UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Scalable domain decomposition methods for finite element approximations of transient and electromagnetic problems

Marc Olm

Universitat Politècnica de Catalunya

Doctoral Thesis

Scalable domain decomposition methods for finite element approximations of transient and electromagnetic problems

Author:
Marc Olm

Supervisor:
Prof. Santiago Badia

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Doctorat en Enginyeria Civil

Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports de Barcelona

Barcelona, January 2019

Abstract

The main object of study of this thesis is the development of scalable and robust solvers based on domain decomposition (DD) methods for the linear systems arising from the finite element (FE) discretization of transient and electromagnetic problems.

The thesis commences with a theoretical review of the curl-conforming edge (or Nédélec) FEs of the first kind and a comprehensive description of a general implementation strategy for $h-$ and $p-$ adaptive elements of arbitrary order on tetrahedral and hexahedral non-conforming meshes. Then, a novel balancing domain decomposition by constraints (BDDC) preconditioner that is robust for multi-material and/or heterogeneous problems posed in curl-conforming spaces is presented. The new method, in contrast to existent approaches, is based on the definition of the ingredients of the preconditioner according to the physical coefficients of the problem and does not require spectral information. The result is a robust and highly scalable preconditioner that preserves the simplicity of the original BDDC method.

When dealing with transient problems, the time direction offers itself an opportunity for further parallelization. Aiming to design scalable space-time solvers, first, parallel-in-time parallel methods for linear and non-linear ordinary differential equations (ODEs) are proposed, based on (non-linear) Schur complement efficient solvers of a multilevel partition of the time interval. Then, these ideas are combined with DD concepts in order to design a two-level preconditioner as an extension to space-time of the BDDC method. The key ingredients for these new methods are defined such that they preserve the time causality, i.e., information only travels from the past to the future. The proposed schemes are weakly scalable in time and space-time, i.e., one can efficiently exploit increasing computational resources to solve more time steps in (approximately) the same time-to-solution.

All the developments presented herein are motivated by the driving application of the thesis, the 3D simulation of the low-frequency electromagnetic response of High Temperature Superconductors (HTS). Throughout the document, an exhaustive set of numerical experiments, which includes the simulation of a realistic 3D HTS problem, is performed in order to validate the suitability and assess the parallel performance of the High Performance Computing (HPC) implementation of the proposed algorithms.

Resum

L'objecte principal d'estudi d'aquesta tesi és el desenvolupament de solucionadors escalables i robustos basats en mètodes de descomposició de dominis (DD) per a sistemes lineals que sorgeixen en la discretització mitjançant elements finits (FE) de problemes transitoris i electromagnètics.

La tesi comença amb una revisió teòrica dels FE d'eix (o de Nédélec) de la primera família i una descripció exhaustiva d'una estratègia d'implementació general per a elements $h-$ i $p-$adaptatius d'ordre arbitrari en malles de tetraedres i hexaedres no-conformes. Llavors, es presenta un nou precondicionador de descomposició de dominis balancejats per restricció (BDDC) que és robust per a problemes amb múltiples materials i/o heterogenis definits en espais curl-conformes. El nou mètode, en contrast amb els enfocaments existents, està basat en la definició dels ingredients del precondicionador segons els coeficients físics del problema i no requereix informació espectral. El resultat és un precondicionador robust i escalable que preserva la simplicitat del mètode original BDDC.

Quan tractem amb problemes transitoris, la direcció temporal ofereix ella mateixa l'oportunitat de seguir explotant paral·lelisme. Amb l'objectiu de dissenyar precondicionadors en espai-temps, primer, proposem solucionadors paral·lels en temps per equacions diferencials lineals i no-lineals, basats en un solucionador eficient del complement de Schur d'una partició multinivell de l'interval de temps. Seguidament, aquestes idees es combinen amb conceptes de DD amb l'objectiu de dissenyar precondicionadors com a extensió a espai-temps dels mètodes de BDDC. Els ingredients clau d'aquests nous mètodes es defineixen de tal manera que preserven la causalitat del temps, on la informació només viatja de temps passats a temps futurs. Els esquemes proposats són dèbilment escalables en temps i en espai-temps, és a dir, es poden explotar eficientment recursos computacionals creixents per resoldre més passos de temps en (aproximadament) el mateix temps transcorregut de càlcul.

