Efficient and Fast Auxiliary Splines Space Preconditioning B-Splines Mixed Finite Elements

A. El-Akri[1], Y. Guçlu[2], N. Hamid[1], I. Kissami[1], N. Ouhaddou[1], A. Ratnani[1]

[1]*MSDA, Mohammed VI Polytechnic University, Green City, Morocco*

[2]*Max-Planck-Institut fur Plasmaphysik, Garching, Germany*

May 29, 2022

Abstract

This paper presents a fast and efficient implementation for Auxiliary Space Preconditioning using B-Splines Finite Elements for $\boldsymbol{H}(\mathbf{curl}, \Omega)$ and $\boldsymbol{H}(\mathbf{div}, \Omega)$ elliptic problems. A parallelization using OpenMP is also presented. rewrite the abstract

TODO:

-
-
-
-

Contents

1 Introduction

TODO: rewrite the introduction

We consider the following variational problems, on a bounded domain $\Omega \subset \mathbb{R}^d$, with $d \in \{2,3\}$:

$$\boldsymbol{u} \in \boldsymbol{H}_0(\mathrm{curl},\Omega): \quad (\nabla\times\boldsymbol{u},\nabla\times\boldsymbol{v})_{\mathbb{L}^2(\Omega)} + \mu\,(\boldsymbol{u},\boldsymbol{v})_{\mathbb{L}^2(\Omega)} = (\boldsymbol{f},\boldsymbol{v})_{\mathbb{L}^2(\Omega)} \quad \forall \boldsymbol{v}\in \boldsymbol{H}_0(\mathrm{curl},\Omega) \tag{1a}$$

$$\boldsymbol{u} \in \boldsymbol{H}_0(\mathrm{div},\Omega): \quad (\nabla\cdot\boldsymbol{u},\nabla\cdot\boldsymbol{v})_{\mathbb{L}^2(\Omega)} + \mu\,(\boldsymbol{u},\boldsymbol{v})_{\mathbb{L}^2(\Omega)} = (\boldsymbol{f},\boldsymbol{v})_{\mathbb{L}^2(\Omega)} \quad \forall \boldsymbol{v}\in \boldsymbol{H}_0(\mathrm{div},\Omega) \tag{1b}$$

where $0 < \mu \ll 1$ and $\boldsymbol{f} \in \left(L^2(\Omega)\right)^3$.

2 Preliminaries

TODO: add missing parts

2.1 IsoGeometric Spaces

In this section we introduce a discrete counterparts of functional spaces $L^2(\Omega)$, $\boldsymbol{H}(\mathbf{curl},\Omega)$, $\boldsymbol{H}(\mathrm{div},\Omega)$, this is done in the context of IsoGeometric Analysis (IGA) [?, ?, ?, ?, ?].

Accordingly, we start by recalling some basic properties of B-spline functions. For a basic introduction to the subject the reader is referred to standard textbooks [?, ?, ?, ?, ?, ?, ?, ?]. Given a knot vector $T = (t_1, t_2, \ldots, t_m)$, namely a nondecreasing sequence of real numbers, the i-th B-spline of order $p \in \mathbb{N}$ is defined recursively by the following *Cox–de Boor formula*

$$B_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \le t < t_{i+1}, \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} B_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(t),$$

for $i = 1, \ldots, n$ with $n = m - p - 1$. Following [?], we introduce also the vector $U = (u_1, \ldots, u_N)$ of breakpoints where N is the number of knots without repetition and the regularity vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N) \in \mathbb{N}^N$ in such a way that for each $i \in \{1, \ldots, N\}$, the B-spline function $B_{i,p}$ is continuously derivable at the breakpoint u_i. Note that $\alpha_i = p - r_i$ where r_i is the multiplicity of the break point u_i. However, we will only consider *non-periodic* knot vectors

$$T = (\underbrace{0, \ldots, 0}_{p+1}, t_{p+2}, \ldots, t_{m-p-1}, \underbrace{1, \ldots, 1}_{p+1}),$$

and we suppose that $0 \le r_i \le p+1$. In this way we guarantee that $-1 \le \alpha_i \le p+1$ where the minimal regularity $\alpha_i = -1$ corresponds to a discontinuity at knot u_i. We also introduce the *Schoenberg space*

$$\mathcal{S}^p_{\boldsymbol{\alpha}} = span\left\{B_{i,p} \,:\, i = 1, \ldots, n\right\}.$$

This definition is generalized to the multivariate case $\Omega = (0,1)^3$ by *tensorization*: With a tridirectional knot vector $\boldsymbol{T} = T_1 \times T_2 \times T_3$ at hand, where

$$T_i = (\underbrace{0, \ldots, 0}_{p_i+1}, t_{i,p_i+2}, \ldots, t_{i,m_i-p_i-1}, \underbrace{1, \ldots, 1}_{p_i+1}), \quad m_i, p_i \in \mathbb{N},\ i = 1,2,3,$$

is an open univariate knot vector, the *three dimensional Schoenberg space* is defined by

$$\boldsymbol{\mathcal{S}}^{p_1,p_2,p_3}_{\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2,\boldsymbol{\alpha}_3} = \mathcal{S}^{p_1}_{\boldsymbol{\alpha}_1} \otimes \mathcal{S}^{p_2}_{\boldsymbol{\alpha}_2} \otimes \mathcal{S}^{p_3}_{\boldsymbol{\alpha}_3},$$

where $\boldsymbol{\alpha}_i$ is the regularity vector related to knot T_i, with $i = 1, 2, 3$. However, we shall also assume our mesh to be *locally quasi-uniform*, meaning, there exists a constant $\theta \geq 1$ such that for all $i \in \{1, 2, 3\}$ we have

$$\frac{1}{\theta} \leq \frac{h_{i,j_i}}{h_{i,j_i+1}} \leq \theta, \quad j_i = 1, \ldots, N_i - 2,$$

where N_i is the number of T_i-knots without repetition and $h_{i,j_i} = t_{i,j_i+1} - t_{i,k_{j_i}}$, with $k_{j_i} = \max\{l : t_l < t_{i,j_i+1}\}$.

With these notations the IsoGeometric spaces read [?, ?]

$$\begin{cases} V_h(\mathbf{grad}, \Omega) := \boldsymbol{\mathcal{S}}^{p_1,p_2,p_3}_{\boldsymbol{\alpha_1},\boldsymbol{\alpha_2},\boldsymbol{\alpha_3}} \\ \boldsymbol{V}_h(\mathbf{curl}, \Omega) := \boldsymbol{\mathcal{S}}^{p_1-1,p_2,p_3}_{\boldsymbol{\alpha_1}-1,\boldsymbol{\alpha_2},\boldsymbol{\alpha_3}} \times \boldsymbol{\mathcal{S}}^{p_1,p_2-1,p_3}_{\boldsymbol{\alpha_1},\boldsymbol{\alpha_2}-1,\boldsymbol{\alpha_3}} \times \boldsymbol{\mathcal{S}}^{p_1,p_2,p_3-1}_{\boldsymbol{\alpha_1},\boldsymbol{\alpha_2},\boldsymbol{\alpha_3}-1} \\ \boldsymbol{V}_h(\mathrm{div}, \Omega) := \boldsymbol{\mathcal{S}}^{p_1,p_2-1,p_3-1}_{\boldsymbol{\alpha_1},\boldsymbol{\alpha_2}-1,\boldsymbol{\alpha_3}-1} \times \boldsymbol{\mathcal{S}}^{p_1-1,p_2,p_3-1}_{\boldsymbol{\alpha_1}-1,\boldsymbol{\alpha_2},\boldsymbol{\alpha_3}-1} \times \boldsymbol{\mathcal{S}}^{p_1-1,p_2-1,p_3}_{\boldsymbol{\alpha_1}-1,\boldsymbol{\alpha_2}-1,\boldsymbol{\alpha_3}}, \end{cases} \tag{2}$$

where h refers to the global mesh size, i.e $h = \max\limits_{\substack{1 \leq j_i \leq N_i - 1 \\ i=1,2,3}} h_{i,j_i}$. These discrete spaces enjoy the following property

