Block recursive LU preconditioners for the thermally coupled incompressible inductionless MHD problem[☆]

Santiago Badia[a,b], Alberto F. Martín[a,b], Ramon Planas[a,b]

[a] *Centre Internacional de Mètodes Numèrics a l'Enginyeria (CIMNE), Parc Mediterrani de la Tecnologia, UPC, Esteve Terradas 5, 08860 Castelldefels, Spain ({sbadia,amartin,rplanas}@cimne.upc.edu).*
[b] *Universitat Politècnica de Catalunya, Jordi Girona 1-3, Edifici C1, 08034 Barcelona, Spain.*

Abstract

The thermally coupled incompressible inductionless magnetohydrodynamics (MHD) problem models the flow of an electrically charged fluid under the influence of an external electromagnetic field with thermal coupling. This system of partial differential equations is strongly coupled and highly nonlinear for real cases of interest. Therefore, fully implicit time integration schemes are very desirable in order to capture the different physical scales of the problem at hand. However, solving the multiphysics linear systems of equations resulting from such algorithms is a very challenging task which requires efficient and scalable preconditioners. In this work, a new family of recursive block *LU* preconditioners is designed and tested for solving the thermally coupled inductionless MHD equations. These preconditioners are obtained after splitting the fully coupled matrix into one-physics problems for every variable (velocity, pressure, current density, electric potential and temperature) that can be optimally solved, e.g., using preconditioned domain decomposition algorithms. The main idea is to arrange the original matrix into an (arbitrary) 2×2 block matrix, and consider a *LU* preconditioner obtained by approximating the corresponding Schur complement. For every one of the diagonal blocks in the *LU* preconditioner, if it involves more than one type of unknown, we proceed the same way in a recursive fashion. This approach is stated in an abstract way, and can be straightforwardly applied to other multiphysics problems. Further, we precisely explain a flexible and general software design for the code implementation of this type of preconditioners.

1. Introduction

The thermally coupled incompressible inductionless magnetohydrodynamics (MHD) model describes the dynamics of an electrically conducting fluid under an external electromagnetic field with thermal coupling where the magnetic field induced by the currents is negligible with respect to the externally applied one. This system of partial differential equations can be applied to simulate a wide range of applications, such as MHD pumps, steel casting processes, crystal growth devices or breeding blankets in nuclear fusion reactors. The Galerkin finite element (FE) approximation of this problem faces several well-known drawbacks. First, convective dominated flows may lead to oscillations because the system loses its elliptic nature. Second, a strong coupling between the two saddle-point subproblems, the hydrodynamic and the magnetic ones, may introduce numerical instabilities when solving the resulting linear systems of equations. Finally, there is the need to satisfy the classical inf-sup conditions between the approximation spaces for the velocity and the pressure and also for the current density and the electric potential in order to have a well-posed problem. There exist several options to circumvent these difficulties being stabilization methods one of the most widely used. In this work, we consider two stabilization techniques based on the variational multiscale ideas in [24, 16, 17].

The discretization of realistic problems with very fine finite element meshes leads to linear systems of equations to be solved with a number of degrees of freedom in the range of $10^6 - 10^9$. For the solution of such huge systems, an efficient and scalable preconditioner is required. There exist several approaches for preconditioning this type of problems. One approach that has been extensively used for preconditioning large-scale multi-physics problems is the algebraic multigrid (AMG) algorithm [26, 34]. This technique is very efficient for Laplacian-type problems but suffers for indefinite and nonsymmetric problems. Another interesting approach consists of an approximate block *LU* factorization of the system matrix. This type of preconditioners have been widely studied for the incompressible

[☆]This work has been funded by the European Research Council under the FP7 Programme Ideas through the Starting Grant No. 258443 - COMFUS: Computational Methods for Fusion Technology. A. F. Martín was also partially funded by the UPC postdoctoral grants under the programme "BKC-5-Atracció i Fidelització de talent al BKC". The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Red Española de Supercomputación in the exploitation of the Marenostrum-3 supercomputer.

Navier-Stokes equations in the fluid mechanics community [22, 21, 18]. Recently, this approach has been applied to the full resistive MHD equations [9] (using the preconditioner as a solver in an operator splitting fashion) and to the 2D incompressible (reduced) resistive MHD formulation [19]. The crucial aspect in these approximate block preconditioners relies in an efficient approximation of the Schur complement that allows the uncoupling between the several physical variables of the problem at the preconditioner level.

These approximate block factorization ideas have been used in this work to design new block recursive *LU* preconditioners for the inductionless MHD and the thermally coupled inductionless MHD problems. As an example, the inductionless MHD system involves four different unknowns (velocity, pressure, current density and electric potential) and leads to a 4×4 block system matrix (one block per unknown). Our approach consists of arranging the multiphysics 4×4 block matrix as a 2×2 block matrix (grouping unknowns) where in turn every block is a 2×2 block matrix. Then, we perform an incomplete *LU* factorization of the 2×2 system block matrix where we consider some (cheap) approximation of the resulting Schur complement and possibly the rest of diagonal blocks. Recursively, since the diagonal blocks are in fact 2×2 block matrices, we approximate these matrices the same way, i.e., using an incomplete *LU* approximation. As a result, the only blocks that have to be *inverted* are one-variable (one-physics) problems. The key point for these preconditioners to be efficient relies in obtaining a good approximation of the Schur complement for all 2×2 systems. In order to define the Schur complement approximations, we have extended ideas from the techniques used for the incompressible Navier-Stokes equations to the inductionless MHD system. Moreover, a study of the exact Schur complement behavior and the effect of cancelling different terms in it has allowed us to propose an improved version of the Pressure-Convection-Diffusion (PCD) preconditioner (see [22]) where we introduce additional stabilization terms. The application of the MHD preconditioner to the thermally coupled problem is straightforward, since the coupling is in one direction only.

The contributions of the article are the following. On one side, we propose new stabilized formulations based on term-by-term projection stabilization for the inductionless MHD problem, and extend this formulation and the one in [32] to the thermally coupled case. The most particular feature of our approach is the explicit introduction of the current density as an additional unknown of the problem. We note that the typical approach is to decouple fluid and electromagnetic problems [12, 28, 31, 30]. Then, the electromagnetic problem is solved in terms of the electric potential only and the current density and Lorentz force are computed. Next, the fluid problem is solved with the previously computed Lorentz force. This approach treats the multiphysics coupling explicitly (for transient problems), and, in the best case, it is only coupled via fixed point iterations (if it converges). However, for large Hartmann numbers (in fusion reaction breeding blanket simulations it is in the order of $10^4 - 10^5$) this approach can only work for extremely small time step values for transient problems and it cannot be used for steady problems. In any case, in our approach the resulting linear systems still require efficient preconditioning for high Hartmann numbers. Thus, we propose an abstract setting to design preconditioners for multiphysics problems, based on a recursive use of block factorization. This general framework is applied to the (thermally coupled) inductionless MHD problem, considering different preconditioners based on approximations of the resulting Schur complement matrices. The efficiency of these preconditioners is assessed via a complete set of numerical experiments. Finally, we give details about an abstract and flexible implementation of block recursive preconditioning. The combination of our FE formulations, with an explicit treatment of the current density, and the recursive *LU* preconditioners we propose, allow us to solve realistic breeding blanket simulations with very high Hartmann numbers.

The article is organized as follows. Section 2 states the problem in both its strong and weak form for the adimensional version of the inductionless MHD equations and the thermally coupled inductionless MHD problem. In Section 3, the stabilization methods used in this work are presented and the block structure of the resulting linear system of equations is highlighted. In Section 4, the underlying recursion in the design of block preconditioners is defined together with a review of classical block preconditioners from fluid mechanics. Moreover, two different recursive block preconditioners are developed for the inductionless MHD problem depending on the grouping of the physical variables. For each preconditioner, the approximation of the Schur complement is chosen. An improved version of PCD block preconditioners is also derived in Section 4 with an experimental justification of its design. To close Section 4, the recursive block preconditioners derived for the inductionless MHD problem are extended for the thermally coupled problem in a straightforward manner. Some numerical experiments to test the properties of the block preconditioners presented in this work are carried out in Section 5. Section 6 deals with some software design and implementation aspects that are crucial to manage the block preconditioners recursion. Finally, some conclusions are drawn in Section 7.

2. Continuous MHD problem

2.1. Inductionless MHD

The incompressible inductionless mangnetohydrodynamics (MHD) system of partial differential equations consists of the incompressible Navier-Stokes problem coupled with Ohm's law and the electric charge conservation equations via the Lorentz force. It reads as: find a velocity field $u(x,t)$, a pressure $p(x,t)$, a current density field $j(x,t)$ and an electric potential $\phi(x,t)$ such that

$$\partial_t u + (u\cdot\nabla)u - \nu\Delta u + \nabla p - \frac{1}{\rho}(j\times B) = f, \tag{1}$$

$$\nabla\cdot u = 0, \tag{2}$$

$$j + \sigma\nabla\phi - \sigma(u\times B) = 0, \tag{3}$$

$$\nabla\cdot j = 0, \tag{4}$$

in $(x,t)\in\Omega\times(0,T)$, where $\Omega\subset\mathbb{R}^d$ is the open bounded domain filled by the fluid and d is the spatial dimension. The partial time derivative is denoted by ∂_t and ν, ρ, σ stand for the viscosity, density and electric conductivity of the fluid, respectively. Finally, f corresponds to the body forces of the flow motion and B to the external magnetic field.

Consider a partition of the domain boundary $\Gamma=\partial\Omega$ into two parts $\Gamma=\Gamma_{E,u}\cup\Gamma_{N,u}$. The boundary conditions for the velocity field are no-slip wall conditions, $u=0$ on $\Gamma_{E,u}$, and zero traction conditions, $-pn+\nu n\cdot\nabla u=0$ on $\Gamma_{N,u}$. Let us define a different partition of the domain boundary for imposing the electromagnetic boundary conditions, $\Gamma=\Gamma_{C,j}\cup\Gamma_{I,j}$, where $\Gamma_{C,j}$ corresponds to perfectly conducting walls and $\Gamma_{I,j}$ to perfectly insulating walls. Perfectly conducting walls do not apply any resistance to the current which implies that the electric currents cross the wall surface orthogonally. This condition is imposed by $j\times n=0$ on $\Gamma_{C,j}$. Taking into account that $u=0$ on $\Gamma_{C,j}$, this condition is equivalent to impose $\phi=0$ on $\Gamma_{C,j}$. On the other hand, insulating walls do not allow the electric currents to cross them, which means that the normal component of the current density field has to vanish, that is, $j\cdot n=0$ on $\Gamma_{I,j}$. Lastly, an initial condition for the velocity field has to be considered, i.e., $u=u_0$ in Ω for $t=0$.

It is convenient for solving real problems with extreme physical properties (as the test blanket module case in Section 5.3) to work with the dimensionless form of the incompressible inductionless MHD system. Let us redefine the problem in (1)-(4) by (see [29])

$$\partial_t u + (u\cdot\nabla)u - \frac{1}{\mathrm{Re}}\Delta u + \nabla p - \mathrm{N}(j\times B) = f, \tag{5}$$

$$\nabla\cdot u = 0, \tag{6}$$

$$j + \nabla\phi - (u\times B) = 0, \tag{7}$$

$$\nabla\cdot j = 0, \tag{8}$$

where the variables and operators have been scaled as

$$\begin{array}{ll} u\to u\frac{1}{u_0}, & \nabla\to\nabla L,\\ t\to t\frac{u_0}{L}, & B\to B\frac{1}{B_0},\\ p\to p\frac{1}{\rho u_0^2}, & f\to f\frac{L}{u_0^2},\\ j\to j\frac{1}{\sigma u_0 B_0}, & \phi\to\phi\frac{1}{u_0 B_0}. \end{array}$$

Here, L is a characteristic length of the domain and u_0, B_0 are the characteristic scales for the velocity and the external magnetic fields, respectively. Let us also introduce the adimensional numbers in system (5)-(8)

$$\text{Reynolds number, } \mathrm{Re} = \frac{u_0 L}{\nu},$$

$$\text{Interaction parameter, } \mathrm{N} = \frac{\sigma L B_0^2}{\rho u_0}.$$

There is also another adimensional number that governs system (5)-(8) behavior, the Hartmann number $\mathrm{Ha}=\sqrt{\mathrm{ReN}}$, that gives the ratio between electromagnetic and viscous forces.

Let us introduce some standard notation here. We denote by $L_0^2(\Omega)$ the set of functions in $L^2(\Omega)$ with zero mean value and by $H^1_{0,\Gamma}(\Omega)^d$ the functions belonging to $H^1(\Omega)$ that vanish on $\Gamma\subset\partial\Omega$. Let us now consider the following

functional spaces: $V_u = H^1_{0,\Gamma_{E,u}}(\Omega)^d$, $Q_u = L^2_0(\Omega)$, $V_j = L^2(\Omega)$ and $Q_\phi = H^1_{0,\Gamma_{C,j}}(\Omega)$. Assuming a smooth external magnetic field B, the weak form of system (5)-(8) reads as follows: find $(u, p, j, \phi) \in V_u \times Q_p \times V_j \times Q_\phi$ such that

$$F_u(u)u + C_u j + G_u p = f_u \quad \text{in} \quad V_u', \qquad D_u u = 0 \quad \text{in} \quad Q_p', \tag{9}$$

$$F_j j + C_j u + G_j \phi = 0 \quad \text{in} \quad V_j', \qquad D_j j = 0 \quad \text{in} \quad Q_\phi', \tag{10}$$

for almost every $t \in (0, T)$. We omit the discussion about the regularity in time of the unknowns for simplicity. The definition of these operators comes from system (5)-(8) and the notion of weak derivatives as follows.

The semi-linear fluid operator $F_u : V_u \times V_u \to V_u'$ is defined as $F_u(w)v := M_u \partial_t v + K_u(w)v$. The fluid mass matrix $M_u : V_u \to V_u'$ is defined as $M_u v := (v, \cdot)$. The semi-linear operator $K_u : V_u \times V_u \to V_u'$ includes the viscous and convective terms and reads as $K_u(w)v := \frac{1}{\mathrm{Re}}(\nabla v, \nabla(\cdot)) + \langle (w \cdot \nabla)v, \cdot \rangle$. $G_u : Q_p \to V_u'$ denotes the (integrated by parts) pressure gradient, i.e., $G_u q := -(q, \nabla \cdot (\cdot))$, and $D_u = -G_u^T$ is the velocity divergence operator; the superscript T indicates the transpose operator. The linear Lorentz force coupling operator $C_u : V_j \to V_u'$ is defined as $C_u k := -\mathrm{N} \langle \cdot, k \times B \rangle$. With regard to the magnetic subproblem, we define the mass matrix operator $F_j : V_j \to V_j'$ as $F_j k := (k, \cdot)$, $C_j := -C_u^T$, the electric potential gradient operator $G_j \psi := (\cdot, \nabla \psi)$ and the corresponding divergence operator $D_j = -G_j^T$. We note that the conditions on $\Gamma_{E,u}$ and $\Gamma_{C,j}$ are strongly enforced whereas those on $\Gamma_{N,u}$ and $\Gamma_{I,j}$ are weakly enforced (see [32] for a more detailed explanation).

2.2. Thermally coupled problem

The strong form of the thermally coupled incompressible inductionless MHD problem is obtained from equations (5)-(8) and Boussinesq approximation for the thermal coupling. It reads as: find a velocity field $u(x,t)$, a pressure $p(x,t)$, a current density field $j(x,t)$, an electric potential $\phi(x,t)$ and a temperature $\theta(x,t)$ such that,

$$\partial_t u + (u \cdot \nabla)u - \frac{1}{\mathrm{Re}} \Delta u + \nabla p - \mathrm{N}(j \times B) + \frac{\mathrm{Gr}}{\mathrm{Re}^2} \theta = f + \frac{\mathrm{Gr}}{\mathrm{Re}^2} \theta_{ref}, \tag{11}$$

$$\nabla \cdot u = 0, \tag{12}$$

$$j + \nabla \phi - (u \times B) = 0, \tag{13}$$

$$\nabla \cdot j = 0, \tag{14}$$

$$\partial_t \theta + (u \cdot \nabla)\theta - \frac{1}{\mathrm{Pe}} \Delta \theta = Q, \tag{15}$$

where the temperature has been scaled as $\theta \to \theta \frac{1}{\Delta \theta}$. The additional adimensional numbers in (11)-(15) are defined as,

$$\text{Grashof number, } \mathrm{Gr} = \frac{g \beta \Delta \theta L^3}{\nu^2},$$

$$\text{Péclet number, } \mathrm{Pe} = \frac{\rho c_p u_0 L}{k_t},$$

where g is the norm of the gravity field, β the thermal expansion coefficient, $\Delta\theta$ a temperature increment, θ_{ref} a reference temperature, Q a heat source, c_p the specific heat at constant pressure, k_t the thermal conductivity and $\kappa = \frac{k_t}{\rho c_p}$ the thermal diffusivity. Moreover, the Prandtl number is an adimensional number that relates the viscous and thermal diffusivities, $\mathrm{Pr} = \frac{\nu}{\kappa}$.

The boundary conditions to be imposed for problem (11)-(15) correspond to the conditions stated in Section 2.1 for the MHD variables plus boundary conditions for the temperature. Consider a partition of the domain boundary such as $\Gamma = \Gamma_{E,\theta} \cup \Gamma_{N,\theta}$. The Dirichlet condition on $\Gamma_{E,\theta}$ implies a fixed temperature $\theta = \theta_D$ whereas the condition on $\Gamma_{N,\theta}$ corresponds to imposing a heat flux on the boundary, i.e., $\frac{k_t}{\rho c_p} n \cdot \nabla \theta = q$. Finally, an initial condition for the temperature field, $\theta = \theta_0$ in Ω for $t = 0$, has to be considered. We assume that $Q = q = \theta_D = 0$ for simplicity.

We define the temperature functional space $V_\theta = H^1_{0,\Gamma_{E,\theta}}(\Omega)$. The weak form of problem (11)-(15) can be stated as: find $(u, p, j, \phi, \theta) \in V_u \times Q_p \times V_j \times Q_\phi \times V_\theta$ such that

$$F_u(u)u + C_u j + G_u p + H_u \theta = f_u \quad \text{in} \quad V_u', \qquad D_u u = 0 \quad \text{in} \quad Q_p', \tag{16}$$

$$F_j j + C_j u + G_j \phi = 0 \quad \text{in} \quad V_j', \qquad D_j j = 0 \quad \text{in} \quad Q_\phi', \tag{17}$$

$$F_\theta(u)\theta = 0 \quad \text{in} \quad V_\theta'. \tag{18}$$

where the operators related to the inductionless MHD problem are defined in Section 2.1. The semi-linear thermal problem operator $F_\theta : V_u \times V_\theta \to V_\theta'$ reads as $F_\theta(w)\varphi := (\partial_t \varphi, \cdot) + \frac{1}{\mathrm{Pe}}(\nabla \varphi, \nabla \cdot) + \langle (w \cdot \nabla)\varphi, \cdot \rangle$, whereas the buoyancy term operator $H_u : V_\theta \to V_u'$ is $H_u \theta := \frac{\mathrm{Gr}}{\mathrm{Re}^2}(\theta, \cdot)$.

