Research Article

Jaime Mora* and Leszek Demkowicz

Fast Integration of DPG Matrices Based on Sum Factorization for all the Energy Spaces

https://doi.org/10.1515/cmam-2018-0205
Received August 9, 2018; revised January 26, 2019; accepted March 1, 2019

Abstract: Numerical integration of the stiffness matrix in higher-order finite element (FE) methods is recognized as one of the heaviest computational tasks in an FE solver. The problem becomes even more relevant when computing the Gram matrix in the algorithm of the Discontinuous Petrov Galerkin (DPG) FE methodology. Making use of 3D tensor-product shape functions, and the concept of sum factorization, known from standard high-order FE and spectral methods, here we take advantage of this idea for the entire exact sequence of FE spaces defined on the hexahedron. The key piece to the presented algorithms is the exact sequence for the one-dimensional element, and use of hierarchical shape functions. Consistent with existing results, the presented algorithms for the integration of H^1, $H(\text{curl})$, $H(\text{div})$, and L^2 inner products, have the $\mathcal{O}(p^7)$ computational complexity in contrast to the $\mathcal{O}(p^9)$ cost of conventional integration routines. Use of Legendre polynomials for shape functions is critical in this implementation. Three boundary value problems under different variational formulations, requiring combinations of H^1, $H(\text{div})$ and $H(\text{curl})$ test shape functions, were chosen to experimentally assess the computation time for constructing DPG element matrices, showing good correspondence with the expected rates.

Keywords: Sum Factorization, Fast Integration, Tensor Product, DPG, Discontinuous Petrov Galerkin, Finite Element Method, Energy Spaces, Element Matrices

MSC 2010: 65N30, 65D30, 65Y20

1 Introduction

During the computation of a numerical solution to a linear boundary value problem using a finite element (FE) method, two major tasks are carried out by the computer processors, namely, calculation and assembly of the stiffness matrix and the load vector, and the solution of the linear system. The latter depends mostly on the size and properties of the global stiffness matrix, while the former depends more heavily on the local characteristics of every element in the mesh, as both the load vector and stiffness matrix are first computed at the element level. In higher-order finite elements, the calculation of the integrals that compose such matrix and vector may become costly since the higher the degree of a polynomial integrand, the larger the number of quadrature points needed to perform an exact (or at least a well approximated) numerical integration. Our interest in this article is to combine several results that can lead to significant savings in three-dimensional DPG computations.

***Corresponding author: Jaime Mora,** Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, 201 E 24th St, Austin, TX 78712, USA, e-mail: jmorapaz@ices.utexas.edu
Leszek Demkowicz, Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, 201 E 24th St, Austin, TX 78712, USA, e-mail: leszek@ices.utexas.edu

In FE, every type of element can be associated to one or more families of shape functions, whether for the trial space or for the test space. In two spatial dimensions, the only conventional tensor-product element is the quadrilateral, which is generated by tensor-multiplying two line segments. In 3D, the conventional element types are the hexahedron, the tetrahedron, the (triangular) prism and the pyramid (of quadrilateral base) [12]. Out of them, the hexahedron and the prism are examples of tensor-product elements. The hexahedron is a triple tensor product of 1D intervals, and the prism is a tensor product of a triangle in 2D and a 1D interval. Thus the standard shape functions for these two types of elements are tensor products of the shape functions associated to their lower-dimension generating elements.

The idea of using sum factorization for computing 2D and 3D integrals has been an established technique for spectral methods in CFD [14, 23], later adopted for p- and hp-FE with higher-order shape functions by Melenk, Gerdes and Schwab [19]. It was additionally implemented for the case of a fully automatic hp-adaptive FE solution of the Helmholtz equation by Kurtz [18], therein delivering explicit steps to compute the matrix while keeping a low memory requirement. More recently, the tensor-product nature of the shape functions in Bernstein–Bézier FE and isogeometric analysis also motivated the extension of this integration for the stiffness matrices of those methods [1, 2].

Use of fast integration algorithms becomes even more critical in an efficient implementation of the Discontinuous Petrov Galerkin (DPG) FE methodology. In order to see that, let us first refer to a classic Galerkin FE method in 3D. Let p denote the order of polynomial basis for discretizing the solution. The algebraic structure of the final linear system reads

$$\mathsf{B}\mathsf{u} = \mathsf{l}$$

with square stiffness matrix B of size $\mathcal{O}(p^3) \times \mathcal{O}(p^3)$, solution vector u and load vector l, both of the same size $\mathcal{O}(p^3)$. In DPG, we enrich the test space usually by increasing the polynomial order in Δp only for the test functions. Here the single equation above is replaced by a larger system

$$\begin{pmatrix} \mathsf{G} & \mathsf{B} & \widetilde{\mathsf{B}} \\ \mathsf{B}^{\mathsf{T}} & 0 & 0 \\ \widetilde{\mathsf{B}}^{\mathsf{T}} & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathsf{s} \\ \mathsf{u} \\ \mathsf{w} \end{pmatrix} = \begin{pmatrix} \mathsf{l} \\ 0 \\ 0 \end{pmatrix}$$

with additional unknowns s (size $\mathcal{O}((p+\Delta p)^3)$) and w (size $\mathcal{O}(p^2)$), matrices G (square, size $\mathcal{O}(p+\Delta p)^3$), B (resized to $\mathcal{O}((p+\Delta p)^3) \times \mathcal{O}(p^3)$), $\widetilde{\mathsf{B}}$ (size $\mathcal{O}((p+\Delta p)^3) \times \mathcal{O}(p^2)$), and load vector l (resized to $\mathcal{O}((p+\Delta p)^3)$). A conventional numerical integration of all these matrices has a complexity with leading terms of

$$\mathcal{O}((p+\Delta p)^9) + \mathcal{O}((p+\Delta p)^6 p^3)$$

floating point operations. Furthermore, we can retrieve a smaller symmetric system by statically condensing s, obtaining

$$\begin{pmatrix} \mathsf{B}^{\mathsf{T}}\mathsf{G}^{-1}\mathsf{B} & \mathsf{B}^{\mathsf{T}}\mathsf{G}^{-1}\widetilde{\mathsf{B}} \\ \widetilde{\mathsf{B}}^{\mathsf{T}}\mathsf{G}^{-1}\mathsf{B} & \widetilde{\mathsf{B}}^{\mathsf{T}}\mathsf{G}^{-1}\widetilde{\mathsf{B}} \end{pmatrix} \begin{pmatrix} \mathsf{u} \\ \mathsf{w} \end{pmatrix} = \begin{pmatrix} \mathsf{B}^{\mathsf{T}}\mathsf{G}^{-1}\mathsf{l} \\ \widetilde{\mathsf{B}}^{\mathsf{T}}\mathsf{G}^{-1}\mathsf{l} \end{pmatrix}.$$

Reaching this point involves a cost of $\mathcal{O}((p+\Delta p)^9)$ operations for the Cholesky factorization and necessary substitution steps to get $G^{-1}\mathsf{l}$, $\mathcal{O}((p+\Delta p)^9 p^3)$ for $G^{-1}\mathsf{B}$ and $G^{-1}\widetilde{\mathsf{B}}$, and a leading cost of $\mathcal{O}(p^6(p+\Delta p)^6)$ for the final matrix-matrix multiplications. These costs are almost unavoidable in order to have a solution using the DPG methodology, and we can handle them in an efficient way using highly specialized linear algebra libraries, which results in a smaller CPU time. Thus, if we want to get time savings in this technique, the part on which we can focus is the construction of G, B, $\widetilde{\mathsf{B}}$ and l. Matrix G, hereinafter referred to as the Gram matrix, is of special interest as it is the largest array within the element level calculations. Therefore, having as a basis the sum factorization approaches mentioned earlier, we aim in this paper to adapt those algorithms to the computation of G and finally obtain a significant saving in the implementation of DPG. In particular, we can easily extend these ideas to the integration of B if the variational formulation is the ultraweak. For more details on the derivation of the shown DPG system of equations, see Section 3.

In the present paper, when deriving the algorithms, we restrict ourselves to working with Gram matrices coming from the standard inner products only; however, it will be shown in one of the application cases that these ideas can also be extended to a more involved type of inner product. Moreover, we restrict our-

selves to real-valued functions for simplicity, but no major modification is anticipated when moving into complex-valued functions. We also work just on the hexahedron case. For the subject of FE shape functions, this article uses as a main reference the thorough review of shape functions for elements of all shapes done by Fuentes et al. [12]. We now outline the document: in Section 2, we explain the process of sum factorization and develop algorithms to apply the technique for all the Hilbert spaces belonging to the exact sequence; in Section 3, we provide several examples of DPG implementations and observe the computing time for the Gram matrix and other matrices, followed by a discussion. We will finally close the article with a few conclusions and mentions to possible future extensions of the present work.

2 Sum Factorization for all the Energy Spaces

The approach sought in this article is related to the concept of exact sequences of energy spaces. With that starting point, we go from infinite-dimensional energy spaces to finite-dimensional subspaces, which in the end are the ones we implement in any finite element method. We later define tensor-product spaces and proceed to study how to compute the Gram matrix for each space of the exact sequence for the hexahedron of the first type [22].

2.1 Exact Sequences

Let $X_0, X_1, \ldots, X_{N_s}$ be a finite family of vector spaces. Let $A_i\colon X_{i-1} \to X_i$ be a linear operator for $i = 1, \ldots, N_s$. We say that the following sequence or *complex*

$$X_0 \xrightarrow{A_1} X_1 \xrightarrow{A_2} \cdots \xrightarrow{A_{N_s}} X_{N_s}$$

is exact if, for $i = 1, 2, \ldots, N_s - 1$, it holds that $\mathsf{R}(A_i) = \mathsf{N}(A_{i+1})$ (where R denotes the range of the operator and N symbolizes the nullspace or kernel). In the context of energy spaces, i.e., Hilbert spaces for the solution of variational formulations of boundary value problems, we will define an exact sequence for our cases of interest, specifying both the energy spaces and operators involved.

In the paper, Ω will denote a bounded and simply connected domain in $\mathbb{R}^N$, $N = 1, 3$.

2.1.1 Exact Sequence in 1D ($\Omega = (a, b)$, $a, b \in \mathbb{R}$, $a < b$)

$$\mathbb{R} \xrightarrow{\mathrm{id}} H^1(\Omega) \xrightarrow{\partial} L^2(\Omega) \xrightarrow{0} \{0\}.$$

The presence of a zero operator in the last link of the exact sequence indicates that ∂ is a surjection. Similarly, the presence of the first link means that the nullspace of ∂ consists only of constant functions. Hereinafter, we omit writing both the first and last links of the sequence, remembering that the last operator must be a surjection, and that the nullspace of the first one is made only by constant functions.

2.1.2 Exact Sequence in 3D

The three-dimensional exact sequence involves all three classical vector calculus' differential operators,

$$H^1(\Omega) \xrightarrow{\nabla} H(\mathrm{curl}, \Omega) \xrightarrow{\mathrm{curl}} H(\mathrm{div}, \Omega) \xrightarrow{\mathrm{div}} L^2(\Omega). \tag{2.1}$$

We are interested in tensor-product three-dimensional finite elements equipped with discrete subspaces of each of the exact sequence spaces (2.1). The most relevant example is the hexahedron since it is a triple tensor product of the 1D simplex, the interval.

2.2 Tensor-Product Finite Element Shape Functions

Let $\mathcal{I} = (0, 1)$ be the master interval in $\mathbb{R}$, and denote the master hexahedron by $\hat{\mathcal{K}} := \mathcal{I}^3$ (in what follows, $\hat{\mathcal{K}}$ and $\mathcal{I}^3$ will be used interchangeably). Let $\mathbf{x}_{\mathcal{K}} : \hat{\mathcal{K}} \to \mathcal{K}$ be the element map, transforming the master element into a physical space element $\mathcal{K}$. Suppose $\mathbf{x}_{\mathcal{K}}$ is a diffeomorphism over $\hat{\mathcal{K}}$. Then the Jacobian matrix $\mathcal{J}$ is well defined,

$$\mathcal{J} := \frac{\partial \mathbf{x}_{\mathcal{K}}}{\partial \boldsymbol{\xi}},$$

where $\boldsymbol{\xi}$ is the position vector in the parametric (master) domain. The determinant of the Jacobian will be denoted $|\mathcal{J}| := \det \mathcal{J}$.

Let $T^{\text{grad}} : H^1(\hat{\mathcal{K}}) \to H^1(\mathcal{K})$ be the map that takes the H^1 finite element space defined on the master domain $\hat{\mathcal{K}}$ to the physical space element $\mathcal{K}$. In the same fashion, consider analogue maps denoted T^{curl}, T^{div}, T; these are the pullback or Piola maps for each function space in (2.1), and they are defined as follows (see [10, § 2.1.5]):

$$H^1(\hat{\mathcal{K}}) \ni \hat{u} \mapsto T^{\text{grad}}\hat{u} := \hat{u} \circ \mathbf{x}_{\mathcal{K}}^{-1} = u \in H^1(\mathcal{K}), \tag{2.2}$$

$$H(\text{curl}, \hat{\mathcal{K}}) \ni \hat{E} \mapsto T^{\text{curl}}\hat{E} := (\mathcal{J}^{-T}\hat{E}) \circ \mathbf{x}_{\mathcal{K}}^{-1} = E \in H(\text{curl}, \mathcal{K}), \tag{2.3}$$

$$H(\text{div}, \hat{\mathcal{K}}) \ni \hat{V} \mapsto T^{\text{div}}\hat{V} := (|\mathcal{J}|^{-1}\mathcal{J}\hat{V}) \circ \mathbf{x}_{\mathcal{K}}^{-1} = V \in H(\text{div}, \mathcal{K}), \tag{2.4}$$

$$L^2(\hat{\mathcal{K}}) \ni \hat{q} \mapsto T\hat{q} := (|\mathcal{J}|^{-1}\hat{q}) \circ \mathbf{x}_{\mathcal{K}}^{-1} = q \in L^2(\mathcal{K}). \tag{2.5}$$

As an important remark concerning notation, we have opted to use a circumflex or "hat" (ˆ) above nearly every function, domain or vector space defined in the master space.

Now it is known that, for an FE method implementation, we do not work with the entire energy space but with a finite-dimensional linear subspace, usually consisting of polynomials. When we work on a *mesh* of multiple elements, there exist different conditions for a finite-dimensional piecewise polynomial subspace to be *conforming* to (i.e., to be a proper subspace of) each energy space. For the case of H^1, the piecewise polynomial functions must be scalar-valued and globally continuous. In $H(\text{curl})$, we must construct vector-valued piecewise polynomials satisfying tangential continuity across elements. $H(\text{div})$-conformity requires continuity of the normal component of the vector-valued piecewise polynomial functions. Finally, L^2-conformity requires no kind of inter-element continuity for its scalar-valued functions. However, in this work, we focus on the computations on a single element; then inter-element continuity is not a concern in our derivations. However, it is valuable to be aware of this fact when choosing the finite element spaces and basis functions.

Following the notation in [10, 12], let us call the finite-dimensional subspaces for $\hat{\mathcal{K}}$ as follows:

$$\begin{aligned}
\hat{W}^p &\subsetneq H^1(\hat{\mathcal{K}}),\\
\hat{Q}^p &\subsetneq H(\text{curl}, \hat{\mathcal{K}}),\\
\hat{V}^p &\subsetneq H(\text{div}, \hat{\mathcal{K}}),\\
\hat{Y}^p &\subsetneq L^2(\hat{\mathcal{K}}),
\end{aligned}$$

where the superindex p symbolizes the nominal polynomial order of the sequence of spaces. Making sure that these finite-dimensional subspaces form themselves an exact sequence is important as it leads to useful properties when studying interpolants and approximability. We take that information as given because proving that fact is not central to this paper; thereby we just refer to those proofs within [10]. Through the Piola transformations, we can get the resulting finite-dimensional subspaces in the physical space element. Those would be

$$\begin{aligned}
T^{\text{grad}}\hat{W}^p &=: W^p \subsetneq H^1(\mathcal{K}),\\
T^{\text{curl}}\hat{Q}^p &=: Q^p \subsetneq H(\text{curl}, \mathcal{K}),\\
T^{\text{div}}\hat{V}^p &=: V^p \subsetneq H(\text{div}, \mathcal{K}),\\
T\hat{Y}^p &=: Y^p \subsetneq L^2(\mathcal{K}).
\end{aligned} \tag{2.6}$$

Due to the element map being a diffeomorphism, there is a unique correspondence between any function of the physical element subspaces and the master element subspaces:

- for all $u \in W^p$ there is exactly one $\hat{u} \in \hat{W}^p$ such that $u = T^{\text{grad}}\hat{u}$,
- for all $E \in Q^p$ there is exactly one $\hat{E} \in \hat{Q}^p$ such that $E = T^{\text{curl}}\hat{E}$,
- for all $V \in V^p$ there is exactly one $\hat{V} \in \hat{V}^p$ such that $V = T^{\text{div}}\hat{V}$,
- for all $q \in Y^p$ there is exactly one $\hat{q} \in \hat{Y}^p$ such that $q = T\hat{q}$.

Those finite-dimensional subspaces for the hexahedron are [10, 22]

$$\begin{aligned}
\hat{W}^p &= \mathcal{Q}^{p_1,p_2,p_3}(\mathcal{J}^3),\\
&\downarrow \nabla\\
\hat{Q}^p &= \mathcal{Q}^{p_1-1,p_2,p_3}(\mathcal{J}^3) \times \mathcal{Q}^{p_1,p_2-1,p_3}(\mathcal{J}^3) \times \mathcal{Q}^{p_1,p_2,p_3-1}(\mathcal{J}^3),\\
&\downarrow \text{curl}\\
\hat{V}^p &= \mathcal{Q}^{p_1,p_2-1,p_3-1}(\mathcal{J}^3) \times \mathcal{Q}^{p_1-1,p_2,p_3-1}(\mathcal{J}^3) \times \mathcal{Q}^{p_1-1,p_2-1,p_3}(\mathcal{J}^3),\\
&\downarrow \text{div}\\
\hat{Y}^p &= \mathcal{Q}^{p_1-1,p_2-1,p_3-1}(\mathcal{J}^3),
\end{aligned} \tag{2.7}$$

where p_1, p_2, p_3 are positive integers and, for any non-negative integers p, q, r, the space

$$\mathcal{Q}^{p,q,r}(\mathcal{J}^3) := \mathcal{P}^p(\mathcal{J}) \otimes \mathcal{P}^q(\mathcal{J}) \otimes \mathcal{P}^r(\mathcal{J})$$

with $\mathcal{P}^p(\mathcal{J})$ being the space of univariate polynomials defined over $\mathcal{J}$ of degree less than or equal to p. A space like $\mathcal{Q}^{p,q,r}(\mathcal{J}^3)$ is known as a tensor-product polynomial space. If $\{f_i^p;\ i = 0, \ldots, p\}$, $\{f_j^q;\ j = 0, \ldots, q\}$, $\{f_k^r;\ k = 0, \ldots, r\}$ are bases for the polynomial spaces $\mathcal{P}^p(\mathcal{J})$, $\mathcal{P}^q(\mathcal{J})$, $\mathcal{P}^r(\mathcal{J})$, respectively, then

$$\{f_{ijk}^{p,q,r};\ i = 0, \ldots, p;\ j = 0, \ldots, q;\ k = 0, \ldots, r\}$$

is a basis for $\mathcal{Q}^{p,q,r}(\mathcal{J}^3)$, where these functions are defined by

$$f_{ijk}^{p,q,r}(\boldsymbol{\xi}) = f_i^p(\xi_1) f_j^q(\xi_2) f_k^r(\xi_3) \quad \text{for all } \boldsymbol{\xi} \in \mathcal{J}^3;\ i = 0, \ldots, p;\ j = 0, \ldots, q;\ k = 0, \ldots, r.$$

Corresponding polynomial subspaces are also defined for the one-dimensional interval's exact sequence. They are

$$\begin{aligned}
\hat{W}_{\mathcal{J}}^p &= \mathcal{P}^p(\mathcal{J}) \subsetneq H^1(\mathcal{J}),\\
&\downarrow \partial\\
\hat{Y}_{\mathcal{J}}^p &= \mathcal{P}^{p-1}(\mathcal{J}) \subsetneq L^2(\mathcal{J}).
\end{aligned}$$

Following the property of the exact sequence explained above, it must hold that $\partial \hat{W}_{\mathcal{J}}^p = \hat{Y}_{\mathcal{J}}^p$, which is easy to verify.

Notice how the hexahedron's exact sequence may be reconstructed by using the 1D interval's exact sequence multiple times.

$$\begin{aligned}
\hat{W}^p &= \hat{W}_{\mathcal{J}}^{p_1} \otimes \hat{W}_{\mathcal{J}}^{p_2} \otimes \hat{W}_{\mathcal{J}}^{p_3},\\
&\downarrow \nabla=(\partial_1,\partial_2,\partial_3)\\
\hat{Q}^p &= \hat{Y}_{\mathcal{J}}^{p_1} \otimes \hat{W}_{\mathcal{J}}^{p_2} \otimes \hat{W}_{\mathcal{J}}^{p_3} \times \hat{W}_{\mathcal{J}}^{p_1} \otimes \hat{Y}_{\mathcal{J}}^{p_2} \otimes \hat{W}_{\mathcal{J}}^{p_3} \times \hat{W}_{\mathcal{J}}^{p_1} \otimes \hat{W}_{\mathcal{J}}^{p_2} \otimes \hat{Y}_{\mathcal{J}}^{p_3},\\
&\downarrow \text{curl}=(\partial_2(\cdot)_3-\partial_3(\cdot)_2,\partial_3(\cdot)_1-\partial_1(\cdot)_3,\partial_1(\cdot)_2-\partial_2(\cdot)_1)\\
\hat{V}^p &= \hat{W}_{\mathcal{J}}^{p_1} \otimes \hat{Y}_{\mathcal{J}}^{p_2} \otimes \hat{Y}_{\mathcal{J}}^{p_3} \times \hat{Y}_{\mathcal{J}}^{p_1} \otimes \hat{W}_{\mathcal{J}}^{p_2} \otimes \hat{Y}_{\mathcal{J}}^{p_3} \times \hat{Y}_{\mathcal{J}}^{p_1} \otimes \hat{Y}_{\mathcal{J}}^{p_2} \otimes \hat{W}_{\mathcal{J}}^{p_3},\\
&\downarrow \text{div}=\partial_1(\cdot)_1+\partial_2(\cdot)_2+\partial_3(\cdot)_3\\
\hat{Y}^p &= \hat{Y}_{\mathcal{J}}^{p_1} \otimes \hat{Y}_{\mathcal{J}}^{p_2} \otimes \hat{Y}_{\mathcal{J}}^{p_3}.
\end{aligned}$$

This shows that, if in an FE element subroutine we replace the calls to 3D shape functions by multiple calls to the 1D shape functions, we can reconstruct all the spaces in the exact sequence. As it will be explained below, doing this can provide great benefits with regard to computational performance.