2.2 Splines Histopolation

In this subsection, we present the histopolation problem and its matrix form. In opposition to the interpolation problem, where we preserve the values of a function on a given nodes, another interesting way to approximate a function, is to preserve the integrals between given points, rather than the value of the function on these points. Given the set of interpolations points $X := \{x_0, \cdots, x_n\}$ and a continuous function f, the histopolation problem writes

Definition 2.1 (Spline histopolation)**.** *Find a spline* $s := \sum_{j=1}^{n} c_j N_j^p \in \mathcal{S}_p(T)$ *such that*

$$\int_{x_i}^{x_{i+1}} s \; dx = \int_{x_i}^{x_{i+1}} f \; dx \qquad 1 \leq i \leq n \tag{3}$$

In a matrix form, the Spline histopolation problem writes $Hc = y$ where H is the **histopolation matrix**, c is the unkown vector of the spline coefficients and y is the given data, are given by

$$c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \quad H = \begin{bmatrix} \int_{x_0}^{x_1} N_1^p \; dx & \ldots & \int_{x_0}^{x_1} N_n^p \; dx \\ \int_{x_1}^{x_2} N_1^p \; dx & \ldots & \int_{x_1}^{x_2} N_n^p \; dx \\ \vdots & \ldots & \vdots \\ \int_{x_n}^{x_{n+1}} N_1^p \; dx & \ldots & \int_{x_n}^{x_{n+1}} N_n^p \; dx \end{bmatrix} \quad y = \begin{bmatrix} \int_{x_0}^{x_1} f \; dx \\ \int_{x_1}^{x_2} f \; dx \\ \vdots \\ \int_{x_n}^{x_{n+1}} f \; dx \end{bmatrix}$$

We recall that the histopolation matrix H is non singular iff $t_i < x_i < t_{i+p+1} \quad \forall i \in \{0, \ldots, n\}$ add citation.

Histopolation using M-Splines

Rather than using the B-Splines for the histopolation problem, one can use the M-Splines. In this case, the histopolation matrix can be computed easily using the following result.

Proposition 2.1. *For every* $0 \leq i \leq n$ *and* $0 \leq j \leq n$, *we have*

$$\int_{x_i}^{x_{i+1}} M_j^p(t) \; dt = \sum_{k=0}^{j-1} \left(N_k^p(x_i) - N_k^p(x_{i+1}) \right) \tag{4}$$

Proof. Integrating the relation $\frac{d}{dt}N_k^p(t) = M_k^p(t) - M_{k+1}^p(t)$ on the interval $[x_i, x_{i+1}]$, we have

$$N_k^p(x_{i+1}) - N_k^p(x_i) = \int_{x_i}^{x_{i+1}} \left(M_k^p(t) - M_{k+1}^p(t)\right)\ dt$$

summing the last equation for $k = 0$ to $k = j-1$, we get

$$\begin{aligned}\sum_{k=0}^{j-1} \left(N_k^p(x_{i+1}) - N_k^p(x_i)\right) &= \int_{x_i}^{x_{i+1}} \sum_{k=0}^{j-1} \left(M_k^p(t) - M_{k+1}^p(t)\right)\ dt \\ &= \int_{x_i}^{x_{i+1}} \Big(M_0^p(t) - M_j^p(t)\Big)\ dt\end{aligned}$$

hence,

$$\int_{x_i}^{x_{i+1}} M_j^p(t)\ dt = \sum_{k=0}^{j-1} \left(N_k^p(x_i) - N_k^p(x_{i+1})\right)$$

□

A special case, which is of interest for our Auxiliary Space Preconditioning method, is when the function lives in $\mathcal{S}_p(T)$ and the histopolation is done on $\mathcal{S}_{p-1}(T)$ using M-Splines. More precisely, we are interested in the case where Homogeneous Dirichlet boundary conditions are imposed and the interpolating degrees of freedom are removed. Let $u := \sum_{1 \le j \le n-1} u_j N_{\boldsymbol{\mu}_1, j_1}^{p_1}$, we have

$$\int_{x_i}^{x_{i+1}} u\ dx = \sum_{1 \le j \le n} u_j \int_{x_i}^{x_{i+1}} N_{\boldsymbol{\mu}_1, j_1}^{p_1}\ dx \qquad 1 \le i \le n-1 \tag{5}$$

which can be written in a matrix form as

$$\begin{aligned}\begin{bmatrix} \int_{x_0}^{x_1} u\ dx \\ \int_{x_1}^{x_2} u\ dx \\ \vdots \\ \int_{x_n}^{x_{n+1}} u\ dx \end{bmatrix} &= \begin{bmatrix} \int_{x_0}^{x_1} N_1^p\ dx & \dots & \int_{x_0}^{x_1} N_{n-1}^p\ dx \\ \int_{x_1}^{x_2} N_1^p\ dx & \dots & \int_{x_1}^{x_2} N_{n-1}^p\ dx \\ \vdots & \dots & \vdots \\ \int_{x_n}^{x_{n+1}} N_1^p\ dx & \dots & \int_{x_n}^{x_{n+1}} N_{n-1}^p\ dx \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{bmatrix} \\ &= \begin{bmatrix} \int_{x_0}^{x_1} M_1^{p-1}\ dx & \dots & \int_{x_0}^{x_1} M_{n-1}^{p-1}\ dx \\ \int_{x_1}^{x_2} M_1^{p-1}\ dx & \dots & \int_{x_1}^{x_2} M_{n-1}^{p-1}\ dx \\ \vdots & \dots & \vdots \\ \int_{x_n}^{x_{n+1}} M_1^{p-1}\ dx & \dots & \int_{x_n}^{x_{n+1}} M_{n-1}^{p-1}\ dx \end{bmatrix} \begin{bmatrix} u_1^\star \\ u_2^\star \\ \vdots \\ u_{n-1}^\star \end{bmatrix}\end{aligned}$$

Finaly, we get

$$U^\star = \left(H^M\right)^{-1} H^B U \tag{6}$$

with

$$H^M := \begin{bmatrix} \int_{x_0}^{x_1} M_1^{p-1}\, dx & \dots & \int_{x_0}^{x_1} M_{n-1}^{p-1}\, dx \\ \int_{x_1}^{x_2} M_1^{p-1}\, dx & \dots & \int_{x_1}^{x_2} M_{n-1}^{p-1}\, dx \\ \vdots & \dots & \vdots \\ \int_{x_n}^{x_{n+1}} M_1^{p-1}\, dx & \dots & \int_{x_n}^{x_{n+1}} M_{n-1}^{p-1}\, dx \end{bmatrix} \tag{7}$$

$$H^B := \begin{bmatrix} \int_{x_0}^{x_1} N_1^{p}\, dx & \dots & \int_{x_0}^{x_1} N_{n-1}^{p}\, dx \\ \int_{x_1}^{x_2} N_1^{p}\, dx & \dots & \int_{x_1}^{x_2} N_{n-1}^{p}\, dx \\ \vdots & \dots & \vdots \\ \int_{x_n}^{x_{n+1}} N_1^{p}\, dx & \dots & \int_{x_n}^{x_{n+1}} N_{n-1}^{p}\, dx \end{bmatrix} \tag{8}$$

3 Kronecker algebra

In this section, we present an overview about an interesting subject, which is the Kronecker Algebra, and which will be of a big interest in the Fast-IGA approach. More details about Kronecker Algebra can be found in [**?**, **?**, **?**].