3. Stabilized finite element formulation

3.1. Inductionless MHD

The variational problem (9)-(10) is linearized, discretized in time and spatially approximated following the same procedure as in [32]. The linearization method chosen is Picard method, the time derivatives are discretized using the θ-method and the spatial approximation is obtained with the standard Galerkin method. Given, e.g., the operator M_u, we denote its finite element restriction as $M_{uh} : V_{uh} \to V'_{uh}$. This way, we define the FE restriction of all the operators in Section 2.1. However, in order to simplify notation, we will omit the h subindex for the discrete operators. Thus, the discrete and linearized form can be stated as

$$F_u(a_h^{n+\theta})u_h^{n+\theta} + C_u j_h^{n+1} + G_u p_h^{n+1} = f_u^{n+\theta} \quad \text{in} \quad V'_{u,h}, \qquad D_u u_h^{n+\theta} = 0 \quad \text{in} \quad Q'_{ph}, \tag{19}$$

$$F_j j_h^{n+1} + C_j u_h^{n+\theta} + G_j \phi_h^{n+1} = 0 \quad \text{in} \quad V'_{jh}, \qquad D_j j_h^{n+1} = 0 \quad \text{in} \quad Q'_{\phi h}, \tag{20}$$

where $a_h^{n+\theta} = u_h^{n+\theta,k}$ being k the previous iteration of the nonlinear Picard iterative loop and the discrete FE spaces V_{uh}, Q_{ph}, V_{jh} and $Q_{\phi h}$ are subspaces of their infinite dimensional counterparts V_u, Q_p, V_j and Q_ϕ. The right-hand-side (RHS) term $f_u^{n+\theta}$ includes the time derivative term $\frac{1}{\delta t} M_u u_h^n$.

The Galerkin approximation of this problem is known to have many drawbacks. First, the discrete problem is well-posed only if the discrete inf-sup conditions for $V_{uh} \times Q_{ph}$ and $V_{jh} \times Q_{\phi h}$ are satisfied:

$$\inf_{q_h \in Q_{ph}} \sup_{v_h \in V_{uh}} \frac{(q_h, \nabla \cdot v_h)}{\|q_h\| \|\nabla v_h\|} \geq \beta^* > 0, \qquad \inf_{\psi_h \in Q_{\phi h}} \sup_{k_h \in V_{jh}} \frac{(\nabla \psi_h, k_h)}{\|\nabla \psi_h\| \|k_h\|} \geq \gamma^* > 0,$$

where β^* and γ^* are positive constants independent of the mesh size h. Depending on how the finite element spaces V_{uh}, Q_{ph}, V_{jh} and $Q_{\phi h}$ are chosen, these conditions might not be satisfied. For instance, equal order spaces do not fulfill the discrete inf-sup conditions. Moreover, when solving problems where the first order derivatives dominate the second order ones in the Navier-Stokes equations, that is, convection dominated cases, oscillations may appear in the solution. Finally, a strong coupling between the hydrodynamic and electromagnetic problems may lead to numerical instabilities.

The solution adopted in this work to avoid these drawbacks consists of stabilization methods. The basic idea under a stabilization method is to add certain terms to the variational form of the problem that allow one to circumvent the previously mentioned difficulties associated to the Galerkin approximation of the problem without spoiling accuracy. Two different stabilization methods are used in this work.

The first one is the algebraic sub-grid scale method (ASGS) following the subgrid scale concept introduced in [24]. We can consider a variational multiscale formulation of the problem, by using the framework in [24, 23]. Let us denote the finite element partition as $\mathcal{T}_h$; $K \in \mathcal{T}_h$ is a FE. This way, we add a term of the type $\sum_{k \in \mathcal{T}_h} \int_K (F - \partial_t \mathcal{M}(U) - \mathcal{L}(U)) \cdot \boldsymbol{\tau} \mathcal{L}^T(V) \mathrm{d}x$ to the left-hand-side (LHS) of (19)-(20), where F is the forcing term, $\mathcal{L}$ the steady-state spatial differential operator of the problem at hand and $\mathcal{M}$ the continuous mass operator. $\boldsymbol{\tau}$ is the matrix of stabilization parameters. Alternatively, when considering a Galerkin/Least-Squares (GLS) stabilization of the problem, we just replace $\mathcal{L}^T(V)$ by $-\mathcal{L}(V)$ in the definition of the stabilization term. The application of this method to the incompressible inductionless MHD problem is deeply explained in [32]. We include this method in Algorithm 1 for the sake of completeness. The symbol Δ_h stands for the broken Laplacian, i.e., $(\Delta_h v_h, \cdot) = \sum_{K \in \mathcal{T}_h} \int_K \Delta v_h(\cdot) \mathrm{d}x$.

The second stabilization method developed for this work is the orthogonal sub-scale stabilization method (OSS) introduced in [16, 17]. We consider a term-by-term formulation, where we only introduce as stabilization terms the quantities we want to stabilize (first order derivative terms) scaled with properly chosen stabilization parameters. These terms alone would destroy the convergence properties of the resulting method. In order to have optimal convergence, we subtract to the quantities to be stabilized a proper projection onto the FE space. The OSS stabilization terms to be added to the Galerkin formulation read as (before linearization)

$$\begin{aligned}
&(\tau_1 \pi^\perp_{hu}((u_h \cdot \nabla)u_h), \pi^\perp_{hu}((u_h \cdot \nabla)v_h)) + (\tau_1 \pi^\perp_{hu}(\nabla p_h), \pi^\perp_{hu}(\nabla q_h)) + (\tau_1 \pi^\perp_{hu}(j_h \times B), \pi^\perp_{hu}(k_h \times B)) \\
&+ (\tau_2 \nabla \cdot u_h, \nabla \cdot v_h) + (\tau_3 \pi^\perp_{hj}(\nabla \phi), \pi^\perp_{hj}(\nabla \psi)) + (\tau_3 \pi^\perp_{hj}(u_h \times B), \pi^\perp_{hj}(v_h \times B)) + (\tau_4 \nabla \cdot j_h, \nabla \cdot k_h),
\end{aligned}$$

where $\pi^\perp_{hu}(\cdot) = \mathrm{Id}(\cdot) - \pi_{hu}(\cdot)$; Id denotes the identity and $\pi_{hu} : L^2(\Omega) \to V_{uh}$ corresponds to a projector onto the FE space. $\pi^\perp_{hj}(\cdot)$ is defined analogously with respect to V_{jh}. We note that the projections are not required for the divergence terms, since both u and j are solenoidal. Different choices for the projector have been proposed so far, e.g., the orthogonal subscales (OSS) formulation considers the L^2 projector and a local nodal Scott-Zhang projector is used in [3]; other local projection stabilization methods can be found in [10, 27]. In this work, we consider the OSS formulation. In practice, the projection in the OSS method is treated explicitly, e.g.,

$$(\tau_1 \pi^\perp_{hu}(\nabla p_h), \pi^\perp_{hu}(\nabla q_h)) \approx (\tau_1 \nabla p_h, \nabla q_h) - (\tau_1 \pi_{hu}(\nabla p_h), \nabla q_h),$$

Algorithm 1: ASGS stabilization for the inductionless MHD problem

Given u_h^n at the previous time step value, find u_h^{n+1}, j_h^{n+1}, p_h^{n+1} and ϕ_h^{n+1} such that

$$
\begin{aligned}
&\left(\delta_t u_h^{n+1,k+1}, v_h\right) + \left\langle (u_h^{n+1,k}\cdot\nabla)u_h^{n+1,k+1}, v_h\right\rangle + \tfrac{1}{\mathrm{Re}}\left(\nabla u_h^{n+1,k+1}, \nabla v_h\right) - \left(p_h^{n+1,k+1}, \nabla\cdot v_h\right)\\
&\quad -\mathrm{N}\left\langle j_h^{n+1,k+1}\times B, v_h\right\rangle + \left\langle (u_h^{n+1,k}\cdot\nabla)v_h + \tfrac{1}{\mathrm{Re}}\Delta_h v_h, \tau_1^{n+1,k}\boldsymbol{R}_{h,u}^{n+1,k+1}\right\rangle\\
&\quad + \left(\nabla\cdot v_h, \tau_2^{n+1,k}R_{h,p}^{n+1,k+1}\right) - \left\langle (v_h\times B), \tau_3^{n+1,k}\boldsymbol{R}_{h,j}^{n+1,k+1}\right\rangle = \left(f^{n+1}, v_h\right),\\
&\left(\nabla\cdot u_h^{n+1,k+1}, q_h\right) + \left\langle \tau_1^{n+1,k}\boldsymbol{R}_{h,u}^{n+1,k+1}, \nabla q_h\right\rangle = 0,\\
&\left(j_h^{n+1,k+1}, k_h\right) + \left(\nabla\phi_h^{n+1,k+1}, k_h\right) - \left\langle u_h^{n+1,k+1}\times B, k_h\right\rangle\\
&\quad -\mathrm{N}\left\langle k_h\times B, \tau_1^{n+1,k}\boldsymbol{R}_{h,u}^{n+1,k+1}\right\rangle - \left\langle k_h, \tau_3^{n+1,k}\boldsymbol{R}_{h,j}^{n+1,k+1}\right\rangle\\
&\quad + \left(\nabla\cdot k_h, \tau_4^{n+1,k}R_{h,\phi}^{n+1,k+1}\right) = 0,\\
&-\left(j_h^{n+1,k+1}, \nabla\psi_h\right) + \left\langle \nabla\psi_h, \tau_3^{n+1,k}\boldsymbol{R}_{h,j}^{n+1,k+1}\right\rangle = 0,
\end{aligned}
$$

where the residuals are:

$$
\begin{aligned}
\boldsymbol{R}_{h,u} &:= \delta_t u_h + (a\cdot\nabla)u_h - \tfrac{1}{\mathrm{Re}}\Delta_h u_h + \nabla p_h - \mathrm{N}(j_h\times B) - f,\\
R_{h,p} &:= \nabla\cdot u_h,\\
\boldsymbol{R}_{h,j} &:= j_h + \nabla\phi_h - (u_h\times B),\\
R_{h,\phi} &:= \nabla\cdot j_h.
\end{aligned}
$$

The stabilization parameters have the following expressions within each element K:

$$
\begin{aligned}
&\alpha := c_1\tfrac{a}{h} + c_2\tfrac{1}{h^2\mathrm{Re}}, \quad \beta := c_3\mathrm{N}B, \quad \gamma := c_4,\\
&\tau_1 = \alpha^{-1}\left(1 + \tfrac{1}{\sqrt{\alpha\gamma}}\beta\right)^{-1}, \quad \tau_2 = c_5\tfrac{h^2}{\tau_1},\\
&\tau_3 = \gamma^{-1}\left(1 + \tfrac{1}{\sqrt{\alpha\gamma}}\beta\right)^{-1}, \quad \tau_4 = c_6\tfrac{h^2}{\tau_3},
\end{aligned}
$$

where $c_1,\ldots,c_6$ are algorithmic constants.

where the last term is treated explicitly, using the value from the previous nonlinear iteration (idem for the rest of terms). We note that the approximation comes from the fact that τ_1 is not constant in general (otherwise the previous re-statement will be exact). In order to make this relation hold for non-constant τ, we can use a τ-weighted L^2 projector (see [17]). With all these ingredients, we end up with the OSS formulation included in Algorithm 2.

Both ASGS and OSS methods can be stated as

$$
F_u(a_h^{n+\theta})u_h^{n+\theta} + C_u j_h^{n+1} + G_u p_h^{n+1} + T_u\phi_h^{n+1} = f_u^{n+\theta}, \qquad D_u u_h^{n+\theta} + C_p p_h^{n+1} + T_p j_h^{n+1} = f_p^{n+1}, \tag{21}
$$

$$
F_j j_h^{n+1} + C_j u_h^{n+\theta} + G_j\phi_h^{n+1} + T_j p_h^{n+1} = f_j^{n+1}, \qquad D_j j_h^{n+1} + C_\phi\phi_h^{n+1} + T_\phi u_h^{n+1} = f_\phi^{n+1}, \tag{22}
$$

where F_u, C_u, F_j and C_j have been properly modified (with respect to the original Galerkin formulation) in order to include the corresponding stabilization terms in Algorithms 1 or 2. Note that for the ASGS formulation, the right-hand-side terms f_p, f_j and f_ϕ are zero (for OSS, they include the projection treated explicitly) whereas for the OSS formulation, the operators T_u, T_p, T_j and T_ϕ are zero because there are no stabilization terms that couple u-ϕ and j-p. The discrete and stabilized formulation (21)-(22) results in a linear system of equations to be solved. This system of equations has a 4×4 block structure, where we consider one block per unknown:

$$
\begin{bmatrix} F_u & G_u & C_u & T_u\\ D_u & C_p & T_p & 0\\ C_j & T_j & F_j & G_j\\ T_\phi & 0 & D_j & C_\phi \end{bmatrix}\begin{bmatrix} u\\ p\\ j\\ \phi\end{bmatrix} = \begin{bmatrix} f_u\\ f_p\\ f_j\\ f_\phi\end{bmatrix}. \tag{23}
$$

We further note that the OSS algorithm does not modify the off-diagonal terms, i.e., G_u, C_u, D_u, G_j, C_j, and D_j, keeping the block skew-symmetric nature of the Galerkin matrix. However, using ASGS these off-diagonal terms are perturbed in such a way that this property is lost. It has important effects as segregated algorithms are not unconditionally stable for non skew-symmetric matrices (see [9]). Finally, note that for the sake of conciseness, Algorithms 1 and 2 have been written using the Backward Euler method ($\theta = 1$) for time integration.

Algorithm 2: OSS stabilization for the inductionless MHD problem

Given u_h^n at the previous time step value, find u_h^{n+1}, j_h^{n+1}, p_h^{n+1} and ϕ_h^{n+1} such that

$$
\begin{aligned}
&\left(\delta_t u_h^{n+1,k+1}, v_h\right) + \left\langle (u_h^{n+1,k}\cdot\nabla)u_h^{n+1,k+1}, v_h\right\rangle + \tfrac{1}{\mathrm{Re}}\left(\nabla u_h^{n+1,k+1}, \nabla v_h\right) - \left(p_h^{n+1,k+1}, \nabla\cdot v_h\right)\\
&\quad -\mathrm{N}\left\langle j_h^{n+1,k+1}\times B, v_h\right\rangle + \left\langle (u_h^{n+1,k}\cdot\nabla)u_h^{n+1,k+1}, \tau_1^{n+1,k}(u_h^{n+1,k}\cdot\nabla)v_h\right\rangle + \left\langle u_h^{n+1,k+1}\times B, \tau_3^{n+1,k}(v_h\times B)\right\rangle\\
&\quad + \left(\nabla\cdot u_h^{n+1,k+1}, \tau_2^{n+1,k}\nabla\cdot v_h\right)\\
&\quad = \left(f^{n+1}, v_h\right) + \left\langle x_{1,h}^{n+1,k}, \tau_1^{n+1,k}(u_h^{n+1,k}\cdot\nabla)v_h\right\rangle - \left\langle y_{2,h}^{n+1,k}, \tau_3^{n+1,k}(v_h\times B)\right\rangle,\\
&\left(\nabla\cdot u_h^{n+1,k+1}, q_h\right) + \left(\nabla p_h^{n+1,k+1}, \tau_1^{n+1,k}\nabla q_h\right)\\
&\quad = \left\langle x_{2,h}^{n+1,k}, \tau_1^{n+1,k}\nabla q_h\right\rangle,\\
&\left(j_h^{n+1,k+1}, k_h\right) + \left(\nabla\phi_h^{n+1,k+1}, k_h\right) - \left\langle u_h^{n+1,k+1}\times B, k_h\right\rangle + \mathrm{N}^2\left\langle j_h^{n+1,k+1}\times B, \tau_1^{n+1,k}(k_h\times B)\right\rangle\\
&\quad + \left(\nabla\cdot j_h^{n+1,k+1}, \tau_4^{n+1,k}\nabla\cdot k_h\right) = -\mathrm{N}\left\langle x_{3,h}^{n+1,k}, \tau_1^{n+1,k}(k_h\times B)\right\rangle,\\
&-\left(j_h^{n+1,k+1}, \nabla\psi_h\right) + \left(\nabla\phi_h^{n+1,k+1}, \tau_3^{n+1,k}\nabla\psi_h\right)\\
&\quad = \left(y_{1,h}^{n+1,k}, \tau_3^{n+1,k}\nabla\psi_h\right),
\end{aligned}
$$

where the projections are computed from:

$$
\begin{aligned}
\left(x_{1,h}^{n+1,k}, v_h\right) &= \left\langle (u_h^{n+1,k}\cdot\nabla)u_h^{n+1,k}, v_h\right\rangle,\\
\left(x_{2,h}^{n+1,k}, v_h\right) &= \left(\nabla p_h^{n+1,k}, v_h\right),\\
\left(x_{3,h}^{n+1,k}, v_h\right) &= -\mathrm{N}\left\langle j_h^{n+1,k}\times B, v_h\right\rangle,\\
\left(y_{1,h}^{n+1,k}, k_h\right) &= \left(\nabla\phi_h^{n+1,k}, k_h\right),\\
\left(y_{2,h}^{n+1,k}, k_h\right) &= -\left\langle u_h^{n+1,k}\times B, k_h\right\rangle.
\end{aligned}
$$

The stabilization parameters have the following expressions within each element K:

$$
\begin{aligned}
&\alpha := c_1\tfrac{a}{h} + c_2\tfrac{1}{h^2\mathrm{Re}}, \quad \beta := c_3\mathrm{N}B, \quad \gamma := c_4,\\
&\tau_1 = \alpha^{-1}\left(1 + \tfrac{1}{\sqrt{\alpha\gamma}}\beta\right)^{-1}, \quad \tau_2 = c_5\tfrac{h^2}{\tau_1},\\
&\tau_3 = \gamma^{-1}\left(1 + \tfrac{1}{\sqrt{\alpha\gamma}}\beta\right)^{-1}, \quad \tau_4 = c_6\tfrac{h^2}{\tau_3},
\end{aligned}
$$

where $c_1, \ldots, c_6$ are algorithmic constants.

3.2. Thermally coupled problem

The weak form of the thermally coupled inductionless MHD problem (16)-(18) is linearized and discretized in both time and space following the same ideas exposed in Section 3.1. The only additional nonlinear term corresponds to the convective term in the temperature equation. This term is also linearized using Picard method, which leads to the discrete and linearized form,

$$
F_u(a_h^{n+\theta})u_h^{n+\theta} + C_u j_h^{n+1} + G_u p_h^{n+1} + H_u\theta_h^{n+\theta} = f_u^{n+\theta} \quad \text{in} \quad V'_{uh}, \qquad D_u u_h^{n+\theta} = 0 \quad \text{in} \quad Q'_{ph}, \tag{24}
$$

$$
F_j j_h^{n+1} + C_j u_h^{n+\theta} + G_j\phi_h^{n+1} = 0 \quad \text{in} \quad V'_{jh}, \qquad D_j j_h^{n+1} = 0 \quad \text{in} \quad Q'_{\phi h}, \tag{25}
$$

$$
F_\theta(a_h^{n+\theta})\theta_h^{n+\theta} = 0 \quad \text{in} \quad V'_{\theta h}, \tag{26}
$$

where the discrete FE space $V_{\theta h}$ is a subspace of V_θ.

The Galerkin approximation (24)-(26) of the problem adds another source of instability, i.e., the presence of the convective term in the temperature equation. This term may introduce oscillations in the solution for convection dominated cases. Thus, we include a SUPG-type stabilization for the thermal problem in the ASGS formulation for the thermally coupled MHD system, i.e.,

$$
(\tau_5(u_h\cdot\nabla)\varphi_h, \delta_t\theta_h + (a\cdot\nabla)\theta_h - \frac{1}{\mathrm{Pe}}\Delta_h\theta_h - Q).
$$

The thermal coupling in system (24)-(26), after Picard's linearization, is in one direction only; the thermal subproblem is independent of the fluid subproblem. Using a full ASGS formulation, this very interesting property would be lost. The final system with the definition of the stabilization parameters is included in Algorithm 3.