Finally, the Gram matrix that is going to be studied throughout this work is explicitly defined as follows. Let $\mathcal{H}$ be a finite-dimensional Hilbert space with inner product $(\cdot,\cdot)_{\mathcal{H}}$, $\dim \mathcal{H} = N_h$, and a basis $\{h_I\}_{I=1}^{N_h}$. The Gram matrix $\mathsf{G}^{\mathcal{H}}$ is given by

$$\mathsf{G}^{\mathcal{H}}_{IJ} := (h_I, h_J)_{\mathcal{H}} \quad \text{for } I, J = 1, \dots, N_h. \tag{2.8}$$

Our goal is to propose algorithms to compute a Gram matrix (2.8) more efficiently than the conventional ones when dealing with tensor-product finite-element spaces of shape functions. All the energy spaces in the 3D exact sequence (2.1) are going to be analyzed (setting $\Omega = \mathcal{K}$); thereby a particular inner product is required to be defined in every case. The finite-dimensional Hilbert spaces will be those defined in (2.6).

The following subsections are presented in order of complexity instead of their position in the exact sequence. We remark that, although the first two cases to be described may be well known (i.e., L^2 and H^1), they introduce notions that are fundamental to familiarize with the procedure and so derive the other two cases ($H(\mathrm{div})$ and $H(\mathrm{curl})$ spaces – harder to find in the literature) in a systematic manner, which is possibly the main contribution of this work.

Remark 2.1. Even though the concept of exact sequence of finite element spaces has implications in approximability, we do not deal with that theory here. Instead, our main reason to invoke it is mostly to show that we are including in this work all four possible energy spaces. Other families of H^1, $H(\mathrm{curl})$, $H(\mathrm{div})$ and L^2 finite element subspaces that do not form exact sequences, but that are tensor-product derived and for which Piola mappings (2.2)–(2.5) are valid, are suitable too for the fast integration algorithms presented below. In particular, if those spaces form a complex (as defined in Section 2.1, i.e., if $\mathsf{R}(A_i) \subset \mathsf{N}(A_{i+1})$ for all i), the results must hold despite the non-exactness. However, we advise a careful implementation when trying the algorithms in such a scenario, as well as checking that our derivations remain valid for the spaces at hand.

Remark 2.2. The following sections comprise an intensive notation dealing with indices, which produce long and complicated expressions. To avoid further complexity and any possible confusion, no summation convention is used anywhere in this paper.

2.3 Space L^2

Let us recall the definition of the $L^2(\mathcal{K})$ space,

$$L^2(\mathcal{K}) = \left\{ \text{Lebesgue-measurable functions } f : \mathcal{K} \to \mathbb{R} : \int_{\mathcal{K}} |f(\boldsymbol{x})|^2 \, d^3\boldsymbol{x} < \infty \right\}.$$

The symbol for the inner product in $L^2(\mathcal{K})$, $(\cdot,\cdot)_{L^2(\mathcal{K})}$, typically incorporates the domain of integration as a subscript,

$$(\varphi, \vartheta)_{\mathcal{K}} := (\varphi, \vartheta)_{L^2(\mathcal{K})} = \int_{\mathcal{K}} \varphi(\boldsymbol{x}) \vartheta(\boldsymbol{x}) \, d^3\boldsymbol{x} \quad \text{for all } \varphi, \vartheta \in L^2(\mathcal{K}).$$

For vector-valued functions living in $\boldsymbol{L}^2(\mathcal{K}) := (L^2(\mathcal{K}))^3$, the associated inner product is defined componentwise, that is,

$$(\Phi, \Theta)_{\mathcal{K}} := (\Phi, \Theta)_{\boldsymbol{L}^2(\mathcal{K})} = \int_{\mathcal{K}} \Phi(\boldsymbol{x})^{\mathsf{T}} \Theta(\boldsymbol{x}) \, d^3\boldsymbol{x} = \sum_{d=1}^{3} (\varphi_d, \vartheta_d)$$
$$\text{for all } \Phi = (\varphi_1, \varphi_2, \varphi_3),\ \Theta = (\vartheta_1, \vartheta_2, \vartheta_3) \in \boldsymbol{L}^2(\mathcal{K}).$$

Let the order of the shape functions for the master hexahedron be (p_1, p_2, p_3), in the sense of the exact sequence (2.7). Consider a basis for Y^p, $\{v_I\}_{I=0}^{\dim Y^p - 1}$, where $\dim Y^p = p_1 p_2 p_3$. Thus, for any pair of integers $0 \le I, J < \dim Y^p$, the corresponding entry for the L^2 Gram matrix G is obtained as follows:

$$\begin{aligned} \mathsf{G}_{IJ} = (v_I, v_J)_{\mathcal{K}} &= \int_{\mathcal{K}} v_I(\boldsymbol{x}) v_J(\boldsymbol{x}) \, d^3\boldsymbol{x} \\ &= \int_{\hat{\mathcal{K}}} v_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi}) v_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})| \, d^3\boldsymbol{\xi} \end{aligned}$$

$$
\begin{aligned}
&= \int_{\hat{\mathcal{K}}} (T\hat{v}_I) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi}) (T\hat{v}_J) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})| \, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} [(|\mathcal{J}|^{-1}\hat{v}_I) \circ \mathbf{x}_{\mathcal{K}}^{-1}] \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi}) [(|\mathcal{J}|^{-1}\hat{v}_J) \circ \mathbf{x}_{\mathcal{K}}^{-1}] \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})| \, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} \hat{v}_I(\boldsymbol{\xi}) \hat{v}_J(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})|^{-1} \, d^3\boldsymbol{\xi} \\
&= \int_0^1 \int_0^1 \int_0^1 \hat{v}_I(\xi_1, \xi_2, \xi_3) \hat{v}_J(\xi_1, \xi_2, \xi_3) |\mathcal{J}(\xi_1, \xi_2, \xi_3)|^{-1} \, d\xi_3 \, d\xi_2 \, d\xi_1, \qquad (2.9)
\end{aligned}
$$

where the definition of the L^2 Piola map (see (2.5)) was applied, and the element map between the master and physical space was invoked to transform the integrand. In the last line of derivation (2.9), the volume integral over the master hexahedron is rewritten as three univariate integrals over the master interval. Now, since $\hat{v}_I$ and $\hat{v}_J$ belong to $\hat{Y}^p$, they are tensor-product polynomials in $\mathcal{Q}^{p_1-1,p_2-1,p_3-1}(\mathcal{I}^3)$; thus, if we take $\hat{v}_I$ as the model case, we have

$$
\hat{v}_I(\xi_1, \xi_2, \xi_3) := v_{1;i_1}(\xi_1) v_{2;i_2}(\xi_2) v_{3;i_3}(\xi_3),
$$

where the univariate polynomials $\{v_{a;i_a}\}_{i_a=0}^{p_a-1}$ form a basis of shape functions for the space $\mathcal{P}^{p_a-1}(\mathcal{I})$, for $a = 1, 2, 3$, and the integer indices $0 \le i_a < p_a$ are given so that they uniquely correspond to the original index I (e.g., through the formula $I = i_1 + p_1 i_2 + p_1 p_2 i_3$). It is important to remark that should we account for a hierarchical basis of polynomials, then we could use that basis for each $\mathcal{P}^{p_a-1}(\mathcal{I})$, and the need for the first identifier in the index of $v_{a;i_a}$ goes away. Assuming that is the case, we can rewrite $\hat{v}_I$ and $\hat{v}_J$ as

$$
\begin{aligned}
\hat{v}_I(\xi_1, \xi_2, \xi_3) &= v_{i_1}(\xi_1) v_{i_2}(\xi_2) v_{i_3}(\xi_3), \\
\hat{v}_J(\xi_1, \xi_2, \xi_3) &= v_{j_1}(\xi_1) v_{j_2}(\xi_2) v_{j_3}(\xi_3).
\end{aligned} \qquad (2.10)
$$

Combining (2.9) and (2.10), we get

$$
\begin{aligned}
(v_I, v_J)_{\mathcal{K}} &= \int_0^1 \int_0^1 \int_0^1 v_{i_1}(\xi_1) v_{i_2}(\xi_2) v_{i_3}(\xi_3) v_{j_1}(\xi_1) v_{j_2}(\xi_2) v_{j_3}(\xi_3) |\mathcal{J}(\xi_1, \xi_2, \xi_3)|^{-1} \, d\xi_3 \, d\xi_2 \, d\xi_1 \\
&= \int_0^1 v_{i_1}(\xi_1) v_{j_1}(\xi_1) \left\{ \int_0^1 v_{i_2}(\xi_2) v_{j_2}(\xi_2) \left[\int_0^1 v_{i_3}(\xi_3) v_{j_3}(\xi_3) |\mathcal{J}(\xi_1, \xi_2, \xi_3)|^{-1} \, d\xi_3 \right] d\xi_2 \right\} d\xi_1. \qquad (2.11)
\end{aligned}
$$

The second line above depicts how the original volume integral turns into three nested interval integrals through Fubini's theorem. Let us define a sequence of auxiliary functions with which we will assemble the triple integral in (2.11).

$$
\mathcal{G}^A_{i_3 j_3}(\xi_1, \xi_2) := \int_0^1 v_{i_3}(\xi_3) v_{j_3}(\xi_3) |\mathcal{J}(\xi_1, \xi_2, \xi_3)|^{-1} \, d\xi_3, \qquad (2.12)
$$

$$
\mathcal{G}^B_{i_2 j_2 i_3 j_3}(\xi_1) := \int_0^1 v_{i_2}(\xi_2) v_{j_2}(\xi_2) \mathcal{G}^A_{i_3 j_3}(\xi_1, \xi_2) \, d\xi_2, \qquad (2.13)
$$

$$
\implies \mathsf{G}_{IJ} = \mathcal{G}_{i_1 j_1 i_2 j_2 i_3 j_3} := \int_0^1 v_{i_1}(\xi_1) v_{j_1}(\xi_1) \mathcal{G}^B_{i_2 j_2 i_3 j_3}(\xi_1) \, d\xi_1. \qquad (2.14)
$$

Given ξ_1, ξ_2, we can evaluate $\mathcal{G}^A_{i_3 j_3}$ with numerical integration. Let ξ_3^n, $n = 1, \ldots, N$, be the collection of quadrature points in the interval $(0, 1)$ that make a polynomial integrand of degree $2N - 1$ be numerically integrated with full accuracy, and let w_3^N be the associated weight. Then the value of (2.12) may be approximated by

$$
\mathcal{G}^A_{i_3 j_3}(\xi_1, \xi_2) \approx \sum_{n=1}^{N} v_{i_3}(\xi_3^n) v_{j_3}(\xi_3^n) |\mathcal{J}(\xi_1, \xi_2, \xi_3^n)|^{-1} w_3^n. \qquad (2.15)
$$

In the same manner, let ξ_2^m, w_2^m with $m = 1, \ldots, M$ and ξ_1^l, w_1^l with $l = 1, \ldots, L$. We can approximate thus expressions (2.13) and (2.14) as

$$\mathcal{G}^B_{i_2 j_2 i_3 j_3}(\xi_1) \approx \sum_{m=1}^{M} \nu_{i_2}(\xi_2^m)\nu_{j_2}(\xi_2^m)\mathcal{G}^A_{i_3 j_3}(\xi_1, \xi_2^m)w_2^m, \tag{2.16}$$

$$\mathcal{G}_{i_1 j_1 i_2 j_2 i_3 j_3} \approx \sum_{l=1}^{L} \nu_{i_1}(\xi_1^l)\nu_{j_1}(\xi_1^l)\mathcal{G}^B_{i_2 j_2 i_3 j_3}(\xi_1^l)w_1^l. \tag{2.17}$$

Making distinction of different sets of quadrature points makes sense when the individual polynomial degrees are not equal, but even in that case, things may get simpler if we choose to work only with the largest of those sets in all cases, and we can therefore establish $L = M = N$, $\xi_1^l = \xi_2^l = \xi_3^l = \zeta^l$ and the weight $w_1^l = w_2^l = w_3^l = w^l$ for every l between 1 and L. Below, it will be clear that even after taking this shortcut, the complexity of the resulting algorithm will not be driven by such a quadrature order.

Suppose we want to approximate the triple integral in the last line of (2.9) without taking advantage of the sum factorization process. Then we will have L^3 quadrature points, which are the triplets $(\xi_1^l, \xi_2^m, \xi_3^n) =: \boldsymbol{\xi}_{lmn}$, with their associated weights $w_{lmn} := w^l w^m w^n$ so the approximate inner product will be

$$\mathsf{G}_{IJ} = (\upsilon_I, \upsilon_J)_{\mathcal{K}} \approx \sum_{l=1}^{L}\sum_{m=1}^{L}\sum_{n=1}^{L} \hat{\upsilon}_I(\boldsymbol{\xi}_{lmn})\hat{\upsilon}_J(\boldsymbol{\xi}_{lmn})|\mathcal{J}(\boldsymbol{\xi}_{lmn})|^{-1}w_{lmn}. \tag{2.18}$$

Clearly, (2.18) and (2.15)–(2.17) are equivalent if we use the same quadrature order per coordinate. However, here lies the entire spirit of the sum-factorization or tensor-product-based integration, as we will see. Firstly, it is a direct observation that the algorithm to compute (2.18) is the conventional one in 3D finite element codes, which is presented in Algorithm 1. Here, as usual in FE algorithms, the symmetry $\mathsf{G}_{IJ} = \mathsf{G}_{JI}$ is taken advantage of so that the off-diagonal entries are computed only once. Please note that whenever a subroutine **call** is made in any of the algorithms below, the first arguments given are inputs, with a semicolon separating them from the outputs. Moreover, outputs written within curly brackets denote arrays.

If we have a uniform polynomial degree $p = p_1 = p_2 = p_3$, we will need at least $L = p$ to compute the approximate integral. Consequently, in Algorithm 1, the accumulation statement is going to be executed $\frac{1}{2}p^6(p^3 + 1)$ times. This represents an operation count of $\mathcal{O}(p^9)$.

Now let us study the algorithm for the tensor-product-based integration. The way we nested the interval integrals in (2.11) suggests we can perform a similar ordering in the algorithm loops. The idea is to fix a quadrature point in the coordinate 1, then compute the corresponding term in the sum of (2.17), for which we need to go over each quadrature point in coordinate 2 in order to obtain the sum in (2.16), and at each step of those it is required to go over all the quadrature points in the third coordinate and evaluate the expression (2.15); all must iterate until the sum to approximate the inner product is completed.

procedure L2GRAM(i_{el}, G) ▷ Compute matrix G for element No. i_{el} – Conventional algorithm
  call setquadrature3D($i_{el}, p_1, p_2, p_3; L, \{\boldsymbol{\xi}_{lmn}, w_{lmn}\}$)
  $\mathsf{G} \leftarrow 0$ ▷ Initialize Gram matrix
  for $l, m, n = 1$ to L **do**
    call Shape3L2($\boldsymbol{\xi}_{lmn}, p_1, p_2, p_3; \{\hat{\upsilon}_I(\boldsymbol{\xi}_{lmn})\}$) ▷ Evaluate 3D shape functions at $\boldsymbol{\xi}_{lmn}$
    call geometry($\boldsymbol{\xi}_{lmn}, i_{el}$; **x**, $\mathcal{J}(\boldsymbol{\xi}_{lmn})$, $\mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn})$, $|\mathcal{J}|$) ▷ Compute **x** and Jacobian
    for $J = 0$ to dim $Y^p - 1$ **do**
      for $I = J$ to dim $Y^p - 1$ **do**
        $\mathsf{G}_{IJ} \leftarrow \mathsf{G}_{IJ} + \hat{\upsilon}_I(\boldsymbol{\xi}_{lmn})\hat{\upsilon}_J(\boldsymbol{\xi}_{lmn})|\mathcal{J}|^{-1}w_{lmn}$ ▷ Accumulate through (2.18)
  return G

Algorithm 1: Conventional computation of the L^2 Gram matrix.

procedure L2GramTensor(i_{el}, G) ▷ Compute matrix G for element No. i_{el} – Sum factorization
  $p_{\max} \leftarrow \max\{p_1, p_2, p_3\}$
  call setquadrature1D($i_{el}, p_{\max} - 1; L, \{\zeta^l, w^l\}$)
  $\mathcal{G} \leftarrow 0$ ▷ Initialize Gram matrix
  for $l = 1$ to L **do**
    call Shape1L2($\zeta^l, p_1; \{v_{i_1}(\zeta^l)\}$) ▷ Evaluate 1D shape functions at ζ^l
    for $j_3 = 0$ to $p_3 - 1$ **do**
      for $i_3 = j_3$ to $p_3 - 1$ **do**
        $\mathcal{G}^B \leftarrow 0$
        for $m = 1, L$ **do**
          call Shape1L2($\zeta^m, p_2; \{v_{i_2}(\zeta^m)\}$) ▷ Evaluate 1D shape functions at ζ^m
          $\mathcal{G}^A \leftarrow 0$
          for $n = 1$ to L **do**
            call Shape1L2($\zeta^n, p_3; \{v_{i_3}(\zeta^n)\}$) ▷ Evaluate 1D shape functions at ζ^n
            $\boldsymbol{\xi}_{lmn} \leftarrow (\zeta^l, \zeta^m, \zeta^n)$
            call geometry($\boldsymbol{\xi}_{lmn}, i_{el}; \mathbf{x}, \mathcal{J}(\boldsymbol{\xi}_{lmn}), \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn}), |\mathcal{J}|$) ▷ Compute $\mathbf{x}$ and Jacobian
            $\mathcal{G}^A_{i_3 j_3} \leftarrow \mathcal{G}^A_{i_3 j_3} + v_{i_3}(\zeta^n) v_{j_3}(\zeta^n) |\mathcal{J}|^{-1} w^n$ ▷ Accumulate through (2.15)
          for $j_2 = 0$ to $p_2 - 1$ **do**
            for $i_2 = j_2$ to $p_2 - 1$ **do**
              $\mathcal{G}^B_{i_2 j_2 i_3 j_3} \leftarrow \mathcal{G}^B_{i_2 j_2 i_3 j_3} + v_{i_2}(\zeta^m) v_{j_2}(\zeta^m) \mathcal{G}^A_{i_3 j_3}(\zeta^l, \zeta^m) w^m$ ▷ From (2.16)
        for $j_2 = 0$ to $p_2 - 1$ **do**
          for $i_2 = j_2$ to $p_2 - 1$ **do**
            for $j_1 = 0$ to $p_1 - 1$ **do**
              for $i_1 = j_1$ to $p_1 - 1$ **do**
                $\mathcal{G}_{i_1 j_1 i_2 j_2 i_3 j_3} \leftarrow \mathcal{G}_{i_1 j_1 i_2 j_2 i_3 j_3} + v_{i_1}(\zeta^l) v_{j_1}(\zeta^l) \mathcal{G}^B_{i_2 j_2 i_3 j_3}(\zeta^l) w^l$ ▷ From (2.17)
  return G

Algorithm 2: Computation of the L^2 Gram matrix by sum factorization.

Additionally, see (2.12)–(2.14) to notice the symmetry in $\mathcal{G}^A_{i_3 j_3} = \mathcal{G}^A_{j_3 i_3}$ and the two minor symmetries in the other auxiliary function $\mathcal{G}^B_{i_2 j_2 i_3 j_3} = \mathcal{G}^B_{j_2 i_2 i_3 j_3} = \mathcal{G}^B_{i_2 j_2 j_3 i_3} = \mathcal{G}^B_{j_2 i_2 j_3 i_3}$. Furthermore, we have the symmetry of the inner product, also noticeable in (2.14). Those symmetries can be exploited in order to make a more efficient computation of the Gram matrix. Algorithm 2 and the ones below include all the symmetry considerations and, whenever these algorithms are coded, it is important to later retrieve those entries that can be accessed doing use of symmetries.

Assuming that Algorithm 2 is used having uniform polynomial degree p, then the cost of the integration is driven by the accumulation for each index combination of $\mathcal{G}_{i_1 j_1 i_2 j_2 i_3 j_3}$, which is executed $p[\frac{1}{2}p(p+1)]^3$ times, making a cost of $\mathcal{O}(p^7)$.

It is worth mentioning that the array $\mathcal{G}_{i_1 i j_1 i_2 j_2 i_3 j_3}$ needs not be constructed, especially if taking into account computer memory issues. Instead, its value can be directly accumulated into the corresponding entry of G.

In case the polynomial degrees are not uniform, different 1D rules could be applied per coordinate. If so, a key point in the implementation of Algorithm 2 would be to reorder the integral nesting to make the quadrature points in the outermost loop correspond to the coordinate with the least polynomial order.

This saving of two orders of magnitude will be observed in all the remaining cases, although some may involve more complicated calculations.