Definition 3.1 (The **vec** operator)**.** *Let $A = (a_{ij}) \in \mathcal{M}_{n\times m}$, the* **vec** *operator is defined as,*

$$\mathbf{vec}A = \begin{pmatrix} A_{:,1} \\ \vdots \\ A_{:,m} \end{pmatrix} \in \mathbb{R}^{mn} \tag{9}$$

which is simply a vector composed by stacking all the columns of A. Where we denote $A_{:,j}$ the j^{th} column of A.
We also define the inverse operator of **vec** *by,*

$$A = \mathbf{vec}^{-1}\mathbf{vec}A \tag{10}$$

Definition 3.2 (Kronecker product)**.** *Let $A = (a_{ij}) \in \mathcal{M}_{m\times n}$ and $B = (b_{ij}) \in \mathcal{M}_{r\times s}$ be two matrices. The Kronecker product of A and B, denoted by $A \otimes B \in \mathcal{M}_{mr\times ns}$, defines the following matrix:*

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix} \tag{11}$$

Example

Let

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

then their Kronecker product is,

$$A \otimes B = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{pmatrix} \tag{12}$$

Properties

In the sequel, we recall some basic properties of the Kronecker product.

Proposition 3.1. *We have the following properties*

(a). If α is a scalar, then

$$A \otimes \alpha B = \alpha A \otimes B \tag{13}$$

(b). Distributivity

$$\begin{cases} (A+B) \otimes C = A \otimes C + B \otimes C \\ A \otimes (B+C) = A \otimes B + A \otimes C \end{cases} \tag{14}$$

(c). Associativity

$$A \otimes B \otimes C = A \otimes (B \otimes C) = (A \otimes B) \otimes C \tag{15}$$

(d). Mixed Product Rule

$$(A \otimes B)(C \otimes D) = AC \otimes BD \tag{16}$$

(e). Transposition

$$(A \otimes B)^T = A^T \otimes B^T \tag{17}$$

(f). Inverse

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{18}$$

(g). The **vec** *operator*

$$\mathbf{vec}(ABC) = (C^T \otimes A)\mathbf{vec}(B) \tag{19}$$

(h). Trace

$$\mathbf{tr}(A \otimes B) = \mathbf{tr}(B \otimes A) = \mathbf{tr}(A)\mathbf{tr}(B) \tag{20}$$

(i). Let $A \in \mathcal{M}_{n\times n}$ and $B \in \mathcal{M}_{m\times m}$, we have,

$$\mathbf{mspec}(A \otimes B) = \{\lambda\mu, \quad \lambda \in \mathbf{mspec}(A),\ \mu \in \mathbf{mspec}(B)\} \tag{21}$$

(j). Let $A \in \mathcal{M}_{n\times n}$ and $B \in \mathcal{M}_{m\times m}$, then we have the following properties,

- *if A and B are diagonal, then $A \otimes B$ is diagonal,*
- *if A and B are upper triangular, then $A \otimes B$ is upper triangular,*
- *if A and B are lower triangular, then $A \otimes B$ is lower triangular,*

4 Matrix form of the variational problem

In the sequel, we derive the matrix form related to the discrete variational formulation of (**??**). We shall consider a computational domain Ω as unit square $(0,1)^2$ or unit cube $(0,1)^3$ and derive the matrix form using the Kronecker Algebra. Before expliciting the matrix forms in 2D and 3D, we start by introducing some 1D matrices that we will need.

$$\begin{cases} (M_s)_{i_s,j_s} &= \int_0^1 N^{p_s}_{\boldsymbol{\mu}_s,i_s} N^{p_s}_{\boldsymbol{\mu}_s,j_s} \,\mathrm{d}x_s \\ (K_s)_{i_s,j_s} &= \int_0^1 N^{p_s\,\prime}_{\boldsymbol{\mu}_s,i_s} N^{p_s\,\prime}_{\boldsymbol{\mu}_s,j_s} \,\mathrm{d}x_s \\ (D_s)_{i_s,j_s} &= \int_0^1 M^{p_s-1}_{\boldsymbol{\mu}_s-1,i_s} M^{p_s-1}_{\boldsymbol{\mu}_s-1,j_s} \,\mathrm{d}x_s \\ (R_s)_{i_s,j_s} &= \int_0^1 N^{p_s\,\prime}_{\boldsymbol{\mu}_s,i_s} M^{p_s-1}_{\boldsymbol{\mu}_s-1,j_s} \,\mathrm{d}x_s \end{cases} \tag{22}$$

4.1 2D case

We consider the discrete variational formulation of (**??**). For the sake of simplicity we shall introduce the scalar functions

$$\Psi^1_{\boldsymbol{j}} = M^{p_1-1}_{\boldsymbol{\mu}_1-1,j_1} N^{p_2}_{\boldsymbol{\mu}_2,j_2}$$
$$\Psi^2_{\boldsymbol{j}} = N^{p_1}_{\boldsymbol{\mu}_1,j_1} M^{p_2-1}_{\boldsymbol{\mu}_2-1,j_2}$$

we also define the vectors $\boldsymbol{e}_1 = \begin{bmatrix}1\\0\end{bmatrix}$ and $\boldsymbol{e}_2 = \begin{bmatrix}0\\1\end{bmatrix}$. Therefor, the expression of $\boldsymbol{u}_h \in \boldsymbol{V}_h(\mathrm{curl},\Omega)$ becomes

$$\boldsymbol{u}_h = \sum_{\boldsymbol{j}} \left(u^1_{\boldsymbol{j}} \Psi^1_{\boldsymbol{j}} \boldsymbol{e}_1 + u^2_{\boldsymbol{j}} \Psi^2_{\boldsymbol{j}} \boldsymbol{e}_2 \right)$$

we find that A is a symmetric 2×2 block matrix of the form

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix} \tag{23}$$

where

$$A_{11\boldsymbol{i},\boldsymbol{j}} = \int_\Omega \partial_{x_2}\Psi^1_{\boldsymbol{j}} \partial_{x_2}\Psi^1_{\boldsymbol{i}} \,\mathrm{d}\mathbf{x} + \int_\Omega \tau \Psi^1_{\boldsymbol{j}} \Psi^1_{\boldsymbol{i}} \,\mathrm{d}\mathbf{x}$$
$$A_{12\boldsymbol{i},\boldsymbol{j}} = -\int_\Omega \partial_{x_2}\Psi^1_{\boldsymbol{j}} \partial_{x_1}\Psi^2_{\boldsymbol{i}} \,\mathrm{d}\mathbf{x}$$
$$A_{22\boldsymbol{i},\boldsymbol{j}} = \int_\Omega \partial_{x_1}\Psi^2_{\boldsymbol{j}} \ \partial_{x_1}\Psi^2_{\boldsymbol{i}} \,\mathrm{d}\mathbf{x} + \int_\Omega \tau \Psi^2_{\boldsymbol{j}} \Psi^2_{\boldsymbol{i}} \,\mathrm{d}\mathbf{x}$$

For the right hand side, the entries associated to each component of the vector $\boldsymbol{f}$ are given by

$$F_{1,i} = \int_\Omega \boldsymbol{f}_1 \Psi^1_i \,\mathrm{d}\mathbf{x}$$
$$F_{2,i} = \int_\Omega \boldsymbol{f}_2 \Psi^2_i \,\mathrm{d}\mathbf{x}$$

Hence, we have,

$$\begin{cases} A_{11\boldsymbol{i},\boldsymbol{j}} & = (D_1 \otimes K_2)_{\boldsymbol{ij}} + \tau \left(D_1 \otimes M_2\right)_{\boldsymbol{ij}} \\ A_{12\boldsymbol{i},\boldsymbol{j}} & = -\left(R_1 \otimes R_2^T\right)_{\boldsymbol{ij}} \\ A_{22\boldsymbol{i},\boldsymbol{j}} & = (K_1 \otimes D_2)_{\boldsymbol{ij}} + \tau \left(M_1 \otimes D_2\right)_{\boldsymbol{ij}} \end{cases}$$