Algorithm 3: ASGS stabilization for the thermally coupled inductionless MHD problem

Given u_h^n at the previous time step value, find u_h^{n+1}, j_h^{n+1}, p_h^{n+1} and ϕ_h^{n+1} such that

$$
\begin{aligned}
&\left(\delta_t u_h^{n+1,k+1}, v_h\right) + \left\langle (u_h^{n+1,k}\cdot\nabla)u_h^{n+1,k+1}, v_h\right\rangle + \tfrac{1}{\mathrm{Re}}\left(\nabla u_h^{n+1,k+1}, \nabla v_h\right) - \left(p_h^{n+1,k+1}, \nabla\cdot v_h\right)\\
&\quad -\mathrm{N}\left\langle j_h^{n+1,k+1}\times B, v_h\right\rangle + \tfrac{\mathrm{Gr}}{\mathrm{Re}^2}\left(\theta_h^{n+1,k+1}, v_h\right) + \left\langle (u_h^{n+1,k}\cdot\nabla)v_h + \tfrac{1}{\mathrm{Re}}\Delta_h v_h, \tau_1^{n+1,k}\boldsymbol{R}_{h,u}^{n+1,k+1}\right\rangle\\
&\quad + \left(\nabla\cdot v_h, \tau_2^{n+1,k}R_{h,p}^{n+1,k+1}\right) - \left\langle (v_h\times B), \tau_3^{n+1,k}\boldsymbol{R}_{h,j}^{n+1,k+1}\right\rangle = \left(f^{n+1}, v_h\right) + \tfrac{\mathrm{Gr}}{\mathrm{Re}^2}\left(\theta_{ref}, v_h\right),\\
&\left(\nabla\cdot u_h^{n+1,k+1}, q_h\right) + \left\langle \tau_1^{n+1,k}\boldsymbol{R}_{h,u}^{n+1,k+1}, \nabla q_h\right\rangle = 0,\\
&\left(j_h^{n+1,k+1}, k_h\right) + \left(\nabla\phi_h^{n+1,k+1}, k_h\right) - \left\langle u_h^{n+1,k+1}\times B, k_h\right\rangle\\
&\quad -\mathrm{N}\left\langle k_h\times B, \tau_1^{n+1,k}\boldsymbol{R}_{h,u}^{n+1,k+1}\right\rangle - \left\langle k_h, \tau_3^{n+1,k}\boldsymbol{R}_{h,j}^{n+1,k+1}\right\rangle + \left(\nabla\cdot k_h, \tau_4^{n+1,k}R_{h,\phi}^{n+1,k+1}\right) = 0,\\
&-\left(j_h^{n+1,k+1}, \nabla\psi_h\right) + \left\langle \nabla\psi_h, \tau_3^{n+1,k}\boldsymbol{R}_{h,j}^{n+1,k+1}\right\rangle = 0,\\
&\left(\delta_t\theta_h^{n+1,k+1}, \varphi_h\right) + \left\langle (u_h^{n+1,k}\cdot\nabla)\theta_h^{n+1,k+1}, \varphi_h\right\rangle + \tfrac{1}{\mathrm{Pe}}\left(\nabla\theta_h^{n+1,k+1}, \nabla\varphi_h\right)\\
&\quad + \left\langle (u_h^{n+1,k}\cdot\nabla)\varphi_h, \tau_5^{n+1,k}\boldsymbol{R}_{h,\theta}^{n+1,k+1}\right\rangle = \left(Q^{n+1}, \varphi_h\right).
\end{aligned}
$$

where the residuals are:

$$
\begin{aligned}
\boldsymbol{R}_{h,u} &:= \delta_t u_h + (a\cdot\nabla)u_h - \tfrac{1}{\mathrm{Re}}\Delta_h u_h + \nabla p_h - \mathrm{N}(j_h\times B) + \tfrac{\mathrm{Gr}}{\mathrm{Re}^2}\theta_h - f,\\
R_{h,p} &:= \nabla\cdot u_h,\\
\boldsymbol{R}_{h,j} &:= j_h + \nabla\phi_h - (u_h\times B),\\
R_{h,\phi} &:= \nabla\cdot j_h,\\
R_{h,\theta} &:= \delta_t\theta_h + (a\cdot\nabla)\theta_h - \tfrac{1}{\mathrm{Pe}}\Delta_h\theta_h - Q.
\end{aligned}
$$

The stabilization parameters have the following expressions within each element K:

$$
\begin{aligned}
&\alpha := c_1\tfrac{a}{h} + c_2\tfrac{1}{h^2\mathrm{Re}}, \quad \beta := c_3\mathrm{N}B, \quad \gamma := c_4,\\
&\tau_1 = \alpha^{-1}\left(1 + \tfrac{1}{\sqrt{\alpha\gamma}}\beta\right)^{-1}, \quad \tau_2 = c_5\tfrac{h^2}{\tau_1},\\
&\tau_3 = \gamma^{-1}\left(1 + \tfrac{1}{\sqrt{\alpha\gamma}}\beta\right)^{-1}, \quad \tau_4 = c_6\tfrac{h^2}{\tau_3},\\
&\tau_5 = \left(c_7\tfrac{a}{h} + c_8\tfrac{1}{h^2\mathrm{Pe}}\right)^{-1},
\end{aligned}
$$

where $c_1,\ldots,c_8$ are algorithmic constants.

On the other hand, we can also use the OSS technique explained in the previous section to the thermally coupled system (24)-(26). In this case, we simply need to add the term

$$(\tau_5\pi_{h\theta}^{\perp}((u_h\cdot\nabla)\phi_h), \pi_{h\theta}^{\perp}((u_h\cdot\nabla)\psi_h)),$$

where $\pi_{h\theta}^{\perp}$ is the L^2-projection onto the FE space $V_{\theta h}$. The resulting OSS algorithm, after time integration and linearization, is stated in Algorithm 4.

The addition of the new variational forms due to the thermal coupling and its associated stabilization terms can be stated compactly as the following 5×5 block linear system of equations,

$$
\begin{bmatrix}
F_u & G_u & C_u & T_u & H_u\\
D_u & C_p & T_p & 0 & H_p\\
C_j & T_j & F_j & G_j & H_j\\
T_\phi & 0 & D_j & C_\phi & 0\\
0 & 0 & 0 & 0 & F_\theta
\end{bmatrix}
\begin{bmatrix} u\\ p\\ j\\ \phi\\ \theta \end{bmatrix}
=
\begin{bmatrix} f_u\\ f_p\\ f_j\\ f_\phi\\ f_\theta \end{bmatrix}, \tag{27}
$$

where the matrices have been modified accordingly to include the stabilization terms. Note that the operators H_p and H_j are zero for the OSS stabilized formulation because there are not coupling terms between p-θ and j-θ.

4. Definition of block recursive preconditioners for the (thermally coupled) inductionless MHD problem

In this section, we design block preconditioners for multiphysics problems based on a recursive use of inexact block LU factorization. This strategy is first presented in a general (abstract) form. The key ingredient of this formulation is the approximation of the Schur complements. Next, we list typical Schur complement approximations for the Navier-Stokes equations. Finally, we apply the abstract setting for the (thermally coupled) inductionless MHD problem.

Algorithm 4: OSS stabilization for the thermally coupled inductionless MHD problem

Given u_h^n at the previous time step value, find u_h^{n+1}, j_h^{n+1}, p_h^{n+1} and ϕ_h^{n+1} such that

$$
\begin{aligned}
&\left(\delta_t u_h^{n+1,k+1}, v_h\right) + \left\langle (u_h^{n+1,k}\cdot\nabla)u_h^{n+1,k+1}, v_h\right\rangle + \tfrac{1}{\mathrm{Re}}\left(\nabla u_h^{n+1,k+1}, \nabla v_h\right) - \left(p_h^{n+1,k+1}, \nabla\cdot v_h\right)\\
&\quad -\mathrm{N}\left\langle j_h^{n+1,k+1}\times B, v_h\right\rangle + \tfrac{\mathrm{Gr}}{\mathrm{Re}^2}\left(\theta_h^{n+1,k+1}, v_h\right)\\
&\quad + \left\langle (u_h^{n+1,k}\cdot\nabla)u_h, \tau_1^{n+1,k}(u_h^{n+1,k}\cdot\nabla)v_h\right\rangle + \left\langle u_h^{n+1,k+1}\times B, \tau_3^{n+1,k}(v_h\times B)\right\rangle + \left(\nabla\cdot u_h^{n+1,k+1}, \tau_2^{n+1,k}\nabla\cdot v_h\right)\\
&\quad = \left(f^{n+1}, v_h\right) + \left\langle x_{1,h}^{n+1,k}, \tau_1^{n+1,k}(u_h^{n+1,k}\cdot\nabla)v_h\right\rangle - \left\langle y_{2,h}^{n+1,k}, \tau_3^{n+1,k}(v_h\times B)\right\rangle,\\
&\left(\nabla\cdot u_h^{n+1,k+1}, q_h\right) + \left(\nabla p_h^{n+1,k+1}, \tau_1^{n+1,k}\nabla q_h\right)\\
&\quad = \left(x_{2,h}^{n+1,k}, \tau_1^{n+1,k}\nabla q_h\right),\\
&\left(j_h^{n+1,k+1}, k_h\right) + \left(\nabla\phi_h^{n+1,k+1}, k_h\right) - \left\langle u_h^{n+1,k+1}\times B, k_h\right\rangle + \mathrm{N}^2\left\langle j_h^{n+1,k+1}\times B, \tau_1^{n+1,k}(k_h\times B)\right\rangle\\
&\quad + \left(\nabla\cdot j_h^{n+1,k+1}, \tau_4^{n+1,k}\nabla\cdot k_h\right) = -\mathrm{N}\left\langle x_{3,h}^{n+1,k}, \tau_1^{n+1,k}(k_h\times B)\right\rangle,\\
&-\left(j_h^{n+1,k+1}, \nabla\psi_h\right) + \left(\nabla\phi_h^{n+1,k+1}, \tau_3^{n+1,k}\nabla\psi_h\right)\\
&\quad = \left(y_{1,h}^{n+1,k}, \tau_3^{n+1,k}\nabla\psi_h\right),\\
&\left(\delta_t\theta_h^{n+1,k+1}, \varphi_h\right) + \left\langle (u_h^{n+1,k}\cdot\nabla)\theta_h^{n+1,k+1}, \varphi_h\right\rangle + \tfrac{1}{\mathrm{Pe}}\left(\nabla\theta_h^{n+1,k+1}, \nabla\varphi_h\right) + \left\langle (u_h^{n+1,k}\cdot\nabla)\theta_h, \tau_5^{n+1,k}(u_h^{n+1,k}\cdot\nabla)\varphi_h\right\rangle\\
&\quad = (Q^{n+1}, \varphi_h) + \left\langle z_{1,h}^{n+1,k}, \tau_5^{n+1,k}(u_h^{n+1,k}\cdot\nabla)\varphi_h\right\rangle,
\end{aligned}
$$

where the projections are computed from:

$$
\begin{aligned}
\left(x_{1,h}^{n+1,k}, v_h\right) &= \left\langle (u_h^{n+1,k}\cdot\nabla)u_h^{n+1,k}, v_h\right\rangle,\\
\left(x_{2,h}^{n+1,k}, v_h\right) &= \left(\nabla p_h^{n+1,k}, v_h\right),\\
\left(x_{3,h}^{n+1,k}, v_h\right) &= -\mathrm{N}\left\langle j_h^{n+1,k}\times B, v_h\right\rangle,\\
\left(y_{1,h}^{n+1,k}, k_h\right) &= \left(\nabla\phi_h^{n+1,k}, k_h\right),\\
\left(y_{2,h}^{n+1,k}, k_h\right) &= -\left\langle u_h^{n+1,k}\times B, k_h\right\rangle,\\
\left(z_{1,h}^{n+1,k}, \varphi_h\right) &= \left\langle (u_h^{n+1,k}\cdot\nabla)\theta_h^{n+1,k}, \varphi_h\right\rangle.
\end{aligned}
$$

The stabilization parameters have the following expressions within each element K:

$$
\begin{aligned}
&\alpha := c_1\tfrac{a}{h} + c_2\tfrac{1}{h^2\mathrm{Re}}, \quad \beta := c_3\mathrm{N}B, \quad \gamma := c_4,\\
&\tau_1 = \alpha^{-1}\left(1 + \tfrac{1}{\sqrt{\alpha\gamma}}\beta\right)^{-1}, \quad \tau_2 = c_5\tfrac{h^2}{\tau_1},\\
&\tau_3 = \gamma^{-1}\left(1 + \tfrac{1}{\sqrt{\alpha\gamma}}\beta\right)^{-1}, \quad \tau_4 = c_6\tfrac{h^2}{\tau_3},\\
&\tau_5 = \left(c_7\tfrac{a}{h} + c_8\tfrac{1}{h^2\mathrm{Pe}}\right)^{-1},
\end{aligned}
$$

where $c_1, \ldots, c_8$ are algorithmic constants.

4.1. Abstract block recursive factorization

Let us consider a generic system

$$
\begin{bmatrix} A_{11} & \cdots & A_{1n_{\mathrm{unk}}} \\ \vdots & \ddots & \vdots \\ A_{n_{\mathrm{unk}}1} & \cdots & A_{n_{\mathrm{unk}}n_{\mathrm{unk}}} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n_{\mathrm{unk}}} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{n_{\mathrm{unk}}} \end{bmatrix}
$$

arising from the discretization of a multiphysics problem that involves n_{unk} physical variables. Our target is to design an efficient preconditioner for the system matrix A such that it only involves the solution of one-variable (one-physics) problems, for which we can find efficient preconditioners of domain decomposition or algebraic multigrid type. In order to do this, we rely on the incomplete block factorization of a 2×2 block matrix. Obviously, the original multiphysics problem can be arranged as a 2×2 system matrix by splitting (and reordering) the n_{unk} variables into two different ordered sets. After this, the original problem is denoted as:

$$
\begin{bmatrix} F & G \\ D & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}.
$$

We have denoted the block matrices and vectors in the arranged 2×2 block system using the typical notation for the incompressible Navier-Stokes equations. (We do not assume that $D = -G^T$ since it is not true in general, e.g., when solving transient incompressible flows with SUPG-type stabilization techniques.) For saddle-point problems, y usually is a Lagrange multiplier, e.g., the pressure. At this point, we can consider an *exact* block LU factorization of the 2×2 block matrix:

$$A = \begin{bmatrix} F & G \\ D & C \end{bmatrix} = \begin{bmatrix} F & 0 \\ D & S \end{bmatrix} \begin{bmatrix} I & F^{-1}G \\ 0 & I \end{bmatrix},$$

where $S = C - DF^{-1}G$ is the Schur complement matrix with respect to y. This matrix cannot be easily handled. In order to obtain an inexact factorization, the key aspect is the design of a good approximation for the Schur complement matrix. We denote this approximation by $S_\sharp$. We can further consider an approximation of F, which we denote by $F_\sharp$, even though it is not essential in many cases. With these two ingredients, namely $S_\sharp$ and $F_\sharp$, we can consider different preconditioners $P(A)$ for A:

$$D - \text{preconditioner}: \qquad P(A) = \begin{bmatrix} F_\sharp & 0 \\ 0 & S_\sharp \end{bmatrix}^{-1} = \begin{bmatrix} F_\sharp^{-1} & 0 \\ 0 & S_\sharp^{-1} \end{bmatrix}, \tag{28}$$

$$U - \text{preconditioner}: \qquad P(A) = \begin{bmatrix} F_\sharp & G \\ 0 & S_\sharp \end{bmatrix}^{-1} = \begin{bmatrix} F_\sharp^{-1} & -F_\sharp^{-1}GS_\sharp^{-1} \\ 0 & S_\sharp^{-1} \end{bmatrix}, \tag{29}$$

$$LU - \text{preconditioner}: \quad P(A) = \begin{bmatrix} I & F_\sharp^{-1}G \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} F & 0 \\ D & S_\sharp \end{bmatrix}^{-1} = \begin{bmatrix} I & -F_\sharp^{-1}G \\ 0 & I \end{bmatrix} \begin{bmatrix} F^{-1} & 0 \\ -S_\sharp^{-1}DF^{-1} & S_\sharp^{-1} \end{bmatrix}. \tag{30}$$

Since we aim at solving the global problem using a preconditioned Krylov iterative solver, we only require to perform matrix-vector multiplications for both A and the preconditioner $P(A)$. The preconditioner $P(A)$ is defined by the inverses of the matrices on the diagonal, i.e., $F_\sharp^{-1}$, $S_\sharp^{-1}$, and possibly F^{-1}. For practical problems, the computation of the action of $F_\sharp^{-1}$ or $S_\sharp^{-1}$ on a vector is not viable via sparse direct solvers, specially in 3D, due to their high memory and computational demands. We use the following notation: given a matrix H, the approximate action of H^{-1} over a vector computed by a Krylov iterative solver preconditioned with $P(H)$ up to a given tolerance tol_H is denoted by $\texttt{precond\_Krylov}(H, P(H), \text{tol}_H)$. Thus, in practical implementations, we replace $F_\sharp^{-1}$ by $\texttt{precond\_Krylov}(F_\sharp, P(F_\sharp), \text{tol})$ in the definition (28),(29) or (30) of the preconditioner (analogously for $S_\sharp^{-1}$ and possibly F^{-1}).

In this setting, with a particular definition of the block matrix and its approximations, we can recover most of the Schur complement preconditioners in the literature (see Section 4.2). For one-physics problems with a saddle-point structure (e.g., the incompressible (Navier)-Stokes equations, electromagnetics with Lorentz gauge, mixed form of Laplacian-type problems, Darcy's law for flow in porous media) matrices $F_\sharp$ and $S_\sharp$ involve one-variable problems (for the field and the Lagrange multiplier respectively) but this is not the general case for multiphysics problems. However, if, e.g., $F_\sharp$ involves two or more variables, we can perform again an incomplete block factorization of this matrix, i.e., approximate $F_\sharp^{-1}$ by $P(F_\sharp)$ with one of the definitions in (28),(29) or (30) (analogously for $S_\sharp^{-1}$ and F^{-1}). This process can be applied recursively till all diagonal block matrices (to be *inverted*) only involve one variable. We state in Algorithm 5 the definition of the *recursive block LU preconditioner*. (We have assumed in this algorithm that the system has been ordered in such a way that the Schur complement is always defined for the second block unknown.) Let us note that it is not required to end the process when the diagonal system matrices are one-variable matrices, as soon as we have at our disposal an efficient preconditioner for a particular multi-variable matrix.

As an alternative, we can also consider only the LU factorization at the first level, and solve every multi-variable diagonal block using a Krylov iterative solver preconditioned with an incomplete inexact LU factorization. Again, we can proceed in a recursive way. As a result, this procedure will involve as many nested iterative loops as levels of recursion. The resulting algorithm is presented in Algorithm 6. Both approaches will be considered in the numerical experiments section.

For a particular multiphysics problem the key ingredient to be defined is an efficient and robust approximation of $S_\sharp$ for every incomplete LU factorization.