Remark 2.3. The position of the loops for $i3, j3$ in Algorithm 2 was adopted from the one in Jason Kurtz's dissertation [18] and allows to save the memory associated to the indices $i3, j3$ in all the auxiliary arrays. We insist in this ordering for all the algorithms presented on this document because in DPG, as in any higher-order FE technique, the polynomial degrees used could make the size of those arrays considerably large.

procedure L2GramTensor(i_{el}, G) ▷ Compute G for element No. i_{el} – Alternative sum factorization
  $p_{\max} \leftarrow \max\{p_1, p_2, p_3\}$
  call setquadrature1D($i_{el}, p_{\max} - 1; L, \{\zeta^l, w^l\}$)
  $\mathcal{G} \leftarrow 0$ ▷ Initialize Gram matrix
  for $l = 1$ to L **do**
    call Shape1L2($\zeta^l, p_1; \{v_{i_1}(\zeta^l)\}$) ▷ Evaluate 1D shape functions at ζ^l
    $\mathcal{G}^B \leftarrow 0$
    for $m = 1, L$ **do**
      call Shape1L2($\zeta^m, p_2; \{v_{i_2}(\zeta^m)\}$) ▷ Evaluate 1D shape functions at ζ^m
      $\mathcal{G}^A \leftarrow 0$
      for $n = 1$ to L **do**
        call Shape1L2($\zeta^n, p_3; \{v_{i_3}(\zeta^n)\}$) ▷ Evaluate 1D shape functions at ζ^n
        for $j_3 = 0$ to $p_3 - 1$ **do**
          for $i_3 = j_3$ to $p_3 - 1$ **do**
            $\boldsymbol{\xi}_{lmn} \leftarrow (\zeta^l, \zeta^m, \zeta^n)$
            call geometry($\boldsymbol{\xi}_{lmn}, i_{el}; \mathbf{x}, \mathcal{J}(\boldsymbol{\xi}_{lmn}), \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn}), |\mathcal{J}|$) ▷ Compute $\mathbf{x}$ and Jacobian
            $\mathcal{G}^A_{i_3 j_3} \leftarrow \mathcal{G}^A_{i_3 j_3} + v_{i_3}(\zeta^n) v_{j_3}(\zeta^n) |\mathcal{J}|^{-1} w^n$ ▷ Accumulate through (2.15)
        for $j_3 = 0$ to $p_3 - 1$ **do**
          for $i_3 = j_3$ to $p_3 - 1$ **do**
            for $j_2 = 0$ to $p_2 - 1$ **do**
              for $i_2 = j_2$ to $p_2 - 1$ **do**
                $\mathcal{G}^B_{i_2 j_2 i_3 j_3} \leftarrow \mathcal{G}^B_{i_2 j_2 i_3 j_3} + v_{i_2}(\zeta^m) v_{j_2}(\zeta^m) \mathcal{G}^A_{i_3 j_3}(\zeta^l, \zeta^m) w^m$ ▷ From (2.16)
    for $j_3 = 0$ to $p_3 - 1$ **do**
      for $i_3 = j_3$ to $p_3 - 1$ **do**
        for $j_2 = 0$ to $p_2 - 1$ **do**
          for $i_2 = j_2$ to $p_2 - 1$ **do**
            for $j_1 = 0$ to $p_1 - 1$ **do**
              for $i_1 = j_1$ to $p_1 - 1$ **do**
                $\mathcal{G}_{i_1 j_1 i_2 j_2 i_3 j_3} \leftarrow \mathcal{G}_{i_1 j_1 i_2 j_2 i_3 j_3} + v_{i_1}(\zeta^l) v_{j_1}(\zeta^l) \mathcal{G}^B_{i_2 j_2 i_3 j_3}(\zeta^l) w^l$ ▷ From (2.17)
  return G

Algorithm 3: Computation of the L^2 Gram matrix – Alternative sum factorization.

Remark 2.4. Algorithm 2 leads to an operation count of $\mathcal{O}(p^5)$ for the statement "call geometry(…)". Notice that for low values of p, this expensive call may cost a portion similar to the final accumulation statement. Bringing into consideration Remark 2.3, if memory limitations are not a major concern, an alternative way of implementing the sum factorization algorithm is shown in Algorithm 3, which reduces by two orders of magnitude the number of times "call geometry(…)" is executed.

2.4 Space H^1

This energy space is defined as

$$H^1(\mathcal{K}) = \{u \in L^2(\mathcal{K}) : \nabla u \in \boldsymbol{L}^2(\mathcal{K})\}.$$

The inner product in $H^1(\mathcal{K})$ can be computed by means of the expression

$$(u, v)_{H^1(\mathcal{K})} := (u, v)_{\mathcal{K}} + (\nabla u, \nabla v)_{\mathcal{K}} \quad \text{for all } u, v \in H^1(\mathcal{K}).$$

The gradient ∇ used above is computed in the physical space, that is,

$$\nabla u = \left(\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial u}{\partial x_3}\right) = (\partial_1 u, \partial_2 u, \partial_3 u).$$

Moreover, we will need the gradient in the parametric space coordinates of a function $\hat{v}$ defined over $\hat{\mathcal{K}}$,

$$\hat{\nabla}\hat{v} = \left(\frac{\partial \hat{v}}{\partial \xi_1}, \frac{\partial \hat{v}}{\partial \xi_2}, \frac{\partial \hat{v}}{\partial \xi_3}\right) = (\hat{\partial}_1 \hat{v}, \hat{\partial}_2 \hat{v}, \hat{\partial}_3 \hat{v}).$$

The exact sequence in 3D described earlier, along with the Piola map definitions, imply that if $u \in H^1(\mathcal{K})$, then $\nabla u \in H(\mathrm{curl}, \mathcal{K})$ and that if $u = T^{\mathrm{grad}}\hat{u}$ for some $\hat{u} \in H^1(\hat{\mathcal{K}})$, then $\nabla u = T^{\mathrm{curl}}\hat{\nabla}\hat{u}$. Of course, we also have $\hat{\nabla}\hat{u} \in H(\mathrm{curl}, \hat{\mathcal{K}})$. In other words, we have the commutativity between the gradients and the Piola maps, i.e., $\nabla T^{\mathrm{grad}} = T^{\mathrm{curl}}\hat{\nabla}$.

Now let the order of the shape functions for the master hexahedron be (p_1, p_2, p_3), in the sense of the exact sequence (2.7). Consider a basis for W^p, $\{\varphi_I\}_{I=0}^{\dim W^p - 1}$, where $\dim W^p = (p_1+1)(p_2+2)(p_3+1)$. Thus, for any pair of integers $0 \le I, J < \dim W^p$, the inner product is obtained as derived in (2.19).

Analogously to the L^2 case, besides recalling the definitions of both T^{grad} and T^{curl}, in (2.19), we take the same steps as above to achieve an expression suitable for applying sum factorization to get the H^1 Gram matrix, herein denoted $\mathsf{G}^{\mathrm{grad}}$.

$$\begin{aligned}
\mathsf{G}^{\mathrm{grad}}_{IJ} &= (\varphi_I, \varphi_J)_{H^1(\mathcal{K})} \\
&= \int_{\mathcal{K}} \varphi_I(\boldsymbol{x})\varphi_J(\boldsymbol{x})\, d^3\boldsymbol{x} + \int_{\mathcal{K}} [\nabla\varphi_I(\boldsymbol{x})]^{\mathsf{T}}\nabla\varphi_J(\boldsymbol{x})\, d^3\boldsymbol{x} \\
&= \int_{\hat{\mathcal{K}}} \varphi_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})\varphi_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} + \int_{\hat{\mathcal{K}}} [\nabla\varphi_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}}\nabla\varphi_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} \big(T^{\mathrm{grad}}\hat{\varphi}_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})\big)\big(T^{\mathrm{grad}}\hat{\varphi}_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})\big)|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&\qquad + \int_{\hat{\mathcal{K}}} [\nabla(T^{\mathrm{grad}}\hat{\varphi}_I) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}}[\nabla(T^{\mathrm{grad}}\hat{\varphi}_J) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} \big((\hat{\varphi}_I \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})\big)\big((\hat{\varphi}_J \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})\big)|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&\qquad + \int_{\hat{\mathcal{K}}} [(T^{\mathrm{curl}}\hat{\nabla}\hat{\varphi}_I) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}}[(T^{\mathrm{curl}}\hat{\nabla}\hat{\varphi}_J) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} \hat{\varphi}_I(\boldsymbol{\xi})\hat{\varphi}_J(\boldsymbol{\xi})|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&\qquad + \int_{\hat{\mathcal{K}}} [((\mathcal{J}^{-T}\hat{\nabla}\hat{\varphi}_I) \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}}[((\mathcal{J}^{-T}\hat{\nabla}\hat{\varphi}_J) \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} \hat{\varphi}_I(\boldsymbol{\xi})\hat{\varphi}_J(\boldsymbol{\xi})|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} + \int_{\hat{\mathcal{K}}} [\hat{\nabla}\hat{\varphi}_I(\boldsymbol{\xi})]^{\mathsf{T}}\mathcal{D}(\boldsymbol{\xi})[\hat{\nabla}\hat{\varphi}_J(\boldsymbol{\xi})]|\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_0^1\int_0^1\int_0^1 \{\hat{\varphi}_I(\boldsymbol{\xi})\hat{\varphi}_J(\boldsymbol{\xi}) + [\hat{\nabla}\hat{\varphi}_I(\boldsymbol{\xi})]^{\mathsf{T}}\mathcal{D}(\boldsymbol{\xi})[\hat{\nabla}\hat{\varphi}_J(\boldsymbol{\xi})]\}|\mathcal{J}(\boldsymbol{\xi})|\, d\xi_3\, d\xi_2\, d\xi_1, \qquad (2.19)
\end{aligned}$$

where the symmetric matrix $\mathcal{D} := \mathcal{J}^{-1}\mathcal{J}^{-T}$ contains the following entries:

$$\mathcal{D} = \begin{pmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{pmatrix}$$

with $D_{12} = D_{21}$, $D_{13} = D_{31}$, $D_{32} = D_{23}$, and all of the entries being dependent on $\boldsymbol{\xi}$. The conventional algorithm for (2.19) is presented as Algorithm 4.

The sum factorization process for the present situation begins with defining a tensor-product shape function. As $\hat{\varphi}_I$, $\hat{\varphi}_I \in \hat{W}^p = \mathcal{Q}^{p_1,p_2,p_3}(\mathcal{I}^3)$, it follows that

$$\begin{aligned}
\hat{\varphi}_I(\xi_1, \xi_2, \xi_3) &= \chi_{i_1}(\xi_1)\chi_{i_2}(\xi_2)\chi_{i_3}(\xi_3), \\
\hat{\varphi}_J(\xi_1, \xi_2, \xi_3) &= \chi_{j_1}(\xi_1)\chi_{j_2}(\xi_2)\chi_{j_3}(\xi_3),
\end{aligned}$$

procedure H1GRAM(i_{el}, G^{grad}) ▷ Compute matrix G^{grad} for element No. i_{el} – Conventional algorithm
  call setquadrature3D($i_{el}, p_1, p_2, p_3; L, \{\boldsymbol{\xi}_{lmn}, w_{lmn}\}$)
  $\mathsf{G}^{\text{grad}} \leftarrow 0$ ▷ Initialize Gram matrix
  for $l, m, n = 1$ to L **do**
    call Shape3H1($\boldsymbol{\xi}_{lmn}, p_1, p_2, p_3; \{\hat{\varphi}_I(\boldsymbol{\xi}_{lmn})\}, \{\hat{\nabla}\hat{\varphi}_I(\boldsymbol{\xi}_{lmn})\}$) ▷ 3D shape functions at $\boldsymbol{\xi}_{lmn}$
    call geometry($\boldsymbol{\xi}_{lmn}, i_{el}; \mathbf{x}, \mathcal{J}(\boldsymbol{\xi}_{lmn}), \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn}), |\mathcal{J}|$) ▷ Compute x and Jacobian
    $\mathcal{D} \leftarrow \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn})\mathcal{J}^{-T}(\boldsymbol{\xi}_{lmn})$
    for $J = 0$ to dim $W^p - 1$ **do**
      for $I = J$ to dim $W^p - 1$ **do**
        $\mathsf{G}^{\text{grad}}_{IJ} \leftarrow \mathsf{G}^{\text{grad}}_{IJ} + \left\{\hat{\varphi}_I(\boldsymbol{\xi}_{lmn})\hat{\varphi}_J(\boldsymbol{\xi}_{lmn}) + [\hat{\nabla}\hat{\varphi}_I(\boldsymbol{\xi}_{lmn})]^{\intercal}\mathcal{D}[\hat{\nabla}\hat{\varphi}_J(\boldsymbol{\xi}_{lmn})]\right\}|\mathcal{J}|w_{lmn}$
  return G^{grad}

Algorithm 4: Conventional computation of the H^1 Gram matrix.

where the univariate polynomials $\{\chi_{i_a}\}_{i_a=0}^{p_a}$ are a hierarchical basis of the polynomial space $\mathcal{P}^{p_a}(\mathcal{I})$, for $a = 1, 2, 3$, and the integer indices $0 \le i_a, j_a \le p_a$ are given so that they hold a unique correspondence to the original indices I, J (such as $I = i_1 + (p_1 + 1)i_2 + (p_1 + 1)(p_2 + 1)i_3$). Notice that the application of the master-domain gradient to the shape functions makes effect to a single 1D basis function per component, that is,

$$\hat{\nabla}\hat{\varphi}_I = \begin{pmatrix} \chi'_{i_1}(\xi_1)\chi_{i_2}(\xi_2)\chi_{i_3}(\xi_3) \\ \chi_{i_1}(\xi_1)\chi'_{i_2}(\xi_2)\chi_{i_3}(\xi_3) \\ \chi_{i_1}(\xi_1)\chi_{i_2}(\xi_2)\chi'_{i_3}(\xi_3) \end{pmatrix}. \tag{2.20}$$

In the calculation of $\mathsf{G}^{\text{grad}}_{IJ}$, the integrand term $[\hat{\nabla}\hat{\varphi}_I(\boldsymbol{\xi}_{lmn})]^{\intercal}\mathcal{D}[\hat{\nabla}\hat{\varphi}_J(\boldsymbol{\xi}_{lmn})]$ represents the main change with respect to the L^2 problem. Using (2.20) for the expanded form of that expression, we derive the following (for intermediate steps, see [10]):

$$\begin{aligned}
[\hat{\nabla}\hat{\varphi}_I]^{\intercal}\mathcal{D}[\hat{\nabla}\hat{\varphi}_J] &= (\chi'_{i_1}\chi_{i_2}\chi_{i_3}, \chi_{i_1}\chi'_{i_2}\chi_{i_3}, \chi_{i_1}\chi_{i_2}\chi'_{i_3}) \begin{pmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{pmatrix} \begin{pmatrix} \chi'_{j_1}\chi_{j_2}\chi_{j_3} \\ \chi_{j_1}\chi'_{j_2}\chi_{j_3} \\ \chi_{j_1}\chi_{j_2}\chi'_{j_3} \end{pmatrix} \\
&= \chi'_{i_1}\chi'_{j_1}\chi_{i_2}\chi_{j_2}\chi_{i_3}\chi_{j_3}D_{11} \\
&\quad + \chi'_{i_1}\chi_{j_1}(\chi_{i_2}\chi'_{j_2}\chi_{i_3}\chi_{j_3}D_{12} + \chi_{i_2}\chi_{j_2}\chi_{i_3}\chi'_{j_3}D_{13}) \\
&\quad + \chi_{i_1}\chi'_{j_1}(\chi'_{i_2}\chi_{j_2}\chi_{i_3}\chi_{j_3}D_{21} + \chi_{i_2}\chi_{j_2}\chi'_{i_3}\chi_{j_3}D_{31}) \\
&\quad + \chi_{i_1}\chi_{j_1}(\chi'_{i_2}\chi'_{j_2}\chi_{i_3}\chi_{j_3}D_{22} + \chi'_{i_2}\chi_{j_2}\chi_{i_3}\chi'_{j_3}D_{23} + \chi_{i_2}\chi'_{j_2}\chi'_{i_3}\chi_{j_3}D_{32} + \chi_{i_2}\chi_{j_2}\chi'_{i_3}\chi'_{j_3}D_{33}).
\end{aligned} \tag{2.21}$$

It is now clear that each term in (2.21) is associated to one entry of $\mathcal{D}$. We can thus propose auxiliary functions identified by the indices of each D_{ab}. Additionally, after seeing the structure of (2.21), it would be quite useful to have a tool that allows, in a systematic way, to shift between the shape function χ_{i_a} and its derivative χ'_{i_a} from one term to another. The inclusion of a binary superscript $\langle s \rangle$ can do such a task,

$$\chi^{\langle s \rangle}_{i_a} = \begin{cases} \chi_{i_a} & \text{if } s = 0, \\ \chi'_{i_a} & \text{if } s = 1. \end{cases} \tag{2.22}$$

Additionally, a really intuitive yet handy function to manage that binary superscript may be a Kronecker delta. For instance, note that any term of (2.21) can be represented as

$$\chi^{\langle \delta_{1a} \rangle}_{i_1}\chi^{\langle \delta_{1b} \rangle}_{j_1}\chi^{\langle \delta_{2a} \rangle}_{i_2}\chi^{\langle \delta_{2b} \rangle}_{j_2}\chi^{\langle \delta_{3a} \rangle}_{i_3}\chi^{\langle \delta_{3b} \rangle}_{j_3}D_{ab} \quad \text{with } a, b = 1, 2, 3.$$

Although this whole article has as an important point of reference, the algorithm of sum factorization for the Helmholtz equation (whose stiffness matrix is really similar to an H^1 Gram matrix) in the dissertation of Jason Kurtz [18], where each term of the factorized form of the integral conveyed a different auxiliary function, here we decide to provide a single definition of auxiliary functions at each level. These are followed by

a sum over a, b, hence returning all the terms for the required Gram matrix. This new perspective was applied because one of the goals of the current work is to develop a general framework for the four spaces in the exact sequence. Consider the definitions

$$\mathcal{G}^{\text{grad}\,A}_{ab;i_3j_3}(\xi_1,\xi_2) := \int_0^1 \chi_{i_3}^{\langle\delta_{3a}\rangle}(\xi_3)\chi_{j_3}^{\langle\delta_{3b}\rangle}(\xi_3)D_{ab}(\xi_1,\xi_2,\xi_3)|\mathcal{J}(\xi_1,\xi_2,\xi_3)|\,d\xi_3, \tag{2.23}$$

$$\mathcal{G}^{\text{grad}\,B}_{ab;i_2j_2i_3j_3}(\xi_1) := \int_0^1 \chi_{i_2}^{\langle\delta_{2a}\rangle}(\xi_2)\chi_{j_2}^{\langle\delta_{2b}\rangle}(\xi_2)\mathcal{G}^{\text{grad}\,A}_{ab;i_3j_3}(\xi_1,\xi_2)\,d\xi_2, \tag{2.24}$$

$$\mathcal{G}^{\text{grad}}_{ab;i_1j_1i_2j_2i_3j_3} := \int_0^1 \chi_{i_1}^{\langle\delta_{1a}\rangle}(\xi_1)\chi_{j_1}^{\langle\delta_{1b}\rangle}(\xi_1)\mathcal{G}^{\text{grad}\,B}_{ab;i_2j_2i_3j_3}(\xi_1)\,d\xi_1. \tag{2.25}$$

On the other hand, the first term of the inner product is computed with a slight variation of (2.23)–(2.25), but with no derivative and without multiplying the integrand by D_{ab}, therefore dropping indices a, b.

$$\bar{\mathcal{G}}^{\text{grad}\,A}_{i_3j_3}(\xi_1,\xi_2) := \int_0^1 \chi_{i_3}(\xi_3)\chi_{j_3}(\xi_3)|\mathcal{J}(\xi_1,\xi_2,\xi_3)|\,d\xi_3, \tag{2.26}$$

$$\bar{\mathcal{G}}^{\text{grad}\,B}_{i_2j_2i_3j_3}(\xi_1) := \int_0^1 \chi_{i_2}(\xi_2)\chi_{j_2}(\xi_2)\bar{\mathcal{G}}^{\text{grad}\,A}_{i_3j_3}(\xi_1,\xi_2)\,d\xi_2, \tag{2.27}$$

$$\bar{\mathcal{G}}^{\text{grad}}_{i_1j_1i_2j_2i_3j_3} := \int_0^1 \chi_{i_1}(\xi_1)\chi_{j_1}(\xi_1)\bar{\mathcal{G}}^{\text{grad}\,B}_{i_2j_2i_3j_3}(\xi_1)\,d\xi_1. \tag{2.28}$$

The addition of the two parts described above yields

$$\mathsf{G}^{\text{grad}}_{IJ} = \bar{\mathcal{G}}^{\text{grad}}_{i_1j_1i_2j_2i_3j_3} + \sum_{a,b=1}^{3} \mathcal{G}^{\text{grad}}_{ab;i_1j_1i_2j_2i_3j_3}.$$

Given the auxiliary functions' definitions for this Gram matrix, some symmetries are possible to be identified so that many extra computations may be avoided.

$$\begin{aligned}
\mathcal{G}^{\text{grad}\,A}_{ab;i_1j_1} &= \mathcal{G}^{\text{grad}\,A}_{ba;j_1i_1} && \text{for all indices,}\\
\mathcal{G}^{\text{grad}}_{ab;i_1j_1i_2j_2i_3j_3} &= \mathcal{G}^{\text{grad}}_{ba;j_1i_1j_2i_2j_3i_3} && \text{for all indices,}\\
\bar{\mathcal{G}}^{\text{grad}\,A}_{i_3j_3} &= \bar{\mathcal{G}}^{\text{grad}\,A}_{j_3i_3} && \text{for all indices,}\\
\bar{\mathcal{G}}^{\text{grad}\,B}_{i_2j_2i_3j_3} &= \bar{\mathcal{G}}^{\text{grad}\,B}_{j_2i_2i_3j_3} = \bar{\mathcal{G}}^{\text{grad}\,B}_{i_2j_2j_3i_3} && \text{for all indices,}\\
\bar{\mathcal{G}}^{\text{grad}}_{i_1j_1i_2j_2i_3j_3} &= \bar{\mathcal{G}}^{\text{grad}}_{j_1i_1i_2j_2i_3j_3} = \bar{\mathcal{G}}^{\text{grad}}_{i_1j_1j_2i_2i_3j_3} = \bar{\mathcal{G}}^{\text{grad}}_{i_1j_1i_2j_2j_3i_3} && \text{for all indices.}
\end{aligned}$$

Having in mind the relations above, along with (2.23)–(2.28), we propose Algorithm 5 as the tensor-product-based method to compute the H^1 Gram matrix, in which a one-dimensional quadrature is applied to every coordinate, as seen in (2.15)–(2.17).

If a uniform order p is assumed in Algorithm 5, we need at least $L = p + 1$ for the integration, so the reader may estimate the operation count by summing the number of times the code line that accumulates $\mathcal{G}_{i_1j_1i_2j_2i_3j_3}$ and verify that it is $\mathcal{O}[(p+1)^7]$, two orders of magnitude below Algorithm 4.

