\eta, \quad \hat{\mathbf{B}} = -a\mathbf{Y}_\xi + b\mathbf{X}_\xi.$$

and which satisfies the SBP property [7],

$$a(\mathcal{Q}_x + \mathcal{Q}_x^T) + b(\mathcal{Q}_y + \mathcal{Q}_y^T) = aE^T P\mathbf{n}_x E + bE^T P\mathbf{n}_y E.$$

From (B.10) we now have,

$$\mathcal{Q}+\mathcal{Q}^T = E^T\hat{P}\hat{\boldsymbol{n}}_\xi E(\boldsymbol{\mathcal{J}}\dot{\boldsymbol{\xi}})+E^T\hat{P}\hat{\boldsymbol{n}}_\eta E(\boldsymbol{\mathcal{J}}\dot{\boldsymbol{\eta}})+aE^TP\mathbf{n}_xE+bE^TP\mathbf{n}_yE. \quad \text{(B.11)}$$

In exactly the same way as (B.6) previously, the first two terms on the right hand side above can be written as,

$$\begin{aligned} E^T\hat{P}\hat{\boldsymbol{n}}_\xi E(\boldsymbol{\mathcal{J}}\dot{\boldsymbol{\xi}}) + E^T\hat{P}\hat{\boldsymbol{n}}_\eta E(\boldsymbol{\mathcal{J}}\dot{\boldsymbol{\eta}}) &= E^T\hat{P}\hat{\mathbf{n}}_\xi(E\boldsymbol{\mathcal{J}}\dot{\boldsymbol{\xi}}E^T)E + E^T\hat{P}\hat{\mathbf{n}}_\eta(E\boldsymbol{\mathcal{J}}\dot{\boldsymbol{\eta}}E^T)E \\ &= E^T\hat{P}\hat{\mathbf{n}}_\xi(-\dot{\mathbf{x}}\mathbf{y}_\eta + \dot{\mathbf{y}}\mathbf{x}_\eta)E + E^T\hat{P}\hat{\mathbf{n}}_\eta(\dot{\mathbf{x}}\mathbf{y}_\xi - \dot{\mathbf{y}}\mathbf{x}_\xi)E, \end{aligned}$$

where in the last step we have used the metric relations (B.9). At the north and south boundaries to $\hat{\Omega}$ we have $\hat{n}_\xi = \pm 1$ and $\hat{n}_\eta = 0$, and a parametrization with positive orientation is there given by $s = \mp\eta$. At the east and west boundaries, we similarly have $\hat{n}_\xi = 0$, $\hat{n}_\eta = \pm 1$ and $s = \pm\xi$. Continuing the analysis, we get,

$$\begin{aligned} E^T\hat{P}\hat{\boldsymbol{n}}_\xi E(\boldsymbol{\mathcal{J}}\dot{\boldsymbol{\xi}}) + E^T\hat{P}\hat{\boldsymbol{n}}_\eta E(\boldsymbol{\mathcal{J}}\dot{\boldsymbol{\eta}}) &= E^T\hat{P}(-\dot{\mathbf{x}}\mathbf{y}_s + \dot{\mathbf{y}}\mathbf{x}_s)E \\ &= -E^T\mathbf{J}\hat{P}(\dot{\mathbf{x}}\mathbf{n}_x + \dot{\mathbf{y}}\mathbf{n}_y)E, \end{aligned}$$

where $\mathbf{J}\hat{P} = P$ is the discrete integration operator along the physical domain boundary. In the last step we have used (9). Inserting this into (B.11) finally yields,

$$\mathcal{Q} + \mathcal{Q}^T = E^TP(-\dot{\mathbf{x}}\mathbf{n}_x - \dot{\mathbf{y}}\mathbf{n}_y + a\mathbf{n}_x + b\mathbf{n}_y)E = E^TP\boldsymbol{\lambda}E, \quad \text{(B.12)}$$

which is exactly the SBP property (13).

Appendix C. Free-stream preservation

In this section we investigate the free-stream preserving properties of the SBP operators introduced in the previous section, assuming rigid body mesh motions (28). Recall that either of the two forms of $\boldsymbol{\lambda}$ in (29) and (30) is sufficient for generating free-stream preserving IPP interpolation operators with an automatic approach.