4.2. Incompressible Navier-Stokes preconditioners

Let us review some of the classical block preconditioners for solving the incompressible Navier-Stokes problem. Consider that the linear system of equations to be solved after discretization and stabilization of the Navier-Stokes equations is written as

$$\begin{bmatrix} F & G \\ D & C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}.$$

Algorithm 5: $P = \texttt{LU\_block\_precond}(A)$

Define a 2×2 block partition of the system matrix (into subsets of physical variables):

$$A = \begin{bmatrix} F & G \\ D & C \end{bmatrix}$$

Define the approximations $S_\sharp \approx C - DF^{-1}G$ and $F_\sharp \approx F$
for $H = \{S_\sharp,\ F_\sharp$ *(and possibly* $F)$ $\}$ **do**
  if H *involves more than one physical variable* **then**
    Replace H^{-1} by its LU approximation, i.e., $H^{-1} \leftarrow \texttt{LU\_block\_precond}(H)$
  else
    Define an effective preconditioner $P(H)$ (e.g., using DDM, multigrid...)
    Define the tolerance tol_H and replace $H^{-1} \leftarrow \texttt{precond\_Krylov}(H, P(H), \text{tol}_H)$
Define $P(A)$ as in (28), (29) or (30)

Algorithm 6: $P = \texttt{LU\_approximation}(A)$

Define a 2×2 block partition of the system matrix (into subsets of physical variables):

$$A = \begin{bmatrix} F & G \\ D & C \end{bmatrix}$$

Define the approximations $S_\sharp \approx C - DF^{-1}G$ and $F_\sharp \approx F$
for $H = \{S_\sharp,\ F_\sharp$ *(and possibly* $F)$ $\}$ **do**
  if H *involves more than one variable* **then**
    Define the LU factorization of H: $P(H) \leftarrow \texttt{LU\_approximation}(H)$
    Define the tolerance tol_H and replace $H^{-1} \leftarrow \texttt{precond\_Krylov}(H, P(H), \text{tol}_H)$
  else
    Define an effective preconditioner $P(H)$ (e.g., using DDM, multigrid...)
    Define the tolerance tol_H and replace $H^{-1} \leftarrow \texttt{precond\_Krylov}(H, P(H), \text{tol}_H)$
Define $P(A)$ as in (28), (29) or (30)

First, let us state the Uzawa method as a U-preconditioner (29) for solving the stationary Stokes problem. Considering that the block matrix $F = K$, i.e., it only contains diffusive terms, the exact Schur complement for the pressure is $S = C - DK^{-1}G$. However, at the continuous level, $DK^{-1}G = \nabla^T \Delta_0^{-1} \nabla$, where Δ_0^{-1} denotes the inverse of the Laplace problem with homogeneous Dirichlet boundary conditions on $\Gamma_{E,u}$. If we replace Δ_0^{-1} by Δ^{-1} (without boundary conditions), the following commutation of operators holds: $\nabla^T (\nabla^2)^{-1} \nabla = (\nabla^2)^{-1} \nabla^T \nabla = 1$. Now, we consider the approximation $\nabla^T \Delta_0^{-1} \nabla \approx 1$; it is exact for periodic boundary conditions or $\Omega = \mathbb{R}^d$. As a result, we approximate $DK^{-1}G \approx \text{Re}\, M_p$, where M_p is a mass matrix for the pressure. This way, we can write the Uzawa block preconditioner as,

$$P_{\text{Uzw}}(A) = \begin{bmatrix} F_\sharp & G \\ 0 & S_\sharp \end{bmatrix}^{-1}, \text{ where } \begin{array}{l} F_\sharp^{-1} = F^{-1} \\ S_\sharp^{-1} = C^{-1} + \frac{1}{\text{Re}}\, M_p^{-1} \end{array} . \tag{31}$$

Note that the expression of the Schur complement approximation also involves the stabilization matrix inverse.

Cahouet and Chabard extended these ideas to the transient Stokes problem in [13]. In this case, the block matrix $F = \frac{1}{\delta t} M + K$ contains the temporal and diffusive terms. If we take into account that $-D(\frac{1}{\delta t} M)^{-1} G$ is spectrally equivalent to a Laplacian matrix, i.e., $-D(\frac{1}{\delta t} M)^{-1} G \approx \delta t\, L$ (L denotes the typical FE discretization for $-\Delta$), the Cahouet-Chabard (CC) block preconditioner is written as,

$$P_{CC}(A) = \begin{bmatrix} F_\sharp & G \\ 0 & S_\sharp \end{bmatrix}^{-1}, \text{ where } \begin{array}{l} F_\sharp^{-1} = F^{-1} \\ S_\sharp^{-1} = (C + \delta t L)^{-1} + \frac{1}{\text{Re}}\, M_p^{-1} \end{array} . \tag{32}$$

In order to introduce the convective term into the preconditioner, the pressure convection-diffusion (PCD) preconditioner was developed in [22, 21]. This preconditioner is based on approximating the original Schur comple-

ment by a commutation of operators, viz. $\nabla^T\mathcal{L}^{-1}\nabla \approx \nabla^T\nabla\mathcal{L}_p^{-1}$, where $\mathcal{L}(\cdot) = \frac{1}{\delta t}(\cdot) + a\cdot\nabla(\cdot) - \frac{1}{\mathrm{Re}}\Delta(\cdot)$ is a CDR operator and $\mathcal{L}_p$ a pressure CDR operator. If we apply this approximation to the discrete level, it leads to $DF^{-1}G \approx D(M^{-1}F)^{-1}M^{-1}G \approx DM^{-1}G(M_p^{-1}F_p)^{-1} \approx L_pF_p^{-1}M_p$, where F_p is the matrix obtained after discretization of the pressure CDR operator $\mathcal{L}_p$. Therefore, the expression of the PCD preconditioner is,

$$P_{PCD}(A) = \begin{bmatrix} F_\sharp & G \\ 0 & S_\sharp \end{bmatrix}^{-1}, \text{ where } \begin{array}{l} F_\sharp^{-1} = F^{-1} \\ S_\sharp^{-1} = M_p^{-1}F_pL_p^{-1} \end{array}. \tag{33}$$

Finally, other classical and well-known algorithms for solving the transient Navier-Stokes equations are the pressure segregation (PC) methods, also known as fractional step schemes. They were first developed independently by Chorin [14, 15] and Temam [35]. Basically, they consist of two steps. First, an intermediate velocity that does not verify the incompressibility condition is obtained from the momentum equation. Then, an end-of-step velocity is computed taking into account the pressure gradient and the velocity divergence terms. These schemes assume that the convective and diffusive terms are negligible with respect to the temporal evolutionary term and therefore $F \approx \frac{1}{\delta t}M$ and $-DF^{-1}G \approx -D(\frac{1}{\delta t}M)^{-1}G \approx \delta t L$, which is a reasonable assumption for δt small. They can be implemented as a LU-preconditioner

$$P_{PC}(A) = \begin{bmatrix} I & F_\sharp^{-1}G \\ 0 & I \end{bmatrix}^{-1}\begin{bmatrix} F & 0 \\ D & S_\sharp \end{bmatrix}^{-1}, \text{ where } \begin{array}{l} F_\sharp^{-1} = \delta t M^{-1} \\ S_\sharp^{-1} = (C+\delta t L)^{-1} \end{array}. \tag{34}$$

4.3. Incompressible inductionless MHD preconditioners

In this section, we consider two different preconditioners for the inductionless MHD problem, based on the recursive block LU factorization introduced above. The first preconditioner is based on an initial factorization of the system matrix into fluid and magnetic subproblems. The second preconditioner segregates at the first level field variables (velocity and current) from Lagrange multiplier-type variables (pressure and electric potential).

4.3.1. Fluid-magnetic subproblem factorization (FMS preconditioner)

Let us consider the 4×4 block system (23) and reorder it in such a way that the electromagnetic variables are written first and then the fluid unknowns:

$$A = \left[\begin{array}{cc|cc} F_j & G_j & C_j & T_j \\ D_j & C_\phi & T_\phi & 0 \\ \hline C_u & T_u & F_u & G_u \\ T_p & 0 & D_u & C_p \end{array}\right]\left[\begin{array}{c} j \\ \phi \\ \hline u \\ p \end{array}\right] = \left[\begin{array}{c} f_j \\ f_\phi \\ \hline f_u \\ f_p \end{array}\right]. \tag{35}$$

Following Algorithms 5 and 6, we arrange the 4×4 block matrix A from (35) as a 2×2 block system grouping together the electromagnetic unknowns, j-ϕ, on one hand and the fluid unknowns, u-p, on the other hand. Moreover, we choose the preconditioner P_{FMS} as a U-preconditioner (29),

$$\begin{bmatrix} F_{j\phi} & C_{j\phi} \\ C_{up} & F_{up} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \quad P_{FMS}(A) = \begin{bmatrix} F_{j\phi} & C_{j\phi} \\ 0 & S_{up\sharp} \end{bmatrix}^{-1}, \tag{36}$$

where $x=[j,\phi]^T$, $y=[u,p]^T$, $f=[f_j,f_\phi]^T$, $g=[f_u,f_p]^T$ and matrices $F_{j\phi}$, F_{up}, $C_{j\phi}$ and C_{up} are the corresponding 2×2 block matrices. The key aspect to derive the preconditioner is the definition of the approximation for $S_{up\sharp} = F_{up} - C_{up}F_{j\phi}^{-1}C_{j\phi}$. We have chosen $S_{up\sharp} = F_{uj}$. Let us justify this (simple) choice for the Schur complement approximation. It comes from the analysis of the continuous problem (5)-(8). We consider the electromagnetic subproblem (7)-(8). From (8), we have that $\phi = \Delta_0^{-1}(\nabla\cdot(u\times B))$, where Δ_0^{-1} is the inverse of the Laplacian with homogeneous boundary conditions on $\Gamma_{C,j}$. Invoking this expression of ϕ in (7) we obtain:

$$j = -\nabla\Delta_0^{-1}(\nabla\cdot(u\times B)) + (u\times B).$$

Using this expression in the momentum equation (5), we get

$$\partial_t u + (u\cdot\nabla)u - \frac{1}{\mathrm{Re}}\Delta u + \nabla p + \mathrm{N}((\nabla\Delta^{-1}\nabla\cdot(u\times B) - (u\times B))\times B) = f.$$

Now, we note that the coupling term from the magnetic problem to the fluid problem, i.e., $((\nabla\Delta^{-1}\nabla\cdot(u\times B))-(u\times B))$, vanishes when we consider the same approximation as for the Uzawa algorithm, $\nabla\Delta^{-1}\nabla\cdot \approx I$. This derivation suggests that the subproblems for the Navier-Stokes (NSI) and Darcy-type (DCY) equations can be solved uncoupled at the

preconditioner level. This is certainly an approximation, but this kind of approximation has been proved to be optimal for the Stokes problem [22], leading to robust preconditioners. No approximation of $F_{j\phi}$ is taken at this level.

The following step in the preconditioner definition consists of adding a second level of recursion with the approximation of the inverses of block matrices $F_{j\phi}$ and F_{up}. For instance, let us replace them by a U-preconditioner, $F_{j\phi}^{-1} \leftarrow \texttt{LU\_block\_precond}(F_{j\phi})$ and $F_{up}^{-1} \leftarrow \texttt{LU\_block\_precond}(F_{up})$,

$$P(F_{j\phi}) = \begin{bmatrix} F_j & G_j \\ 0 & S_{\phi\sharp} \end{bmatrix}^{-1}, \qquad P(F_{up}) = \begin{bmatrix} F_u & G_u \\ 0 & S_{p\sharp} \end{bmatrix}^{-1}.$$

The last missing ingredient is to define the approximation of $S_{\phi\sharp}$ and $S_{p\sharp}$. For $S_{p\sharp}$, we can consider any of the methods in Section 4.2. On the other hand, since $S_\phi = C_\phi - D_\phi F_j^{-1} G_j$ and F_j is a mass matrix for the Galerkin approximation, we can naturally use the same approximation as for PC or PCD problems, i.e., $S_{\phi\sharp}^{-1} \approx (C_\phi + L_\phi)^{-1}$. Summarizing, we consider the following options:

$$\text{Uzawa}: \begin{cases} S_{p\sharp}^{-1} = C_p^{-1} + \frac{1}{\mathrm{Re}} M_p^{-1} \\ S_{\phi\sharp}^{-1} = (C_\phi + L_\phi)^{-1} \end{cases}, \tag{37}$$

$$CC: \begin{cases} S_{p\sharp}^{-1} = (C_p + \delta t L_p)^{-1} + \frac{1}{\mathrm{Re}} M_p^{-1} \\ S_{\phi\sharp}^{-1} = (C_\phi + L_\phi)^{-1} \end{cases}, \tag{38}$$

$$PCD: \begin{cases} S_{p\sharp}^{-1} = M_p^{-1} F_p L_p^{-1} \\ S_{\phi\sharp}^{-1} = M_\phi^{-1} F_\phi L_\phi^{-1} \end{cases} \quad \text{where} \quad \begin{array}{l} F_p(p,q) = \frac{1}{\delta t}(p,q) + ((u\cdot\nabla)p, q) + \frac{1}{\mathrm{Re}}(\nabla p, \nabla q) \\ F_\phi(\phi,\psi) = (\phi,\psi) \end{array}, \tag{39}$$

$$PC: \begin{cases} S_{p\sharp}^{-1} = (C_p + \delta t L_p)^{-1} \\ S_{\phi\sharp}^{-1} = (C_\phi + L_\phi)^{-1} \end{cases}. \tag{40}$$

Note that for using the PC method, we have to define $P_{FMS}(A)$ in (36) as a LU-preconditioner (30). We do not write the details here for the sake of conciseness. Finally, we can write the 4×4 block preconditioner $P_{FMS}(A)$ for (35) as

$$P_{FMS}(A) = \left[\begin{array}{cc|cc} F_j & G_j & C_j & T_j \\ 0 & S_\phi & T_\phi & 0 \\ \hline 0 & 0 & F_u & G_u \\ 0 & 0 & 0 & S_p \end{array}\right]^{-1}. \tag{41}$$

Summarizing, we have defined a recursive block preconditioner P_{FMS} that allows us to decouple the computation of a multiphysics problem such as the inductionless MHD problem into one-physics problems for every physical variable (velocity, pressure, current density and electric potential) at the preconditioner level. Figure 1 displays a tree diagram to highlight how the coupled problem is uncoupled into one-physics problems along the two levels of recursion. First, the preconditioner splits the magnetic unknowns from the fluid ones whereas the second level allows us to decouple the computation of j and ϕ on one hand, and the computation of u and p on the other hand. At every splitting, we create two new problems, namely the F and S problems, and we have defined how to approximate every one of these matrices.

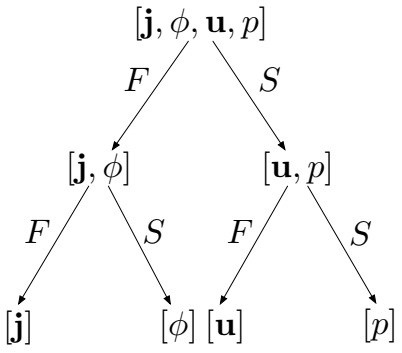


Figure 1: Schematic representation of the tree-like structure of the FMS block preconditioner.

4.3.2. Field-Lagrange multiplier factorization (FLM preconditioner)

Let us consider now a different reordering of system (23) where the vectorial fields u and j are written first and the Lagrange multipliers p and ϕ next,

$$\left[\begin{array}{cc|cc} F_u & C_u & G_u & T_u \\ C_j & F_j & T_j & G_j \\ \hline D_u & T_p & C_p & 0 \\ T_\phi & D_j & 0 & C_\phi \end{array}\right] \left[\begin{array}{c} u \\ j \\ \hline p \\ \phi \end{array}\right] = \left[\begin{array}{c} f_u \\ f_j \\ \hline f_p \\ f_\phi \end{array}\right]. \tag{42}$$

Let us write the 4×4 block matrix A from (42) as a 2×2 block system splitting the vectorial fields from the Lagrange multipliers, that is, grouping together u-j and p-ϕ,

$$\left[\begin{array}{cc} F_{uj} & G_{uj} \\ D_{uj} & C_{p\phi} \end{array}\right] \left[\begin{array}{c} x \\ y \end{array}\right] = \left[\begin{array}{c} f \\ g \end{array}\right], \tag{43}$$

where now $x = [u, j]^T$, $y = [p, \phi]^T$, $f = [f_u, f_j]^T$, $g = [f_p, f_\phi]^T$ and matrices F_{uj}, G_{uj}, D_{uj} and $C_{p\phi}$ are the corresponding 2×2 block matrices. Let us now define a U-preconditioner for the 2×2 block system (43). The required ingredients are a good approximation for the Schur complement $S_{p\phi} = C_{p\phi} - D_{p\phi} F_{uj}^{-1} G_{uj}$ and possibly for F_{uj}^{-1}. For the Schur complement matrix, we consider the following approximations. First, we neglect the off-diagonal (coupling) blocks in F_{uj}^{-1}, $D_{p\phi}$ and G_{uj} (we note that for $D_{p\phi}$ and G_{uj} the off-diagonal blocks are zero for Galerkin and OSS approximations). This way, we only need to approximate the fluid Schur complement $S_p = C_p - D_p F_u^{-1} G_u$ and the magnetic Schur complement $S_\phi = C_\phi - D_\phi F_j^{-1} G_j$. As approximation of S_p and S_ϕ, we can use any of the preconditioners presented in (37)-(40).

However, the 2×2 block matrix F_{uj} still couples the computation of u and j. At this point, we can add a second level of recursion and approximate the inverse of F_{uj} by, e.g., its U-factorization, $F_{uj}^{-1} \leftarrow$ `LU_block_precond`(F_{uj}). Following the same ideas presented in [9] where the matrix F_u is approximated by $\delta t^{-1} M_u$, the Schur complement $S_j = F_j - \delta t C_j F_u^{-1} C_u$ can be approximated by the term

$$S_j \approx F_j - \delta t C_j M_u^{-1} C_u \approx F_j + R_j, \quad \text{where} \quad R_j = \delta t \mathrm{N}^2(j \times B, k \times B).$$

Finally, let us write the P_{FLM} preconditioner expression,

$$P_{FLM}(A) = \left[\begin{array}{cc|cc} F_u & C_u & G_u & T_u \\ 0 & S_j & T_j & G_j \\ \hline 0 & 0 & S_p & 0 \\ 0 & 0 & 0 & S_\phi \end{array}\right]^{-1}. \tag{44}$$

This preconditioner decouples the computation of the four physical variables into one-physics problems in a recursive way. The first level of recursion decouples the vectorial fields, velocity and current density, from the Lagrange multipliers, pressure and electric potential. In the second recursive level, the computation of u and j is also decoupled. Note that the Lagrange multipliers p and ϕ are decoupled because of the diagonal structure of the Schur complement approximation for $S_{p\phi}$ considered above. Figure 2 shows these two levels of recursive uncoupling in a tree structure.

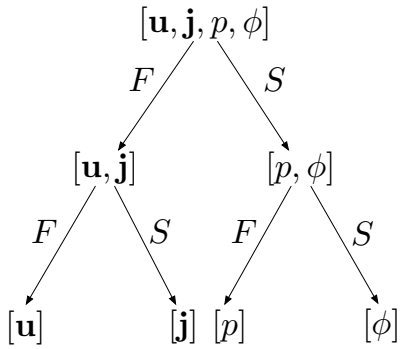


Figure 2: Schematic representation of the tree-like structure of FLM block preconditioner.