2.5 Space $H(\mathbf{div})$

This third space contains functions whose divergence is square integrable, i.e.,

$$H(\text{div},\mathcal{K}) = \{V \in \boldsymbol{L}^2(\mathcal{K}) : \text{div}\,V \in L^2(\mathcal{K})\}.$$

procedure H1GramTensor(i_{el}, G^{grad}) ▷ Get G^{grad} for element No. i_{el} – Sum factorization
  $p_{\max} \leftarrow \max\{p_1, p_2, p_3\}$
  call setquadrature1D(i_{el}, $p_{\max}$; L, $\{\zeta^l, w^l\}$)
  $\bar{\mathcal{G}}^{\text{grad}}, \mathcal{G}^{\text{grad}} \leftarrow 0$ ▷ Initialize Gram matrix
  for $l = 1$ to L **do**
    call Shape1H1($\zeta^l, p_1; \{\chi_{i_1}(\zeta^l), \chi'_{i_1}(\zeta^l)\}$) ▷ Evaluate 1D shape functions at ζ^l
    for $j_3 = 0$ to p_3 **do**
      for $i_3 = j_3$ to p_3 **do**
        $\bar{\mathcal{G}}^{\text{grad}\,B}, \mathcal{G}^{\text{grad}\,B} \leftarrow 0$
        for $m = 1$ to L **do**
          call Shape1H1($\zeta^m, p_2; \{\chi_{i_2}(\zeta^m), \chi'_{i_2}(\zeta^m)\}$) ▷ Evaluate 1D shape functions at ζ^m
          $\bar{\mathcal{G}}^{\text{grad}\,A}, \mathcal{G}^{\text{grad}\,A} \leftarrow 0$
          for $n = 1$ to L **do**
            call Shape1H1($\zeta^n, p_3; \{\chi_{i_3}(\zeta^n), \chi'_{i_3}(\zeta^n)\}$) ▷ Evaluate 1D shape functions at ζ^n
            $\boldsymbol{\xi}_{lmn} \leftarrow (\zeta^l, \zeta^m, \zeta^n)$
            call geometry($\boldsymbol{\xi}_{lmn}, i_{el}$; $\mathbf{x}, \mathcal{J}(\boldsymbol{\xi}_{lmn}), \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn}), |\mathcal{J}|$) ▷ Compute $\mathbf{x}$ and Jacobian
            $\mathcal{D} \leftarrow \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn})\mathcal{J}^{-T}(\boldsymbol{\xi}_{lmn})$
            $\bar{\mathcal{G}}^{\text{grad}\,A}_{i_3 j_3} \leftarrow \bar{\mathcal{G}}^{\text{grad}\,A}_{i_3 j_3} + \chi_{i_3}(\zeta^n)\chi_{j_3}(\zeta^n)|\mathcal{J}|w^n$ ▷ Accumulate to obtain (2.26)
            for $a, b = 1$ to 3 **do**
              $\mathcal{G}^{\text{grad}\,A}_{ab;i_3 j_3} \leftarrow \mathcal{G}^{\text{grad}\,A}_{ab;i_3 j_3} + \chi^{\langle \delta_{3a}\rangle}_{i_3}(\zeta^n)\chi^{\langle \delta_{3b}\rangle}_{j_3} D_{ab}(\zeta^n)|\mathcal{J}|w^n$ ▷ Accumulate to obtain (2.23)
          for $j_2, i_2 = 0$ to p_2 **do**
            $\bar{\mathcal{G}}^{\text{grad}\,B}_{i_2 j_2 i_3 j_3} \leftarrow \bar{\mathcal{G}}^{\text{grad}\,B}_{i_2 j_2 i_3 j_3} + \chi_{i_2}(\zeta^m)\chi_{j_2}(\zeta^m)\bar{\mathcal{G}}^{\text{grad}\,A}_{i_3 j_3}(\zeta^l, \zeta^m)w^m$ ▷ By (2.27)
            for $a, b = 1$ to 3 **do**
              $\mathcal{G}^{\text{grad}\,B}_{ab;i_2 j_2 i_3 j_3} \leftarrow \mathcal{G}^{\text{grad}\,B}_{ab;i_2 j_2 i_3 j_3} + \chi^{\langle \delta_{2a}\rangle}_{i_2}(\zeta^m)\chi^{\langle \delta_{2b}\rangle}_{j_2}(\zeta^m)\mathcal{G}^{\text{grad}\,A}_{ab;i_3 j_3}(\zeta^l, \zeta^m)w^m$ ▷ By (2.24)
      for $j_2, i_2 = 0$ to p_2 **do**
        for $j_1, i_1 = 0$ to p_1 **do**
          $I = i_1 + (p_1 + 1)i_2 + (p_1 + 1)(p_2 + 1)i_3$
          $J = j_1 + (p_1 + 1)j_2 + (p_1 + 1)(p_2 + 1)j_3$
          if $J \geq I$ **then**
            $\bar{\mathcal{G}}^{\text{grad}}_{i_1 j_1 i_2 j_2 i_3 j_3} \leftarrow \bar{\mathcal{G}}^{\text{grad}}_{i_1 j_1 i_2 j_2 i_3 j_3} + \chi_{i_1}(\zeta^l)\chi_{j_1}(\zeta^l)\bar{\mathcal{G}}^{\text{grad}\,B}_{i_2 j_2 i_3 j_3}(\zeta^l)w^l$ ▷ (2.28)
            for $a, b = 1$ to 3 **do**
              $\mathcal{G}^{\text{grad}}_{ab;i_1 j_1 i_2 j_2 i_3 j_3} \leftarrow \mathcal{G}^{\text{grad}}_{ab;i_1 j_1 i_2 j_2 i_3 j_3} + \chi^{\langle \delta_{1a}\rangle}_{i_1}(\zeta^l)\chi^{\langle \delta_{1b}\rangle}_{j_1}(\zeta^l)\mathcal{G}^{\text{grad}\,B}_{ab;i_2 j_2 i_3 j_3}(\zeta^l)w^l$ ▷ (2.25)
  return G^{grad}

Algorithm 5: Computation of the H^1 Gram matrix by sum factorization.

The corresponding inner product is

$$(V, W)_{H(\text{div},\mathcal{K})} := (V, W)_{\mathcal{K}} + (\text{div}\, V, \text{div}\, W)_{\mathcal{K}} \quad \text{for all } V, W \in H(\text{div}, \mathcal{K}),$$

where the spatial-coordinates divergence is

$$\text{div}\, V = \partial_1 V_1 + \partial_2 V_2 + \partial_3 V_3.$$

Furthermore, the divergence in the master element coordinates of a function $\hat{W}$ is

$$\widehat{\text{div}}\, \hat{W} = \hat{\partial}_1 \hat{W}_1 + \hat{\partial}_2 \hat{W}_2 + \hat{\partial}_3 \hat{W}_3.$$

In a similar way to the previous problem, we have that if $V \in H(\text{div}, \mathcal{K})$ (and by definition $\text{div}\, V \in L^2(\mathcal{K})$), if $V = T^{\text{div}}\hat{V}$ for some $\hat{V} \in H(\text{div}, \hat{\mathcal{K}})$ (with $\widehat{\text{div}}\, \hat{V} \in L^2(\hat{\mathcal{K}})$), then $\text{div}\, V = T\, \widehat{\text{div}}\, \hat{V}$ thanks to another commutativity that holds for this space, $\text{div}\, T^{\text{div}} = T\, \widehat{\text{div}}$.

Now let the order of the shape functions for the master hexahedron be (p_1, p_2, p_3), in the sense of the exact sequence (2.7). Consider a basis for V^p, $\{\vartheta_I\}_{I=0}^{\dim V^p - 1}$, where

$$\dim V^p = (p_1+1)p_2p_3 + (p_2+1)p_3p_1 + (p_3+1)p_1p_2.$$

Central to the understanding of this third space is to remind that its elements are vector-valued functions. As in the H^1 case, the definitions of both T^{div} and T need to be invoked in (2.29). With this in mind, we take the same steps as above to achieve an expression suitable for sum factorization on the $H(\mathrm{div})$ Gram matrix, below denoted $\mathsf{G}^{\mathrm{div}}$. Thus, for any pair of integers $0 \le I, J < \dim V^p$, the inner product is obtained as follows:

$$\begin{aligned}
\mathsf{G}^{\mathrm{div}}_{IJ} &= (\vartheta_I, \vartheta_J)_{H(\mathrm{div},\mathcal{K})} \\
&= \int_{\mathcal{K}} \vartheta_I(\boldsymbol{x})^{\mathsf{T}} \vartheta_J(\boldsymbol{x})\, d^3\boldsymbol{x} + \int_{\mathcal{K}} \operatorname{div}\vartheta_I(\boldsymbol{x}) \operatorname{div}\vartheta_J(\boldsymbol{x})\, d^3\boldsymbol{x} \\
&= \int_{\hat{\mathcal{K}}} [\vartheta_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [\vartheta_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} + \int_{\hat{\mathcal{K}}} \operatorname{div}\vartheta_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi}) \operatorname{div}\vartheta_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} [T^{\mathrm{div}}\hat{\vartheta}_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [T^{\mathrm{div}}\hat{\vartheta}_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&\qquad + \int_{\hat{\mathcal{K}}} [\operatorname{div}(T^{\mathrm{div}}\hat{\vartheta}_I) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] [\operatorname{div}(T^{\mathrm{div}}\hat{\vartheta}_J) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} [(|\mathcal{J}|^{-1}\mathcal{J}\hat{\vartheta}_I \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [(|\mathcal{J}|^{-1}\mathcal{J}\hat{\vartheta}_J \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&\qquad + \int_{\hat{\mathcal{K}}} [(T\,\widehat{\operatorname{div}}\,\hat{\vartheta}_I) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] [(T\,\widehat{\operatorname{div}}\,\hat{\vartheta}_J) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} \hat{\vartheta}_I(\boldsymbol{\xi})^{\mathsf{T}} \mathcal{C}(\boldsymbol{\xi}) \hat{\vartheta}_J(\boldsymbol{\xi}) |\mathcal{J}^{-1}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&\qquad + \int_{\hat{\mathcal{K}}} [((|\mathcal{J}|^{-1}\,\widehat{\operatorname{div}}\,\hat{\vartheta}_I) \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] [((|\mathcal{J}|^{-1}\,\widehat{\operatorname{div}}\,\hat{\vartheta}_J) \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} \hat{\vartheta}_I(\boldsymbol{\xi})^{\mathsf{T}} \mathcal{C}(\boldsymbol{\xi}) \hat{\vartheta}_J(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})|^{-1}\, d^3\boldsymbol{\xi} + \int_{\hat{\mathcal{K}}} \widehat{\operatorname{div}}\,\hat{\vartheta}_I(\boldsymbol{\xi})\, \widehat{\operatorname{div}}\,\hat{\vartheta}_J(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})|^{-1}\, d^3\boldsymbol{\xi} \\
&= \int_0^1 \int_0^1 \int_0^1 \{\hat{\vartheta}_I(\boldsymbol{\xi})^{\mathsf{T}} \mathcal{C}(\boldsymbol{\xi}) \hat{\vartheta}_J(\boldsymbol{\xi}) + \widehat{\operatorname{div}}\,\hat{\vartheta}_I(\boldsymbol{\xi})\, \widehat{\operatorname{div}}\,\hat{\vartheta}_J(\boldsymbol{\xi})\} |\mathcal{J}(\boldsymbol{\xi})|^{-1}\, d\xi_3\, d\xi_2\, d\xi_1,
\end{aligned} \tag{2.29}$$

where $\mathcal{C} := \mathcal{J}^{\mathsf{T}}\mathcal{J} = (\mathcal{J}^{-1}\mathcal{J}^{-T})^{-1} = \mathcal{D}^{-1}$ contains the following entries:

$$\mathcal{C} = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{pmatrix}$$

with $C_{12} = C_{21}$, $C_{13} = C_{31}$, $C_{32} = C_{23}$, and every entry depends on $\boldsymbol{\xi}$. The conventional algorithm for (2.29) is presented as Algorithm 6.

The sum factorization process for the present situation begins by defining a tensor-product shape function. As $\hat{\vartheta}_I, \hat{\vartheta}_I \in \hat{V}^p = \mathcal{Q}^{p_1,p_2-1,p_3-1}(\mathcal{J}^3) \times \mathcal{Q}^{p_1-1,p_2,p_3-1}(\mathcal{J}^3) \times \mathcal{Q}^{p_1-1,p_2-1,p_3}(\mathcal{J}^3)$, it follows that

$$\begin{aligned}
\hat{\vartheta}_I(\xi_1, \xi_2, \xi_3) &= \mu_{a,1;i_1}(\xi_1)\mu_{a,2;i_2}(\xi_2)\mu_{a,3;i_3}(\xi_3)\hat{\mathbf{e}}_a, \\
\hat{\vartheta}_J(\xi_1, \xi_2, \xi_3) &= \mu_{b,1;j_1}(\xi_1)\mu_{b,2;j_2}(\xi_2)\mu_{b,3;j_3}(\xi_3)\hat{\mathbf{e}}_b,
\end{aligned} \tag{2.30}$$

where $\hat{\mathbf{e}}_a$, $a = 1, 2, 3$, is the a-th canonical Cartesian unit vector in the master space; the univariate polynomials $\mu_{a,d;i_d}(\xi_d)$, $d = 1, 2, 3$, are determined through the rule

$$\mu_{a,d;i_d} = \begin{cases} \chi_{i_d} & \text{if } a = d, \\ \nu_{i_d} & \text{otherwise.} \end{cases} \tag{2.31}$$

procedure HDIVGRAM(i_{el}, $\mathsf{G}^{\mathrm{div}}$) ▷ Compute matrix $\mathsf{G}^{\mathrm{div}}$ for element No. i_{el} – Conventional algorithm
  call setquadrature3D($i_{el}, p_1, p_2, p_3; L, \{\boldsymbol{\xi}_{lmn}, w_{lmn}\}$)
  $\mathsf{G}^{\mathrm{div}} \leftarrow 0$ ▷ Initialize Gram matrix
  for $l, m, n = 1$ to L **do**
    call Shape3Hdiv($\boldsymbol{\xi}_{lmn}, p_1, p_2, p_3; \{\hat{\vartheta}_I(\boldsymbol{\xi}_{lmn})\}, \{\widehat{\mathrm{div}}\,\hat{\vartheta}_I(\boldsymbol{\xi}_{lmn})\}$) ▷ 3D shape functions at $\boldsymbol{\xi}_{lmn}$
    call geometry($\boldsymbol{\xi}_{lmn}, i_{el}; \mathbf{x}, \mathcal{J}(\boldsymbol{\xi}_{lmn}), \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn}), |\mathcal{J}|$) ▷ Compute $\mathbf{x}$ and Jacobian
    $\mathcal{C} \leftarrow \mathcal{J}^{\mathsf{T}}(\boldsymbol{\xi}_{lmn})\mathcal{J}(\boldsymbol{\xi}_{lmn})$
    for $J = 0$ to $\dim V^p - 1$ **do**
      for $I = J$ to $\dim V^p - 1$ **do**
        $\mathsf{G}^{\mathrm{div}}_{IJ} \leftarrow \mathsf{G}^{\mathrm{div}}_{IJ} + \{\hat{\vartheta}_I(\boldsymbol{\xi}_{lmn})^{\mathsf{T}}\mathcal{C}\hat{\vartheta}_J(\boldsymbol{\xi}_{lmn}) + \widehat{\mathrm{div}}\,\hat{\vartheta}_I(\boldsymbol{\xi}_{lmn})\,\widehat{\mathrm{div}}\,\hat{\vartheta}_J(\boldsymbol{\xi}_{lmn})\}|\mathcal{J}|^{-1}w_{lmn}$
  return $\mathsf{G}^{\mathrm{div}}$

Algorithm 6: Conventional computation of the $H(\mathrm{div})$ Gram matrix.

where it is recalled that $\{\chi_{i_a}\}_{i_a=0}^{p_a}$ are a hierarchical basis of $W_{\mathcal{J}}^{p_a}$, while $\{v_{i_a}\}_{i_a=0}^{p_a-1}$ form a basis for $Y_{\mathcal{J}}^{p_a}$ (for $a = 1, 2, 3$). In this new scenario, the integer indices i_a, j_a depend on the vector component where $\hat{\vartheta}_I$ lies. The correspondence formula between the tensor-product index I and the one-dimensional ones could be of the form

$$I = \begin{cases} i_1 + (p_1+1)i_2 + (p_1+1)p_2 i_3 & \text{if } a = 1, \\ (p_1+1)p_2p_3 + i_1 + p_1 i_2 + p_1(p_2+1)i_3 & \text{if } a = 2, \\ (p_1+1)p_2p_3 + p_1(p_2+1)p_3 + i_1 + p_1 i_2 + p_1 p_2 i_3 & \text{if } a = 3. \end{cases} \tag{2.32}$$

Furthermore, it should be acknowledged that if $\{\chi_{i_a}\}_{i_a=0}^{p_a}$ is a hierarchical basis of $\mathcal{P}^{p_a}(\mathcal{J})$, then $\{\chi'_{i_a}\}_{i_a=1}^{p_a}$ is a hierarchical basis of $\mathcal{P}^{p_a-1}(\mathcal{J}) = Y_{\mathcal{J}}^{p_a}$. We can therefore use this basis by letting

$$v_{i_d} = \chi'_{i_d+1} \quad \text{for } i_d = 0, \dots, p_d - 1, \tag{2.33}$$

having $p_d = p_1, p_2, p_3$.

By combining (2.30), (2.31) and (2.33), the tensor-product $H(\mathrm{div})$ shape functions and its master-coordinate divergence can be rewritten as

$$\hat{\vartheta}_I(\xi_1, \xi_2, \xi_3) = \chi_{i_a}(\xi_a)\chi'_{i_{a+1}+1}(\xi_{a+1})\chi'_{i_{a+2}+1}(\xi_{a+2})\hat{\mathbf{e}}_a, \tag{2.34}$$

$$\widehat{\mathrm{div}}\,\hat{\vartheta}_I(\xi_1, \xi_2, \xi_3) = \chi'_{i_a}(\xi_a)\chi'_{i_{a+1}+1}(\xi_{a+1})\chi'_{i_{a+2}+1}(\xi_{a+2}) \tag{2.35}$$

with $0 \le i_a \le p_a$, $0 \le i_{a+1} < p_{a+1}$, $0 \le i_{a+2} < p_{a+2}$, where $a = 1, 2, 3$ and the indices $(a, a+1, a+2)$ are interpreted as a cyclic permutation of $(1, 2, 3)$.

Notice that if we introduce an additional identifier to the index, we can restate the shape function in terms of (ξ_1, ξ_2, ξ_3), which is desirable before the integral nesting of the tensor-product-based algorithm. As above, a Kronecker delta associated to the vector component where the shape function lies can do that work for us. But in this case, as seen in (2.34), we require the derivative in the factors depending on the coordinates other than ξ_a, so we need a *complement* of the Kronecker delta. For that purpose we make a new definition:

$$\gamma_{cd} = 1 - \delta_{cd} \quad \text{for } c, d = 1, 2, 3.$$

Additionally, the notation convention to know if we need χ or its derivative turns again useful. Making use of (2.22) jointly with the Kronecker delta complement for the index, we can obtain new equations for $\hat{\vartheta}_I$ and $\widehat{\mathrm{div}}\,\hat{\vartheta}_I$, for which the effect of (the component index) a on each univariate polynomial is obtained more systematically than in (2.34) and (2.35), respectively.

$$\hat{\vartheta}_I(\xi_1, \xi_2, \xi_3) = \chi^{\langle \gamma_{1a} \rangle}_{i_1+\gamma_{1a}}(\xi_1)\chi^{\langle \gamma_{2a} \rangle}_{i_2+\gamma_{2a}}(\xi_2)\chi^{\langle \gamma_{3a} \rangle}_{i_3+\gamma_{3a}}(\xi_3)\hat{\mathbf{e}}_a, \tag{2.36}$$

$$\widehat{\mathrm{div}}\,\hat{\vartheta}_I(\xi_1, \xi_2, \xi_3) = \chi'_{i_1+\gamma_{1a}}(\xi_1)\chi'_{i_2+\gamma_{2a}}(\xi_2)\chi'_{i_3+\gamma_{3a}}(\xi_3). \tag{2.37}$$

Using (2.37) in the second term of the Gram matrix entry derived in (2.29), we get to

$$\int_{\hat{\mathcal{K}}} \widehat{\operatorname{div}}\,\hat{\vartheta}_I \widehat{\operatorname{div}}\,\hat{\vartheta}_J |\mathcal{J}|^{-1}\, d^3\boldsymbol{\xi} = \int_{\hat{\mathcal{K}}} \chi'_{i_1+\gamma_{1a}} \chi'_{i_2+\gamma_{2a}} \chi'_{i_3+\gamma_{3a}} \chi'_{j_1+\gamma_{1b}} \chi'_{j_2+\gamma_{2b}} \chi'_{j_3+\gamma_{3b}} |\mathcal{J}|^{-1}\, d^3\boldsymbol{\xi}, \tag{2.38}$$

which, due to (2.33), is indeed an L^2 inner product just as in (2.11). Notice that in (2.38) we abolished the writing of arguments in order to shorten the number of symbols present, but it clearly holds that each univariate shape function is dependent on the $\boldsymbol{\xi}$ component with its same index. As we have presented an algorithm for this type of integral, let us focus on the first term of (2.29) instead, and we will later incorporate both of them in a single algorithm.

Take (2.36) as a representation of the $H(\operatorname{div})$ shape functions $\hat{\vartheta}_I$, $\hat{\vartheta}_J$. Insert this representation into the first term of (2.29), and rearrange in 1D integrals to obtain a nested integral

$$\begin{aligned}\int_{\hat{\mathcal{K}}} \hat{\vartheta}_I^{\mathsf{T}}\, \mathcal{C}\, \hat{\vartheta}_J |\mathcal{J}|^{-1}\, d^3\boldsymbol{\xi} &= \int_{\hat{\mathcal{K}}} \chi^{\langle\gamma_{1a}\rangle}_{i_1+\gamma_{1a}} \chi^{\langle\gamma_{2a}\rangle}_{i_2+\gamma_{2a}} \chi^{\langle\gamma_{3a}\rangle}_{i_3+\gamma_{3a}} \hat{\mathbf{e}}_a^{\mathsf{T}} \mathcal{C} \chi^{\langle\gamma_{1b}\rangle}_{j_1+\gamma_{1b}} \chi^{\langle\gamma_{2b}\rangle}_{j_2+\gamma_{2b}} \chi^{\langle\gamma_{3b}\rangle}_{j_3+\gamma_{3b}} \hat{\mathbf{e}}_b |\mathcal{J}|^{-1}\, d^3\boldsymbol{\xi} \\ &= \int_0^1 \chi^{\langle\gamma_{1a}\rangle}_{i_1+\gamma_{1a}} \chi^{\langle\gamma_{1b}\rangle}_{j_1+\gamma_{1b}} \left\{ \int_0^1 \chi^{\langle\gamma_{2a}\rangle}_{i_2+\gamma_{2a}} \chi^{\langle\gamma_{2b}\rangle}_{j_2+\gamma_{2b}} \left[\int_0^1 \chi^{\langle\gamma_{3a}\rangle}_{i_3+\gamma_{3a}} \chi^{\langle\gamma_{3b}\rangle}_{j_3+\gamma_{3b}} C_{ab} |\mathcal{J}|^{-1}\, d\xi_3 \right] d\xi_2 \right\} d\xi_1,\end{aligned}$$

where we have used the fact that $\hat{\mathbf{e}}_a^{\mathsf{T}} \mathcal{C} \hat{\mathbf{e}}_b = C_{ab}$.