Tots els desenvolupaments presentats aquí són motivats pel problema d'aplicació de la tesi, la simulació de la resposta electromagnètica de baixa freqüència dels superconductors d'alta temperatura (HTS) en 3D. Al llarg del document, es realitza un conjunt exhaustiu d'experiments numèrics, els quals inclouen la simulació d'un problema de HTS realista en 3D, per validar la idoneïtat i el rendiment paral·lel de la implementació per a computació d'alt rendiment dels algorismes proposats.

Acknowledgements

This work would never have been done without the guidance of my supervisor Santiago Badia. I am very grateful for the opportunity of developing a thesis with such a brilliant researcher and for the availability, patience and support showed towards his PhD candidates. I had the pleasure to share this long journey with a wonderful group of researchers at the Large Scale Scientific Computing group of CIMNE. A sincere thanks to Javier, for his valuable advice and kind support, and specially to Alberto, who made things easier from the first day until the last one with his infinite patience sharing his deep expertise and his personal support.

A deep thanks to all the group members who have passed during these years by the office 214. Specially to Oriol and Alba, who introduced me to the field of research and the PhD community in the first half of this journey, and to Jesús and Eric, who have been insuperable day by day workmates in this (more conscious) second stage of the PhD. Thanks to Ramon, Víctor, Francesc, Àlex, Pere, Manuel, they all have been always ready for discussion and willing to help. I am proud to say that I could develop this thesis in an environment of excellence and cooperation, and all the group members contributed to this fact.

In a more personal side, I would like first to thank Josep and Gerard. It has been a while, but 5 years back I began this journey with their unconditional support. There are two groups of people, university and home town friends, who were there before the starting of this thesis, have been there throughout its development and I hope will be there for many more years. Thanks to all of them for making life serious but fun. Thanks to all the folks from CIMNE, specially to those ones from the Pigs team, with whom I could share other passions beyond research.

A deep thanks to my beloved family, specially to my parents Josep M^{a} and Dolors, for giving me all the love, unconditional support and freedom to take my own path. This thesis would have not even commenced without them. To my twin, who has been there day by day from the first day till the last one bringing home wherever we are. And finally to Mercè, for her kind patience, love, encouragement and joy, building home wherever we are.

I gratefully acknowledge the financial support received from the Catalan government through the FI AGAUR grant.

Contents

Abstract iii

Resum v

Acknowledgements vii

1 Introduction 1
- 1.1 Motivation . . . 6
- 1.2 Thesis objectives . . . 7
- 1.3 Structure of the document . . . 10
- 1.4 Conference and workshop presentations . . . 12

2 Curl-conforming Finite Elements 15
- 2.1 Introduction . . . 15
- 2.2 Problem statement . . . 18
  - 2.2.1 Notation . . . 18
  - 2.2.2 Formulation . . . 19
- 2.3 Edge FEs . . . 20
  - 2.3.1 Reference cell . . . 21
  - 2.3.2 Polynomial spaces . . . 21
    - Construction of polynomial spaces . . . 21
    - Hexahedra . . . 22
    - Tetrahedra . . . 23
  - 2.3.3 Edge moments in the reference element . . . 24
    - Hexahedra . . . 25
    - Tetrahedra . . . 26
  - 2.3.4 Construction of edge shape functions . . . 27
- 2.4 Global FE spaces and conformity . . . 29
  - 2.4.1 Moments in the physical space and the Piola map . . . 29
    - Hexahedra . . . 30
    - Tetrahedra . . . 30
  - 2.4.2 Nédélec interpolator . . . 31
  - 2.4.3 Global degrees of freedom (DOFs) . . . 31
- 2.5 Edge FEs in h-adaptive meshes . . . 33
  - 2.5.1 Hierarchical adaptive mesh refinement (AMR) on octree-based meshes 33