Note that the same derivation can be made for system (42) if the position of u and j is interchanged. We do not write the details for the sake of conciseness but it is important to highlight that the approximation of F_{uj} implies in this case the approximation of the Schur complement with respect to u instead of j, i.e.,

$$P(F_{uj}) = \begin{bmatrix} F_j & C_j \\ 0 & S_u \end{bmatrix}^{-1} \quad \text{where} \quad S_u \approx F_u - C_u M_j^{-1} C_j \approx F_u + R_u \quad \text{and} \quad R_u = (u \times B, v \times B).$$

4.4. New stabilized PCD preconditioners for inductionless MHD

In this section, we motivate the FLM preconditioner, via some numerical experiments. Let us consider system (42) preconditioned with the U-preconditioner with no approximations, i.e., using the *exact* Schur complement $S_{p\phi}$ and F_{uj}. Note that this is not affordable for a real simulation but for a very small problem, it will give us an insight on the importance of every term that is included in the Schur complement. The exact expression of the Schur complement for the OSS formulation presented in Algorithm 2 is, (recall that blocks T_* in (42) are zero for OSS stabilization),

$$S_\sharp = C_{p\phi} - D_{uj} F_{uj}^{-1} G_{uj} = \begin{bmatrix} C_p & 0 \\ 0 & C_\phi \end{bmatrix} - \begin{bmatrix} D_u & 0 \\ 0 & D_j \end{bmatrix} \begin{bmatrix} F_u & C_u \\ C_j & F_j \end{bmatrix}^{-1} \begin{bmatrix} G_u & 0 \\ 0 & G_j \end{bmatrix}. \tag{45}$$

Table 1 presents the number of iterations needed for solving the inductionless MHD problem using an iterative solver like GMRES. We have solved the test problem for two Hartmann numbers, Ha $= 10, 1000$. To assess the importance of the terms in F_{uj}^{-1} in the Schur complement, two different versions of F_{uj}^{-1} in (45) have been tested. First, we want to evaluate the importance of the stabilization terms at the preconditioner. In order to do so, we have considered the exact Schur complement, denoted as "Stabilized Coupled" in Table 1 and the case in which we do eliminate the stabilization terms at the preconditioner, denoted as "Galerkin Coupled." Next, we want to evaluate the importance of the coupling terms, eliminating at the preconditioner the coupling terms; this case is denoted as "Stabilized Uncoupled". Finally, we consider the case without stabilization and coupling terms, denoted as "Galerkin Uncoupled".

We have solved the 3D inductionless MHD cavity flow problem (see Section 5.2 for a detailed description) using a very coarse mesh of $8 \times 8 \times 8$ linear hexahedral elements and the numerical method with OSS stabilization presented in Algorithm 2. The results obtained when solving the linear system of equations from the first nonlinear iteration for the four possible combinations are reported in Table 1. These results indicate that the coupling blocks C_u and C_j do not have an important effect on the Schur complement definition. This fact is the numerical evidence that has motivated the FLM preconditioners above.

	Stabilized Coupled	Galerkin Coupled	Stabilized Uncoupled	Galerkin Uncoupled
Ha=10	2	5	7	7
Ha=1000	2	31	8	36

Table 1: Number of iterations when solving the exact Schur complement.

On the other hand, the results in Table 1 suggest that the stabilization terms are crucial for a robust Schur complement, especially when the Hartmann number increases. It has motivated us to introduce the stabilization terms in the pressure operator for the Schur complement approximation defined in (41) for the PCD preconditioner in order to improve its efficiency. Therefore, we propose the following *stabilized* pressure CDR operators

$$\hat{F}_p(p,q) = \frac{1}{\delta t}(p,q) + ((u\cdot\nabla)p, q) + \frac{1}{\mathrm{Re}}(\nabla p, \nabla q) + (\tau_1(u\cdot\nabla)p, (u\cdot\nabla)q) + (\tau_3 p|B|, q|B|), \tag{46}$$

$$\hat{F}_\phi(\phi,\psi) = (\phi,\psi) + \mathrm{N}^2(\tau_1\phi|B|, \psi|B|), \tag{47}$$

instead of (39). It is important to note that the stabilization terms with vectorial products for u and j have been approximated by scalar multiplications for the scalar variables p and ϕ where $|B|$ is the norm of the external magnetic field B. The improvement associated to this modification will be shown with numerical tests in Section 5.2.3. Note also that this new definition of the Schur complement approximation for the Lagrange multipliers p and ϕ can be used to improve the FMS block preconditioner in (41).

4.5. Thermally coupled inductionless MHD preconditioners

The design of recursive block preconditioners for solving the thermally coupled inductionless MHD equations follows the ideas presented in Section 4.1, since the system matrix (27) already has a U structure; the coupling between θ and the rest of MHD unknowns is in one direction only. Therefore, the definition of, e.g., a U-preconditioner for (27) reads as

$$P(A) = \begin{bmatrix} F_{upj\phi\sharp}^{-1} & -F_{upj\phi\sharp}^{-1} C_{upj\phi} F_{\theta}^{-1} \\ 0 & F_{\theta}^{-1} \end{bmatrix}, \tag{48}$$

where $F_{upj\phi\sharp}^{-1}$ can be the FMS preconditioner (41) or the FLM preconditioner (44).

5. Numerical experiments

This section is devoted to numerically test the behavior of the block preconditioners exposed in previous sections. On one hand, the 3D lid-driven magnetohydrodynamic cavity flow has been used to test and evaluate the properties of the several block preconditioners designed previously. On the other hand, a simulation of a real application such as a Test Blanket Module (TBM) for nuclear fusion reactors has been carried out in order to check the preconditioner behavior when solving a very challenging problem due to its extreme physical working conditions, i.e., a very high Hartmann number. Finally, a flow into a vertical enclosure subject to a temperature gradient has been simulated to test the block preconditioners derived for the thermally coupled inductionless MHD equations.

5.1. Experimental framework

The block recursive LU preconditioners subject of study were implemented and tested within FEMPAR. FEMPAR is an in-house, parallel hybrid OpenMP/MPI, object-oriented (OO) framework, developed in Fortran90/95, for the massively parallel stabilized FE simulation of multiphysics problems governed by systems of PDEs. FEMPAR provides the tools to drive all the steps required in a typical massively parallel FE multiphysics simulation. These steps comprise the partition (via multilevel graph partitioning) of the underlying unstructured computational mesh into submeshes for distributed-memory computation, the definition of a multi-physics coupled problem and its FE time and space discretization, the nonlinear solution of the problem, the parallel assembly of the underlying blocked large and sparse linear system, the definition of block preconditioners for its preconditioned Krylov subspace solution, and the use of optimal parallel solvers for each block problem at hand. As an optimal parallel solver, among others, FEMPAR provides highly scalable distributed-memory implementations of the Balancing Domain Decomposition by Constraints (BDDC) preconditioner [8, 7, 20].

All experiments reported in the sequel were obtained on a large-scale multicore-based distributed-memory machine, Marenostrum III, located at the Barcelona Supercomputing Center. The Marenostrum III is a FDR10 Infiniband interconnected cluster with 36 IBM System x iDataPlex racks devoted to computations. Each rack is composed of 84 IBM dx360 M4 compute nodes, each equipped with two Intel Xeon E5-2670 EightCore processors running at 2.6 GHz (16 computational cores in total) and 32 GBytes of DDR3 memory (2 GBytes per core), and runs a full-featured Linux OS (SuSe distribution 11 SP2). The codes were compiled using Intel Fortran compiler (13.0.1) with recommended optimization flags and we used OpenMPI (1.5.4) tools and libraries for message-passing. The codes were linked against the BLAS/LAPACK and PARDISO available on the Intel MKL library (version 11.0, update 1). All floating-point calculations were performed in IEEE double precision.

5.2. Three-dimensional (3D) MHD cavity flow

The experiments in this subsection deal with the three-dimensional (3D) lid-driven magnetohydrodynamic cavity flow. The computational domain is a unit cube $[0,1]^3$ discretized by a series of uniform meshes composed of linear hexahedral elements with 2^n elements by dimension, where $n = 3, ..., 7$. These meshes were uniformly partitioned into/distributed over (proportionally) increasing number of subdomains/computational cores.

The velocity boundary conditions for this problem consist of a moving upper lid in x-direction, $u = (1,0,0)$ on $(0,1) \times (0,1) \times \{1\}$, and a no-slip condition $u = (0,0,0)$ elsewhere on the boundary. The pressure is fixed to zero in one point, in order to fix the mean value. The boundary condition for the current density is $j \cdot n = 0$ on the whole boundary. The external magnetic field B is chosen to be orthogonal to the moving lid and therefore, only its z-component is nonzero. Note that it can be written in terms of the Reynolds and Hartmann numbers as

$$B = (0,0,B_z) \text{ with } B_z = \frac{\mathrm{Ha}}{\sqrt{\mathrm{Re}}}.$$

For the underlying preconditioned iterative solvers, the iteration is stopped whenever the residual r_k at a given iteration k satisfies $\|r_k\|_2 \leq \text{atol} + \text{rtol}\|r_0\|_2$, with atol and rtol being, respectively, the absolute and relative residual tolerances. Unless specified, atol $= 0.0$ and rtol $= 10^{-8}$ for both the external and internal iterative solution processes in block recursive preconditioners, i.e., we use Algorithm 6. In particular, FGMRES will be used as the iterative solver for the topmost and intermediate levels, and BDDC preconditioning for the bottommost level (one-physics problems).

The following subsections 5.2.1 and 5.2.2 will compare the behavior of the FLM preconditioner (44) depending on the approximation of the Schur complement with respect to the Lagrange multipliers, p and ϕ, defined in (37)-(40). Therefore, the several versions of the FLM preconditioner tested in the following subsections will be called after the approximation of the Schur complement for p and ϕ, i.e., Uzawa, Cahouet-Chabard (CC), Pressure Convection-Diffusion (PCD) and Pressure Correction (PC).

The following numerical experiments will also study the effect of the stabilization technique, ASGS or OSS, in the system matrix. It is important to stress that, for the following tests, we have only included the terms that are strictly needed for stabilization purposes in the OSS formulation. Therefore, terms $(\tau_2 \nabla \cdot u_h, \nabla \cdot v_h)$ and $(\tau_4 \nabla \cdot j_h, \nabla \cdot k_h)$ have not been used in Algorithm 2.

5.2.1. Comparison between Schur complement approximations for the FLM preconditioner (stationary case)

This subsection compares the efficiency of two block preconditioners for the stationary problem. Moreover, another goal of this study is to assess the stabilization method influence in the preconditioner behavior. The block preconditioners used are Pressure Convection-Diffusion (PCD) and Uzawa, together with the two stabilization techniques, ASGS and OSS, see Algorithms 1 and 2.

The 3D cavity problem has been solved with a Reynolds number Re $= 10$ and for three Hartmann numbers Ha $= 10, 100, 1000$. Figure 3 shows the number of iterations needed for the linear solver to converge. In the top row, the number of iterations corresponds to the external solver whereas the bottom row displays the iterations to converge the inner block u-j. On one hand, the Uzawa preconditioner does not optimally converge with h; the number of external iterations increases when the mesh is refined for every Hartmann number and for both stabilization methods. On the other hand, the PCD preconditioner has a much better behavior for small Hartmann numbers, Ha=10. For Larger Hartmann numbers, such as Ha $= 100, 1000$, the external solver requires more iterations to converge. However, as the mesh is refined, the number of iterations is reduced because for smaller h the diffusive term is more important than the convective one. Moreover, the plots in the bottom row in Figure 3 show that the inner solver for the coupling u-j has a much better behavior. The iterations do not increase when reducing h, or even they are reduced for the PCD preconditioner. There exists a slight increase when increasing the Hartmann number because the coupling between the fluid and magnetic subproblems becomes stronger.

5.2.2. Comparison between Schur complement approximations for the FLM preconditioner (transient case)

This subsection compares the number of iterations needed to solve the transient magnetohydrodynamic cavity problem in three dimensions (3D) using different block preconditioners for the inductionless MHD equations. Three block preconditioners have been used, namely Pressure Convection-Diffusion (PCD), Cahouet-Chabard (CC), and Pressure Correction (PC). Similarly to the previous subsection, two different stabilization formulations have been solved, ASGS and OSS methods.

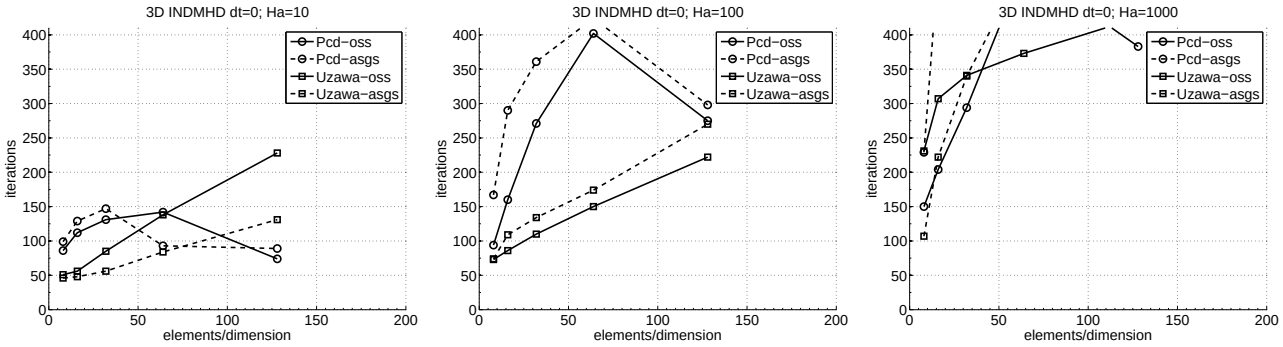
Figures 4 and 5 display the results obtained for two time step sizes $\delta t = 1.0$s and $\delta t = 0.01$s, respectively. The top row shows the plots for the number of external iterations needed to solve the linear system for three Hartmann numbers, Ha $= 10, 100, 1000$ from left to right. Similarly, the bottom row contains the results for the internal block u-j solver, also for Ha $= 10, 100, 1000$ from left to right.

In general, the number of external iterations required to solve OSS linear systems is smaller than that required for ASGS ones for both choices of δt, especially for the highest Hartmann number, i.e., Ha $= 1000$. Regarding preconditioner efficiency, the PCD and CC preconditioners have a very similar good behavior, both for the number of iterations and the convergence with h, except for the highest Hartmann number Ha $= 1000$ where the number of iterations increases when reducing h. The PC preconditioner only works reasonably well for small δt but it shows a degradation with h.

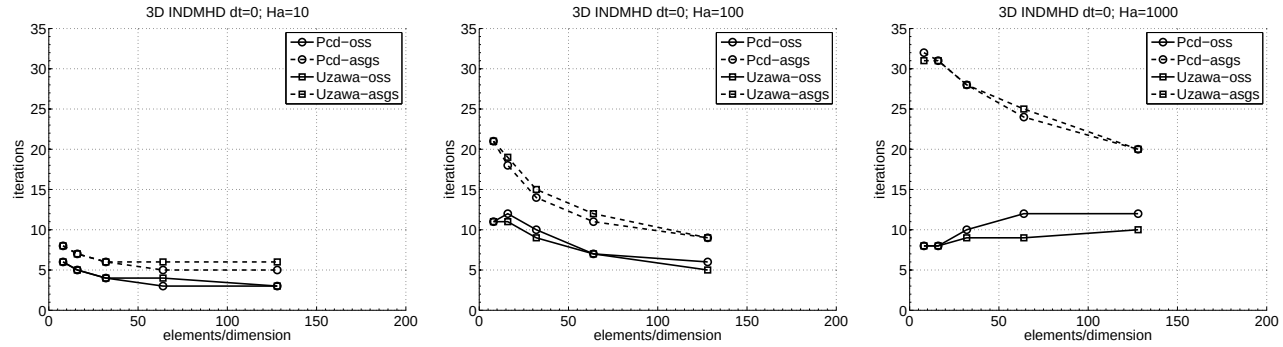
On the other hand, the internal block u-j has a better behavior when reducing h, independently of δt. However, there is a mild increase in the number of iterations for larger Hartmann numbers.

5.2.3. Improved PCD Schur complement approximation for the FLM preconditioner

In Section 4.4 we have proposed an improvement of the original PCD preconditioner by introducing some stabilization terms in the Schur complement approximation for both the pressure and the electric potential. This subsection studies its properties solving the transient 3D magnetohydrodynamic cavity flow problem with $\delta t = 0.1, 0.01, 0.001$s for the FLM preconditioner version.



(a) External iterations (FLM preconditioner). Left: Ha=10, Center: Ha=100, Right: Ha=1000.



(b) Internal iterations *u-j* (FLM preconditioner). Left: Ha=10, Center: Ha=100, Right: Ha=1000.

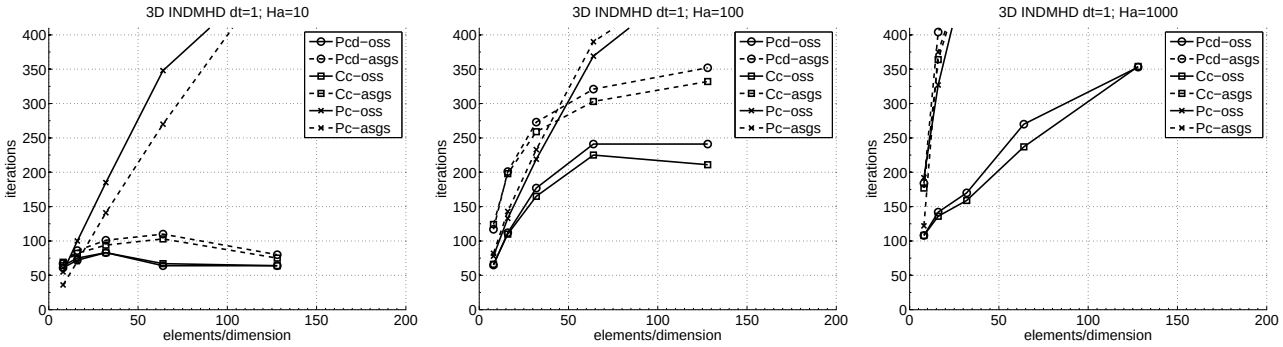
Figure 3: 3D stationary magnetohydrodynamic cavity problem.

Figure 6 shows the number of external iterations for solving the linear system of equations using the stabilized PCD preconditioner. It is clear that the preconditioner behavior with respect to h improves when the time step size is reduced. However, although the number of iterations is also reduced when reducing the time step size δt for Ha = 100, the behavior with respect to h is not as good as the rest of Hartmann numbers tested. This behavior is similar to the original PCD preconditioner, as shown in the previous subsection. However, the addition of the stabilization terms into the Schur complement approximation is crucial for improving the preconditioner efficiency and reducing the number of iterations needed to solve the system. Figure 7 shows the number of iterations versus the elements per dimension for both the original PCD and the stabilized PCD preconditioners for 3D uniform meshes with number of elements per dimension from 2^3 to 2^8. The subplot in the left displays the results for a small Hartmann number Ha = 10 where both preconditioners have almost the same behavior. However, for larger Hartmann numbers Ha = $100, 1000$, the center and right subplots show that the stabilized PCD preconditioner reduces the number of iterations needed to solve the problem and also speeds up the convergence to an asymptotic state. Although the behavior of the stabilized PCD is better than the original PCD, for a Hartmann number of Ha=1000 and $\delta t = 0.01$s the number of iterations increases when reducing h in meshes composed by up to 2^8 elements per dimension.

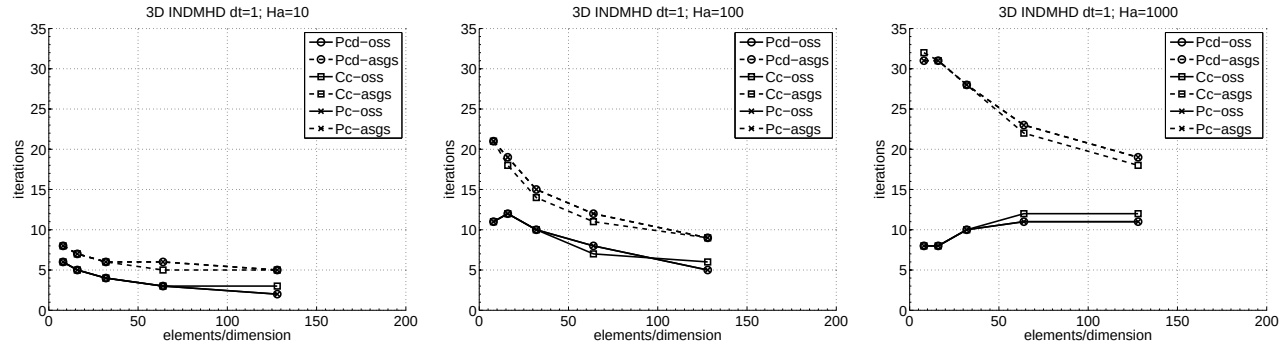
5.2.4. Improved PCD Schur complement approximation for the FMS preconditioner

The improvement of the Schur complement approximation for the Lagrange multipliers for the PCD block preconditioner proposed in Section 4.4 has also been applied together with the FMS preconditioner defined in Section 4.3.1. It involves the internal blocks splitting between subproblems, that is, between the fluid (NSI) and magnetic (DCY) subproblems. This section deals with the numerical tests done using this approach.