Therefor, we have the following matrix form

$$\mathcal{A}^\tau = \mathcal{A}^0 + \tau \mathcal{M} \tag{24}$$

where

$$\mathcal{A}^0 = \begin{bmatrix} D_1 \otimes K_2 & -R_1 \otimes R_2^T \\ -R_1^T \otimes R_2 & K_1 \otimes D_2 \end{bmatrix} \tag{25}$$

and

$$\mathcal{M} = \begin{bmatrix} D_1 \otimes M_2 & 0 \\ 0 & M_1 \otimes D_2 \end{bmatrix} \tag{26}$$

4.2 3D case

We consider the discrete variational formulation of (**??**). For the sake of simplicity we shall introduce the scalar functions

$$\begin{aligned} \Psi_{\boldsymbol{j}}^1 &= M_{\boldsymbol{\mu}_1-1,j_1}^{p_1-1} N_{\boldsymbol{\mu}_2,j_2}^{p_2} N_{\boldsymbol{\mu}_3,j_3}^{p_3} \\ \Psi_{\boldsymbol{j}}^2 &= N_{\boldsymbol{\mu}_1,j_1}^{p_1} M_{\boldsymbol{\mu}_2-1,j_2}^{p_2-1} N_{\boldsymbol{\mu}_3,j_3}^{p_3} \\ \Psi_{\boldsymbol{j}}^3 &= N_{\boldsymbol{\mu}_1,j_1}^{p_1} N_{\boldsymbol{\mu}_2,j_2}^{p_2} M_{\boldsymbol{\mu}_3-1,j_3}^{p_3-1} \end{aligned}$$

we also define the vectors $\boldsymbol{e}_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}$, $\boldsymbol{e}_2 = \begin{bmatrix}0\\1\\0\end{bmatrix}$ and $\boldsymbol{e}_3 = \begin{bmatrix}0\\0\\1\end{bmatrix}$. Therefor, the expression of $\boldsymbol{u}_h \in \boldsymbol{V}_h(\mathrm{curl}, \Omega)$ becomes

$$\boldsymbol{u}_h = \sum_{\boldsymbol{j}} \left(u_{\boldsymbol{j}}^1 \Psi_{\boldsymbol{j}}^1 \boldsymbol{e}_1 + u_{\boldsymbol{j}}^2 \Psi_{\boldsymbol{j}}^2 \boldsymbol{e}_2 + u_{\boldsymbol{j}}^3 \Psi_{\boldsymbol{j}}^3 \boldsymbol{e}_3 \right)$$

On the other hand, we have,

$$\begin{aligned} \nabla \times \Psi_{\boldsymbol{j}}^1 \boldsymbol{e}_1 &= \partial_{x_3} \Psi_{\boldsymbol{j}}^1 \boldsymbol{e}_2 - \partial_{x_2} \Psi_{\boldsymbol{j}}^1 \boldsymbol{e}_3 \\ \nabla \times \Psi_{\boldsymbol{j}}^2 \boldsymbol{e}_2 &= -\partial_{x_3} \Psi_{\boldsymbol{j}}^2 \boldsymbol{e}_1 + \partial_{x_1} \Psi_{\boldsymbol{j}}^2 \boldsymbol{e}_3 \\ \nabla \times \Psi_{\boldsymbol{j}}^3 \boldsymbol{e}_3 &= \partial_{x_2} \Psi_{\boldsymbol{j}}^3 \boldsymbol{e}_1 - \partial_{x_1} \Psi_{\boldsymbol{j}}^3 \boldsymbol{e}_2 \end{aligned}$$

Because $\boldsymbol{e}_i \cdot \boldsymbol{e}_j = \delta_{ij}$, we get,

$$\begin{aligned} \nabla \times \Psi_{\boldsymbol{j}}^1 \boldsymbol{e}_1 \cdot \nabla \times \Psi_{\boldsymbol{i}}^1 \boldsymbol{e}_1 &= \partial_{x_3} \Psi_{\boldsymbol{j}}^1 \ \partial_{x_3} \Psi_{\boldsymbol{i}}^1 + \partial_{x_2} \Psi_{\boldsymbol{j}}^1 \ \partial_{x_2} \Psi_{\boldsymbol{i}}^1 \\ \nabla \times \Psi_{\boldsymbol{j}}^1 \boldsymbol{e}_1 \cdot \nabla \times \Psi_{\boldsymbol{i}}^2 \boldsymbol{e}_2 &= -\partial_{x_2} \Psi_{\boldsymbol{j}}^1 \ \partial_{x_1} \Psi_{\boldsymbol{i}}^2 \\ \nabla \times \Psi_{\boldsymbol{j}}^1 \boldsymbol{e}_1 \cdot \nabla \times \Psi_{\boldsymbol{i}}^3 \boldsymbol{e}_3 &= -\partial_{x_3} \Psi_{\boldsymbol{j}}^1 \ \partial_{x_1} \Psi_{\boldsymbol{i}}^3 \\ \nabla \times \Psi_{\boldsymbol{j}}^2 \boldsymbol{e}_2 \cdot \nabla \times \Psi_{\boldsymbol{i}}^2 \boldsymbol{e}_2 &= \partial_{x_3} \Psi_{\boldsymbol{j}}^2 \ \partial_{x_3} \Psi_{\boldsymbol{i}}^2 + \partial_{x_1} \Psi_{\boldsymbol{j}}^2 \ \partial_{x_1} \Psi_{\boldsymbol{i}}^2 \\ \nabla \times \Psi_{\boldsymbol{j}}^2 \boldsymbol{e}_2 \cdot \nabla \times \Psi_{\boldsymbol{i}}^3 \boldsymbol{e}_3 &= -\partial_{x_3} \Psi_{\boldsymbol{j}}^2 \ \partial_{x_2} \Psi_{\boldsymbol{i}}^3 \\ \nabla \times \Psi_{\boldsymbol{j}}^3 \boldsymbol{e}_3 \cdot \nabla \times \Psi_{\boldsymbol{i}}^3 \boldsymbol{e}_3 &= \partial_{x_2} \Psi_{\boldsymbol{j}}^3 \ \partial_{x_2} \Psi_{\boldsymbol{i}}^3 + \partial_{x_1} \Psi_{\boldsymbol{j}}^3 \ \partial_{x_1} \Psi_{\boldsymbol{i}}^3 \end{aligned}$$

we find that A is a symmetric 3×3 block matrix of the form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{12}^T & A_{22} & A_{23} \\ A_{13}^T & A_{23}^T & A_{33} \end{bmatrix} \tag{27}$$

where

$$\begin{aligned}
A_{11\boldsymbol{i},\boldsymbol{j}} &= \int_\Omega \partial_{x_3}\Psi^1_{\boldsymbol{j}}\partial_{x_3}\Psi^1_{\boldsymbol{i}} + \partial_{x_2}\Psi^1_{\boldsymbol{j}}\partial_{x_2}\Psi^1_{\boldsymbol{i}}\,\mathrm{d}\mathbf{x} + \int_\Omega \tau\Psi^1_{\boldsymbol{j}}\Psi^1_{\boldsymbol{i}}\,\mathrm{d}\mathbf{x} \\
A_{12\boldsymbol{i},\boldsymbol{j}} &= -\int_\Omega \partial_{x_2}\Psi^1_{\boldsymbol{j}}\partial_{x_1}\Psi^2_{\boldsymbol{i}}\,\mathrm{d}\mathbf{x} \\
A_{13\boldsymbol{i},\boldsymbol{j}} &= -\int_\Omega \partial_{x_3}\Psi^1_{\boldsymbol{j}}\partial_{x_1}\Psi^3_{\boldsymbol{i}}\,\mathrm{d}\mathbf{x} \\
A_{22\boldsymbol{i},\boldsymbol{j}} &= \int_\Omega \partial_{x_3}\Psi^2_{\boldsymbol{j}}\ \partial_{x_3}\Psi^2_{\boldsymbol{i}} + \partial_{x_1}\Psi^2_{\boldsymbol{j}}\ \partial_{x_1}\Psi^2_{\boldsymbol{i}}\,\mathrm{d}\mathbf{x} + \int_\Omega \tau\Psi^2_{\boldsymbol{j}}\Psi^2_{\boldsymbol{i}}\,\mathrm{d}\mathbf{x} \\
A_{23\boldsymbol{i},\boldsymbol{j}} &= -\int_\Omega \partial_{x_3}\Psi^2_{\boldsymbol{j}}\ \partial_{x_2}\Psi^3_{\boldsymbol{i}}\,\mathrm{d}\mathbf{x} \\
A_{33\boldsymbol{i},\boldsymbol{j}} &= \int_\Omega \partial_{x_2}\Psi^3_{\boldsymbol{j}}\ \partial_{x_2}\Psi^3_{\boldsymbol{i}} + \partial_{x_1}\Psi^3_{\boldsymbol{j}}\ \partial_{x_1}\Psi^3_{\boldsymbol{i}}\,\mathrm{d}\mathbf{x} + \int_\Omega \tau\Psi^3_{\boldsymbol{j}}\Psi^3_{\boldsymbol{i}}\,\mathrm{d}\mathbf{x}
\end{aligned}$$