2.5.2 Conformity of the global FE space . . . 34
Constraints for Lagrangian FEs . . . 35
Constraints for edge FEs . . . 36
Global assembly for non-conforming meshes . . . 38
2.6 Numerical experiments . . . 38
2.6.1 Uniform mesh refinement . . . 39
Hexahedral meshes . . . 40
Tetrahedral meshes . . . 41
2.6.2 h-adaptive mesh refinement . . . 41
2.7 Conclusions . . . 45

3 Balancing Domain Decomposition by Constraints solvers for heterogeneous problems in $H(\mathbf{curl})$ 47
3.1 Introduction . . . 47
3.2 Problem setting . . . 50
3.2.1 Domain partition . . . 51
3.2.2 Finite Element spaces . . . 51
3.2.3 Objects . . . 52
3.3 Physics-Based BDDC . . . 52
3.3.1 Change of basis . . . 52
3.3.2 Preconditioner . . . 54
3.3.3 Perturbed physics-based BDDC (PB-BDDC) preconditioner . . . 56
3.3.4 Relaxed PB-BDDC . . . 56
3.4 Implementation aspects . . . 57
3.4.1 Coarse Edge partition . . . 58
3.4.2 Change of basis . . . 59
3.4.3 BDDC constraints . . . 63
3.4.4 Building Θ^r_{pb} . . . 64
3.5 Numerical results . . . 65
3.5.1 Experimental framework . . . 66
3.5.2 Homogeneous problem . . . 66
3.5.3 Multi-material problem . . . 67
Checkerboard distribution . . . 67
Multiple channels . . . 70
3.5.4 Heterogeneous problems . . . 72
Periodic analytical functions . . . 72
High Temperature Superconductors . . . 74
3.6 Conclusions . . . 76

4 Simulation of High Temperature Superconductors and experimental validation 79
4.1 Introduction . . . 79

4.2 The system of equations . . . 82
4.2.1 Notation . . . 82
4.2.2 Maxwell equations in electromagnetics . . . 82
4.2.3 The H-formulation . . . 83
4.2.4 Transmission conditions . . . 84
4.2.5 Material modelling . . . 84
4.3 Numerical approximation . . . 85
4.3.1 Time discretization . . . 86
4.3.2 Dirichlet Boundary Conditions . . . 86
4.3.3 Mesh generation by hierarchical AMR . . . 86
4.4 Nonlinear transient solver . . . 87
4.4.1 Algebraic form . . . 87
4.4.2 Adaptive time stepping . . . 88
4.4.3 Linearization . . . 88
4.4.4 Parallel linear solver . . . 90
4.5 Numerical experiments . . . 91
4.5.1 Software . . . 91
4.5.2 Experimental framework . . . 92
4.5.3 Comparison against experimental data . . . 92
4.5.4 3D benchmark . . . 94
4.6 Conclusions . . . 102

5 Nonlinear parallel-in-time solvers for ordinary differential equations 103
5.1 Introduction . . . 103
5.2 Statement of the problem . . . 105
5.3 An ODE direct solver . . . 106
5.3.1 Application to different time integrators . . . 108
5.3.2 Parareal interpretation . . . 109
5.3.3 Parallel efficiency . . . 110
5.4 Iterative solvers for nonlinear ODEs . . . 112
5.4.1 Newton-Schur complement methods . . . 112
5.4.2 Nonlinear Schur complement-Newton methods . . . 113
5.5 Numerical experiments . . . 116
5.5.1 Experimental set-up . . . 116
5.5.2 Two-level solvers . . . 117
5.5.3 Multilevel solver . . . 118
5.6 Conclusions . . . 120

6 Space-Time Balancing Domain Decomposition by Constraints preconditioners 123
6.1 Introduction . . . 123
6.2 Problem setting . . . 125

6.2.1 Domain partitions . . . 125
6.2.2 Space-time discretization . . . 125
6.3 Space BDDC preconditioning . . . 126
6.3.1 Sub-assembled problem . . . 127
6.3.2 Coarse DOFs . . . 127
6.3.3 Transfer operator . . . 128
6.3.4 Space-parallel preconditioner . . . 128
6.4 Space-time BDDC preconditioning . . . 129
6.4.1 Sub-assembled problem . . . 129
6.4.2 Coarse DOFs . . . 131
6.4.3 Transfer operator . . . 131
6.4.4 Space-time-parallel preconditioner . . . 133
6.4.5 Implementation aspects . . . 133
6.5 Numerical experiments . . . 135
6.5.1 Experimental framework . . . 135
6.5.2 Weak scalability setting . . . 135
6.5.3 Weak scalability in Time . . . 136
Time-parallel Poisson solver . . . 137
Time-parallel CDR solver . . . 137
6.5.4 Weak scalability in space-time . . . 138
Space-time Poisson solver . . . 140
Space-time CDR solver . . . 142
6.5.5 Nonlinear space-time p-Laplacian solver . . . 143
6.6 Conclusions . . . 144