Figure 8 shows the number of external iterations needed to solve the linear system of equations for three different time step sizes, namely $\delta t = 0.1, 0.01, 0.001$s. The plots display the results obtained for four Hartmann numbers Ha = $1, 10, 100, 1000$. It is clear that this preconditioner has a very good behavior and it is optimally convergent with the mesh size h. Further, the results are quite insensitive to δt and the Hartmann number.



(a) External iterations. Left: Ha=10, Center: Ha=100, Right: Ha=1000.



(b) Internal iterations u-j. Left: Ha=10, Center: Ha=100, Right: Ha=1000.

Figure 4: 3D transient magnetohydrodynamic cavity problem with $\delta t = 1.0$s (FLM preconditioner).

5.2.5. Effect of the internal blocks precision over the external solver

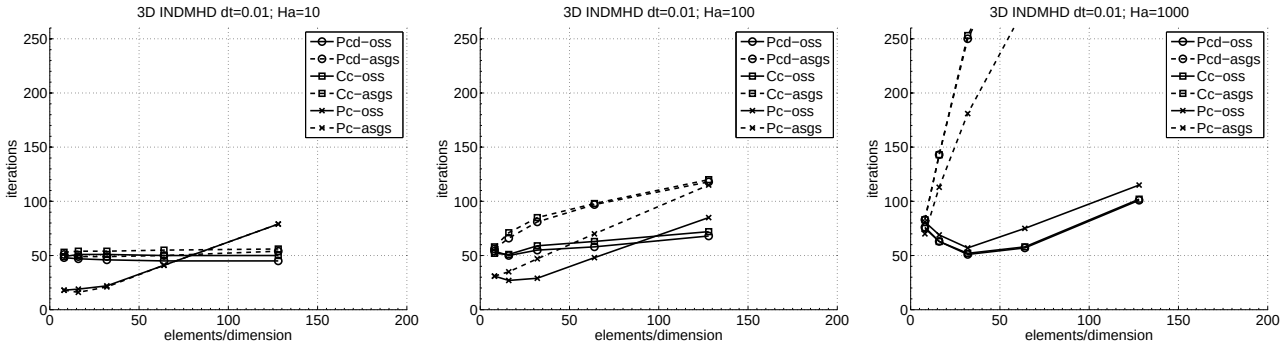
This study aims to understand which is the effect of the precision when solving the internal blocks, u-j (FLM) or by subproblems (FMS), over the external iterations for the stabilized PCD block preconditioners. We seek a reduction of the total computation time by relaxing the internal tolerance (and, consequently, the time spent by the internal solver), despite the potential increase in the number of external iterations.

On one hand, Figure 9 shows the results obtained using the PCD FLM preconditioner for $\delta t = 0.1, 0.01, 0.001$s and Hartmann numbers Ha $= 1, 10, 100, 1000$. Each combination has been solved with three different internal block approaches, just applying once the internal preconditioner, i.e., using the block recursive preconditioner as stated in Algorithm 5, or iterating (Algorithm 6) with rtol $= 10^{-2}, 10^{-4}$. The results indicate that for small Hartmann numbers, the number of external iterations is insensitive to the internal blocks precision. However, when the Hartmann number is larger, just applying the internal preconditioner leads to an important increase in the number of iterations, although they are reduced when reducing the mesh size h. Therefore, these results allow a relaxation in the solution of the internal blocks which greatly reduces the total computation time, as it will be shown later on this subsection.

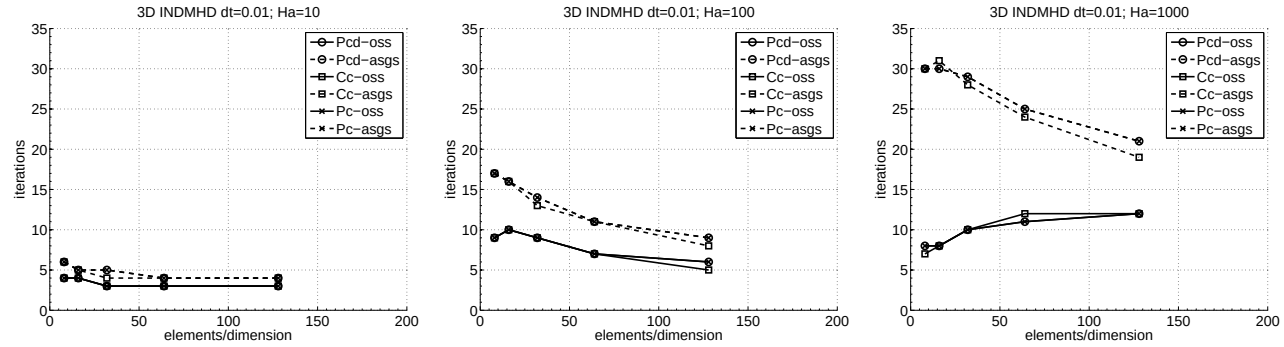
On the other hand, the same study has been done for the PCD FMS preconditioner. The results are presented in Figure 10. In this case, the number of external iterations is much more sensitive to just applying the internal preconditioner or iterating until convergence of the internal blocks (NSI and DCY). It is very clear that iterating the internal blocks reduces drastically the number of external iterations for every combination of time step size δt and Hartmann number Ha.

However, the number of external iterations is not conclusive when deciding which is the best option to solve the problem. This study has to be completed with time measurements of the external iterative solver (FGMRES) for both preconditioners. It has been done for the finest mesh composed by $128 \times 128 \times 128$ elements, uniformly partitioned into/distributed over $8 \times 8 \times 8 = 512$ subdomains/cores. The computational times for the PCD FLM preconditioner are shown in Table 2 whereas the results for the PCD FMS version are presented in Table 3.

The results for the PCD FLM indicate that the fastest option is to iterate the internal blocks with a relative tolerance of 10^{-2}, even though the number of external iterations is very similar for just applying once the internal preconditioner. The internal solver for the u-j block requires very few iterations to converge. Despite this moderate additional internal solver cost (compared to a single application of the internal preconditioner), the (slight) reduction



(a) External iterations. Left: Ha=10, Center: Ha=100, Right: Ha=1000.



(b) Internal iterations u-j. Left: Ha=10, Center: Ha=100, Right: Ha=1000.

Figure 5: 3D transient magnetohydrodynamic cavity problem with $\delta t = 0.01$s (FLM preconditioner).

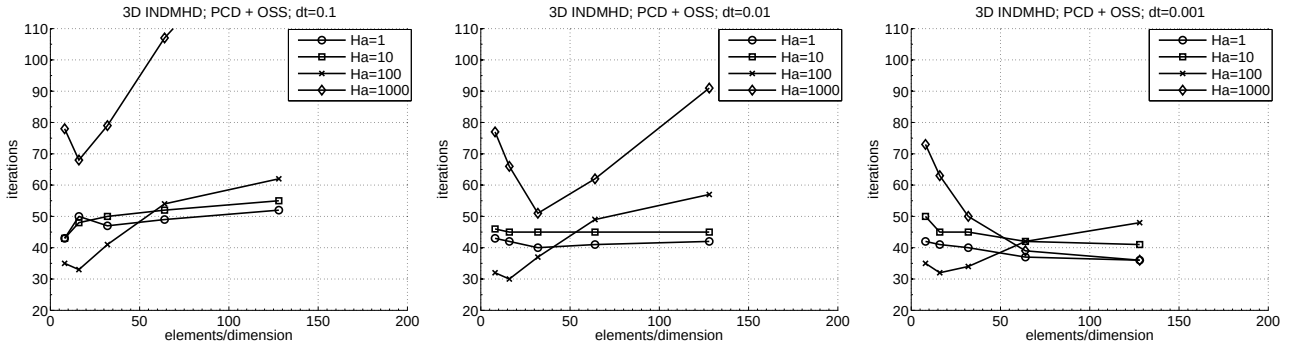


Figure 6: Stabilized PCD FLM iterations. Left: $\delta t = 0.1$s, Center: $\delta t = 0.01$s, Right: $\delta t = 0.001$s.

of the number of external iterations pays off for reducing the total computation time. Note that the block of the Lagrange multipliers p and ϕ is block diagonal and therefore there is no need to iterate it.

On the other hand, the best option for solving the magnetohydrodynamic cavity flow problem using as preconditioner the PCD FMS is just applying once the internal preconditioners, for both subproblems NSI and DCY. In this case, the internal solver for both subproblems requires a high number of iterations to converge, so that it more than pays off a single application of the internal preconditioner (despite the significant increase in the number of external iterations). However, for the largest Hartmann number considered in this study, Ha = 1000, we expect iterating the internal subproblems to be the method of choice for finer meshes, given the dramatic increase of the number of external iterations with h for a single application of the internal preconditioner.

5.3. Simulation of a Test Blanket Module (TBM) for nuclear fusion reactors

In recent years, a great worldwide effort has been put on the design and development of new nuclear fusion reactors, materializing in ITER, the International Thermonuclear Experimental Reactor, that is currently being constructed in

$h = \frac{1}{128}$		Apply M^{-1}	Iter. tol=10^{-2}	Iter. tol=10^{-4}
δt=0.001	Ha=1	38.04	25.81	55.83
	Ha=10	40.99	31.13	68.61
	Ha=100	53.16	58.12	77.66
	Ha=1000	78.34	55.07	69.16
δt=0.01	Ha=1	51.43	34.28	81.57
	Ha=10	53.37	39.52	86.87
	Ha=100	81.91	81.25	123.16
	Ha=1000	206.99	153.72	202.35
δt=0.1	Ha=1	67.77	47.20	109.38
	Ha=10	73.05	59.59	116.42
	Ha=100	103.33	91.44	141.97
	Ha=1000	357.03	257.45	377.55

Table 2: Computational time (s) for the PCD FLM preconditioner.

$h = \frac{1}{128}$		Apply M^{-1}	Iter. tol=10^{-3}	Iter. tol=10^{-4}
δt=0.001	Ha=1	44.45	27.04	44.66
	Ha=10	50.72	36.21	61.80
	Ha=100	60.01	74.52	110.64
	Ha=1000	63.75	281.99	361.56
δt=0.01	Ha=1	60.05	41.80	85.82
	Ha=10	63.92	72.76	124.58
	Ha=100	81.46	159.81	268.28
	Ha=1000	134.11	384.04	613.33
δt=0.1	Ha=1	76.44	97.88	114.23
	Ha=10	87.56	104.86	184.19
	Ha=100	99.93	273.65	366.13
	Ha=1000	228.79	632.55	1057.98

Table 3: Computational time (s) for the PCD FMS preconditioner.

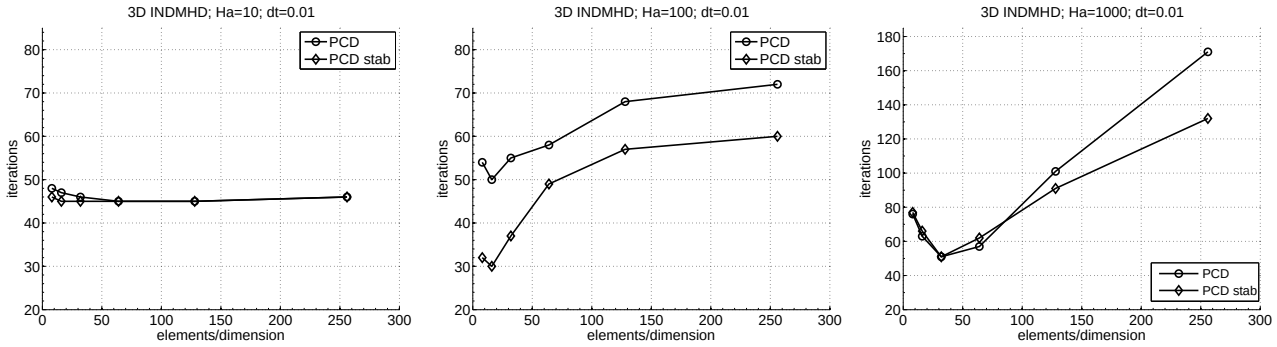


Figure 7: Comparison between PCD and stab. PCD FLM. Left: Ha=10, Center: Ha=100, Right: Ha=1000.

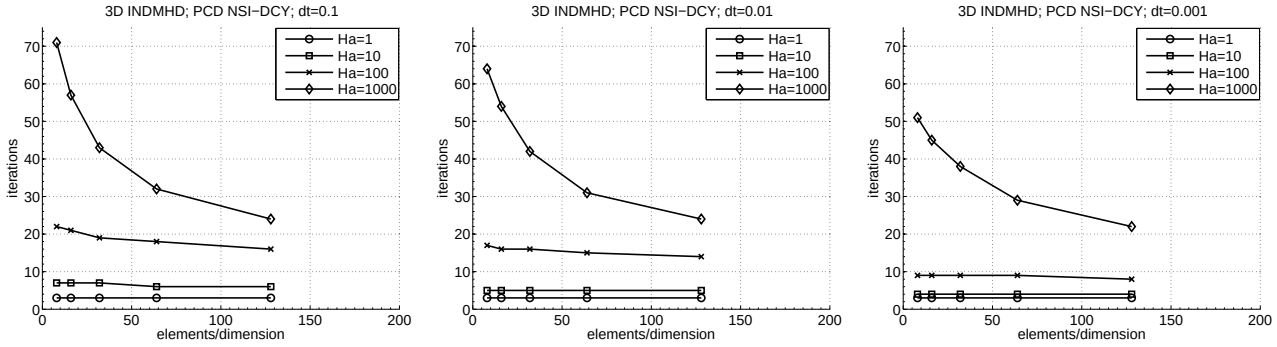


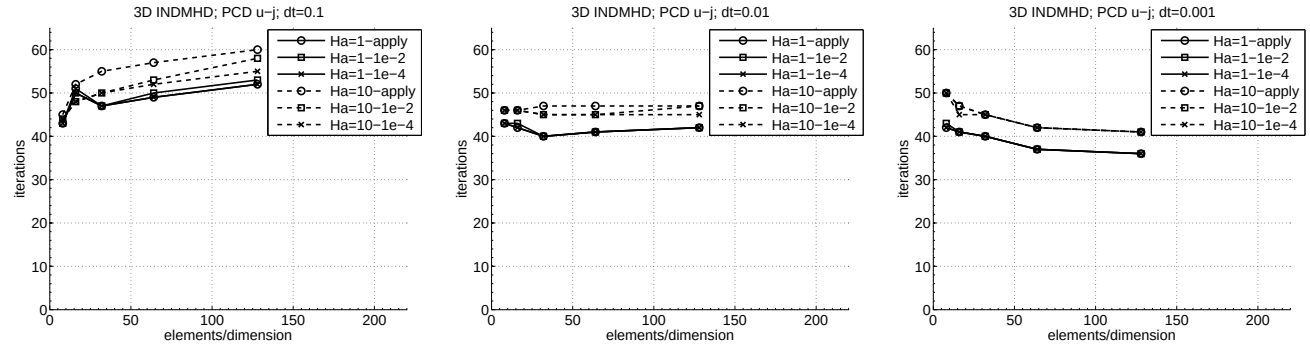
Figure 8: Stabilized PCD FMS iterations. Left: $\delta t = 0.1$s, Center: $\delta t = 0.01$s, Right: $\delta t = 0.001$s.

Cadarache (France).[1] ITER is thought to be an experimental laboratory to test the very complex and new technology needed for energy generation of future plants like DEMO. One of the main components to be studied in ITER are the Breeding Blankets (BB). These devices have crucial functions for the fusion reactor to work properly, i.e., heat power extraction from the plasma, tritium generation and shielding of the magnets from neutron and gamma radiation. In the frame of the Spanish Breeding Blanket Technology Programme TECNOFUS (see www.tecnofus.net), a dual-coolant liquid metal blanket has been designed. The liquid metal flow inside the blanket channels can be modelled through the incompressible inductionless MHD equations because its magnetic Reynolds number is very low (see [29]).

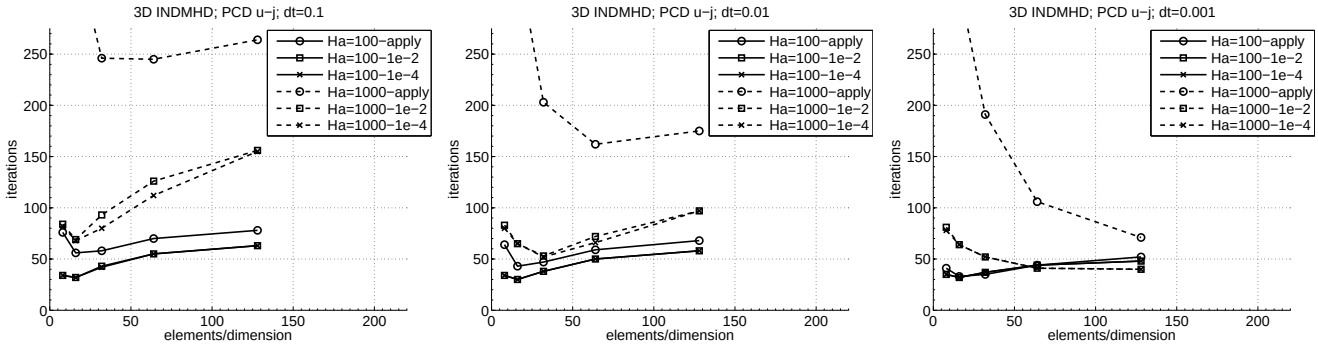
The simulation of the Tecnofus TBM is a very challenging task due to its extreme physical conditions. The fluid is a liquid metal, the alloy Pb-15.7Li, which has a density of $\rho = 9660\,\mathrm{kg/m^3}$, a viscosity of $\nu = 1.3 \cdot 10^{-7}\,\mathrm{m^2/s}$ and an electric conductivity of $\sigma = 751280\,(\mathrm{Ohm \cdot m})^{-1}$. The adimensional numbers corresponding to these physical properties are a Reynolds number of $\mathrm{Re} = 4.55 \cdot 10^6$ and a Hartmann number of $\mathrm{Ha} = 5.14 \cdot 10^4$. Note that the external magnetic field magnitude is $B = 10\,\mathrm{T}$, the velocity magnitude at the channel core is designed as $U = 0.2\,\mathrm{m/s}$ and the characteristic length of the channel section is $L = 0.305\,\mathrm{m}$.

Two meshes have been used to solve this problem. The first one, MESH-12.5M, is composed by 2,509,705 nodes and 12,395,008 linear tetrahedral elements. The second mesh, MESH-100M, is obtained after an uniform refinement of the first one where every tetrahedra is divided into 8 smaller tetrahedral elements. This way, the second mesh has 20,017,537 nodes and 99,160,064 elements. Using an automatic mesh partitioner [25], MESH-12.5M and MESH-100M were partitioned into/distributed over 512 and 4096 subdomains/cores, respectively. The boundary conditions have been set to no-slip conditions at the walls ($u = 0$) and a fixed velocity at the inlet ($u = (0, 0, -4.0)\,\mathrm{m/s}$) for the velocity field and perfectly conducting walls ($\phi = 0$) for the electromagnetic variables. Furthermore, the tolerance for the nonlinear iterations has been set to $\mathrm{nltol} = 10^{-3}$. The iterative solver (FGMRES) tolerances are $\mathrm{rtol} = 10^{-6}$ for the relative part and the absolute one is chosen as $\mathrm{atol} = \mathrm{nlres} \cdot \mathrm{nltol}/10$, where nlres is the norm of the system residual and nltol is the tolerance of the nonlinear iterative loop. This way, the iterative solver converges to a residual an order of magnitude lower than what is imposed for the nonlinear iterations to converge.