This is where the corresponding auxiliary functions for $H(\operatorname{div})$ are introduced.

$$\mathcal{G}^{\operatorname{div} A}_{ab;i_3 j_3}(\xi_1, \xi_2) := \int_0^1 \chi^{\langle\gamma_{3a}\rangle}_{i_3+\gamma_{3a}}(\xi_3) \chi^{\langle\gamma_{3b}\rangle}_{j_3+\gamma_{3b}}(\xi_3) C_{ab}(\xi_1, \xi_2, \xi_3) |\mathcal{J}(\xi_1, \xi_2, \xi_3)|^{-1}\, d\xi_3, \tag{2.39}$$

$$\mathcal{G}^{\operatorname{div} B}_{ab;i_2 j_2 i_3 j_3}(\xi_1) := \int_0^1 \chi^{\langle\gamma_{2a}\rangle}_{i_2+\gamma_{2a}}(\xi_2) \chi^{\langle\gamma_{2b}\rangle}_{j_2+\gamma_{2b}}(\xi_2) \mathcal{G}^{\operatorname{div} A}_{ab;i_3 j_3}(\xi_1, \xi_2)\, d\xi_2, \tag{2.40}$$

$$\mathcal{G}^{\operatorname{div}}_{ab;i_1 j_1 i_2 j_2 i_3 j_3} := \int_0^1 \chi^{\langle\gamma_{1a}\rangle}_{i_1+\gamma_{1a}}(\xi_1) \chi^{\langle\gamma_{1b}\rangle}_{j_1+\gamma_{1b}}(\xi_1) \mathcal{G}^{\operatorname{div} B}_{ab;i_2 j_2 i_3 j_3}(\xi_1)\, d\xi_1. \tag{2.41}$$

In these newly defined auxiliary function arrays for the first term of the $H(\operatorname{div})$ Gram matrix, some symmetries arise as well as they did in the L^2 and H^1 cases. We do not explicitly state them now, but the reader is encouraged to explore such properties, having as a main criterion the fact that the Gram matrix as a whole is symmetric.

Recall that the second term of the inner product, written in tensor-product form in (2.38), must be added to (2.41). The calculation of that term is practically a copy of Algorithm 2, but some caution has to be taken with the indices. We will denote those slightly modified auxiliary arrays $\widetilde{\mathcal{G}}^{\operatorname{div} A}$, $\widetilde{\mathcal{G}}^{\operatorname{div} B}$, $\widetilde{\mathcal{G}}^{\operatorname{div}}$. Next, by implementing a 1D quadrature for the approximation of each auxiliary function (2.39)–(2.41), similar to those in (2.15)–(2.17), we get the third sum factorization algorithm for the computation of a hexahedral element's Gram matrix, Algorithm 7.

2.6 Space $H(\operatorname{curl})$

The last energy space needs its functions to have a square-integrable curl. It is defined as

$$H(\operatorname{curl}, \mathcal{K}) = \{E \in \boldsymbol{L}^2(\mathcal{K}) : \operatorname{curl} E \in \boldsymbol{L}^2(\mathcal{K})\}.$$

The inner product for this space is

$$(E, F)_{H(\operatorname{curl},\mathcal{K})} := (E, F)_{\mathcal{K}} + (\operatorname{curl} E, \operatorname{curl} F)_{\mathcal{K}} \quad \text{for all } E, F \in H(\operatorname{curl}, \mathcal{K}).$$

In this definition, the curl is given by

$$\operatorname{curl} E = (\partial_2 E_3 - \partial_3 E_2, \partial_3 E_1 - \partial_1 E_3, \partial_1 E_2 - \partial_2 E_1).$$

procedure HDIVGRAMTENSOR(i_{el}, $\mathsf{G}^{\mathrm{div}}$) ▷ $\mathsf{G}^{\mathrm{div}}$ for element No. i_{el} – Sum factorization

- $p_{\max} \leftarrow \max\{p_1, p_2, p_3\}$
- **call** setquadrature1D($i_{el}, p_{\max}; L, \{\zeta^l, w^l\}$)
- $\mathcal{G}^{\mathrm{div}}, \widetilde{\mathcal{G}}^{\mathrm{div}} \leftarrow 0$ ▷ Initialize Gram matrix
- **for** $l = 1$ to L **do**
  - **call** Shape1H1($\zeta^l, p_1; \{\chi_{i_1}(\zeta^l), \chi'_{i_1}(\zeta^l)\}$) ▷ Evaluate 1D shape functions at ζ^l
  - **for** $j_3 = 0$ to p_3 **do**
    - **for** $i_3 = j_3$ to p_3 **do**
      - $\mathcal{G}^{\mathrm{div}\,B}, \widetilde{\mathcal{G}}^{\mathrm{div}\,B} \leftarrow 0$
      - **for** $m = 1$ to L **do**
        - **call** Shape1H1($\zeta^m, p_2; \{\chi_{i_2}(\zeta^m), \chi'_{i_2}(\zeta^m)\}$) ▷ Evaluate 1D shape functions at ζ^m
        - $\mathcal{G}^{\mathrm{div}\,A}, \widetilde{\mathcal{G}}^{\mathrm{div}\,A} \leftarrow 0$
        - **for** $n = 1$ to L **do**
          - **call** Shape1H1($\zeta^n, p_3; \{\chi_{i_3}(\zeta^n), \chi'_{i_3}(\zeta^n)\}$) ▷ Evaluate 1D shape functions at ζ^n
          - $\boldsymbol{\xi}_{lmn} \leftarrow (\zeta^l, \zeta^m, \zeta^n)$
          - **call** geometry($\boldsymbol{\xi}_{lmn}, i_{el}; \mathbf{x}, \mathcal{J}(\boldsymbol{\xi}_{lmn}), \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn}), |\mathcal{J}|$) ▷ Compute $\mathbf{x}$ and Jacobian
          - $\mathcal{C} \leftarrow \mathcal{J}^{\mathsf{T}}(\boldsymbol{\xi}_{lmn})\mathcal{J}(\boldsymbol{\xi}_{lmn})$
          - **for** $a, b = 1$ to 3 **do**
            - **if** $j_3 + \gamma_{3b} \le p_3$ and $i_3 + \gamma_{3a} \le p_3$ **then** ▷ Avoids extra computations
              - $\mathcal{G}^{\mathrm{div}\,A}_{ab;i_3j_3} \leftarrow \mathcal{G}^{\mathrm{div}\,A}_{ab;i_3j_3} + \chi^{\langle\gamma_{3a}\rangle}_{i_3+\gamma_{3a}}(\zeta^n)\chi^{\langle\gamma_{3b}\rangle}_{j_3+\gamma_{3b}}(\zeta^n)C_{ab}|\mathcal{J}|^{-1}w^n$ ▷ To obtain (2.39)
              - $\widetilde{\mathcal{G}}^{\mathrm{div}\,A}_{ab;i_3j_3} \leftarrow \widetilde{\mathcal{G}}^{\mathrm{div}\,A}_{ab;i_3j_3} + \chi'_{i_3+\gamma_{3a}}(\zeta^n)\chi'_{j_3+\gamma_{3b}}(\zeta^n)|\mathcal{J}|^{-1}w^n$
        - **for** $j_2, i_2 = 0$ to p_2 **do**
          - **for** $a, b = 1$ to 3 **do**
            - **if** $j_2 + \gamma_{2b} \le p_2$ and $i_2 + \gamma_{2a} \le p_2$ **then** ▷ Avoids extra computations
              - $\mathcal{G}^{\mathrm{div}\,B}_{ab;i_2j_2i_3j_3} \leftarrow \mathcal{G}^{\mathrm{div}\,B}_{ab;i_2j_2i_3j_3} + \chi^{\langle\gamma_{2a}\rangle}_{i_2+\gamma_{2a}}(\zeta^m)\chi^{\langle\gamma_{2b}\rangle}_{j_2+\gamma_{2b}}(\zeta^m)\mathcal{G}^{\mathrm{div}\,A}_{ab;i_3j_3}w^m$ ▷ (2.40)
              - $\widetilde{\mathcal{G}}^{\mathrm{div}\,B}_{ab;i_2j_2i_3j_3} \leftarrow \widetilde{\mathcal{G}}^{\mathrm{div}\,B}_{ab;i_2j_2i_3j_3} + \chi'_{i_2+\gamma_{2a}}(\zeta^m)\chi'_{j_2+\gamma_{2b}}(\zeta^m)\widetilde{\mathcal{G}}^{\mathrm{div}\,A}_{ab;i_3j_3}w^m$
      - **for** $j_2, i_2 = 0$ to p_2 **do**
        - **for** $j_1, i_1 = 0$ to p_1 **do**
          - Determine I, J through (2.32)
          - **if** $J \ge I$ **then**
            - **for** $a, b = 1$ to 3 **do**
              - **if** $j_1 + \gamma_{1b} \le p_1$ and $i_1 + \gamma_{1a} \le p_1$ **then** ▷ Avoids extra computations
                - $\mathcal{G}^{\mathrm{div}}_{ab;i_1j_1i_2j_2i_3j_3} \leftarrow \mathcal{G}^{\mathrm{div}}_{ab;i_1j_1i_2j_2i_3j_3} + \chi^{\langle\gamma_{1a}\rangle}_{i_1+\gamma_{1a}}(\zeta^l)\chi^{\langle\gamma_{1b}\rangle}_{j_1+\gamma_{1b}}(\zeta^l)\mathcal{G}^{\mathrm{div}\,B}_{ab;i_2j_2i_3j_3}w^l$
                - $\widetilde{\mathcal{G}}^{\mathrm{div}}_{ab;i_1j_1i_2j_2i_3j_3} \leftarrow \widetilde{\mathcal{G}}^{\mathrm{div}}_{ab;i_1j_1i_2j_2i_3j_3} + \chi'_{i_1+\gamma_{1a}}(\zeta^l)\chi'_{j_1+\gamma_{1b}}(\zeta^l)\widetilde{\mathcal{G}}^{B}_{ab;i_2j_2i_3j_3}w^l$
- **return** $\mathsf{G}^{\mathrm{div}}$

Algorithm 7: Computation of the $H(\mathrm{div})$ Gram matrix by sum factorization.

Its counterpart in master coordinates is

$$\widehat{\mathrm{curl}}\,\hat{F} = (\hat{\partial}_2\hat{F}_3 - \hat{\partial}_3\hat{F}_2, \hat{\partial}_3\hat{F}_1 - \hat{\partial}_1\hat{F}_3, \hat{\partial}_1\hat{F}_2 - \hat{\partial}_2\hat{F}_1).$$

In this last space, the commutativity that is applicable to our derivations is given by $\mathrm{curl}\, T^{\mathrm{curl}} = T^{\mathrm{div}}\,\widehat{\mathrm{curl}}$.

Now let the order of the shape functions for the master hexahedron be (p_1, p_2, p_3), in the sense of the exact sequence (2.7). Consider a basis for Q^p, $\{\psi_I\}_{I=0}^{\dim Q^p - 1}$, where

$$\dim Q^p = p_1(p_2+1)(p_3+1) + (p_1+1)p_2(p_3+1) + (p_1+1)(p_2+1)p_3.$$

In this fourth energy space, the elements are vector-valued functions too. As in the H^1 and $H(\mathrm{div})$ cases, the definitions of both T^{curl} and T^{div} need to be invoked during the derivation of (2.42). Taking in consideration the previous problems, we take the same steps in order to obtain an expression suitable for the sum factor-

ization of the $H(\mathrm{curl})$ Gram matrix, hereinafter called $\mathsf{G}^{\mathrm{curl}}$. Thus, for any pair of integers $0 \le I, J < \dim Q^p$, the inner product is equivalent to the following expressions:

$$
\begin{aligned}
\mathsf{G}^{\mathrm{curl}}_{IJ} &= (\psi_I, \psi_J)_{H(\mathrm{curl},\mathcal{K})} \\
&= \int_{\mathcal{K}} \psi_I(\boldsymbol{x})^{\mathsf{T}} \psi_J(\boldsymbol{x})\, d^3\boldsymbol{x} + \int_{\mathcal{K}} [\mathrm{curl}\, \psi_I(\boldsymbol{x})]^{\mathsf{T}} [\mathrm{curl}\, \psi_J(\boldsymbol{x})]\, d^3\boldsymbol{x} \\
&= \int_{\hat{\mathcal{K}}} [\psi_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [\psi_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} + \int_{\hat{\mathcal{K}}} [\mathrm{curl}\, \psi_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [\mathrm{curl}\, \psi_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} [T^{\mathrm{curl}} \hat{\psi}_I \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [T^{\mathrm{curl}} \hat{\psi}_J \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&\qquad + \int_{\hat{\mathcal{K}}} [\mathrm{curl}(T^{\mathrm{curl}} \hat{\psi}_I) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [\mathrm{curl}(T^{\mathrm{curl}} \hat{\psi}_J) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} [(\mathcal{J}^{-T} \hat{\psi}_I \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [(\mathcal{J}^{-T} \hat{\psi}_J \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&\qquad + \int_{\hat{\mathcal{K}}} [(T^{\mathrm{div}}\, \widehat{\mathrm{curl}}\, \hat{\psi}_I) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [(T^{\mathrm{div}}\, \widehat{\mathrm{curl}}\, \hat{\psi}_J) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} \hat{\psi}_I(\boldsymbol{\xi})^{\mathsf{T}} \mathcal{D}(\boldsymbol{\xi}) \hat{\psi}_J(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&\qquad + \int_{\hat{\mathcal{K}}} [((|\mathcal{J}|^{-1} \mathcal{J}\, \widehat{\mathrm{curl}}\, \hat{\psi}_I) \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})]^{\mathsf{T}} [((|\mathcal{J}|^{-1} \mathcal{J}\, \widehat{\mathrm{curl}}\, \hat{\psi}_J) \circ \mathbf{x}_{\mathcal{K}}^{-1}) \circ \mathbf{x}_{\mathcal{K}}(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} \\
&= \int_{\hat{\mathcal{K}}} \hat{\psi}_I(\boldsymbol{\xi})^{\mathsf{T}} \mathcal{D}(\boldsymbol{\xi}) \hat{\psi}_J(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})|\, d^3\boldsymbol{\xi} + \int_{\hat{\mathcal{K}}} [\widehat{\mathrm{curl}}\, \hat{\psi}_I(\boldsymbol{\xi})]^{\mathsf{T}} \mathcal{C}(\boldsymbol{\xi}) [\widehat{\mathrm{curl}}\, \hat{\psi}_J(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|^{-1}\, d^3\boldsymbol{\xi} \\
&= \int_0^1 \int_0^1 \int_0^1 \{\hat{\psi}_I(\boldsymbol{\xi})^{\mathsf{T}} \mathcal{D}(\boldsymbol{\xi}) \hat{\psi}_J(\boldsymbol{\xi}) |\mathcal{J}(\boldsymbol{\xi})| + [\widehat{\mathrm{curl}}\, \hat{\psi}_I(\boldsymbol{\xi})]^{\mathsf{T}} \mathcal{C}(\boldsymbol{\xi}) [\widehat{\mathrm{curl}}\, \hat{\psi}_J(\boldsymbol{\xi})] |\mathcal{J}(\boldsymbol{\xi})|^{-1}\}\, d\xi_3\, d\xi_2\, d\xi_1. \qquad (2.42)
\end{aligned}
$$

With the assumption that a 3D finite element code enabled for $H(\mathrm{curl})$ shape functions has the $\widehat{\mathrm{curl}}$ operation incorporated, the conventional algorithm for (2.42) is presented as Algorithm 8.

Now we proceed to define a tensor-product shape function for this energy space. As

$$\hat{\psi}_I, \hat{\psi}_I \in \hat{Q}^p = \mathcal{Q}^{p_1-1,p_2,p_3}(\mathcal{I}^3) \times \mathcal{Q}^{p_1,p_2-1,p_3}(\mathcal{I}^3) \times \mathcal{Q}^{p_1,p_2,p_3-1}(\mathcal{I}^3),$$

it follows that

$$
\begin{aligned}
\hat{\psi}_I(\xi_1, \xi_2, \xi_3) &= \rho_{a,1;i_1}(\xi_1) \rho_{a,2;i_2}(\xi_2) \rho_{a,3;i_3}(\xi_3) \hat{\mathbf{e}}_a, \\
\hat{\psi}_J(\xi_1, \xi_2, \xi_3) &= \rho_{b,1;j_1}(\xi_1) \rho_{b,2;j_2}(\xi_2) \rho_{b,3;j_3}(\xi_3) \hat{\mathbf{e}}_b,
\end{aligned}
$$

where $\hat{\mathbf{e}}_a$, $a = 1, 2, 3$, is the a-th canonical Cartesian unit vector in the master space; the univariate polynomials $\rho_{a,d;i_d}(\xi_d)$, $d = 1, 2, 3$, are determined through the rule

$$\rho_{a,d;i_d} = \begin{cases} \nu_{i_d} & \text{if } a = d, \\ \chi_{i_d} & \text{otherwise.} \end{cases} \qquad (2.43)$$

Also here the values that the integer indices i_a, j_a may take depend on the vector component where $\hat{\psi}_I$ lies (a). The correspondence formula between the tensor-product index I and the one-dimensional ones could be of the form

$$I = \begin{cases} i_1 + p_1 i_2 + p_1(p_2+1) i_3 & \text{if } a = 1, \\ p_1(p_2+1)(p_3+1) + i_1 + (p_1+1) i_2 + (p_1+1) p_2 i_3 & \text{if } a = 2, \\ p_1(p_2+1)(p_3+1) + (p_1+1) p_2 (p_3+1) & \\ \qquad + i_1 + (p_1+1) i_2 + (p_1+1)(p_2+1) i_3 & \text{if } a = 3. \end{cases} \qquad (2.44)$$

procedure HCURLGRAM(i_{el}, G^{curl}) ▷ Compute matrix G^{curl} for element No. i_{el} – Conventional algorithm
  call setquadrature3D($i_{el}, p_1, p_2, p_3; L, \{\boldsymbol{\xi}_{lmn}, w_{lmn}\}$)
  $\mathsf{G}^{\text{curl}} \leftarrow 0$ ▷ Initialize Gram matrix
  for $l, m, n = 1$ to L **do**
    call Shape3Hcurl($\boldsymbol{\xi}_{lmn}, p_1, p_2, p_3; \{\hat{\psi}_I(\boldsymbol{\xi}_{lmn})\}, \{\widehat{\text{curl}}\,\hat{\psi}_I(\boldsymbol{\xi}_{lmn})\}$) ▷ 3D shape functions at $\boldsymbol{\xi}_{lmn}$
    call geometry($\boldsymbol{\xi}_{lmn}, i_{el}; \mathbf{x}, \mathcal{J}(\boldsymbol{\xi}_{lmn}), \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn}), |\mathcal{J}|$) ▷ Compute $\mathbf{x}$ and Jacobian
    $\mathcal{D} \leftarrow \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn})\mathcal{J}^{-T}(\boldsymbol{\xi}_{lmn})$
    $\mathcal{C} \leftarrow \mathcal{J}^{\mathsf{T}}(\boldsymbol{\xi}_{lmn})\mathcal{J}(\boldsymbol{\xi}_{lmn})$
    for $J = 0$ to $\dim Q^p - 1$ **do**
      for $I = J$ to $\dim Q^p - 1$ **do**
        $\mathsf{G}^{\text{curl}}_{IJ} \leftarrow \mathsf{G}^{\text{curl}}_{IJ} + \{\hat{\psi}_I(\boldsymbol{\xi}_{lmn})^{\mathsf{T}}\mathcal{D}\hat{\psi}_J(\boldsymbol{\xi}_{lmn})|\mathcal{J}| + [\widehat{\text{curl}}\,\hat{\psi}_I(\boldsymbol{\xi}_{lmn})]^{\mathsf{T}}\mathcal{C}[\widehat{\text{curl}}\,\hat{\vartheta}_J(\boldsymbol{\xi}_{lmn})]|\mathcal{J}|^{-1}\}w_{lmn}$
  return G^{curl}

Algorithm 8: Conventional computation of the $H(\text{curl})$ Gram matrix.