7 Conclusions and future work **147**
7.1 Conclusions . . . 147
7.2 Open lines of research . . . 149

List of Figures

2.1 3D hexahedral reference FE. 26
2.2 3D tetrahedral reference FE. 27
2.3 Vector-field plots of the shape functions in the lowest order 2D hexahedra. The indices of edges follow the geometrical entities indices for the reference hexahedron in Fig. 2.1a restricted to the plane $z = 0$. Auxiliary circles denote the geometrical entity where the moment is defined. 28
2.4 Vector-field plots of the shape functions in the second order hexahedral edge element. The indices of edges follow the geometrical entities indices for the reference hexahedron in Fig. 2.1a restricted to the plane $z = 0$. Auxiliary circles denote the geometrical entity where the moment is defined. 28
2.5 Vector-field plots of the shape functions in the lowest order tetrahedral element. The indices of edges follow the geometrical entities indices for the reference tetrahedron in Fig. 2.2a restricted to the plane $z = 0$. Auxiliary circles denote the geometrical entity where the moment is defined. 29
2.6 Vector-field plots of the shape functions in the second order edge element. The indices of edges follow the geometrical entities indices for the reference tetrahedron in Fig. 2.2a restricted to the plane $z = 0$. Auxiliary circles denote the geometrical entity where the moment is defined. 29
2.7 Oriented mesh. Global information (left) and its local, oriented, counterpart (right) for two elements K_i and K_j. Common edges for two adjacent tetrahedra will always agree in its direction, as well as chosen tangent vectors $\boldsymbol{\tau_1}$ and $\boldsymbol{\tau_2}$ to the common face. 32
2.8 h-adaptive refined (single octree) *non-conforming* meshes. 34
2.9 The coarse cell K_{2h} is refined in order to meet the highest level of refinement. Then, the index $s(g)$ can be determined. 35
2.10 Scheme for enforcing continuity at common entities for two cells with different level of refinement with second order Lagrangian FEs. 36
2.11 Scheme for enforcing continuity at common entities for two cells with different level of refinement with second order edge FEs. 37
2.12 Sequence of spaces in order to compute the restriction operator entries for K_h^2 with second order edge FEs. Representation of edge and inner DOFs by arrows and nodes, resp. 37
2.13 Analytical functions. 40
2.14 Error norms for different orders 2D hexahedral edge FEs. 40
2.15 Error norms for different orders 3D hexahedral edge FEs. 40

2.16 Error norms for different orders 2D tetrahedral edge FEs. 41
2.17 Error norms for different orders 3D tetrahedral edge FEs. 41
2.18 fichera 2D problem. At each mesh refinement step the 5% of cells with highest local cell contribution to the L_2-norm of the error are refined. . . . 42
2.19 Error norms for the fichera 2D problem with uniform refinement for $n = 1$. 43
2.20 Error norms for the fichera 2D problem with uniform refinement for $n = 4$. 43
2.21 Error norms for the fichera 2D with $n = 1$ and adaptive refinement, which at every iterate marks for refinement the 5% of cells that show the highest local cell L_2-norm of the error. Lines without markers show the error convergence with uniform refinement process for every FE order. 44
2.22 Error norms for the Fichera 3D problem with uniform (straight line) and adaptive refinement. 5% of cells that have the highest local L_2-norm of the error are marked for refinement at every refinement step. 44
2.23 Adaptive meshes for the Fichera 3D problem. Cells with highest 5% Local L_2-error(K) are refined at each refinement step. 45