For the right hand side, the entries associated to each component of the vector $\boldsymbol{f}$ are given by

$$\begin{aligned}
F_{1,i} &= \int_\Omega \boldsymbol{f}_1\Psi^1_i\,\mathrm{d}\mathbf{x} \\
F_{2,i} &= \int_\Omega \boldsymbol{f}_2\Psi^2_i\,\mathrm{d}\mathbf{x} \\
F_{3,i} &= \int_\Omega \boldsymbol{f}_3\Psi^3_i\,\mathrm{d}\mathbf{x}
\end{aligned}$$

Hence, we have,

$$\begin{cases}
A_{11\boldsymbol{i},\boldsymbol{j}} & = (D_1 \otimes M_2 \otimes K_3)_{\boldsymbol{ij}} + (D_1 \otimes K_2 \otimes M_3)_{\boldsymbol{ij}} + \tau\,(D_1 \otimes M_2 \otimes M_3)_{\boldsymbol{ij}} \\
A_{12\boldsymbol{i},\boldsymbol{j}} & = -\left(R_1 \otimes R_2^T \otimes M_3\right)_{\boldsymbol{ij}} \\
A_{13\boldsymbol{i},\boldsymbol{j}} & = -\left(R_1 \otimes M_2 \otimes R_3^T\right)_{\boldsymbol{ij}} \\
A_{22\boldsymbol{i},\boldsymbol{j}} & = (M_1 \otimes D_2 \otimes K_3)_{\boldsymbol{ij}} + (K_1 \otimes D_2 \otimes M_3)_{\boldsymbol{ij}} + \tau\,(M_1 \otimes D_2 \otimes M_3)_{\boldsymbol{ij}} \\
A_{23\boldsymbol{i},\boldsymbol{j}} & = -\left(M_1 \otimes R_2 \otimes R_3^T\right)_{\boldsymbol{ij}} \\
A_{33\boldsymbol{i},\boldsymbol{j}} & = (M_1 \otimes K_2 \otimes D_3)_{\boldsymbol{ij}} + (K_1 \otimes M_2 \otimes D_3)_{\boldsymbol{ij}} + \tau\,(M_1 \otimes M_2 \otimes D_3)_{\boldsymbol{ij}}
\end{cases}$$

Therefor, we have the following matrix form

$$\mathcal{A}^\tau = \mathcal{A}^0 + \tau\mathcal{M} \tag{28}$$

where

$$\mathcal{A}^0 = \begin{bmatrix} D_1 \otimes M_2 \otimes K_3 + D_1 \otimes K_2 \otimes M_3 & -R_1 \otimes R_2^T \otimes M_3 & -R_1 \otimes M_2 \otimes R_3^T \\ -R_1^T \otimes R_2 \otimes M_3 & M_1 \otimes D_2 \otimes K_3 + K_1 \otimes D_2 \otimes M_3 & -M_1 \otimes R_2 \otimes R_3^T \\ -R_1^T \otimes M_2 \otimes R_3 & -M_1 \otimes R_2^T \otimes R_3 & M_1 \otimes K_2 \otimes D_3 + K_1 \otimes M_2 \otimes D_3 \end{bmatrix} \tag{29}$$

and

$$\mathcal{M} = \begin{bmatrix} D_1 \otimes M_2 \otimes M_3 & 0 & 0 \\ 0 & M_1 \otimes D_2 \otimes M_3 & 0 \\ 0 & 0 & M_1 \otimes M_2 \otimes D_3 \end{bmatrix} \tag{30}$$

5 Auxiliary Space Preconditioners

The objective of this section is the construction of an appropriate auxiliary space preconditioner for our **curl-curl** problem. As already mentioned, the main challenge is the derivation of a discrete version of the regular decomposition of Theorem **??**, namely the Hitmair-Xu decomposition. The presentation is in three subsections. First, in Subsection **??** we focus on the case without mapping where Ω is simply a parametric domain given by $\Omega = (0,1)^3$ and we show the discrete Hitmair-Xu decomposition in this case. The result of this subsection is extended to the case of a physical domain in Subsection **??**. The construction of the preconditioner is developed in Subsection **??**.

Through this section $A \lesssim B$ means that there exists some constant $C > 0$, which is independent of h, such that $A \leq CB$.

5.1 Auxiliary Space Preconditioners

Now we have all the ingredients to apply the abstract ASP theory of Section **??**. Indeed, armed with the above stable discrete decomposition result, using the notations of Section **??**, we consider $V = \boldsymbol{V}_h(\mathbf{curl}, \Omega)$ equipped with the bilinear form a related to equation (**??**), namely $a(\boldsymbol{w}_h, \widetilde{\boldsymbol{w}_h}) = (\mathbf{curl}\,(\boldsymbol{w}_h), \mathbf{curl}\,(\widetilde{\boldsymbol{w}_h}))_{\mathbb{L}^2(\Omega)} + \mu(\boldsymbol{w}_h, \widetilde{\boldsymbol{w}_h})_{\mathbb{L}^2(\Omega)}$ for $\boldsymbol{w}_h, \widetilde{\boldsymbol{w}_h} \in V$, and auxiliary spaces $W_1 = \mathbb{V}_h(\mathbf{curl}, \Omega)$ and $W_2 = V_h(\mathbf{grad}, \Omega)$ equipped with the following inner products

$$a_1(\boldsymbol{\varphi}_h, \widetilde{\boldsymbol{\varphi}_h}) = (\boldsymbol{\varphi}_h, \widetilde{\boldsymbol{\varphi}_h})_{\mathbb{H}^1(\Omega)} + \mu(\boldsymbol{\varphi}_h, \widetilde{\boldsymbol{\varphi}_h})_{\mathbb{L}^2(\Omega)}, \quad \boldsymbol{\varphi}_h, \widetilde{\boldsymbol{\varphi}_h} \in W_1,$$

and

$$a_2(\phi_h, \widetilde{\phi_h}) = \mu\,(\mathbf{grad}\,(\phi_h), \mathbf{grad}\,(\widetilde{\phi_h}))_{\mathbb{L}^2(\Omega)}, \quad \phi_h, \widetilde{\phi_h} \in W_2,$$

respectively. The corresponding transfer operators are $\pi_1 = \Pi_h^{\mathbf{curl}} : W_1 \longrightarrow V$ and $\pi_2 = \mathbf{grad} : W_2 \longrightarrow V$ and the smoother is simply the inner product a.