Appendix C.1. Structured methods

For rigid body motions (28), the metric relations (B.9) simplify into

$$\mathcal{J}\xi_t = \left[-\bar{a}y + \bar{b}x + \omega(x^2+y^2)/2\right]_\eta \quad \mathcal{J}\eta_t = \left[\bar{a}y - \bar{b}x - \omega(x^2+y^2)/2\right]_\xi . \tag{C.1}$$

On the discrete side, we correspondingly define

$$\begin{aligned} \boldsymbol{\mathcal{J}\xi}_t &= \mathrm{Diag}(\mathcal{D}_\eta \left[-\bar{a}\mathbf{Y} + \bar{b}\mathbf{X} + (\mathbf{X}^2+\mathbf{Y}^2)/2\right] \mathbb{1}) \\ \boldsymbol{\mathcal{J}\eta}_t &= \mathrm{Diag}(\mathcal{D}_\xi \left[\bar{a}\mathbf{Y} - \bar{b}\mathbf{X} - (\mathbf{X}^2+\mathbf{Y}^2)/2\right] \mathbb{1}). \end{aligned} \tag{C.2}$$

With this definition, $\boldsymbol{\lambda}$ in (B.12) has exactly the form (30).

Now assume that the reference grid operators are consistent, i.e.

$$Q_\xi \mathbf{1} = 0, \quad Q_\eta \mathbf{1} = 0.$$

For the first part of the operator in (B.10), we then have,

$$\begin{aligned} \boldsymbol{\mathcal{J}}^{-1}((\boldsymbol{\mathcal{J}\xi}_t)\mathcal{Q}_\xi + \mathcal{Q}_\xi(\boldsymbol{\mathcal{J}\xi}_t) + (\boldsymbol{\mathcal{J}\eta}_t)\mathcal{Q}_\eta + \mathcal{Q}_\eta(\boldsymbol{\mathcal{J}\eta}_t))\mathbb{1} &= \boldsymbol{\mathcal{J}}^{-1}(\mathcal{Q}_\xi(\boldsymbol{\mathcal{J}\xi}_t) + \mathcal{Q}_\eta(\boldsymbol{\mathcal{J}\eta}_t))\mathbb{1} \\ &= \mathcal{P}(\mathcal{D}_\xi(\boldsymbol{\mathcal{J}\xi}_t) + \mathcal{D}_\eta(\boldsymbol{\mathcal{J}\eta}_t))\mathbb{1}. \end{aligned}$$

Now, since $\mathcal{D}_\xi\mathcal{D}_\eta = \mathcal{D}_\eta\mathcal{D}_\xi$ due to the tensor product structure, inserting (C.2) into the above yields

$$\begin{aligned} (\mathcal{D}_\xi(\boldsymbol{\mathcal{J}\xi}_t) + \mathcal{D}_\eta(\boldsymbol{\mathcal{J}\eta}_t))\mathbb{1} = (&\mathcal{D}_\xi\mathcal{D}_\eta \left[-\bar{a}\mathbf{Y} + \bar{b}\mathbf{X} + \omega(\mathbf{X}^2+\mathbf{Y}^2)/2\right] \\ &+\mathcal{D}_\eta\mathcal{D}_\xi \left[\bar{a}\mathbf{Y} - \bar{b}\mathbf{X} - \omega(\mathbf{X}^2+\mathbf{Y}^2)/2\right]) \mathbb{1} = 0. \end{aligned}$$

Similar results can easily be derived for $\mathcal{D}_x$ and $\mathcal{D}_y$ as well, showing that the SBP operator in (B.10) is free-stream preserving (22).