[1] See `http://www.iter.org` for details.



(a) Ha=1, 10. Left: $\delta t = 0.1$s, Center: $\delta t = 0.01$s, Right: $\delta t = 0.001$s.



(b) Ha=100, 1000. Left: $\delta t = 0.1$s, Center: $\delta t = 0.01$s, Right: $\delta t = 0.001$s.

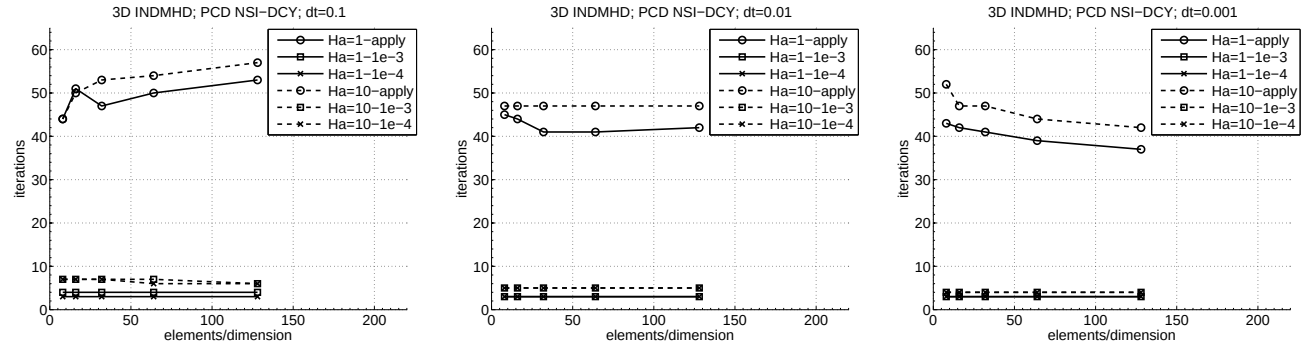
Figure 9: External iterations for the PCD FLM block preconditioner.

The computation has been carried out using the block recursive preconditioner PCD FMS explained in Section 4.3.1 with just applying once the internal recursive preconditioners for the subproblems, Navier-Stokes (NSI) and electromagnetic problem (DCY), which has proven to be the fastest preconditioner for high Hartmann numbers, see Section 5.2.5. The time step size has been chosen to $\delta t = 0.1$ s for MESH-12.5M and $\delta t = 0.025$ s for MESH-100M. This way, the Courant-Friedrichs-Lewy (CFL) number for MESH-100M is twice the CFL number of MESH-12.5M because the mesh size h is reduced by a factor 8 from MESH-12.5M to MESH-100M. Let us recall that the CFL number is defined as $\text{CFL} = U\frac{\delta t}{h}$ where U is a characteristic velocity of the problem. Table 4 shows the number of solver iterations needed in every nonlinear iteration for the first three time steps of the simulation for both meshes. The results show that, even for a CFL number twice larger, the number of solver iterations does not increase, actually they slightly decrease, when refining the mesh and reducing the mesh size h by a factor of 8.

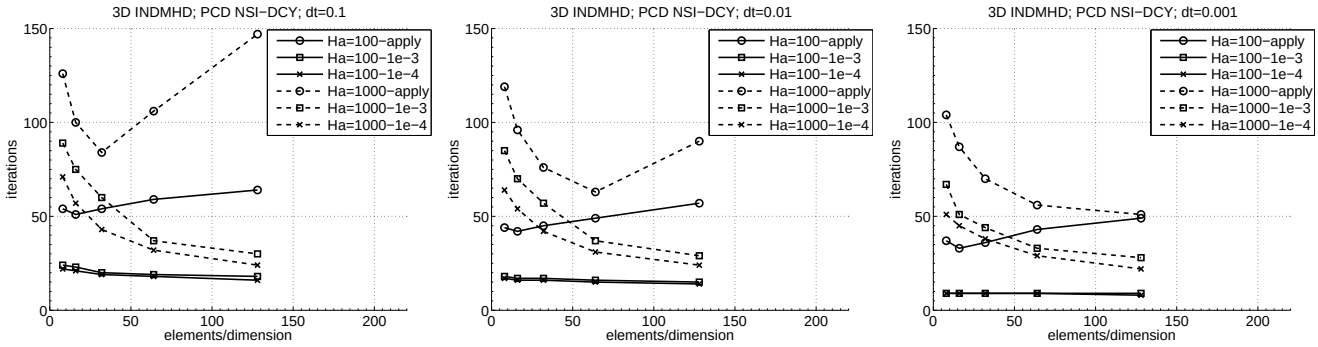
Finally, the transient computation converges to a stationary solution that is shown in Figure 11 for MESH-12.5M. In the top row, Figure 11(a) displays the pressure field in a vertical section and the velocity field at several cross sections of the channels whereas Figure 11(b) shows the electric potential field in the same vertical section as the previous plot and the current density field in cross sections of the channels. Figure 11(c) plots the velocity streamlines along the TBM with a zoom of such streamlines around the turn in Figure 11(d).

5.4. Thermally coupled inductionless MHD flow in a vertical enclosure

This subsection deals with the simulation of the thermally coupled inductionless MHD problem in a vertical enclosure with square section. The computational domain is the volume $[0, 1] \times [0, 1] \times [0, 7.5]$ and it has been discretized with a structured uniform mesh of lineal hexahedral elements containing 48, 48 and 360 elements in x, y and z direction, respectively. This mesh was uniformly partitioned into/distributed over $4 \times 4 \times 10 = 160$ subdomains/cores. Following



(a) Ha=1, 10. Left: $\delta t = 0.1$s, Center: $\delta t = 0.01$s, Right: $\delta t = 0.001$s.



(b) Ha=100, 1000. Left: $\delta t = 0.1$s, Center: $\delta t = 0.01$s, Right: $\delta t = 0.001$s.

Figure 10: External iterations for the PCD FMS block preconditioner.

the indications in [2], the boundary conditions have been set as,

$$
\begin{aligned}
&u_x = u_y = u_z = 0 \text{ at } x = 0, 1 \text{ and } z = 0, 1,\\
&u_y = 0 \text{ at } y = 0, 1,\\
&j_x = 0 \text{ at } x = 0, 1,\\
&j_y = \frac{\Omega}{\mathrm{Ha}} \text{ at } y = 0,\\
&j_y = -\frac{\Omega}{\mathrm{Ha}} \text{ at } y = 1,\\
&j_z = 0 \text{ at } z = 0, 1,\\
&\theta = 0.5 \text{ at } x = 0 \text{ and } \theta = -0.5 \text{ at } x = 1.
\end{aligned}
$$

The gravity field is applied in the z direction and the external magnetic field is horizontal and perpendicular to the temperature gradient $B = (0, B, 0)$ where the magnitude B is computed from the Hartmann number. The adimensional numbers that govern the flow are taken as a Grashof number of $\mathrm{Gr} = 4 \cdot 10^6$, a Prandtl number of $\mathrm{Pr} = 0.025$ and two different Hartmann numbers of $\mathrm{Ha} = 100, 500$.

The linear systems of equations to be solved for every time step and nonlinear iteration have been approximately solved with the preconditioned FGMRES iterative scheme using as preconditioner the recursive block preconditioner $P(A)$ defined in (48) using the block preconditioner P_{FMS} from Section 4.3.1 to approximate the inductionless MHD block. Figure 12 shows the results for a Hartmann number of $\mathrm{Ha} = 100$. Figure 12(a) displays the velocity field whereas the electric potential and the temperature fields are shown in Figures 12(b) and 12(c), respectively. Similarly, the results for the simulation with a Hartmann number of $\mathrm{Ha} = 500$ are plotted in Figures 13(a), 13(b) and 13(c) for the velocity, the electric potential and the temperature fields. In Figure 13 we observe that three vortices appear for low Hartmann numbers, whereas for larger Hartmann numbers the solution presents only one vortex.

	MESH-12.5M	MESH-100M
1st time step	404	366
	304	229
	205	147
	121	111
	74	75
	39	41
2nd time step	248	239
	196	156
	85	70
3rd time step	189	185
	169	123
	53	50

Table 4: Number of solver iterations for the Tecnofus TBM.

6. Software design and implementation

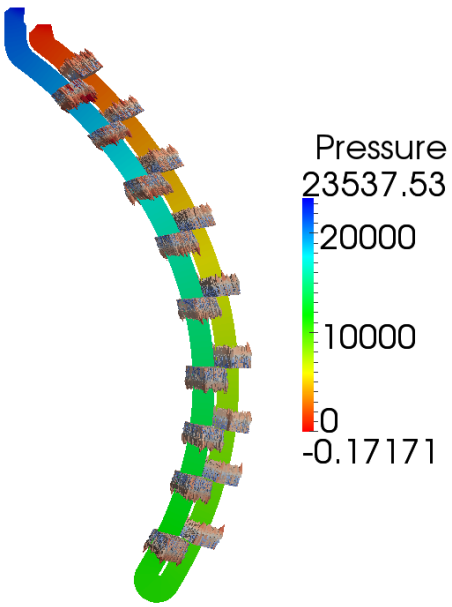
In this section we describe key design guidelines of the software that provides the tools for the code implementation of block recursive preconditioners within FEMPAR. While being applied to the particular context of our simulation software, we expect these guidelines to be very useful for practitioners willing to implement block recursive preconditioning within their computer simulation codes. Some requirements for this software are:

1. It must be *abstract* and *flexible* enough so that it accommodates a large bunch of different block preconditioning strategies in a unified framework, while it is easy to extend with new functionalities without the need of modifying existing code.
2. It must support *recursion* so that the action of the inverses of the diagonals blocks in an approximate block factorization (as might be required e.g., for step 5 of Algorithm 5 at any inner level of the preconditioner hierarchy) can be in turn computed recursively using an approximate block factorization.
3. It must be built on top of the preconditioned iterative solvers available in FEMPAR [8, 6, 7] for the computation of the action of the inverse of a matrix into a vector (as required, e.g., for step 5 of Algorithm 5 in the bottommost level of the preconditioner hierarchy), so that the large bunch code available in FEMPAR for such purpose can be *reused*.

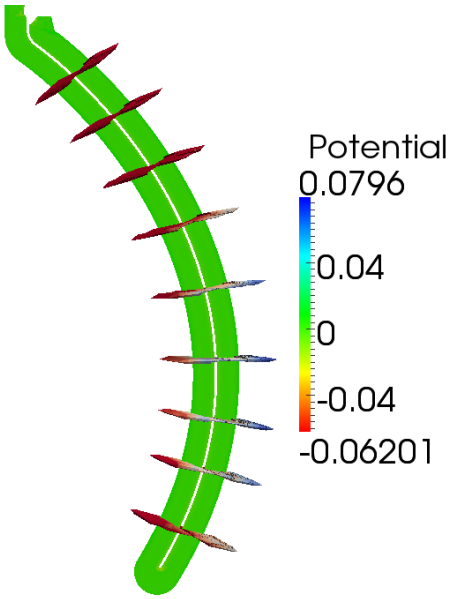
We found the abstractions, design principles and mechanisms provided by the Object-Oriented (OO) software development approach [33] to be particularly useful (if not essential) to meet the aforementioned requirements. Figure 14 illustrates the OO design of the software for block recursive preconditioning as a standard Unified Modeling Language (UML) class diagram. This diagram shows a static, structural view of the software being designed, focusing on its main elements: classes and their relationship. A class is represented as a rectangle containing two compartments, with the top and bottom ones showing the class's name and its methods (also called operations), respectively. Two types of relationships are depicted in Figure 14: *realization* and *aggregation*.[2] We refer the reader to [11] for a comprehensive treatment of UML class diagrams.

To keep the presentation simple, we omitted from Figure 14 the definition of classes (and their relationship) for vectors and sparse matrices. In particular, the Vector class represents a single (discrete) scalar or vector field, e.g., u or p, while Blk_vector represents a block aggregation of several (discrete) scalar or vectorial fields, e.g., $[u, p]$, $[j, \phi]$, or $[j, \phi, u, p]$. In UML terminology, an instance of the Blk_vector class is *composed of* several instances of the Vector class [11]. For reasons made clear below, it is essential for the overall framework that an instance of the Vector class

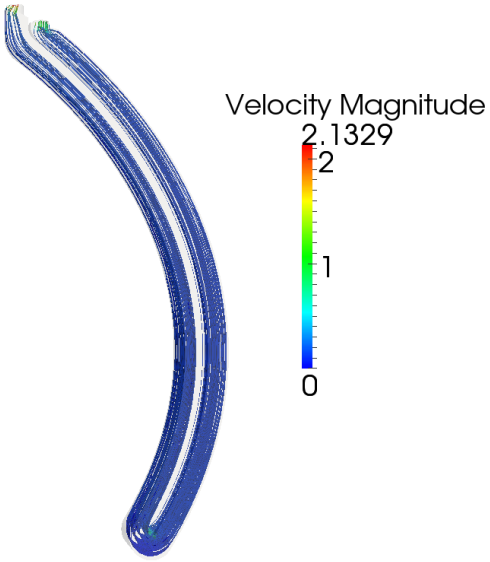
[2] A realization is a relationship among an abstract class and a class (an implementor) that realizes (i.e., implements) the abstract methods provided by the abstract class. Realizations are represented as a solid line among the abstract class and its implementor, with an unfilled triangle pointing to the abstract class; abstract classes and methods have their name depicted *in italics*. On the other hand, in an aggregation, one class (the whole) is a collection or container of another class (the part). An aggregation is depicted as a solid line connecting the whole and the part, with an unfilled diamond on the whole side, and the multiplicity of the aggregation in the part side, i.e., how many instances of the part class are contained in one instance of the whole class.



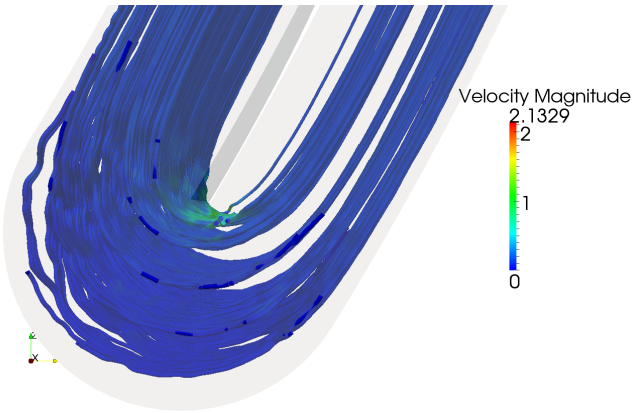
(a) Velocity and Pressure fields



(b) Current Density and Electric Potential fields



(c) Velocity streamlines

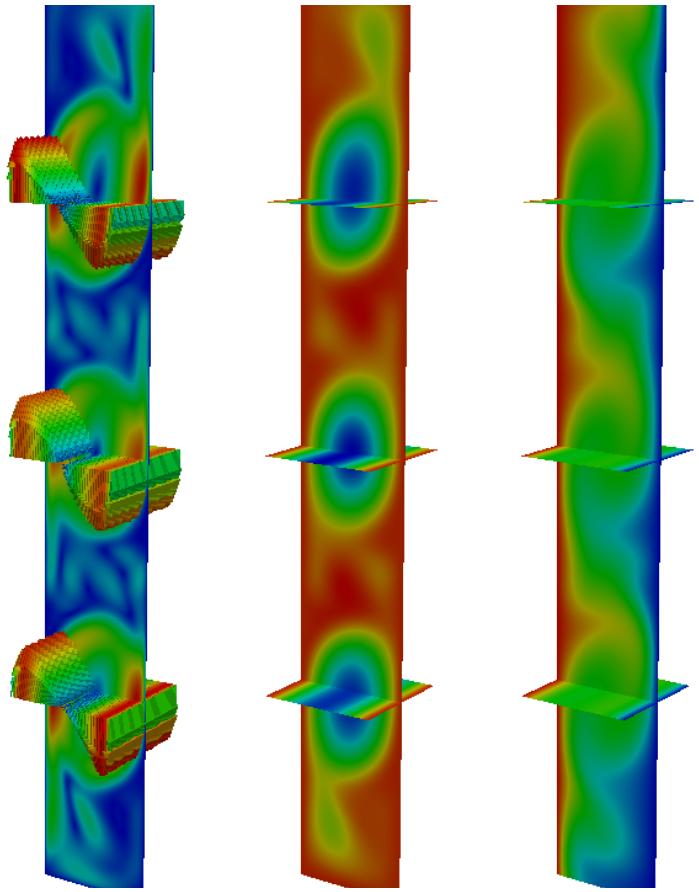


(d) Zoom of velocity streamlines around the turn

Figure 11: Simulation results for the Tecnofus TBM.

can be either created from scratch or from another already created instance. In the former case, new storage space is allocated for the new instance, while in the latter case the new instance shares the storage space with the instance from which it is created. In other words, the former instance is a *view* of the latter instance. A Blk_vector instance can be therefore composed of instances of the Vector class which are *views* of instances previously created. On the other hand, the Matrix class encapsulates a single sparse matrix. The set of methods of these three classes (i.e., Vector, Blk_vector, and Matrix) provide the basic functionality for the implementation of Krylov subspace methods, no matter e.g., how they are stored, nor laid out in a distributed-memory environment.

Central to the design depicted in Figure 14 is the *Operator* abstract class, which represents any linear mapping among vector spaces in its strict mathematical sense. Its application to an instance x of the Vector or Blk_vector classes is returned as an instance y by the *apply_vector* or *apply_blk_vector* abstract methods, respectively. This is the minimal functionality that implementor classes have to realize (i.e., implement) to act as a coefficient matrix or preconditioner

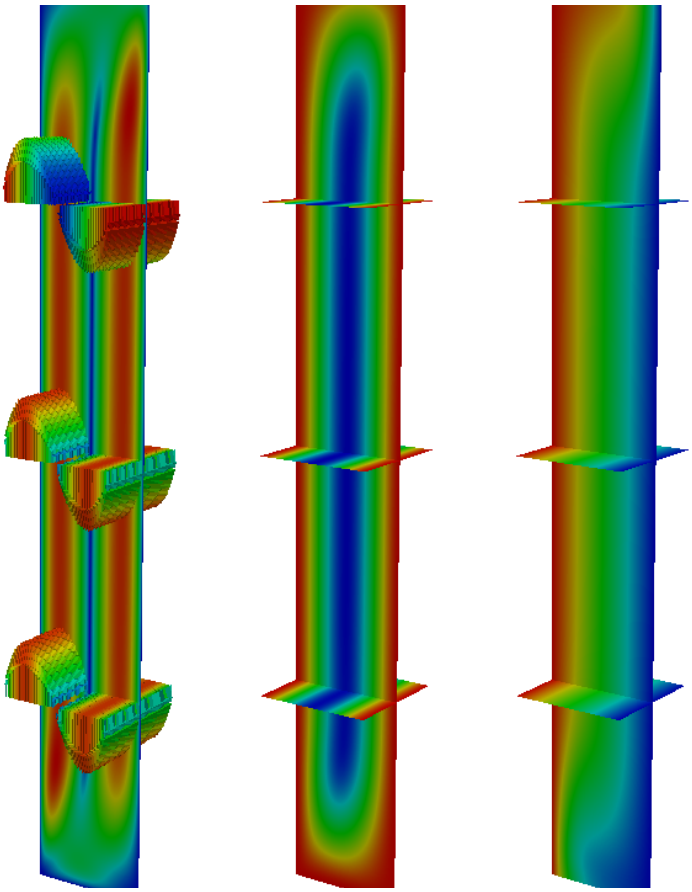


(a) Velocity (b) Electric Potential (c) Temperature

Figure 12: Simulation results for Ha=100.

in preconditioned iterative solvers. The rest of abstract methods of *Operator* are required by implementor classes to implement the *apply_vector* and *apply_blk_vector* abstract methods. The former methods are better grasped by sketching the realization of the latter ones in the implementor classes, which is considered next.