Perhaps it becomes clear by rewriting $\hat{\psi}_I$ using (2.43) just like it was done in the $H(\text{div})$ problem,

$$\begin{aligned}\hat{\psi}_I(\xi_1,\xi_2,\xi_3) &= \nu_{i_a}(\xi_a)\chi_{i_{a+1}}(\xi_{a+1})\chi_{i_{a+2}}(\xi_{a+2})\hat{\mathbf{e}}_a\\ &= \chi'_{i_a+1}(\xi_a)\chi_{i_{a+1}}(\xi_{a+1})\chi_{i_{a+2}}(\xi_{a+2})\hat{\mathbf{e}}_a,\end{aligned} \tag{2.45}$$

where the relation between ν and χ, (2.33), was applied. Moreover, the curl operator is the most complicated case of all in this section. Firstly, instead of using explicit indices 1, 2, 3, we can write the result of this operator having as a reference the component index a in $\hat{\psi}_I$ and making use of the cyclic permutation concept shown above.

$$\begin{aligned}\widehat{\text{curl}}\,\hat{\psi}_I &= [\partial_{a+1}(\hat{\psi}_I)_{a+2} - \partial_{a+2}(\hat{\psi}_I)_{a+1}]\hat{\mathbf{e}}_a\\ &\quad + [\partial_{a+2}(\hat{\psi}_I)_a - \partial_a(\hat{\psi}_I)_{a+2}]\hat{\mathbf{e}}_{a+1}\\ &\quad + [\partial_a(\hat{\psi}_I)_{a+1} - \partial_{a+1}(\hat{\psi}_I)_a]\hat{\mathbf{e}}_{a+2}\\ &= \partial_{a+2}(\hat{\psi}_I)_a\hat{\mathbf{e}}_{a+1} - \partial_{a+1}(\hat{\psi}_I)_a\hat{\mathbf{e}}_{a+2}\\ &= \chi'_{i_a+1}\chi_{i_{a+1}}\chi'_{i_{a+2}}\hat{\mathbf{e}}_{a+1} - \chi'_{i_a+1}\chi'_{i_{a+1}}\chi_{i_{a+2}}\hat{\mathbf{e}}_{a+2}.\end{aligned} \tag{2.46}$$

Now, using the Kronecker delta as a means to manage the superscript (2.22) and the subindex where we need $i_a + 1$ instead of just i_a, and introducing this into (2.45), (2.46), we can write $\hat{\psi}_I$ and its curl in an appropriate form to perform the nesting of integrals of (2.42).

$$\hat{\psi}_I(\xi_1,\xi_2,\xi_3) = \chi^{\langle\delta_{1a}\rangle}_{i_1+\delta_{1a}}(\xi_1)\chi^{\langle\delta_{2a}\rangle}_{i_2+\delta_{2a}}(\xi_2)\chi^{\langle\delta_{3a}\rangle}_{i_3+\delta_{3a}}(\xi_3)\hat{\mathbf{e}}_a, \tag{2.47}$$

$$\begin{aligned}\widehat{\text{curl}}\,\hat{\psi}_I(\xi_1,\xi_2,\xi_3) &= \chi^{\langle\gamma_{1(a+1)}\rangle}_{i_1+\delta_{1a}}(\xi_1)\chi^{\langle\gamma_{2(a+1)}\rangle}_{i_2+\delta_{2a}}(\xi_2)\chi^{\langle\gamma_{3(a+1)}\rangle}_{i_3+\delta_{3a}}(\xi_3)\hat{\mathbf{e}}_{a+1}\\ &\quad - \chi^{\langle\gamma_{1(a+2)}\rangle}_{i_1+\delta_{1a}}(\xi_1)\chi^{\langle\gamma_{2(a+2)}\rangle}_{i_2+\delta_{2a}}(\xi_2)\chi^{\langle\gamma_{3(a+2)}\rangle}_{i_3+\delta_{3a}}(\xi_3)\hat{\mathbf{e}}_{a+2}.\end{aligned} \tag{2.48}$$

By applying (2.47) and (2.48) to the first and second terms of (2.42), respectively, we converge to a definite expression that we can finally implement for the computation of the Gram matrix entry $\mathsf{G}^{\text{curl}}_{IJ}$. Putting aside momentarily the $|\mathcal{J}|$ factors, we have the following equations for each term:

$$\hat{\psi}_I(\boldsymbol{\xi})^{\mathsf{T}}\mathcal{D}(\boldsymbol{\xi})\hat{\psi}_J(\boldsymbol{\xi}) = \chi^{\langle\delta_{1a}\rangle}_{i_1+\delta_{1a}}(\xi_1)\chi^{\langle\delta_{1b}\rangle}_{j_1+\delta_{1b}}(\xi_1)\chi^{\langle\delta_{2a}\rangle}_{i_2+\delta_{2a}}(\xi_2)\chi^{\langle\delta_{2b}\rangle}_{j_2+\delta_{2b}}(\xi_2)\chi^{\langle\delta_{3a}\rangle}_{i_3+\delta_{3a}}(\xi_3)\chi^{\langle\delta_{3b}\rangle}_{j_3+\delta_{3b}}(\xi_3)\hat{\mathbf{e}}_a^{\mathsf{T}}\mathcal{D}(\boldsymbol{\xi})\hat{\mathbf{e}}_b, \tag{2.49}$$

$$\begin{aligned}[\widehat{\text{curl}}\,\hat{\psi}_I]^{\mathsf{T}}\mathcal{C}[\widehat{\text{curl}}\,\hat{\psi}_J] &= \chi^{\langle\gamma_{1(a+1)}\rangle}_{i_1+\delta_{1a}}\chi^{\langle\gamma_{1(b+1)}\rangle}_{j_1+\delta_{1b}}\chi^{\langle\gamma_{2(a+1)}\rangle}_{i_2+\delta_{2a}}\chi^{\langle\gamma_{2(b+1)}\rangle}_{j_2+\delta_{2b}}\chi^{\langle\gamma_{3(a+1)}\rangle}_{i_3+\delta_{3a}}\chi^{\langle\gamma_{3(b+1)}\rangle}_{j_3+\delta_{3b}}\hat{\mathbf{e}}^{\mathsf{T}}_{a+1}\mathcal{C}\hat{\mathbf{e}}_{b+1}\\ &\quad - \chi^{\langle\gamma_{1(a+2)}\rangle}_{i_1+\delta_{1a}}\chi^{\langle\gamma_{1(b+1)}\rangle}_{j_1+\delta_{1b}}\chi^{\langle\gamma_{2(a+2)}\rangle}_{i_2+\delta_{2a}}\chi^{\langle\gamma_{2(b+1)}\rangle}_{j_2+\delta_{2b}}\chi^{\langle\gamma_{3(a+2)}\rangle}_{i_3+\delta_{3a}}\chi^{\langle\gamma_{3(b+1)}\rangle}_{j_3+\delta_{3b}}\hat{\mathbf{e}}^{\mathsf{T}}_{a+2}\mathcal{C}\hat{\mathbf{e}}_{b+1}\\ &\quad - \chi^{\langle\gamma_{1(a+1)}\rangle}_{i_1+\delta_{1a}}\chi^{\langle\gamma_{1(b+2)}\rangle}_{j_1+\delta_{1b}}\chi^{\langle\gamma_{2(a+1)}\rangle}_{i_2+\delta_{2a}}\chi^{\langle\gamma_{2(b+2)}\rangle}_{j_2+\delta_{2b}}\chi^{\langle\gamma_{3(a+1)}\rangle}_{i_3+\delta_{3a}}\chi^{\langle\gamma_{3(b+2)}\rangle}_{j_3+\delta_{3b}}\hat{\mathbf{e}}^{\mathsf{T}}_{a+1}\mathcal{C}\hat{\mathbf{e}}_{b+2}\\ &\quad + \chi^{\langle\gamma_{1(a+2)}\rangle}_{i_1+\delta_{1a}}\chi^{\langle\gamma_{1(b+2)}\rangle}_{j_1+\delta_{1b}}\chi^{\langle\gamma_{2(a+2)}\rangle}_{i_2+\delta_{2a}}\chi^{\langle\gamma_{2(b+2)}\rangle}_{j_2+\delta_{2b}}\chi^{\langle\gamma_{3(a+2)}\rangle}_{i_3+\delta_{3a}}\chi^{\langle\gamma_{3(b+2)}\rangle}_{j_3+\delta_{3b}}\hat{\mathbf{e}}^{\mathsf{T}}_{a+2}\mathcal{C}\hat{\mathbf{e}}_{b+2}.\end{aligned}$$

Note that, in the second equation, the arguments were omitted for practical reasons, but it is easy to determine what is the argument of each univariate polynomial by comparing to (2.49). We can further recall that $\hat{\mathbf{e}}_a^{\mathsf{T}}\mathcal{D}(\boldsymbol{\xi})\hat{\mathbf{e}}_b = D_{ab}$, $\hat{\mathbf{e}}_{a+1}^{\mathsf{T}}\mathcal{C}\hat{\mathbf{e}}_{b+1} = C_{(a+1)(b+1)}$, and similarly we can retrieve $C_{(a+2)(b+1)}$, $C_{(a+1)(b+2)}$, $C_{(a+2)(b+2)}$. The second term of the Gram matrix is more intricate as it comprises four subterms, each of which needs to be computed separately. However, it is possible to generalize the definition of the auxiliary functions adding a new pair of indices. Then let

$$\mathcal{G}^{\operatorname{curl} A;\alpha\beta}_{ab;i_3j_3}(\xi_1,\xi_2) = \int_0^1 \chi^{\langle\gamma_{3(a+\alpha)}\rangle}_{i_3+\delta_{3a}}(\xi_3)\chi^{\langle\gamma_{3(b+\beta)}\rangle}_{j_3+\delta_{3b}}(\xi_3)C_{(a+\alpha)(b+\beta)}(\xi_1,\xi_2,\xi_3)|\mathcal{J}(\xi_1,\xi_2,\xi_3)|^{-1}\,d\xi_3, \tag{2.50}$$

$$\mathcal{G}^{\operatorname{curl} B;\alpha\beta}_{ab;i_2j_2i_3j_3}(\xi_1) = \int_0^1 \chi^{\langle\gamma_{2(a+\alpha)}\rangle}_{i_2+\delta_{2a}}(\xi_2)\chi^{\langle\gamma_{2(b+\beta)}\rangle}_{j_2+\delta_{2b}}(\xi_2)\mathcal{G}^{\operatorname{curl} A;\alpha\beta}_{ab;i_3j_3}(\xi_1,\xi_2)\,d\xi_2, \tag{2.51}$$

$$\mathcal{G}^{\operatorname{curl};\alpha\beta}_{ab;i_1j_1i_2j_2i_3j_3} = \int_0^1 \chi^{\langle\gamma_{1(a+\alpha)}\rangle}_{i_1+\delta_{1a}}(\xi_1)\chi^{\langle\gamma_{1(b+\beta)}\rangle}_{j_1+\delta_{1b}}(\xi_1)\mathcal{G}^{\operatorname{curl} B;\alpha\beta}_{ab;i_2j_2i_3j_3}(\xi_1)\,d\xi_1, \tag{2.52}$$

where $\alpha, \beta = 1, 2$. In the same fashion, for the first term, the auxiliary functions are

$$\check{\mathcal{G}}^{\operatorname{curl} A}_{ab;i_3j_3}(\xi_1,\xi_2) = \int_0^1 \chi^{\langle\delta_{3a}\rangle}_{i_3+\delta_{3a}}(\xi_3)\chi^{\langle\delta_{3b}\rangle}_{j_3+\delta_{3b}}(\xi_3)D_{ab}(\xi_1,\xi_2,\xi_3)\,|\mathcal{J}(\xi_1,\xi_2,\xi_3)|\,d\xi_3, \tag{2.53}$$

$$\check{\mathcal{G}}^{\operatorname{curl} B}_{ab;i_2j_2i_3j_3}(\xi_1) = \int_0^1 \chi^{\langle\delta_{2a}\rangle}_{i_2+\delta_{2a}}(\xi_2)\chi^{\langle\delta_{2b}\rangle}_{j_2+\delta_{2b}}(\xi_2)\check{\mathcal{G}}^{\operatorname{curl} A}_{ab;i_3j_3}(\xi_1,\xi_2)\,d\xi_2, \tag{2.54}$$

$$\check{\mathcal{G}}^{\operatorname{curl}}_{ab;i_1j_1i_2j_2i_3j_3} = \int_0^1 \chi^{\langle\delta_{1a}\rangle}_{i_1+\delta_{1a}}(\xi_1)\chi^{\langle\delta_{1b}\rangle}_{j_1+\delta_{1b}}(\xi_1)\check{\mathcal{G}}^{\operatorname{curl} B}_{ab;i_2j_2i_3j_3}(\xi_1)\,d\xi_1. \tag{2.55}$$

Finally, every Gram matrix entry is computed as

$$\mathsf{G}^{\operatorname{curl}}_{IJ} = \check{\mathcal{G}}^{\operatorname{curl}}_{ab;i_1j_1i_2j_2i_3j_3} + \sum_{\alpha,\beta=1}^{2}(-1)^{\alpha+\beta}\mathcal{G}^{\operatorname{curl};\alpha\beta}_{ab;i_1j_1i_2j_2i_3j_3}. \tag{2.56}$$

Algorithm 9 implements (2.50)–(2.56) using one-dimensional numerical integration for each auxiliary function. If we have a uniform-degree polynomial space (in the sense of the exact sequence), then this algorithm has a complexity of $\mathcal{O}[p^3(p+1)^4] \sim \mathcal{O}(p^7)$, whereas Algorithm 8 accounts for $\mathcal{O}[p^5(p+1)^4] \sim \mathcal{O}(p^9)$, assuming we use $L = p$ quadrature points.

2.7 Use of Legendre Polynomials

Legendre polynomials are a family of hierarchical and L^2 orthogonal polynomials defined over the interval $[-1, 1]$ with the average zero property (except the constant function corresponding to 0-th degree). By shifting the polynomial argument, we can keep all those properties over the interval $[0, 1]$. We are denoting by P_i the i-th-degree Legendre polynomial with $i = 0, 1, 2, \ldots$. Just by defining the first two members of the set along with a recursion formula, we can obtain all of the Legendre polynomial family over $[0, 1]$.

$$\begin{aligned} P_0(\xi) &= 1,\\ P_1(\xi) &= 2\xi - 1,\\ iP_i(\xi) &= (2i-1)(2\xi-1)P_{i-1}(\xi) - (i-1)P_{i-2}(\xi), \quad i \geq 2. \end{aligned} \tag{2.57}$$

In (2.57), the recursion formula is built such that $P_i(1) = 1$ for every non-negative i.

procedure HCURLGRAMTENSOR(i_{el}, $\mathsf{G}^{\mathrm{curl}}$) ▷ $\mathsf{G}^{\mathrm{curl}}$ for element No. i_{el} - Sum factorization

- $p_{\max} \leftarrow \max\{p_1, p_2, p_3\}$
- **call** setquadrature1D(i_{el}, $p_{\max}$; L, $\{\zeta^l, w^l\}$)
- $\breve{\mathcal{G}}^{\mathrm{curl}}$, $\mathcal{G}^{\mathrm{curl}} \leftarrow 0$ ▷ Initialize Gram matrix
- **for** $l = 1$ to L **do**
  - **call** Shape1H1(ζ^l, p_1; $\{\chi_{i_1}(\zeta^l), \chi'_{i_1}(\zeta^l)\}$) ▷ Evaluate 1D shape functions at ζ^l
  - **for** $j_3, i_3 = 0$ to p_3 **do**
    - $\breve{\mathcal{G}}^{\mathrm{curl}\,B}$, $\mathcal{G}^{\mathrm{curl}\,B} \leftarrow 0$
    - **for** $m = 1$ to L **do**
      - **call** Shape1H1(ζ^m, p_2; $\{\chi_{i_2}(\zeta^m), \chi'_{i_2}(\zeta^m)\}$) ▷ Evaluate 1D shape functions at ζ^m
      - $\breve{\mathcal{G}}^{\mathrm{curl}\,A}$, $\mathcal{G}^{\mathrm{curl}\,A} \leftarrow 0$
      - **for** $n = 1$ to L **do**
        - **call** Shape1H1(ζ^n, p_3; $\{\chi_{i_3}(\zeta^n), \chi'_{i_3}(\zeta^n)\}$) ▷ Evaluate 1D shape functions at ζ^n
        - $\boldsymbol{\xi}_{lmn} \leftarrow (\zeta^l, \zeta^m, \zeta^n)$
        - **call** geometry($\boldsymbol{\xi}_{lmn}$, i_{el}; $\mathbf{x}$, $\mathcal{J}(\boldsymbol{\xi}_{lmn})$, $\mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn})$, $|\mathcal{J}|$) ▷ Compute $\mathbf{x}$ and Jacobian
        - $\mathcal{D} \leftarrow \mathcal{J}^{-1}(\boldsymbol{\xi}_{lmn})\mathcal{J}^{-T}(\boldsymbol{\xi}_{lmn})$
        - $\mathcal{C} \leftarrow \mathcal{J}^{\mathrm{T}}(\boldsymbol{\xi}_{lmn})\mathcal{J}(\boldsymbol{\xi}_{lmn})$
        - **for** $a, b = 1$ to 3 **do**
          - **if** $j_3 + \delta_{3b} \le p_3$ and $i_3 + \delta_{3a} \le p_3$ **then** ▷ Avoids extra computations
            - $\breve{\mathcal{G}}^{\mathrm{curl}\,A}_{ab;i_3j_3} \leftarrow \breve{\mathcal{G}}^{\mathrm{curl}\,A}_{ab;i_3j_3} + \chi^{\langle\delta_{3a}\rangle}_{i_3+\delta_{3a}}(\zeta^n)\chi^{\langle\delta_{3b}\rangle}_{j_3+\delta_{3b}}(\zeta^n)D_{ab}|\mathcal{J}|w^n$
            - **for** $\alpha, \beta = 1$ to 2 **do**
              - $\mathcal{G}^{\mathrm{curl}\,A;\alpha\beta}_{ab;i_3j_3} \leftarrow \mathcal{G}^{\mathrm{curl}\,A;\alpha\beta}_{ab;i_3j_3} + \chi^{\langle\gamma_{3a}\rangle}_{i_3+\delta_{3a}}(\zeta^n)\chi^{\langle\gamma_{3b}\rangle}_{j_3+\delta_{3b}}(\zeta^n)C_{(a+\alpha)(b+\beta)}|\mathcal{J}|^{-1}w^n$ ▷ (2.50)
      - **for** $j_2, i_2 = 0$ to p_2 **do**
        - **for** $a, b = 1$ to 3 **do**
          - **if** $j_2 + \delta_{2b} \le p_2$ and $i_2 + \delta_{2a} \le p_2$ **then** ▷ Avoids extra computations
            - $\breve{\mathcal{G}}^{\mathrm{curl}\,B}_{ab;i_2j_2i_3j_3} \leftarrow \breve{\mathcal{G}}^{\mathrm{curl}\,B}_{ab;i_2j_2i_3j_3} + \chi^{\langle\delta_{2a}\rangle}_{i_2+\delta_{2a}}(\zeta^m)\chi^{\langle\delta_{2b}\rangle}_{j_2+\delta_{2b}}(\zeta^m)\breve{\mathcal{G}}^{\mathrm{curl}\,A}_{ab;i_3j_3}w^m$
            - **for** $\alpha, \beta = 1$ to 2 **do**
              - $\mathcal{G}^{\mathrm{curl}\,B;\alpha\beta}_{ab;i_2j_2i_3j_3} \leftarrow \mathcal{G}^{\mathrm{curl}\,B;\alpha\beta}_{ab;i_2j_2i_3j_3} + \chi^{\langle\gamma_{2a}\rangle}_{i_2+\delta_{2a}}(\zeta^m)\chi^{\langle\gamma_{2b}\rangle}_{j_2+\delta_{2b}}(\zeta^m)\mathcal{G}^{\mathrm{curl}\,A;\alpha\beta}_{ab;i_3j_3}w^m$ ▷ (2.51)
    - **for** $j_2, i_2 = 0$ to p_2 **do**
      - **for** $j_1, i_1 = 0$ to p_1 **do**
        - Determine I, J through (2.44)
        - **if** $J \ge I$ **then**
          - **for** $a, b = 1$ to 3 **do**
            - **if** $j_1 + \delta_{1b} \le p_1$ and $i_1 + \delta_{1a} \le p_1$ **then** ▷ Avoids extra computations
              - $\breve{\mathcal{G}}^{\mathrm{curl}}_{ab;i_1j_1i_2j_2i_3j_3} \leftarrow \breve{\mathcal{G}}^{\mathrm{curl}}_{ab;i_1j_1i_2j_2i_3j_3} + \chi^{\langle\delta_{1a}\rangle}_{i_1+\delta_{1a}}(\zeta^l)\chi^{\langle\delta_{1b}\rangle}_{j_1+\delta_{1b}}(\zeta^l)\breve{\mathcal{G}}^{\mathrm{curl}\,B}_{ab;i_2j_2i_3j_3}w^l$
              - **for** $\alpha, \beta = 1$ to 2 **do**
                - $\mathcal{G}^{\mathrm{curl};\alpha\beta}_{ab;i_1j_1i_2j_2i_3j_3} \leftarrow \mathcal{G}^{\mathrm{curl};\alpha\beta}_{ab;i_1j_1i_2j_2i_3j_3} + \chi^{\langle\gamma_{1a}\rangle}_{i_1+\delta_{1a}}(\zeta^l)\chi^{\langle\gamma_{1b}\rangle}_{j_1+\delta_{1b}}(\zeta^l)\mathcal{G}^{\mathrm{curl}\,B;\alpha\beta}_{ab;i_2j_2i_3j_3}w^l$
- **return** $\mathsf{G}^{\mathrm{curl}}$

Algorithm 9: Computation of the $H(\mathrm{curl})$ Gram matrix by sum factorization.

Additionally, the mentioned properties of the Legendre polynomials are formally rewritten like this:

(i) hierarchical polynomial basis

$$\mathcal{P}^n([0,1]) = \mathrm{span}\{P_0, \ldots, P_n\}, \quad n = 0, 1, 2, 3, \ldots,$$

(ii) orthogonality

$$(P_i, P_j)_{L^2(\mathcal{J})} = \int_0^1 P_i(\xi)P_j(\xi)\,d\xi = \delta_{ij}\frac{1}{2i+1}, \quad i, j \ge 0,$$

(iii) average zero

$$\int_0^1 P_i(\xi)\, d\xi = 0, \quad i \geq 1. \tag{2.58}$$

However, we usually need to evaluate derivatives of the shape functions. In that sense, it would be preferable to have all the nice properties of Legendre polynomials in the derivatives of our shape functions. This motivates the introduction of the integrated Legendre polynomials, defined as

$$L_i(\xi) = \int_0^\xi P_{i-1}(t)\, dt = 0, \quad i \geq 1. \tag{2.59}$$

This implies

$$L_i'(\xi) = P_{i-1}(\xi), \quad i \geq 1. \tag{2.60}$$

Notice that (2.58) and (2.59) imply that $L_i(0) = L_i(1) = 0$ for all $i \geq 2$. This means that all higher-order integrated Legendre polynomials are bubbles (smooth functions vanishing at the boundary of its support). Now, with some elementary calculus and the definitions and properties above, we can find a corresponding recursion formula for this new family of polynomials,

$$\begin{aligned} L_1(\xi) &= \xi, \\ 2(2i-1)L_i(\xi) &= P_i(\xi) - P_{i-2}(\xi), \quad i \geq 2. \end{aligned}$$

Finally, we can also do a bit of notation abuse and add a new function to this family,

$$L_0(\xi) = 1 - \xi = 1 - L_1(\xi).$$

The definition of this additional function has the purpose of completing a full basis with the L_i polynomials that is also hierarchical (as long as $i \geq 1$), just like in the original Legendre family. Thus we have

$$\mathcal{P}^n([0,1]) = \operatorname{span}\{L_0, \ldots, L_n\}, \quad n = 1, 2, 3, \ldots.$$

For the 1D W_J^p space, take $\chi_i = L_i$, $i = 0, 1, \ldots, p$; therefore, we are satisfying the requirement (2.33) thanks to (2.60). In other words, if the 1D H^1 shape functions are integrated Legendre polynomials, we comply with the hierarchical basis property, making the algorithms developed above directly applicable. This polynomial family will therefore be the constructing block of every finite element space that we have presented above.