3.1 Standard (old) and new basis DOFs for 3D hexahedra second order edge FE over E. Additional depicted DOFs values are affected by the change of basis for E, while the remaining DOFs are invariant under the change of basis. 62
3.2 Partitions for scalar coefficients described by $\log(\alpha) = 3\sin(3\pi x)$ and $\log(\beta) = 3\sin(3\pi y)$ with an initial $3 \times 3 \times 3$ partition of the unit cube. . . 65
3.3 Aggregation of cells into subsets based on $log(\beta) = 3\sin(3\pi(y))$ with threshold $r = 10^3$. 65
3.4 Weak scalability results for first order edge FE with a constant distribution of materials for different local problem $\frac{H}{h}$ sizes. 67
3.5 Weak scalability results for different order edge FE with a constant distribution of materials and a local problem $\frac{H}{h} = 10$. 68
3.6 Bi-material distribution cases in a structured $3 \times 3 \times 3$ partition of the unit cube $[0, 1]^3$. 68
3.7 Number of iterations for first order edge FE with a $3 \times 3 \times 3$ partition of the unit cube and $H/h = 8$. A checkerboard arrangement of materials is defined: $\alpha_{\text{white}}1.0$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^i$ and $\beta_{\text{black}} = 10^{-i}$, leading to a contrast $= \frac{\alpha_{\text{black}}}{\beta_{\text{black}}} = \frac{10^i}{10^{-i}} = 10^{2i}$. Labels include scaling information, where P denotes perturbation of the preconditioner. 69
3.8 Weak scalability results for first order edge FE with a checkerboard distribution of materials: $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$. 70
3.9 Weak scalability results for different order edge FE and $H/h = 4$ with a checkerboard distribution of materials: $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$. 70

3.10 Sphere partition and distribution of materials. 71
3.11 Robustness for a tetrahedral mesh for different scalings. Material parameters defined as $\alpha_{\text{white}} = 10^i$, $\alpha_{\text{black}} = 1.0$, $\beta_{\text{white}} = 10^{-i}$ and $\beta_{\text{black}} = 1.0$, contrast defined $\frac{\alpha_{\text{white}}}{\beta_{\text{white}}}$. Mesh partitioned into 20 subdomains and random assignment of materials. Labels include scaling information, where P denotes perturbation of the preconditioner. 71
3.12 Weak scalability results for first order edge FE with a channel distribution of materials: $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$. 73
3.13 Weak scalability results for different order edge FE and $H/h = 4$ with a channel distribution of materials: $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$. 73
3.14 Weak scalability for the relaxed PB-BDDC (rPB-BDDC) when only an heterogeneous α is considered, $\beta = 1.0$. 74
3.15 Weak scalability for the rPB-BDDC when only an heterogeneous β is considered, $\alpha = 1.0$. 74
3.16 Weak scalability for the rPB-BDDC when both coefficients are heterogeneous. 74
3.17 Weak scalability for the first linear solver in the HTS problem with $r = 10^2$. 76
3.18 Domain and HTS device for $t = 4$ ms. 77

4.1 Validation problem inputs definition. 94
4.2 Magnetic field $B_y = \mu_0 H_y$ profiles for experimental (Hall probe mapping) and computed data in a full load-unload cycle for the validation test. . . . 95
4.3 Illustration of Mesh 2 (see Tab. 4.2). Refinement pattern is common to all meshes. Adaptive refinement technique results in a structured mesh for the HTS device, while a smart coarsening following the 2:1 ratio can be observed in the dielectric domain surrounding the superconducting region. 97
4.4 Distribution of normalized current densities for the bulk (top) and the stack (bottom). 98
4.5 Magnetization loops for the models. Magnitudes are normalized with critical current density and HTS device size. 98
4.6 Instantaneous power dissipation in the bulk and the stack. 99
4.7 Normalized $\boldsymbol{J}$ components over a line in the y-axis direction that passes through $z = 0$ with a fixed mesh and variable FE orders, i.e., p-refinement. 100
4.8 Normalized $\boldsymbol{J}$ components over a line in the y-axis direction that passes through $z = 0$ with first order FEs and different uniformly refined meshes, i.e., h-refinement. 100
4.9 $\boldsymbol{J}$ components over a line in the y-axis direction that passes through $z = 0$ for different FE orders (p-refinement) and meshes (h-refinement). 100