The most basic form of an *Operator* is a matrix, represented by the MatOp implementor class. An instance of the MatOp class is created from an instance of the Matrix class, so that the realization of the *Operator* abstract methods *apply_vector* or *apply_blk_vector* is naively performed as an invocation of the corresponding methods in the Matrix class. A more involved realization of an *Operator* is the InvMatOp class, which represents the (possible approximate) action of the inverse of a matrix computed by means of a preconditioned iterative solver. This class is the common entry point to all preconditioners and iterative solvers available in FEMPAR (see requirement #3 at the beginning of the section). An InvMatOp instance is created from an instance of the Matrix class, and a set of preconditioner and iterative solver parameters. In this work, we make intensive use of the BDDC preconditioner, for which FEMPAR provides highly efficient distributed-memory implementations [8, 7]. Preconditioner parameters for the BDDC method are e.g., the type of continuity constraints enforced, the corner-detection mechanism that ensures the invertibility of the local Neumann and global coarse-grid problems, or the strategy used to deal with the coarse-grid problem (e.g., serialized or overlapped with fine-grid duties [7]). On the other hand, FEMPAR provides templated implementations of fixed-point (e.g., the Richardson method) and Krylov subspace methods (e.g., PCG and GMRES for symmetric positive definite and general unsymmetric linear systems, respectively, and FGMRES to support variable preconditioning). The former methods are used for the computation of rough approximations of the action of the inverse of a matrix into a vector, which will be shown to be useful for some block recursive preconditioners in Section 5. Iterative solver parameters are essentially those that control the convergence criteria (e.g., absolute and relative residual tolerance, maximum number of iterations, etc.), although there are solver-specific parameters, such as the orthogonalization method and the number of iterations for each restart of GMRES.



(a) Velocity (b) Electric Potential (c) Temperature

Figure 13: Simulation results for Ha=500.

The implementor classes SumOp, MulOp, and ScalOp provide the basic building blocks to construct new linear mappings from existing ones as follows: Given two linear mappings A and B, the SumOp and MulOp classes represent the $A+B$ and AB linear mappings, respectively, while given A and a scalar α, the ScalOp represents αA. An instance of each of these classes is built from its *Operators*, as represented by the aggregation relationship among these three classes and the *Operator* abstract class. The reader should now notice that any implementor class (e.g., MatOp or InvMatOp) can appear in place of an *Operator* (this is precisely the potential behind the *realization* relationship), so that *by means of the SumOp, MulOp and ScalOp implementor classes one may build any linear mapping that involves a combination of these three operations*. For example, to build $M_p^{-1}F_pL_p^{-1}$ in (33) using our software design, one first construct two instances of InvMatOp for M_p^{-1} and L_p^{-1}, respectively, and one instance of MatOp for F_p. Then, an instance of the MulOp class is created from F_p and L_p^{-1} to build $F_pL_p^{-1}$, and finally, another MulOp instance is created to build $M_p^{-1}F_pL_p^{-1}$ from a InvMatOp instance (i.e., M_p^{-1}) and a MulOp instance (i.e., $F_pL_p^{-1}$).

The *create_domain_vector* (*create_domain_blk_vector*) and *create_range_vector* (*create_range_blk_vector*) abstract methods play a major role for the realization of the *apply_vector* (*apply_blk_vector*) by the SumOp, MulOp and ScalOp classes. Let us consider for e.g., the implementation of $y := ABx$, where x and y are instances of the Vector class (this is performed by the *apply_vector* method realized by the MulOp class). This operation can in turn be decomposed into $w := Bx$, and $y := Aw$, where w is a workspace instance of the Vector class. For the product to be well-defined, w must be compatible with the range space of B, or equivalently, with the domain space of A. For example, low-level details such as the size, storage or distributed-memory layout must match. The *create_range_vector* and *create_domain_vector* abstract methods provide a new Vector instance w compatible with the range and domain spaces of the *Operator*, respectively. The implementation of these abstract methods for the MatOp, and InvMatOp classes is the one that ultimately determines the size, storage or distributed-memory layout for w (that in turn are extracted from the Matrix class, that internally encapsulates all these low-level details). The rest of implementor classes, such as SumOp, MulOp, and ScalOp, just implement *create_range_vector* and *create_domain_vector* by means of a call to the corresponding

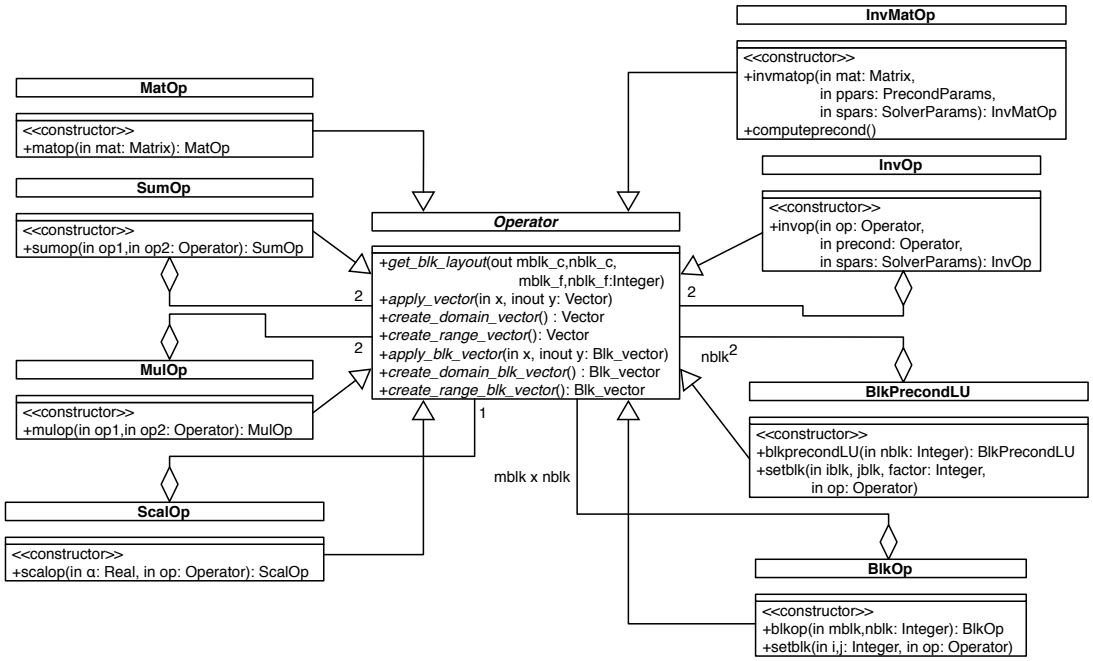


Figure 14: UML class diagram representing the OO design of the software that accommodates block recursive preconditioning within FEMPAR.

method on its leftmost and rightmost *Operator*, respectively.

The BlkOp, BlkPrecondLU, and InvOP classes are the ones that provide our software design with the ability to construct block recursive preconditioners (see requirement #2 at the beginning of the section). The BlkOp implementor class represents a linear mapping which is blocked into $mblk \times nblk$ blocks. A BlkOp instance is built from as many *Operators* as blocks, as represented by the aggregation relationship among BlkOp and the *Operator* abstract method. For example, the coefficient matrix in (23) can be built as a single 4×4 BlkOp instance, with each block being a MatOp. However, as the BlkOp instance in turn realizes the *Operator* abstract class, this coefficient matrix can be alternatively built with a three-level tree-like structure. In the bottommost level, one first builds a MatOp per each of the 4×4 blocks. In an intermediate level, four 2×2 BlkOp instances are then built, each of them corresponding to the four 2×2 blocks delimited by the partitioning lines in (23). Finally, at the topmost level, another 2×2 BlkOp instance is built from the four BlkOp instances in the intermediate level. The realization of the *apply_blk_vector* by the BlkOp class just performs the operation $y := Ax$, where x, y are instances of the Blk_vector class. On the other hand, the BlkPrecondLU represents an approximate block LU factorization (see $P(A)$ in (30)).[3] Its realization of the *apply_blk_vector* performs the operation "Solve $(LU)y = r$", where L and U are lower and upper block triangular factors, respectively, built from $nblk \times nblk$ *Operators* each, and r, y are Blk_vector instances. Finally, the InvOP class realizes *apply_blk_vector* as the action of the (possible approximate) inverse of an *Operator* on a vector, using a preconditioned iterative method with a prescribed preconditioner provided as an *Operator*.

The *get_blk_layout* abstract method is essential for the realization of the *apply_blk_vector* by the BlkOp and BlkPrecondLU classes. Given an *Operator*, this abstract method returns in $mblk_c$ and $nblk_c$ the number of row and column blocks in the coarsest-grain block partitioning of the *Operator*, while $mblk_f$ and $nblk_f$ provide those in the finest-grain one. For instance, in the example of the previous paragraph, it returns $mblk_c = nblk_c = 2$ and $mlbk_f = nblk_f = 4$ for the BlkOp instance in the topmost level of the hierarchy, $mblk_c = nblk_c = mblk_f = nblk_f = 2$ for any of the four BlkOp instances in the intermediate level, and $mblk_c = nblk_c = mblk_f = nblk_f = 1$ for any of the MatOp instances in the bottommost level of the hierarchy. Let us now consider the *apply_blk_vector* in the topmost BlkOp instance, let us call it A. On entry, this method expects x and y to be partitioned into as many blocks as those present in the finest-grain partitioning of the *Operator*, i.e., $nblk_f = 4$ and $mblk_f = 4$, respectively, in this case. In preparation to the call of the *apply_blk_vector* on a given intermediate BlkOp instance, say A_{ij}, the *apply_blk_vector* in A creates a pair of

[3] One can similarly define BlkPrecondD and BlkPrecondU in Figure 14 in order to represent the D-preconditioner and U-preconditioner, respectively, in (28) and (29), although they are omitted from the figure for simplicity.

temporary Blk_vector instances, say x_j and y_i, that are built from those blocks of x and y corresponding to A_{ij}. In other words, the number of blocks of x_j and y_i is given by $nblk_f$ and $mblk_f$ resulting from a call to *get_blk_layout* on A_{ij}, while, x_j (y_i) starts from the block of x (y) with identifier given by the sum of those $nblk_f$'s ($mblk_f$'s) resulting from $j-1$ ($i-1$) calls to *get_blk_layout* on A_{ik} (A_{kj}), with $k = 1, 2, \ldots, j-1$ ($k = 1, 2, \ldots, i-1$). Notice that the blocks of x_j and y_i are created as *views* (see above for the notion of a view) of the corresponding blocks in x and j, so that this mechanism allows the blocks of x and y received on entry to the root of the hierarchy to flow top-bottom thorough the hierarchic deployment of the *apply_blk_vector* method.

The reader might have already observed that this software design accommodates, e.g., the two approximate block recursive preconditioners discussed in Section 4.3. In the topmost level, one creates a 2×2 BlkPrecondU instance, with the two leading diagonal blocks $F_\sharp^{-1}$ and $S_\sharp^{-1}$ defined as InvOp instances, and the upper off-diagonal block G as a BlockOp instance. In the intermediate level, $F_\sharp^{-1}$ is in turn built as an InvOp instance from a 2×2 BlockOp instance (i.e., built from the blocks of $F_\sharp$ as MatOp instances), and a further 2×2 BlkPrecondU to be used as a preconditioner for the preconditioned iterative computation of the action of $F_\sharp^{-1}$ on a vector (see step 6 of Algorithm 6).

Finally, for those developers reluctant to pure OO languages, such as C++ or Fortran2003, let us stress that FEMPAR is a Fortran90/95 code. We were able to implement the design in Figure 14 within FEMPAR, with no loss of functionality, using the techniques discussed in [1] for generic programming and run-time polymorphism emulation in Fortran90/95.

7. Conclusions

In this article we have extended block preconditioning techniques used in computational fluid dynamics to the (thermally coupled) incompressible inductionless magnetohydrodynamics problem. Our approach considers the explicit introduction of the current density as an additional unknown of the problem, in order to end up with a formulation that will be suitable for problems involving large Hartmann numbers, e.g., breeding blanket simulations.

We propose an abstract setting to design preconditioners for multiphysics problems, based on a recursive use of block factorization, that allows us to decouple the computation of every physical variable in a multiphysics problem at the preconditioner level. This idea has been applied to our target problem, (thermally coupled) inductionless MHD problem, (where the unknowns are the velocity, pressure, current density and electric potential) but can also be applied to other problems like resistive MHD [4] or liquid crystal problems [5]. We consider different preconditioners based on approximations of the resulting Schur complement matrices. The robustness of these preconditioners relies on good approximations of the Schur complement matrices that appear in the recursive factorization process. An study of the inductionless MHD system has motivated a first preconditioner that initially decouples fluid and magnetic problems. Next, we propose a method that instead decouples vector fields and Lagrange multipliers at the first level. The assumptions undertaken in this last case have been justified via numerical evidences. As a result of this work, we have also observed the importance to consider stabilization terms in the PCD preconditioners. The recursive preconditioners for both subproblems allow us to obtain block preconditioners with good properties with respect to the mesh size h and the Hartmann number. We give details about an abstract and flexible implementation of block recursive preconditioning.

A detailed set of numerical tests has been performed to assess the properties of the different methods proposed herein. The combination of our FE formulations, with an explicit treatment of the current density, and the recursive *LU* preconditioners we propose, finally allow us to solve realistic breeding blanket simulations with very high Hartmann numbers.

References

[1] J. E. Akin. *Object-oriented programming via Fortran 90/95*. Cambridge University Press, Cambridge; New York, 2003.

[2] G. Authié, T. Tagawa, and R. Moreau. Buoyant flow in long vertical enclosures in the presence of a strong horizontal magnetic field. Part 2. Finite enclosures. *European Journal of Mechanics - B/Fluids*, 22(3):203–220, 2003.

[3] S. Badia. On stabilized finite element methods based on the Scott-Zhang projector. Circumventing the inf-sup condition for the Stokes problem. *Computer Methods in Applied Mechanics and Engineering*, 247-248(0):65–72, 2012.

[4] S. Badia, R. Codina, and R. Planas. On an unconditionally convergent stabilized finite element approximation of resistive magnetohydrodynamics. *Journal of Computational Physics*, 234(0):399–416, 2013.

[5] S. Badia, F. Guillén-González, and J. V. Gutiérrez-Santacreu. Finite element approximation of nematic liquid crystal flows using a saddle-point structure. *Journal of Computational Physics*, 230(4):1686–1706, 2011.

[6] S. Badia, A. F. Martín, and J. Principe. Enhanced balancing Neumann-Neumann preconditioning in computational fluid and solid mechanics. *International Journal for Numerical Methods in Engineering*, 2013. In press.

[7] S. Badia, A. F. Martín, and J. Principe. A highly scalable parallel implementation of balancing domain decomposition by constraints. 2013. Submitted.

[8] S. Badia, A. F. Martín, and J. Principe. Implementation and scalability analysis of balancing domain decomposition methods. *Archives of Computational Methods in Engineering*, 20(3):239–262, 2013.

[9] S. Badia, R. Planas, and J. V. Gutiérrez-Santacreu. Unconditionally stable operator splitting algorithms for the incompressible magnetohydrodynamics system discretized by a stabilized finite element formulation based on projections. *International Journal for Numerical Methods in Engineering*, 93(3):302-328, 2013.

[10] R. Becker and M. Braack. A finite element pressure gradient stabilization for the Stokes equations based on local projections. *Calcolo*, 38(4):173–199, 2001.

[11] G. Booch, J. Rumbaugh, and I. Jacobson. *The unified modeling language user guide*. Addison-Wesley, Upper Saddle River, NJ, 2005.

[12] L. Bühler. Liquid metal magnetohydrodynamics for fusion blankets. In R. M. S. Molokov and H. Moffat, editors, *Magnetohydrodynamics. Historical evolution and Trends*, pages 171–194. Springer, 2007.

[13] J. Cahouet and J. Chabard. Some fast 3D finite element solvers for the generalized Stokes problem. *International Journal for Numerical Methods in Fluids*, 8(8):869–895, 1988.

[14] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2(1):12–26, 1967.

[15] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.

[16] R. Codina. Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 190(13-14):1579–1599, 2000.

[17] R. Codina. Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191(39-40):4295–4321, 2002.

[18] E. C. Cyr, J. N. Shadid, and R. S. Tuminaro. Stabilization and scalable block preconditioning for the Navier-Stokes equations. *Journal of Computational Physics*, 231(2):345–363, 2012.

[19] E. C. Cyr, J. N. Shadid, R. S. Tuminaro, R. P. Pawlowski, and L. Chacón. A new approximate block factorization preconditioner for two-dimensional incompressible (reduced) resistive MHD. *SIAM Journal on Scientific Computing*, 35(3):B701–B730, 2013.

[20] C. R. Dohrmann. A preconditioner for substructuring based on constrained energy minimization. *SIAM Journal on Scientific Computing*, 25(1):246–258, 2003.

[21] H. Elman, V. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 227(3):1790–1808, 2008.

[22] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press, 2005.

[23] T. Hughes, G. Feijóo, L. Mazzei, and J. Quincy. The variational multiscale method–a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166:3–24, 1998.

[24] T. J. Hughes. Multiscale phenomena: Green's functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, 127(1-4):387–401, 1995.

[25] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[26] P. T. Lin, J. N. Shadid, R. S. Tuminaro, M. Sala, G. L. Hennigan, and R. P. Pawlowski. A parallel fully coupled algebraic multilevel preconditioner applied to multiphysics PDE applications: Drift-diffusion, flow/transport/reaction, resistive MHD. *International Journal for Numerical Methods in Fluids*, 64(10-12):1148-1179, 2010.

[27] G. Matthies, P. Skrzypacz, and L. Tobiska. A unified convergence analysis for local projection stabilisations applied to the Oseen problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(4):713–742, 2007.

[28] C. Mistrangelo. Magnetohydrodynamic flow in a mock-up of a HCLL blanket. Part I. Numerical analysis. Technical Report FZKA 7312, Forschungszentrum Karlsruhe, 2008.

[29] U. Müller and L. Bühler. *Magnetofluiddynamics in Channels and Containers*. Springer, 2001.

[30] M.-J. Ni, R. Munipalli, P. Huang, N. Morley, and M. Abdou. A current density conservative scheme for incompressible MHD flows at a low magnetic Reynolds number. Part I: On a rectangular collocated grid system. *Journal of Computational Physics*, 227:174–204, 2007.

[31] M.-J. Ni, R. Munipalli, P. Huang, N. Morley, and M. Abdou. A current density conservative scheme for incompressible MHD flows at a low magnetic Reynolds number. Part II: On an arbitrary collocated mesh. *Journal of Computational Physics*, 227:205–228, 2007.

[32] R. Planas, S. Badia, and R. Codina. Approximation of the inductionless MHD problem using a stabilized finite element method. *Journal of Computational Physics*, 230(8):2977–2996, 2011.

[33] D. Rouson, J. Xia, and X. Xu. *Scientific software design: the object-oriented way*. Cambridge University Press, New York, 2011.

[34] J. Shadid, R. Pawlowski, J. Banks, L. Chacón, P. Lin, and R. Tuminaro. Towards a scalable fully-implicit fully-coupled resistive MHD formulation with stabilized FE methods. *Journal of Computational Physics*, 229(20):7649–7671, 2010.

[35] R. Temam. Sur la stabilité et la convergence de la méthode des pas fractionnaires. *Annali di Matematica Pura ed Applicata*, 79(1):191–379, 1968.