Before we verify the validity of assumptions of Theorem **??**, we transition to a matrix notation. So we write C_h^1 for the matrix related to the restriction of $\mathbb{H}^1(\Omega)$ inner product to $\mathbb{V}_h(\mathbf{grad}, \Omega)$, that is the matrix representation of the bilinear form

$$(\boldsymbol{\varphi}_h, \widetilde{\boldsymbol{\varphi}_h}) \in \mathbb{V}_h(\mathbf{grad}, \Omega) \times \mathbb{V}_h(\mathbf{grad}, \Omega) \mapsto (\boldsymbol{\varphi}_h, \widetilde{\boldsymbol{\varphi}_h})_{\mathbb{H}^1(\Omega)}.$$

Similarly, we write C_h^2 for the matrix related to the restriction of the $\mathbb{L}^2(\Omega)$ inner product to $\mathbb{V}_h(\mathbf{grad}, \Omega)$. Let L_h be the discrete Laplacian matrix, namely the matrix related to the mapping

$$(\boldsymbol{\phi}_h, \widetilde{\boldsymbol{\phi}_h}) \in V_h(\mathbf{grad}, \Omega) \times V_h(\mathbf{grad}, \Omega) \longmapsto \Big(\mathbf{grad}\,(\boldsymbol{\phi}_h), \mathbf{grad}\,(\widetilde{\boldsymbol{\phi}_h})\Big)_{\mathbb{L}^2(\Omega)}.$$

We also write $P_h^{\mathbf{curl}}$ and G_h for the matrices related to the transform operators π_1 and π_2 respectively while R_h further stands for the matrix related to the smoother. With these notations, a simple computation shows that ASP preconditioner for problem (**??**) reads

$$B_h = R_h^{-1} + P_h^{\mathbf{curl}} \left(C_h^1 + \mu C_h^2\right)^{-1} \left(P_h^{\mathbf{curl}}\right)^T + \mu^{-1} G_h L_h^{-1} \left(G_h\right)^T.$$

A direct consequence of Theorem **??** and Theorem **??** the following fundamental result which show the mesh-independent of the preconditioner.

Theorem 5.1. *For $0 < \mu \leq 1$, the spectral condition number $\kappa\,(B_h A_h)$ is bounded, with respect to h.*

6 Algorithms

Algorithm 1: ASP: Auxiliary Space Preconditioning for $\boldsymbol{V}_h(\mathbf{curl}, \Omega)$

```
Input  : $\mathcal{A}^\tau, \mathcal{B}^\tau, b, u_0, \nu_1, \nu_{\mathtt{ASP}}$
Output: $u_k$
1 $k \leftarrow 0$
2 while $k \leq \nu_{ASP}$ and not convergence do
3     $u_k \leftarrow$ gauss_seidel($\mathcal{A}^\tau, b, u_k, \nu_1$)       // Apply Gauss-Seidel smoother
4     $d_k \leftarrow b - \mathcal{A}^\tau u_k$                               // Compute the defect
5     $u_c \leftarrow \mathcal{B}^\tau d_k$                                   // ASP correction
6     $u_k \leftarrow u_k + u_c$                                              // Update the solution
7     $k \leftarrow k + 1$
8 end
```

6.1 Fast Diagonalization method

In order to device a fast solver the Poisson and Laplace problems, we choose to follow the work of Sangalli and Tani [**?**], we describe in the sequel the fast diagonalization method in the case of Isogeometric Analysis. This method was first introduced in [**?**].
For the sack of simplicity, we shall consider the following Laplace problem,

$$\begin{cases} -\nabla^2 u + \tau u = f, & \Omega \\ u = 0, & \partial\Omega \end{cases} \tag{31}$$

The Poisson problem and its solver shall be retrieved with $\tau = 0$. After discretizing the Laplace problem, we get the following linear system

$$\mathcal{L}_\tau x := \left(K_1 \otimes M_2 \otimes M_3 + M_1 \otimes K_2 \otimes M_3 + M_1 \otimes M_2 \otimes K_3 + \tau M_1 \otimes M_2 \otimes M_3 \right) x = b \tag{32}$$

We first consider the generalized eigendecompositions problems

$$K_1 U_1 = M_1 U_1 D_1, \quad K_2 U_2 = M_2 U_2 D_2, \quad K_3 U_3 = M_3 U_3 D_3, \tag{33}$$

where D_1, D_2 and D_3 are diagonal matrices such that

$$U_1^T M_1 U_1 = I_1, \quad U_2^T M_2 U_2 = I_2, \quad U_3^T M_3 U_3 = I_3 \tag{34}$$

Therefor, (**??**) can be written as

$$\left(U_1 \otimes U_2 \otimes U_3 \right)^{-1} \left(D_1 \otimes I_2 \otimes I_3 + I_1 \otimes D_2 \otimes I_3 + I_1 \otimes I_2 \otimes D_3 + \tau I_1 \otimes I_2 \otimes I_3 \right) \left(U_1 \otimes U_2 \otimes U_3 \right)^{-T} x = b \tag{35}$$

The direct solver for the Laplace problem (**??**), is then given by the following algorithm, where we omit the initialization step achieved by solving the generalized eigendecompositions in (**??**): We consider now the vector Laplace problem

$$\begin{cases} -\nabla^2 \mathbf{u} + \tau \mathbf{u} = \mathbf{f}, & \Omega \\ \mathbf{u} = 0, & \partial\Omega \end{cases} \tag{36}$$

Which can be written in a matrix form as

$$\boldsymbol{\mathcal{L}}_\tau = \begin{bmatrix} \mathcal{L}_\tau & 0 & 0 \\ 0 & \mathcal{L}_\tau & 0 \\ 0 & 0 & \mathcal{L}_\tau \end{bmatrix}. \tag{37}$$

Algorithm 2: `fast_diag`: Fast diagonalization method for Laplace problem

```
Input   : $\mathcal{L}_\tau, b$
Output: $x$
1 $\tilde{b} \leftarrow (U_1 \otimes U_2 \otimes U_3)\, b$
2 $\tilde{x} \leftarrow (D_1 \otimes I_2 \otimes I_3 + I_1 \otimes D_2 \otimes I_3 + I_1 \otimes I_2 \otimes D_3 + \tau I_1 \otimes I_2 \otimes I_3)^{-1} \tilde{b}$
3 $x \leftarrow (U_1 \otimes U_2 \otimes U_3)^T \tilde{x}$
```

Therefor, a fast solver for the vector Laplace problem (**??**) can be given by the following algorithm

Algorithm 3: `fast_diag`: Fast diagonalization method for the vector Laplace problem

```
Input   : $\boldsymbol{\mathcal{L}}_\tau, b$
Output: $x$
1 $b_1, b_2, b_3 \leftarrow$ unfold$(b)$
2 $x_1 \leftarrow$ fast_diag$(\mathcal{L}_\tau, b_1)$
3 $x_2 \leftarrow$ fast_diag$(\mathcal{L}_\tau, b_2)$
4 $x_3 \leftarrow$ fast_diag$(\mathcal{L}_\tau, b_3)$
5 $x \leftarrow$ fold$(x_1, x_2, x_3)$
```

6.2 Discrete derivatives

We denote by $\mathbb{I}$ the identity matrix. The discrete derivatives in 2D are given by

$$\begin{cases} \mathbb{G} &= \begin{bmatrix} \mathcal{D}_1 \otimes \mathbb{I}_2 \\ \mathbb{I}_1 \otimes \mathcal{D}_2 \end{bmatrix} \\ \boldsymbol{\mathbb{C}} &= \begin{bmatrix} \mathbb{I}_1 \otimes \mathcal{D}_2 \\ -\mathcal{D}_1 \otimes \mathbb{I}_2 \end{bmatrix} \\ \mathbb{C} &= \begin{bmatrix} -\mathbb{I}_1 \otimes \mathcal{D}_2 & \mathcal{D}_1 \otimes \mathbb{I}_2 \end{bmatrix} \\ \mathbb{D} &= \begin{bmatrix} \mathcal{D}_1 \otimes \mathbb{I}_2 & \mathbb{I}_1 \otimes \mathcal{D}_2 \end{bmatrix} \end{cases} \tag{38}$$