5.1 Multilevel time partition of $[0, T]$. The subindex in t_α^β denotes the partition level, whereas the superindex denotes the time step in such partition. In order to relate time values of two constitutive levels, we use the notation $t_\alpha^\beta = t_{\alpha-1}^{m(\beta)}$, i.e., the time value t_α^β corresponds to the time step $m(\beta)$ at the previous level. 105

5.2 Coarse shape functions $\mathbf{e}_{(1)}$ and $\mathbf{f}_{(1)}$ ($t_1^1 = \pi/3$) for the simple transport operator $\frac{du}{dt}$. A partition of an interval $[0, \pi]$ into three subdomains is considered. Each subdomain is partitioned into 5 time elements. Subintervals are depicted in consecutive different colour in order to aid visualization, and dots aid to identify coarse DOFs position (in the center of the element for DG(0)). 110

5.3 Decomposition of $\mathbf{u}_0$ into fine $\mathbf{E}_0\mathbf{u}_0$ and coarse $\mathbf{u}_1$ component for the simple problem $\partial_t u = \cos(t)$, on $[0, \pi]$. The time domain is partitioned into 10 subdomains and each subdomain is an aggregation of 50 elements. 111

5.4 Iterations for the solution of the nonlinear equation $\partial_t u - u^2 = \cos(\mathrm{t}) - \sin^2(\mathrm{t})$ on $t = [0, 2\pi]$ using Newton's method and the parallel ODE direct solver. Fine and coarse solutions for the first nonlinear solution update δu^1. The time interval is discretized with 500 time steps divided into 15 subdomains. DG(0) (equivalent to Backward-Euler) is used. 113

5.5 Solution to the Lotka-Volterra equations when different scenarios are considered. 117

5.6 Weak scalability for local problem size $n_0/n_1 = 50$. The problem is solved with $\alpha = 3$, $\beta = 0.2$, $\delta = 0.1$, $\gamma = 2$ and $t \in (0, 3]$. Initial guess $u(0) = 10$, $v(0) = 40$. 118

5.7 Weak scalability for $n_0/n_1 = 100$. The problem is solved with $\alpha = 3$, $\beta = 0.2$, $\delta = 0.1$, $\gamma = 2$ and $t \in (0, 3]$. Initial guess $u(0) = 10$, $v(0) = 40$. 118

5.8 Three-level ($\ell = 2$) Schur complement solver with different local sizes. . . 119

5.9 Comparison between the two-level ($\ell = 1$) and the three-level ($\ell = 2$) Schur complement solver with different local sizes. Times for local solvers in the level-0, local solvers in the level-1 and the Schur complement solve in level-2. 120

5.10 Comparison between the fixed coarsening and the adaptive coarsening between level-1 and level-2 for different level-0 local problem sizes. The adaptive coarsening uses $n_2 = n_1/n_2 = \sqrt{n_1}$. 120

6.1 Continuity to be enforced among space-time subdomains, for the 1-dimensional spatial domain case. The sets of nodes in red are related to the space constraints time averages over time sub-intervals in Eq. (6.11), the ones in blue are the space mean value constraints on time interfaces in Eq. (6.12), and the ones in orange are spatial constraints on time interfaces in Eq. (6.13). 132