Remark 2.5 (Side note on conformity using integrated Legendre polynomials). Notice that, across two adjacent elements, the union of functions L_0 and L_1 would form what is known as a *hat* function. In an H_1-conforming discretization, this kind of function is in charge of enforcing continuity at vertices. On the other hand, the higher-order integrated Legendre polynomials (L_i, $i \geq 2$) are *bubbles*, that is, they vanish at endpoints. Consequently, the finite element solution value at vertices is fully determined by the coefficients for L_0 and L_1. Conformity in 1D is so guaranteed at the lowest order case and at any higher order. For the 3D case, by recalling the ideas in Section 2.2, it can be shown that, after constructing the 3D spaces with the 1D spaces in the way therein shown, conformity is also attained.

3 Applications and Results

As stated earlier, the main motivation for proposing these integration algorithms has been to use them within the computational implementation of DPG finite element methods, as they require the computation of the Gram matrix for a high-order discrete test space for each element. Next we summarize the principles of DPG to illustrate the benefit of applying a fast integration technique in its implementation.

The DPG methodology is a family of finite element techniques that are applicable with several variational formulations [6], a growing number of physical applications [8, 11, 13, 15, 16, 24, 25], and holds desirable mathematical properties like getting "automatic" stability in any norm [5], being able to conformingly handle general polygonal meshes [26] and having a built-in *a posteriori* error estimator [3, 9], among others. In general terms, we can describe any DPG method by referring to an abstract variational formulation like the following one: find $u \in \mathscr{U}$ such that

$$b(u, v) = \ell(v) \quad \text{for all } v \in \mathscr{V},$$

where $\mathscr{U}, \mathscr{V}$ are Hilbert spaces of measurable functions defined over a domain Ω, $b(\cdot, \cdot)\colon \mathscr{U} \times \mathscr{V} \to \mathbb{R}$ (or $\mathbb{C}$) is a continuous bilinear (or sesquilinear) form, and $\ell\colon \mathscr{V} \to \mathbb{R}$ (or $\mathbb{C}$) is a continuous linear (or antilinear) functional on $\mathscr{V}$. Suppose that $\mathcal{T}$ is an open partition of our domain Ω, which coincides with the geometry discretization (mesh) used in the finite element method context. Define the broken test space as $\mathscr{V}(\mathcal{T}) := \{f\colon \Omega \to \mathbb{R} \text{ (or } \mathbb{C}, \text{ or } \mathbb{R}^3, \dots) \text{ measurable, such that } f|_{\mathcal{K}} \in \mathscr{V}|_{\mathcal{K}},\ \mathcal{K} \in \mathcal{T}\}$. If testing with functions from the latter space, an additional set of unknowns is necessarily added. Let $\partial\mathcal{T}$ be the set of mesh interfaces, or mesh "skeleton"; let $\mathscr{W}(\partial\mathcal{T})$ be a space of functions defined on $\partial\mathcal{T}$. The resulting problem is: find $u \in \mathscr{U}$, $\widetilde{u} \in \mathscr{W}(\partial\mathcal{T})$ such that

$$b(u, v) + \widetilde{b}(\widetilde{u}, v) = \ell(v) \quad \text{for all } v \in \mathscr{V}(\mathcal{T}), \tag{3.1}$$

where $\widetilde{b}(\cdot, \cdot)$ is a new bilinear functional acting on elements of $\mathscr{W}(\partial\mathcal{T})$ paired with traces of $\mathscr{V}(\mathcal{T})$ on $\partial\mathcal{T}$. When going to the discrete (finite-dimensional) problem, we would like to find $u^h \in \mathscr{U}^h$, $\widetilde{u}^h \in \mathscr{W}^h(\partial\mathcal{T})$, and for the sake of stability, an appropriate finite-dimensional subspace of $\mathscr{V}(\mathcal{T})$ must be picked. DPG's theoretical derivation leads to an exact way of performing this, but it involves the inversion of the Riesz map of $\mathscr{V}(\mathcal{T})$, which in most cases is impossible. Then we limit ourselves to working with an enriched test space, satisfying $\dim \mathscr{V}^r(\mathcal{T}) > \dim \mathscr{U}^h + \dim \mathscr{W}^h(\partial\mathcal{T})$. When implementing this, a convenient way to enrich the test space is to use polynomial spaces of nominal order $p + \Delta p$, with certain increment $\Delta p > 0$ with respect to the nominal polynomial order $p > 0$ of the exact sequence associated to the discrete trial spaces (i.e., $\mathscr{U}^h$ and $\mathscr{W}^h(\partial\mathcal{T})$). Once this enriched test space is formed, the final numerical problem of DPG becomes: find $\psi \in \mathscr{V}^r(\mathcal{T})$, $u^h \in \mathscr{U}^h$, $\widetilde{u}^h \in \mathscr{W}^h(\partial\mathcal{T})$ such that

$$\begin{cases} (\psi, v)_{\mathscr{V}^r(\mathcal{T})} + b(u^h, v) + \widetilde{b}(\widetilde{u}^h, v) = \ell(v) & \text{for all } v \in \mathscr{V}^r(\mathcal{T}), \\ b(\delta u, \psi) = 0 & \text{for all } \delta u \in \mathscr{U}^h, \\ \widetilde{b}(\delta\widetilde{u}, \psi) = 0 & \text{for all } \delta\widetilde{u} \in \mathscr{W}^h(\partial\mathcal{T}), \end{cases} \tag{3.2}$$

where $(\cdot, \cdot)_{\mathscr{V}^r(\mathcal{T})}$ is the inner product of the test space. Let $\{u_j\}_{j=1}^{n}$ be a basis for the first component of the trial space with $n = \dim \mathscr{U}^h$; $\{\widetilde{u}_j\}_{j=n+1}^{n+\widetilde{n}}$ be a basis for the second part of the trial space with $\widetilde{n} = \dim \mathscr{W}^h(\partial\mathcal{T})$; and $\{\phi_i\}_{i=1}^{m}$ be a basis for $\mathscr{V}^r(\mathcal{T})$, where m is its space dimension. With these bases, we represent the unknowns as follows:

$$u^h = \sum_{j=1}^{n} u_j \mathsf{u}_j, \tag{3.3}$$

$$\widetilde{u}^h = \sum_{j=n+1}^{n+\widetilde{n}} \widetilde{u}_j \mathsf{w}_j, \tag{3.4}$$

$$\psi = \sum_{j=1}^{m} \phi_j \mathsf{s}_j.$$

Following this, we can arrive at the augmented linear system shown in the introduction of this paper. Here we restate it since the context may now demonstrate that (3.2) is equivalent to this system (in the real case): find $\mathsf{s} \in \mathbb{R}^m$, $\mathsf{u} \in \mathbb{R}^n$, $\mathsf{w} \in \mathbb{R}^{\widetilde{n}}$ such that

$$\begin{cases} \mathsf{G}\mathsf{s} + \mathsf{B}\mathsf{u} + \widetilde{\mathsf{B}}\mathsf{w} = \mathsf{l}, \\ \mathsf{B}^{\mathsf{T}}\mathsf{s} = 0, \\ \widetilde{\mathsf{B}}^{\mathsf{T}}\mathsf{s} = 0, \end{cases} \tag{3.5}$$

where $\mathsf{G} \in \mathbb{R}^{m \times m}$ is the Gram matrix of $\mathscr{V}^r(\mathcal{T})$, that is, $\mathsf{G}_{ik} = (\phi_i, \phi_k)_{\mathscr{V}^r(\mathcal{T})}$; $\mathsf{B} \in \mathbb{R}^{m \times n}$ is the enriched stiffness

matrix from the field unknowns, given by $\mathsf{B}_{ij} = b(u_j, \phi_i)$; $\widetilde{\mathsf{B}} \in \mathbb{R}^{m\times\tilde{n}}$ is the enriched stiffness matrix from the skeleton unknowns with $\widetilde{\mathsf{B}}_{ij} = \widetilde{b}(\widetilde{u}_j, \phi_i)$; and $\mathsf{l} \in \mathbb{R}^m$ is the enriched load vector, obtained through $\mathsf{l}_i = \ell(\phi_i)$.

In order to find the approximate solution u^h using (3.3), and similarly the solution on interfaces $\widetilde{u}^h$ through (3.4), it is necessary to statically condense s from (3.5) by bringing into consideration that the Gram matrix is invertible (it represents a discrete Riesz map, which is an isomorphism). Then the new discrete system (also repeating from the introduction) yields

$$\begin{pmatrix} \mathsf{B}^\mathsf{T}\mathsf{G}^{-1}\mathsf{B} & \mathsf{B}^\mathsf{T}\mathsf{G}^{-1}\widetilde{\mathsf{B}} \\ \widetilde{\mathsf{B}}^\mathsf{T}\mathsf{G}^{-1}\mathsf{B} & \widetilde{\mathsf{B}}^\mathsf{T}\mathsf{G}^{-1}\widetilde{\mathsf{B}} \end{pmatrix} \begin{pmatrix} \mathsf{u} \\ \mathsf{w} \end{pmatrix} = \begin{pmatrix} \mathsf{B}^\mathsf{T}\mathsf{G}^{-1}\mathsf{l} \\ \widetilde{\mathsf{B}}^\mathsf{T}\mathsf{G}^{-1}\mathsf{l} \end{pmatrix}. \tag{3.6}$$

An alternative way of solving the discrete DPG problem is to apply the so-called Discrete Least Squares framework [17], which has been devised to get, depending on the technique therein implemented, either a more efficient assembly process or a better conditioned linear system than when the full system (3.6) is constructed.

Even though there are not results for hexahedral elements, some stability analysis on triangles and tetrahedra have been made [4, 21]. Such studies showed that the higher the dimension of the enriched test space is with respect to the dimension of the trial spaces, the better is the possibility for the practical DPG method of having stability guaranteed. This implies that the Gram matrix in DPG is the largest array whose computation is required at every element of the partition; therefore a technique to accelerate its construction will favor the overall implementation of this finite element methodology. However, the fact of using a broken test space makes G block diagonal, with each block associated to one and only one element of the mesh. This means that G can be assembled and even "inverted" elementwise, and this is per se a big computational relief. We encourage the reader to refer to another publication on the matter to know more about the specifics of the DPG family of methods [7].

To show the effect in time savings that the proposed tensor-product-based integration generates, three different PDEs (along with their corresponding variational formulation) were picked, and we compared the time needed to calculate the Gram matrix using conventional and sum factorization algorithms. Moreover, we compared also the time elapsed for the formation of G, B and l altogether. For all the examples below, the physical domain was equal to the master hexahedron, and a single-element mesh was considered. The last statement does not mean that these algorithms have not been tested with multiple elements or a more challenging physical geometry. In [20], the ultraweak version of Maxwell's equations was implemented under adaptive refinements and a curvilinear geometry, in the modeling of fiber laser amplifiers. Due to the very costly computations at the local level, the sum factorization ideas derived in this work were incorporated into the code to allow for a finer mesh to be resolved in a reasonable time. With $p_0 = 5$, there was an observed speed-up of about 80 times in the computation of G, B and l on every element. In the cases to be described next, similar speed-ups are obtained (at least for G).

As a final but important remark, the matrices G, B and l obtained in each of the examples below are equal (up to machine precision) independently of the algorithm used, which verifies the proper implementation of the different sum factorization algorithms. In order to make such a verification step, the resulting values from the conventional integration algorithm were used as the reference. In the following examples, the ordering of the loops within the algorithms that was preferred is the alternative one presented in Algorithm 3.

3.1 Primal Formulation for the Poisson Problem

Poisson's equation is given by

$$-\operatorname{div}(k\nabla u) = r,$$

where k is a diffusivity coefficient and r is a source term. This PDE, in the context of DPG, will lead to the broken primal variational formulation, in the terms of (3.1),

$$\mathscr{U} = H^1(\Omega), \quad \mathscr{W}(\partial\mathcal{T}) = H^{-\frac{1}{2}}(\partial\mathcal{T}), \quad \mathscr{V} = H^1(\mathcal{T}),$$
$$b(u, v) = (k\nabla u, \nabla v)_{\mathcal{T}}, \quad \widetilde{b}(\widetilde{u}, v) = \langle \widetilde{u}, v\rangle_{\partial\mathcal{T}}, \quad \ell(v) = (r, v)_{\mathcal{T}},$$

p_0	Δp	p_r	Conventional	Sum factorization
1	1	2	1.76E−03	1.64E−03
1	2	3	3.14E−03	1.95E−03
2	2	4	1.20E−02	3.31E−03
3	2	5	4.31E−02	6.70E−03
4	2	6	0.163	1.49E−02
5	2	7	0.522	3.05E−02

Table 1: Average computation time (seconds) of G^{grad} in the Primal Poisson DPG implementation for different polynomial orders and two integration algorithms.

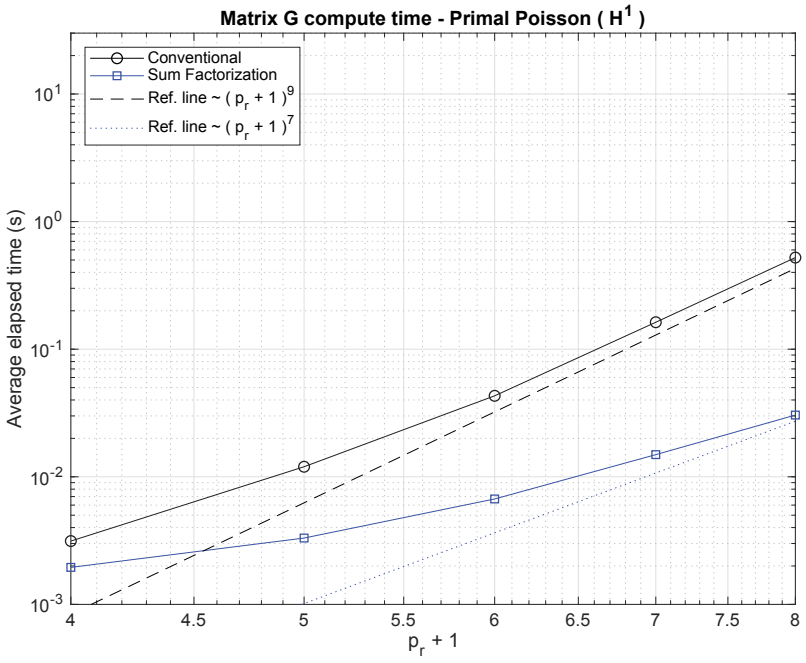

Figure 1: Average computation time (seconds) of G^{grad} in the Primal Poisson DPG implementation for different polynomial orders and two integration algorithms with respect to $p_r + 1$.

where $(\cdot,\cdot)_{\mathcal{T}} = \sum_{\mathcal{K}\in\mathcal{T}}(\cdot,\cdot)_{\mathcal{K}}$, $\langle\cdot,\cdot\rangle_{\partial\mathcal{T}}$ is a duality pairing between $H^{-\frac{1}{2}}(\partial\mathcal{T})$ and $H^{\frac{1}{2}}(\partial\mathcal{T})$. Recalling the Piola maps defined above, the main discrete element subspaces are $\mathscr{U}^h = W^{p_0} = T^{\text{grad}}\mathcal{Q}^{p_0,p_0,p_0}(\mathcal{J}^3)$ and $\mathscr{V}^r = W^{p_r} = T^{\text{grad}}\mathcal{Q}^{p_r,p_r,p_r}(\mathcal{J}^3)$, where $p_r = p_0 + \Delta p$. In order to compute the enriched stiffness matrix and the enriched load vector, a new portion of code must be added to Algorithms 4 and 5. The procedure carried out to compute these additional arrays so far is not optimal, so in a later work the performance of this task can be improved. Moreover, the computation of $\widetilde{\mathsf{B}}$ is done in the conventional way at all times; hence we leave it out of the computing time measurements.

Table 1 shows the average wall clock time of 50 runs of computing the H^1 Gram matrix for this problem, for different values of p_r. This plot is made with $p_r + 1$ as the abscissae because this is the parameter with respect to which the computational cost estimates were presented. The expected trend lines accompany the two data plots in Figure 1 as a reference to compare with. It is noticeable that, in the conventional case, this trend fits better to the data than in the new algorithm.

Additionally, Table 2 and Figure 2 show the average time for computing G, B and l within the code's element subroutine (the component of the finite element code that constructs the element matrices), having fixed $\Delta p = 2$. Since B and l are constructed using a conventional integration approach, this result intends to inform a closer estimate of the actual time savings when using sum factorization in a DPG computational solution. Notice that the reference line on this plot indicates that, in both cases, the trend is again approximately of ninth order.

p_0	Δp	p_r	Conventional	Sum factorization
1	2	3	3.55E–03	2.41E–03
2	2	4	1.60E–02	7.20E–03
3	2	5	6.62E–02	2.81E–02
4	2	6	0.294	0.113
5	2	7	1.01	0.404

Table 2: Average computation time (seconds) of G^{grad}, B, l in the Primal Poisson DPG implementation for different polynomial orders and two integration algorithms.

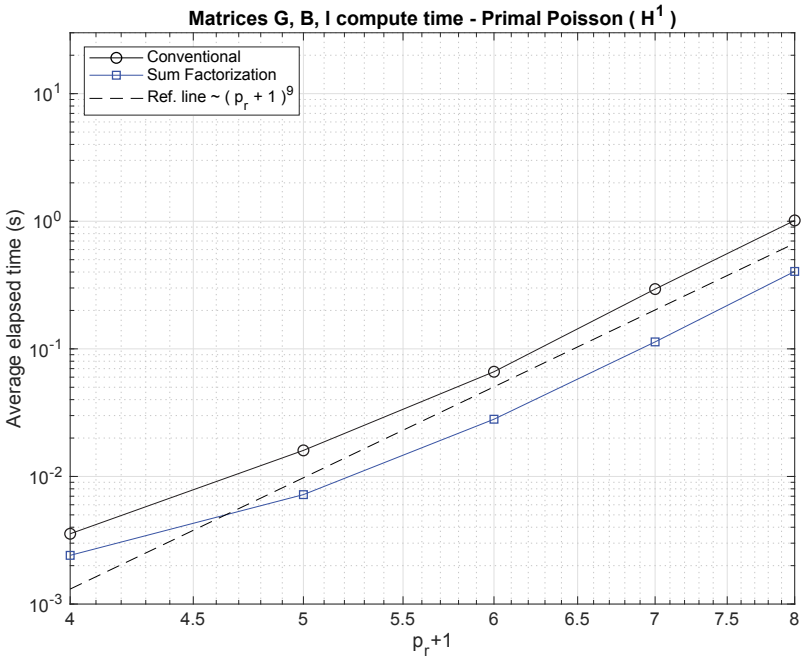

Figure 2: Average computation time (seconds) of G^{grad}, B, l in the Primal Poisson DPG implementation for different polynomial orders and two integration algorithms with respect to $p_r + 1$, with fixed $\Delta p = 2$.

3.2 Primal Formulation for Maxwell's Equations

A second illustrative problem is the time-harmonic Maxwell's wave equation

$$\operatorname{curl}(\mu^{-1}\operatorname{curl} E) - \omega^2 \epsilon E = \mathrm{i}\omega J, \tag{3.7}$$

where E is the unknown electric field, J is the imposed electric current, ω is the angular frequency, and μ and ϵ are electromagnetic constants (the permeability and the permittivity, respectively). In terms of the abstract formulation (3.1), the primal version of the broken variational form for (3.7) is

$$\mathscr{U} = H(\operatorname{curl}, \Omega), \quad \mathscr{W}(\partial\mathcal{T}) = H^{-\frac{1}{2}}(\operatorname{div}, \partial\mathcal{T}), \quad \mathscr{V} = H(\operatorname{curl}, \mathcal{T}),$$
$$b(E, F) = (\mu^{-1}\operatorname{curl} E, \operatorname{curl} F)_{\mathcal{T}} - \omega^2(\epsilon E, F)_{\mathcal{T}}, \quad \widetilde{b}(\widetilde{E}, F) = -\mathrm{i}\omega\langle \widetilde{E}, F\rangle_{\partial\mathcal{T}}, \quad \ell(F) = \mathrm{i}\omega(J, F)_{\mathcal{T}},$$

where E and $\widetilde{E}$ are the trial variables, F is the test variable, and $\langle\cdot,\cdot\rangle_{\partial\mathcal{T}}$ in this case represents the duality pairing between trace spaces $H^{-\frac{1}{2}}(\operatorname{div}, \partial\mathcal{T})$ and $H^{-\frac{1}{2}}(\operatorname{curl}, \partial\mathcal{T})$ (see [4]).

The discrete enriched test subspace is $Q^{p_r} = T^{\operatorname{curl}}(\mathbb{Q}^{p_r-1,p_r,p_r}(\mathcal{J}^3) \times \mathbb{Q}^{p_r,p_r-1,p_r}(\mathcal{J}^3) \times \mathbb{Q}^{p_r,p_r,p_r-1}(\mathcal{J}^3))$. The Gram matrix that corresponds to this problem is the one for $H(\operatorname{curl})$; hereby Algorithms 8 and 9 are going to be compared.

Similarly to the previous case, we present results for the construction of the Gram matrix in Table 3 and Figure 3. In this case, the time averages correspond to 20 runs. When taking into account also the integration of B and l, the results are those shown in Table 4 and in Figure 4.

p_0	Δp	p_r	Conventional	Sum factorization
1	1	2	3.09E−02	3.41E−02
1	2	3	4.22E−02	3.98E−02
2	2	4	0.129	5.39E−02
3	2	5	0.571	9.42E−02
4	2	6	2.30	0.203
5	2	7	7.82	0.464

Table 3: Average computation time (seconds) of G^{curl} in the Primal Maxwell DPG implementation for different polynomial orders and two integration algorithms.