The discrete derivatives in 3D are given by

$$\begin{cases} \mathbb{G} &= \begin{bmatrix} \mathcal{D}_1 \otimes \mathbb{I}_2 \otimes \mathbb{I}_3 \\ \mathbb{I}_1 \otimes \mathcal{D}_2 \otimes \mathbb{I}_3 \\ \mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \mathcal{D}_3 \end{bmatrix} \\ \mathbb{C} &= \begin{bmatrix} 0 & -\mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \mathcal{D}_3 & \mathbb{I}_1 \otimes \mathcal{D}_2 \otimes \mathbb{I}_3 \\ \mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \mathcal{D}_3 & 0 & -\mathcal{D}_1 \otimes \mathbb{I}_2 \otimes \mathbb{I}_3 \\ -\mathbb{I}_1 \otimes \mathcal{D}_2 \otimes \mathbb{I}_3 & \mathcal{D}_1 \otimes \mathbb{I}_2 \otimes \mathbb{I}_3 & 0 \end{bmatrix} \\ \mathbb{D} &= \begin{bmatrix} \mathcal{D}_1 \otimes \mathbb{I}_2 \otimes \mathbb{I}_3 & \mathbb{I}_1 \otimes \mathcal{D}_2 \otimes \mathbb{I}_3 & \mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \mathcal{D}_3 \end{bmatrix} \end{cases} \tag{39}$$

Remark 6.1. *The actual implementation is based on a matrix form, but we should avoid it and implement these operators as functions. This will reduce the memory usage.*

6.3 The $P_h^{\mathbf{curl}}$ Operator

Let us define the following 1D matrices

$$\begin{cases} \mathcal{H}_1 &= \left(H_1^M\right)^{-1} H_1^B \\ \mathcal{H}_2 &= \left(H_2^M\right)^{-1} H_2^B \\ \mathcal{H}_3 &= \left(H_3^M\right)^{-1} H_3^B \end{cases} \tag{40}$$

where H_k^M and H_k^B for $k \in \{1, 2, 3\}$ are defined for each direction k using **??**. In the 2D case, the P_h^{curl} is defined as

$$P_h^{\mathbf{curl}} = \begin{bmatrix} \mathcal{H}_1 \otimes \mathbb{I}_2 \\ \mathbb{I}_1 \otimes \mathcal{H}_2 \end{bmatrix} \tag{41}$$

In the 3D case, the P_h^{curl} is defined as

$$P_h^{\mathrm{curl}} = \begin{bmatrix} \mathcal{H}_1 \otimes \mathbb{I}_2 \otimes \mathbb{I}_3 \\ \mathbb{I}_1 \otimes \mathcal{H}_2 \otimes \mathbb{I}_3 \\ \mathbb{I}_1 \otimes \mathbb{I}_2 \otimes \mathcal{H}_3 \end{bmatrix} \tag{42}$$

Remark 6.2. *In the actual implementation, we compute the matrices $\left(H_k^M\right)^{-1} H_k^B, k \in \{1, 2, 3\}$ which are represented as dense matrices. It would be better to apply the product then solve for $\left(H_k^M\right)^{-1}$ using band solvers from `LAPACK` , see `DGBTRS` and `DGBTRF`.*

6.4 The symmetric Gauss-Seidel method

In Algo. **??**, we recall the symmetric Gauss-Seidel method. In this algorithm, we use our `spsolve` driver, which has different implementations depending on the type of the matrix $\mathcal{A}$. These implementation will be given in the next subsection.

Algorithm 4: `gauss_seidel`: Symmetric Gauss Seidel solver

```
Input  : $\mathcal{A}, x, b, \nu_1$
Output: $x$
1 for $i \leftarrow 1$ to $\nu_1$ do
2     $x \leftarrow x + \texttt{spsolve}(\mathcal{A}, b - \mathcal{A}x, \texttt{lower} = \texttt{True})$
3 end
4 for $i \leftarrow 1$ to $\nu_1$ do
5     $x \leftarrow x + \texttt{spsolve}(\mathcal{A}, b - \mathcal{A}x, \texttt{lower} = \texttt{False})$
6 end
```

2D case

We consider in the following a block matrix

$$\mathcal{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{43}$$

Algorithm 5: `spsolve`: Lower triangular solver for 2×2 block matrix

```
Input  : $\mathcal{A}, b$
Output: $x$
1 $b_1, b_2 \leftarrow \texttt{unfold}(b)$
2 $x_1 \leftarrow \texttt{spsolve}(A_{11}, b_1, \texttt{lower} = \texttt{True})$
3 $\tilde{b}_2 \leftarrow b_2 - A_{21}x_1$
4 $x_2 \leftarrow \texttt{spsolve}(A_{22}, \tilde{b}_2, \texttt{lower} = \texttt{True})$
5 $x \leftarrow \texttt{fold}(x_1, x_2)$
```

Algorithm 6: `spsolve`: Upper triangular solver for 2×2 block matrix

```
Input  : $\mathcal{A}, b$
Output: $x$
1 $b_1, b_2 \leftarrow \texttt{unfold}(b)$
2 $x_2 \leftarrow \texttt{spsolve}(A_{22}, b_2, \texttt{lower} = \texttt{False})$
3 $\tilde{b}_1 \leftarrow b_1 - A_{12}x_2$
4 $x_1 \leftarrow \texttt{spsolve}(A_{11}, \tilde{b}_1, \texttt{lower} = \texttt{False})$
5 $x \leftarrow \texttt{fold}(x_1, x_2)$
```

3D case

We consider in the following a block matrix

$$\mathcal{A} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{22} & A_{33} \end{bmatrix}. \tag{44}$$

Algorithm 7: `spsolve`: Lower triangular solver for 3×3 block matrix

```
Input  : $\mathcal{A}, b$
Output: $x$
1 $b_1, b_2, b_3 \leftarrow$ unfold($b$)
2 $x_1 \leftarrow$ spsolve($A_{11}, b_1$, lower = True)
3 $\tilde{b}_2 \leftarrow b_2 - A_{21} x_1$
4 $x_2 \leftarrow$ spsolve($A_{22}, \tilde{b}_2$, lower = True)
5 $\tilde{b}_3 \leftarrow b_3 - A_{31} x_1 - A_{32} x_2$
6 $x_3 \leftarrow$ spsolve($A_{33}, \tilde{b}_3$, lower = True)
7 $x \leftarrow$ fold($x_1, x_2, x_3$)
```

Algorithm 8: `spsolve`: Upper triangular solver for 3×3 block matrix

```
Input  : $\mathcal{A}, b$
Output: $x$
1 $b_1, b_2, b_3 \leftarrow$ unfold($b$)
2 $x_3 \leftarrow$ spsolve($A_{33}, b_3$, lower = False)
3 $\tilde{b}_2 \leftarrow b_2 - A_{23} x_3$
4 $x_2 \leftarrow$ spsolve($A_{22}, \tilde{b}_2$, lower = False)
5 $\tilde{b}_1 \leftarrow b_1 - A_{12} x_2 - A_{13} x_3$
6 $x_1 \leftarrow$ spsolve($A_{11}, \tilde{b}_1$, lower = False)
7 $x \leftarrow$ fold($x_1, x_2, x_3$)
```

6.5 Triangular solvers

In this part, we provide different implementations for the upper and lower triangular solver that are used in our method. We refer to our driver as `spsolve`. Since the diagonal block matrices can be either a Kronecker product of 3 matrices or the sum a Kronecker product of 3 matrices, we can then derive efficient matrix-free implementation as described in the following algorithms.