Appendix C.2. The unstructured finite volume method

The standard nodal finite volume method yields operators $\mathcal{D}_x = \mathcal{P}^{-1}\mathcal{Q}_x$ and $\mathcal{D}_y = \mathcal{P}^{-1}\mathcal{Q}_y$ which can be shown to satisfy [14],

$$\mathcal{Q}_x + \mathcal{Q}_x^T = -E^T \Delta\mathbf{y} E, \quad \mathcal{Q}_y + \mathcal{Q}_y^T = E^T \Delta\mathbf{x} E, \tag{C.3}$$

where in the interior we have

$$\Delta x_i = x_{i+1/2} - x_{i-1/2} = \frac{x_{i+1} - x_{i-1}}{2}.$$

If we consider the whole boundary forming a closed curve, periodic boundary closures apply,

$$\Delta x_1 = \tfrac{x_2 - x_N}{2}, \quad \Delta x_N = \tfrac{x_1 - x_{N-1}}{2}.$$

If we on the other hand consider just one smooth segment of the boundary, then

$$\Delta x_1 = \tfrac{x_2 - x_1}{2}, \quad \Delta x_N = \tfrac{x_N - x_{N-1}}{2},$$

and corresponding formulas of course also apply to $\Delta\mathbf{y}$. It follows that $\Delta\mathbf{x} = \mathrm{Diag}(Q_s\mathbf{x}\mathbf{1})$, $\Delta\mathbf{y} = \mathrm{Diag}(Q_s\mathbf{y}\mathbf{1})$, where the periodic and non-periodic version of the 1D SBP operator Q_s is respectively,

$$Q_s = \begin{pmatrix} 0 & \frac{1}{2} & & & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & & & -\frac{1}{2} & 0 \end{pmatrix}, \quad Q_s = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & -\frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

Next we define the parametrized integration operator as $\hat{P} = \Delta\mathbf{s} = \mathrm{Diag}(Q_s\mathbf{s})$, where $\mathbf{s}$ contains the nodal values of the parameter s. We now have $\Delta\mathbf{x} = \hat{P}\mathbf{x}_s$, $\Delta\mathbf{y} = \hat{P}\mathbf{y}_s$, where

$$\mathbf{x}_s = \mathrm{Diag}(\hat{P}^{-1}Q_s\mathbf{x}\mathbf{1}), \quad \mathbf{y}_s = \mathrm{Diag}(\hat{P}^{-1}Q_s\mathbf{y}\mathbf{1}).$$

Further, we can rewrite $\hat{P}$ into,

$$\hat{P} = \Delta\mathbf{s} = \frac{\sqrt{\Delta\mathbf{x}^2 + \Delta\mathbf{y}^2}}{\sqrt{\Delta\mathbf{x}^2 + \Delta\mathbf{y}^2}}\Delta\mathbf{s} = \frac{\sqrt{\Delta\mathbf{x}^2 + \Delta\mathbf{y}^2}}{\sqrt{(\frac{\Delta\mathbf{x}}{\Delta\mathbf{s}})^2 + (\frac{\Delta\mathbf{y}}{\Delta\mathbf{s}})^2}} = \frac{\sqrt{\Delta\mathbf{x}^2 + \Delta\mathbf{y}^2}}{\sqrt{\mathbf{x}_s^2 + \mathbf{y}_s^2}}.$$

This yields,

$$\Delta\mathbf{x} = \hat{P}\mathbf{x}_s = P\mathbf{J}^{-1}\mathbf{x}_s, \quad \Delta\mathbf{y} = \hat{P}\mathbf{x}_s = P\mathbf{J}^{-1}\mathbf{y}_s, \tag{C.4}$$

where

$$P = \sqrt{\Delta\mathbf{x}^2 + \Delta\mathbf{y}^2}, \quad \mathbf{J} = \sqrt{\mathbf{x}_s^2 + \mathbf{y}_s^2}.$$

The SBP operator definition (B.4) for a rigid body motion (28) together with (C.3) and (C.4) now leads to the form (29).

To see that the global operator $\mathcal{P}^{-1}\mathcal{Q}$ is in itself free-stream preserving, note that the finite volume method yields operators in space satisfying

$$\mathcal{Q}_x\mathbb{1} = 0, \quad \mathcal{Q}_y\mathbb{1} = 0.$$

This yields, again for (B.4) with (28),

$$\mathcal{Q}\mathbb{1} = \mathcal{Q}_y(\mathbf{x}\mathbb{1}) - \mathcal{Q}_x(\mathbf{y}\mathbb{1}),$$

and we state without proof that this expression is always zero with the finite volume method.