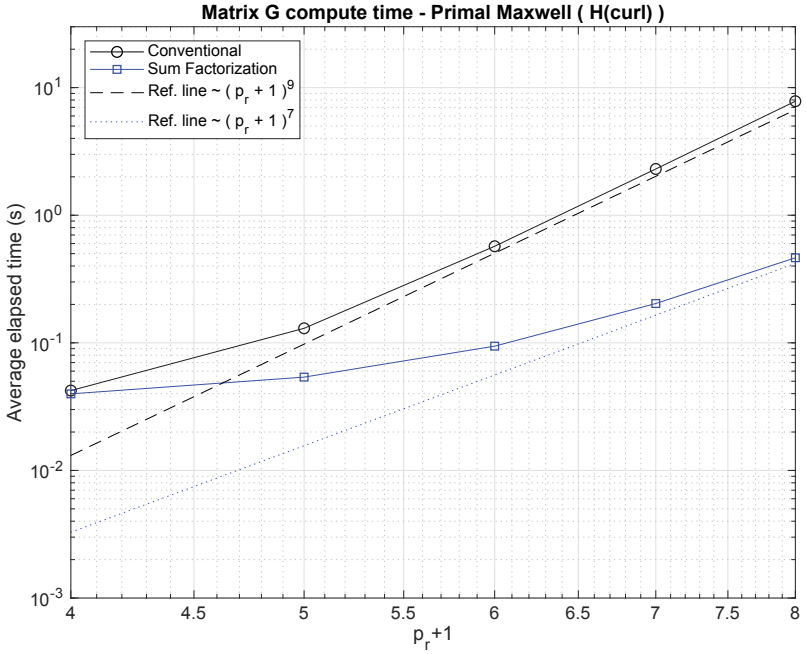

Figure 3: Average computation time (seconds) of G^{curl} in the Primal Maxwell DPG implementation for different polynomial orders and two integration algorithms with respect to $p_r + 1$.

p_0	Δp	p_r	Conventional	Sum factorization
1	2	3	4.65E−02	3.37E−02
2	2	4	0.167	7.75E−02
3	2	5	0.889	0.340
4	2	6	4.02	1.57
5	2	7	15.36	6.02

Table 4: Average computation time (seconds) of G^{curl}, B, l, in the Primal Maxwell DPG implementation for different polynomial orders and two integration algorithms.

3.3 Acoustics and the Ultraweak Variational Formulation

The time-harmonic acoustics system of equations reads as follows:

$$\begin{cases} \mathrm{i}\omega u + \nabla p = 0, \\ \mathrm{i}\omega p + \operatorname{div} u = f, \end{cases}$$

where u is the velocity, p the pressure, and ω the angular frequency. As a first order system, each equation can be tested independently (with broken test functions), and after applying integration by parts, we get the

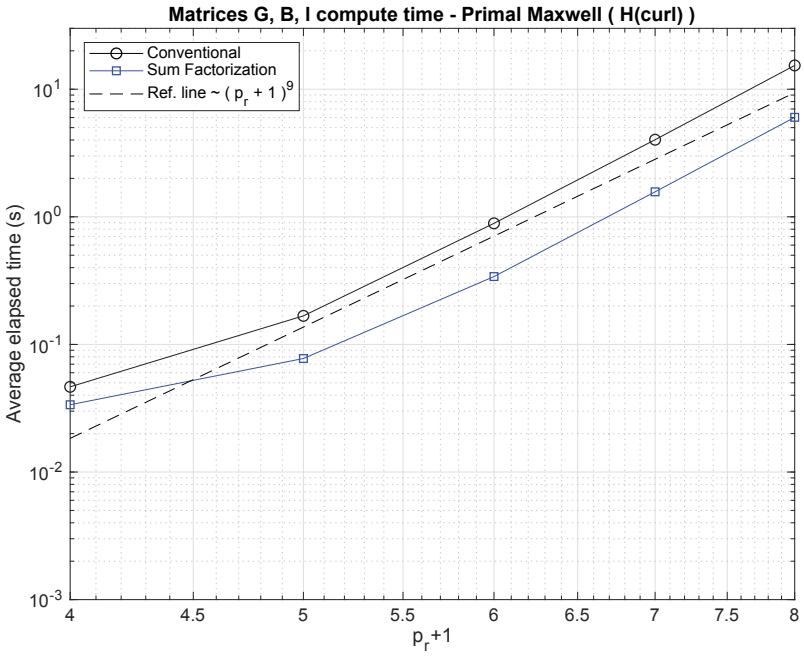

Figure 4: Average computation time (seconds) of $\mathsf{G}^{\mathrm{curl}}$, B, l, in the Primal Maxwell DPG implementation for different polynomial orders and two integration algorithms with respect to $p_r + 1$, with fixed $\Delta p = 2$.

so-called ultraweak variational formulation [24]

$$\begin{aligned}
\mathscr{U} &= L^2(\Omega) \times \boldsymbol{L}^2(\Omega),\\
\mathscr{W}(\partial\mathcal{T}) &= H^{\frac{1}{2}}(\mathrm{div}, \partial\mathcal{T}) \times H^{-\frac{1}{2}}(\mathrm{div}, \partial\mathcal{T}),\\
\mathscr{V} &= H^1(\mathcal{T}) \times H(\mathrm{div}, \mathcal{T}),\\
b((p,u),(q,v)) &= (\mathrm{i}\omega u, v)_{\mathcal{T}} - (p, \mathrm{div}\, v)_{\mathcal{T}} + (\mathrm{i}\omega p, q)_{\mathcal{T}} - (u, \nabla q)_{\mathcal{T}},\\
\widetilde{b}((\widetilde{p},\widetilde{u}_n),(q,v)) &= \langle \widetilde{p}, v\cdot n\rangle_{\partial\mathcal{T}} + \langle \widetilde{u}_n, q\rangle_{\partial\mathcal{T}},\\
\ell((q,v)) &= (f,v)_{\mathcal{T}}.
\end{aligned} \tag{3.8}$$

Interestingly, this variational formulation is characterized by having its field trial variables lying in L^2 spaces. This immediately implies that there is no requirement on continuity across elements for the functions belonging to those spaces. Other variational formulations, where the trial space needs some kind of continuity (e.g., the primal formulations above), and therefore the handling of orientation and the distinction among the type of shape function (vertex, edge, face, interior) make the tensor-product decomposition less direct (see [12]). Unlike those, the ultraweak variational formulation directly allows the correspondence of its discrete trial basis with tensor-product shape functions. As a result, we can apply very similar ideas as above to compute the volume integrals needed for B and l, in addition to G.

For a single element, notice that the stiffness matrix B arising from the bilinear functional in (3.8) can be computed as follows:

$$\mathsf{B}_{IK} = (\mathrm{i}\omega u_{K_R}, v_{I_V})_{\mathcal{K}} - (p_{K_Q}, \mathrm{div}\, v_{I_V})_{\mathcal{K}} + (\mathrm{i}\omega p_{K_Q}, q_{I_H})_{\mathcal{K}} - (u_{K_R}, \nabla q_{I_H})_{\mathcal{K}},$$

where I is the product index of I_H and I_V, K is the product index of K_Q and K_R, $p_{K_Q} \in Y^{p_0}$, $u_{K_R} \in (Y^{p_0})^3$, $q_{I_H} \in W^{p_r}$, and $v_{I_V} \in V^{p_r}$. By mapping to the master hexahedron and using the Piola transformations, we obtain

$$\mathsf{B}_{IK} = \int_{\hat{\mathcal{K}}} \left(\mathrm{i}\omega \hat{u}_{K_R}^{\mathsf{T}} \mathcal{J}\hat{v}_{I_V} |\mathcal{J}|^{-1} - \hat{p}_{K_Q}, \hat{\mathrm{div}}\hat{v}_{I_V} |\mathcal{J}|^{-1} + \mathrm{i}\omega \hat{p}_{K_Q}\hat{q}_{I_H} - \hat{u}_{K_R}^{\mathsf{T}} \mathcal{J}^{-\mathsf{T}} \hat{\nabla}\hat{q}_{I_H}\right) d^3\boldsymbol{\xi}. \tag{3.9}$$

For each term of (3.9), we can express the involved shape functions as tensor products of univariate polynomials and develop similar algorithms as the ones derived above. A similar idea can be applied for the simpler situation of l.

Additionally, the choice of a special norm in the ultraweak formulation is of great benefit to be able to control the solution error in the desired trial norm [6]. Such a norm is the so-called *adjoint graph norm* with a corresponding inner product that is defined as follows:

$$((\delta q, \delta v), (q, v))_{\mathcal{V}} = [\omega^2 + \alpha](\delta q, q)_{\mathcal{T}} + (\nabla \delta q, \nabla q)_{\mathcal{T}} - \mathrm{i}\omega(\operatorname{div} \delta v, q)_{\mathcal{T}} + \mathrm{i}\omega(\delta v, \nabla q)_{\mathcal{T}} + \mathrm{i}\omega(\delta q, \operatorname{div} v)_{\mathcal{T}} - \mathrm{i}\omega(\nabla \delta q, v)_{\mathcal{T}} + [\omega^2 + \alpha](\delta v, v)_{\mathcal{T}} + (\operatorname{div} \delta v, \operatorname{div} v)_{\mathcal{T}} \quad \text{for all } (\delta q, \delta v), (q, v) \in \mathcal{V}. \tag{3.10}$$

In (3.10), α is some positive real number that is used as a correction due to the fact of using broken test spaces [24]. Notice that four terms in this inner product were already studied in the H^1 and $H(\mathrm{div})$ spaces. The remaining four terms form a Hermitian array so that we need to implement only two of them. Considering a single element, the third and fourth terms of (3.10) lead to

$$-\mathrm{i}\omega(\operatorname{div} \delta v, q)_{\mathcal{K}} + \mathrm{i}\omega(\delta v, \nabla q)_{\mathcal{K}} = \mathrm{i}\omega \int_{\hat{\mathcal{K}}} [-(\hat{\operatorname{div}} \delta\hat{v})(\hat{q}) + (\delta\hat{v})^{\mathsf{T}}(\hat{\nabla}\hat{q})]\, d^3\boldsymbol{\xi} \tag{3.11}$$

after implementing the respective Piola maps. Finally, besides using Algorithms 5 and 7 for the four associated terms in (3.10), an analogous sum factorization algorithm must be built for the two integrals in (3.11).

This problem was solved using both conventional integration and tensor-product based integration, translating the ideas above for the stiffness matrix, the load vector and the Gram matrix produced by the adjoint graph norm, into the code. Table 5 and Figure 5 show the results of the average computing time (of

p_0	Δp	p_r	Conventional	Sum factorization
1	1	2	1.48E−03	7.15E−04
1	2	3	1.05E−02	3.40E−03
2	2	4	7.46E−02	9.62E−03
3	2	5	0.437	2.84E−02
4	2	6	1.900	8.08E−02
5	2	7	6.985	0.214
6	2	8	21.88	0.496

Table 5: Average computation time (seconds) of G, B and l in the ultraweak acoustics DPG implementation for different polynomial orders and two integration algorithms.

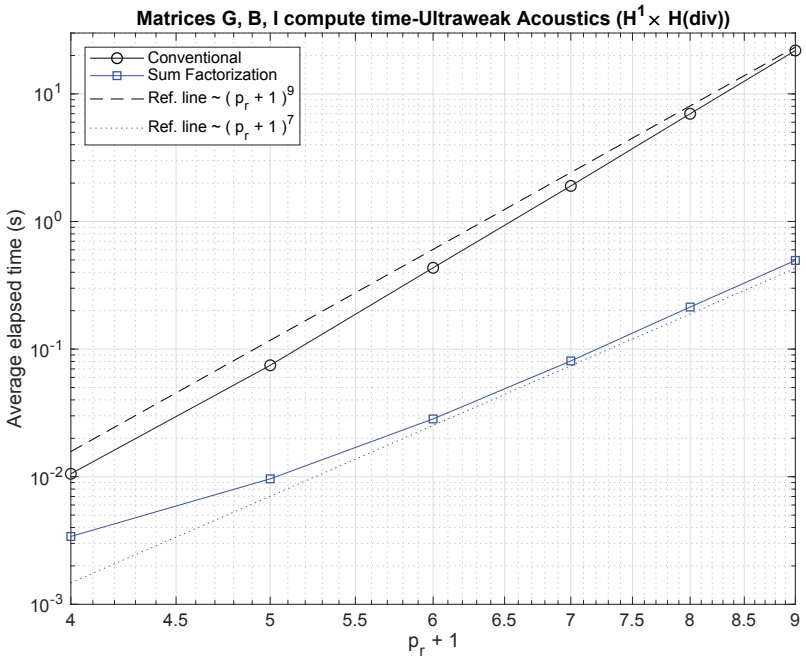

Figure 5: Average computation time (seconds) of G, B, l in the ultraweak acoustics DPG implementation for different polynomial orders and two integration algorithms with respect to $p_r + 1$, with fixed $\Delta p = 2$.

20 runs) of G, B and l for several polynomial degrees. Here, because of the different trends observed, two reference lines are presented, one of ninth order and one of seventh order.

3.4 Discussion

The results observed above allow to identify several achievements, while also making clear what aspects need improvement. Notice that all five plots have the same scales in their axes in order to facilitate visual comparison.

Firstly, Figures 1 and 3 communicate well that the expected asymptotic reference lines approach the numerical experiments results as p_r increases. Unlike the dashed reference line (corresponding to $(p_r + 1)^9$), which is almost parallel to the plot for conventional integration, the fine dotted line (corresponding to $(p_r + 1)^7$) seems to grow slightly faster than the data for the sum factorization results. It may be expected that at higher polynomial degrees the rate becomes closer to 7, but at the moment the important outcome is that the integration of G indeed became two orders of magnitude faster.

As a second observation, we see that Figures 2 and 4 reveal that, when looking at the elapsed time of computing all the matrices, the sum factorization approach overcomes the performance of the conventional algorithm at all degrees p_r. Such a result is also visible in Tables 2 and 4, in which the difference between columns 4 and 5 corresponds to the actual time saving when going from the conventional integration scheme to the herein introduced algorithm for every time the element subroutine is invoked in a DPG code. For the highest values of p_r, this saving can be of the order of a second per element in Poisson, or nine seconds in Maxwell, therefore leading to a big reduction of computation time when a refined mesh is in use. It can be seen nevertheless that the complexity rate for the compute time of all the matrices, in both types of integration, is still 9. This happens because computing the stiffness matrix B with the conventional algorithm has a leading term of $\mathcal{O}((p_0 + 1)^9)$ in its computational complexity, which will obviously begin dominating the cost of all the integration as p_0 grows.

Earning all the advantages of sum factorization for numerical integration has therefore the requirement of finding a good procedure for the construction of B and l. Such a procedure must be able to take profit of the nesting of integrals, as developed above for so diverse situations, for multiple variational formulations.

For the ultraweak formulation, that was clearly observed. In that case, there was a way to extend the ideas of sum factorization and take advantage of them, as noticed in Figure 5. There it can be seen that the complexity rates were successfully taken to the desired order.

As a closing remark, we can state that having proposed and utilized a tensor-product-based algorithm for the integration of less usual Gram matrices, like that of $H(\mathrm{curl})$ or the adjoint-graph inner product for $H^1 \times H(\mathrm{div})$, proved to be very useful regarding the really meaningful relative savings: 16 times faster when $p_0 = 5$ and $\Delta p = 2$, in Maxwell's primal form; or 44 times faster in the ultraweak formulation for acoustics with $p_0 = 6$ and $\Delta p = 2$. This kind of benefit is thereby expected to extend to the rest of energy spaces and inner products frequently used in DPG and other FE methods.

4 Conclusions

A complete set of algorithms for fast integration of Gram matrices for H^1, $H(\mathrm{curl})$, $H(\mathrm{div})$ and L^2 spaces for a general parametric hexahedral element has been presented. The algorithms are based on the sum factorization of tensor-product shape functions. Critical for efficiency is the use of hierarchical shape functions – Legendre polynomials and their integrals – for the 1D exact sequence. The expected reduction of computational complexity from $\mathcal{O}(p^9)$ to $\mathcal{O}(p^7)$ was verified numerically for the H^1, $H(\mathrm{div})$ and $H(\mathrm{curl})$ cases. Of special significance was the implementation of a problem with the ultraweak variational formulation, that allowed the full application of the tensor-product-based integration to both the stiffness matrix and load vector (Figure 5), unlike the other problems studied, for which only the Gram matrix could be treated under this

scheme (as observed in Figures 1, 3), which made the overall integration time improvement less substantial (Figures 2, 4).

Although in some cases, the time saving in obtaining the Gram matrix is as dramatic as going from 22 seconds to half a second (acoustics, $p_r = 8$), and this result should persuade some users to consider the implementation of this set of algorithms; some extra work on this idea seems relevant in order to consolidate the concept of fast integration in DPG thanks to a tensor-product-based integration. Such future work may include a specific way of dealing with the stiffness and load arrays when the trial shape functions require compatibility between adjacent elements (e.g., in unstructured meshes); extending these ideas to the computation of the element boundary contributions to the stiffness matrix and load vector; the application of these algorithms to more boundary-value problems that have been studied with DPG so that more test spaces, as well as more non-conventional norms, can be tried with this approach; and finally, propose the algorithms for other types of 2D and 3D elements.

Acknowledgment: The authors thank the anonymous reviewers for their valuable contributions to the quality of the paper.

Funding: Jaime Mora has been sponsored by a 2015 Colciencias-Fulbright scholarship, granted by the Government of Colombia and the Fulbright Commission-Colombia. Leszek Demkowicz has been supported by a grant from AFOSR (FA9550-12-1-0484).

References

[1] M. Ainsworth, G. Andriamaro and O. Davydov, Bernstein–Bézier finite elements of arbitrary order and optimal assembly procedures, *SIAM J. Sci. Comput.* **33** (2011), no. 6, 3087–3109.

[2] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli and G. Sangalli, Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization, *Comput. Methods Appl. Mech. Engrg.* **285** (2015), 817–828.

[3] C. Carstensen, L. Demkowicz and J. Gopalakrishnan, A posteriori error control for DPG methods, *SIAM J. Numer. Anal.* **52** (2014), no. 3, 1335–1353.

[4] C. Carstensen, L. Demkowicz and J. Gopalakrishnan, Breaking spaces and forms for the DPG method and applications including Maxwell equations, *Comput. Math. Appl.* **72** (2016), no. 3, 494–522.

[5] L. Demkowicz and J. Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. II. Optimal test functions, *Numer. Methods Partial Differential Equations* **27** (2011), no. 1, 70–105.

[6] L. Demkowicz and J. Gopalakrishnan, An overview of the discontinuous Petrov Galerkin method, in: *Recent Developments in Discontinuous Galerkin Finite Element Methods for Partial Differential Equations*, IMA Vol. Math. Appl. 157, Springer, Cham (2014), 149–180.

[7] L. Demkowicz and J. Gopalakrishnan, Discontinuous Petrov–Galerkin (DPG) method, ICES Report 15-20, The University of Texas at Austin, 2015.

[8] L. Demkowicz, J. Gopalakrishnan, S. Nagaraj and P. Sepúlveda, A spacetime DPG method for the Schrödinger equation, *SIAM J. Numer. Anal.* **55** (2017), no. 4, 1740–1759.

[9] L. Demkowicz, J. Gopalakrishnan and A. H. Niemi, A class of discontinuous Petrov–Galerkin methods. Part III: Adaptivity, *Appl. Numer. Math.* **62** (2012), no. 4, 396–427.

[10] L. Demkowicz, J. Kurtz, D. Pardo, M. Paszyński, W. Rachowicz and A. Zdunek, *Computing with hp Finite Elements. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications*, Chapman & Hall/CRC, New York, 2007.

[11] F. Fuentes, L. Demkowicz and A. Wilder, Using a DPG method to validate DMA experimental calibration of viscoelastic materials, *Comput. Methods Appl. Mech. Engrg.* **325** (2017), 748–765.

[12] F. Fuentes, B. Keith, L. Demkowicz and S. Nagaraj, Orientation embedded high order shape functions for the exact sequence elements of all shapes, *Comput. Math. Appl.* **70** (2015), no. 4, 353–458.

[13] F. Hellwig, *Three low-order dPG methods for linear elasticity*, Master's thesis, Humboldt-Universität zu Berlin, Berlin, 2014.

[14] G. E. Karniadakis and S. J. Sherwin, *Spectral/hp Element Methods for Computational Fluid Dynamics*, 2nd ed., Numer. Math. Sci. Comput., Oxford University, New York, 2005.

[15] B. Keith, F. Fuentes and L. Demkowicz, The DPG methodology applied to different variational formulations of linear elasticity, *Comput. Methods Appl. Mech. Engrg.* **309** (2016), 579–609.

[16] B. Keith, P. Knechtges, N. V. Roberts, S. Elgeti, M. Behr and L. Demkowicz, An ultraweak DPG method for viscoelastic fluids, *J. Non-Newton. Fluid Mech.* **247** (2017), 107–122.

[17] B. Keith, S. Petrides, F. Fuentes and L. Demkowicz, Discrete least-squares finite element methods, *Comput. Methods Appl. Mech. Engrg.* **327** (2017), 226–255.
[18] J. P. Kurtz, *Fully Automatic hp-Adaptivity for Acoustic and Electromagnetic Scattering in Three Dimensions*, ProQuest LLC, Ann Arbor, 2007.
[19] J. M. Melenk, K. Gerdes and C. Schwab, Fully discrete hp-finite elements: Fast quadrature, *Comput. Methods Appl. Math.* **190** (2001), no. 32, 4339–4364.
[20] S. Nagaraj, J. Grosek, S. Petrides, L. F. Demkowicz and J. Mora, A 3D DPG Maxwell approach to nonlinear Raman gain in fiber laser amplifiers, *J. Comput. Phys. X* **2** (2019), Article ID 100002.
[21] S. Nagaraj, S. Petrides and L. F. Demkowicz, Construction of DPG Fortin operators for second order problems, *Comput. Math. Appl.* **74** (2017), no. 8, 1964–1980.
[22] J.-C. Nédélec, Mixed finite elements in $\mathbf{R}^3$, *Numer. Math.* **35** (1980), no. 3, 315–341.
[23] S. A. Orszag, Spectral methods for problems in complex geometries, *J. Comput. Phys.* **37** (1980), no. 1, 70–92.
[24] S. Petrides and L. F. Demkowicz, An adaptive DPG method for high frequency time-harmonic wave propagation problems, *Comput. Math. Appl.* **74** (2017), no. 8, 1999–2017.
[25] N. V. Roberts, *A discontinuous Petrov–Galerkin methodology for incompressible flow problems*, PhD thesis, The University of Texas at Austin, Austin, 2013.
[26] A. Vaziri Astaneh, F. Fuentes, J. Mora and L. Demkowicz, High-order polygonal discontinuous Petrov–Galerkin (PolyDPG) methods using ultraweak formulations, *Comput. Methods Appl. Mech. Engrg.* **332** (2018), 686–711.