Algorithm 9: `spsolve`: Lower triangular solver for sum of Kronecker product [CSR] matrices.

```
Input  : $\mathcal{A} = \alpha A_1 \otimes A_2 \otimes A_3 + \beta B_1 \otimes B_2 \otimes B_3 + \gamma C_1 \otimes C_2 \otimes C_3, b$
Output: $x$
for $i_1 \leftarrow 1$ to $n_1$ do
    for $i_2 \leftarrow 1$ to $n_2$ do
        for $i_3 \leftarrow 1$ to $n_3$ do
            $i \leftarrow$ multi_index$(i_1, i_2, i_3)$
            $y_i \leftarrow 0$
            $a_d \leftarrow 1$
            for $k_1 \leftarrow A_1.ptr[i_1]$ to $A_1.ptr[i_1+1]-1$ do
                $j_1 \leftarrow A_1$.indices$[k_1]$
                $a_1 \leftarrow A_1$.data$[k_1]$
                for $k_2 \leftarrow A_2.ptr[i_2]$ to $A_2.ptr[i_2+1]-1$ do
                    $j_2 \leftarrow A_2$.indices$[k_2]$
                    $a_2 \leftarrow A_2$.data$[k_2]$
                    for $k_3 \leftarrow A_3.ptr[i_3]$ to $A_3.ptr[i_3+1]-1$ do
                        $j_3 \leftarrow A_3$.indices$[k_3]$
                        $a_3 \leftarrow A_3$.data$[k_3]$
                        $j \leftarrow$ multi_index$(j_1, j_2, j_3)$
                        if $i < j$ then
                            $y_i \leftarrow y_i + a_1 a_2 a_3 x[j]$
                        else
                            $a_d \leftarrow a_1 a_2 a_3$
                        end
                    end
                end
            end
            $z_i \leftarrow 0$
            $b_d \leftarrow 1$
            for $k_1 \leftarrow B_1.ptr[i_1]$ to $B_1.ptr[i_1+1]-1$ do
                $j_1 \leftarrow B_1$.indices$[k_1]$
                $a_1 \leftarrow B_1$.data$[k_1]$
                for $k_2 \leftarrow B_2.ptr[i_2]$ to $B_2.ptr[i_2+1]-1$ do
                    $j_2 \leftarrow B_2$.indices$[k_2]$
                    $a_2 \leftarrow B_2$.data$[k_2]$
                    for $k_3 \leftarrow B_3.ptr[i_3]$ to $B_3.ptr[i_3+1]-1$ do
                        $j_3 \leftarrow B_3$.indices$[k_3]$
                        $a_3 \leftarrow B_3$.data$[k_3]$
                        $j \leftarrow$ multi_index$(j_1, j_2, j_3)$
                        if $i < j$ then
                            $z_i \leftarrow z_i + a_1 a_2 a_3 x[j]$
                        else
                            $b_d \leftarrow a_1 a_2 a_3$
                        end
                    end
                end
            end
            $w_i \leftarrow 0$
            $c_d \leftarrow 1$
            for $k_1 \leftarrow C_1.ptr[i_1]$ to $C_1.ptr[i_1+1]-1$ do
                $j_1 \leftarrow C_1$.indices$[k_1]$
                $a_1 \leftarrow C_1$.data$[k_1]$
                for $k_2 \leftarrow C_2.ptr[i_2]$ to $C_2.ptr[i_2+1]-1$ do
                    $j_2 \leftarrow C_2$.indices$[k_2]$
                    $a_2 \leftarrow C_2$.data$[k_2]$
                    for $k_3 \leftarrow C_3.ptr[i_3]$ to $C_3.ptr[i_3+1]-1$ do
                        $j_3 \leftarrow C_3$.indices$[k_3]$
                        $a_3 \leftarrow C_3$.data$[k_3]$
                        $j \leftarrow$ multi_index$(j_1, j_2, j_3)$
                        if $i < j$ then
                            $w_i \leftarrow w_i + a_1 a_2 a_3 x[j]$
                        else
                            $c_d \leftarrow a_1 a_2 a_3$
                        end
                    end
                end
            end
            $x[i] \leftarrow \frac{1}{\alpha a_d + \beta b_d \gamma c_d}(b[i] - \alpha y_i - \beta z_i - \gamma w_i)$
        end
    end
end
```

Algorithm 10: `spsolve`: Upper triangular solver for sum of Kronecker product [CSR] matrices.

```
Input  : $\mathcal{A} = \alpha A_1 \otimes A_2 \otimes A_3 + \beta B_1 \otimes B_2 \otimes B_3 + \gamma C_1 \otimes C_2 \otimes C_3, b$
Output: $x$
for $i_1 \leftarrow 1$ to $n_1$ do
    for $i_2 \leftarrow 1$ to $n_2$ do
        for $i_3 \leftarrow 1$ to $n_3$ do
            $i \leftarrow$ multi_index$(i_1, i_2, i_3)$
            $y_i \leftarrow 0$
            $a_d \leftarrow 1$
            for $k_1 \leftarrow A_1.ptr[i_1]$ to $A_1.ptr[i_1+1]-1$ do
                $j_1 \leftarrow A_1$.indices$[k_1]$
                $a_1 \leftarrow A_1$.data$[k_1]$
                for $k_2 \leftarrow A_2.ptr[i_2]$ to $A_2.ptr[i_2+1]-1$ do
                    $j_2 \leftarrow A_2$.indices$[k_2]$
                    $a_2 \leftarrow A_2$.data$[k_2]$
                    for $k_3 \leftarrow A_3.ptr[i_3]$ to $A_3.ptr[i_3+1]-1$ do
                        $j_3 \leftarrow A_3$.indices$[k_3]$
                        $a_3 \leftarrow A_3$.data$[k_3]$
                        $j \leftarrow$ multi_index$(j_1, j_2, j_3)$
                        if $i \geq j$ then
                            $y_i \leftarrow y_i + a_1 a_2 a_3 x[j]$
                        end
                        if $i = j$ then
                            $a_d \leftarrow a_1 a_2 a_3$
                        end
                    end
                end
            end
            $z_i \leftarrow 0$
            $b_d \leftarrow 1$
            for $k_1 \leftarrow B_1.ptr[i_1]$ to $B_1.ptr[i_1+1]-1$ do
                $j_1 \leftarrow B_1$.indices$[k_1]$
                $a_1 \leftarrow B_1$.data$[k_1]$
                for $k_2 \leftarrow B_2.ptr[i_2]$ to $B_2.ptr[i_2+1]-1$ do
                    $j_2 \leftarrow B_2$.indices$[k_2]$
                    $a_2 \leftarrow B_2$.data$[k_2]$
                    for $k_3 \leftarrow B_3.ptr[i_3]$ to $B_3.ptr[i_3+1]-1$ do
                        $j_3 \leftarrow B_3$.indices$[k_3]$
                        $a_3 \leftarrow B_3$.data$[k_3]$
                        $j \leftarrow$ multi_index$(j_1, j_2, j_3)$
                        if $i \geq j$ then
                            $z_i \leftarrow z_i + a_1 a_2 a_3 x[j]$
                        end
                        if $i = j$ then
                            $b_d \leftarrow a_1 a_2 a_3$
                        end
                    end
                end
            end
            $w_i \leftarrow 0$
            $c_d \leftarrow 1$
            for $k_1 \leftarrow C_1.ptr[i_1]$ to $C_1.ptr[i_1+1]-1$ do
                $j_1 \leftarrow C_1$.indices$[k_1]$
                $a_1 \leftarrow C_1$.data$[k_1]$
                for $k_2 \leftarrow C_2.ptr[i_2]$ to $C_2.ptr[i_2+1]-1$ do
                    $j_2 \leftarrow C_2$.indices$[k_2]$
                    $a_2 \leftarrow C_2$.data$[k_2]$
                    for $k_3 \leftarrow C_3.ptr[i_3]$ to $C_3.ptr[i_3+1]-1$ do
                        $j_3 \leftarrow C_3$.indices$[k_3]$
                        $a_3 \leftarrow C_3$.data$[k_3]$
                        $j \leftarrow$ multi_index$(j_1, j_2, j_3)$
                        if $i \geq j$ then
                            $w_i \leftarrow w_i + a_1 a_2 a_3 x[j]$
                        end
                        if $i = j$ then
                            $c_d \leftarrow a_1 a_2 a_3$
                        end
                    end
                end
            end
            $x[i] \leftarrow \frac{1}{\alpha a_d + \beta b_d \gamma c_d}(b[i] - \alpha y_i - \beta z_i - \gamma w_i)$
        end
    end
end
```

6.6 Computational Cost

7 Numerical Results