6.2 Weak scalability for the iterations count (left) and total elapsed time (right) of the TBDDC-generalized minimal residual method (GMRES) solver in the solution of the unsteady 2D Poisson problem on HLRN. Fixed element size $h = 1/30$ while time partition on P_t subdomains. . . . 138
6.3 Weak scalability for the total elapsed time (right) and number of GMRES iterations (left) of the TBDDC solver in the solution of the 2D convection-diffusion-reaction (CDR) equation on HLRN (sinusoidal solution). Fixed element size to $h = 1/30$. 139
6.4 Weak scalability for the total elapsed time (right) and number of GMRES iterations (left) of the TBDDC solver in the solution of the 2D CDR equation (boundary layer) on HLRN. Fixed element size to $h = 1/30$. . . . 139
6.5 Weak scalability for the total elapsed time (right) and number of GMRES iterations (left) of the space-time (ST)BDDC solver in the solution of the 2D Poisson problem on HLRN. 140
6.6 Weak scalability for the total elapsed time (right) and number of GMRES iterations (left) of the STBDDC solver in the solution of the 2D Poisson problem on HLRN. Partition is done equally in time and space, i.e., $P_x = P_t$. 141
6.7 Comparison between the sequential and space-time solvers for the transient Poisson problem on MareNostrum supercomputer. Spatial partition is fixed to $P_x = P_y = 4$. The space-time approach is using $P = 4^2 P_t$ MPI fine-level tasks. 142
6.8 Comparison between the sequential and space-time solvers for the p-Laplacian transient problem on MareNostrum supercomputer. Spatial partition is fixed to $P_x = P_y = 4$. The space-time approach is using $P = 4^2 P_t$ MPI fine-level tasks. 143

List of Tables

3.1 Weak scalability in terms of number of iterations for both preconditioners. Checkerboard distribution of materials with $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$. 69

3.2 Comparison in number of iterations for both preconditioners in a $3 \times 3 \times 3$ partition. Channel distribution of materials with $\gamma = 0.5$ and $\alpha_{\text{black}} = \beta_{\text{black}} = 1.0$, $\alpha_{\text{white}} = 10^i$ and $\beta_{\text{white}} = 10^{-i}$. Contrast defined as $c = \frac{\alpha_{\text{white}}}{\beta_{\text{white}}}$. 72

3.3 Average metrics for the simulation of the time interval $T = [0, 5]$ ms. c_0 denotes the num. of coarse DOFs of the original, geometrical partition. . 76

4.1 Geometric and electrical parameters of the HTS tape used in different problems. Domain units depend on domain dimension d. 93

4.2 Summary of meshes used to reproduce the benchmark. The superscript r denotes the number of uniform refinements applied to every cell of the original mesh. 96

4.3 Comparison of alternating currrents (AC) losses (in mJ) in the bulk and in the stack calculated with two different methods against benchmark results. 97

4.4 Computing times for problem solved using Mesh 2. Iteration and free DOF counters show average values. Simulation ends after 246 converged time steps involving 976 linearized problem solves. 101

4.5 Computing times for problem solved using Mesh 3. Iteration and free DOF counters show average values. Simulation ends after 459 converged time steps involving 2374 linearized problem solves. *Serial run time is computed with an extrapolation with the number of linearized problem solves after 3 days of computation due to limited computing time in the access to MN-IV. 101

6.1 Iteration count for CDR equation. Convective CFL equal to 1.0 in all cases. CFL_ν represents the diffusive CFL. 143

List of Abbreviations

AC	**A**lternating **C**urrent
AMR	**A**daptive **M**esh **R**efinement
BDDC	**B**alancing **D**omain **D**ecomposition by **C**onstraints
BDF	**B**ackward **D**ifferentiation **F**ormula
BE	**B**ackward **E**uler
CDR	**C**onvection **D**iffusion **R**eaction
CB	**C**ubic **B**acktracking
CFL	**C**ourant **F**riedrichs-**L**ewy
CG	**C**onjugate **G**radient
CPU	**C**entral **P**rocessing **U**nit
CSE	**C**omputational **S**cience and **E**ngineering
DC	**D**irect **C**urrent
DD	**D**omain **D**ecomposition
DG	**D**iscontinuous **G**alerkin
DOF	**D**egre **O**f **F**reedom
DP	**D**ual **P**rimal
FE	**F**inite **E**lement
FEM	**F**inite **E**lement **M**ethod
FETI	**F**inite **E**lement **T**earing and **I**nterconnect
FLOPs	**FL**oating point **OP**erations
GLOB	**Gl**obal **Ob**ject
GMRES	**G**eneralized **M**inimal **R**esidual **M**ethod
HPC	**H**igh **P**erformance **C**omputing
HTS	**H**igh **T**emperature **S**uperconductors
MG	**M**ulti**G**rid
MLBDDC	**M**ulti**L**evel **B**alancing **D**omain **D**ecomposition by **C**onstraints
MPI	**M**essage **P**assing **I**nterface
MRI	**M**agnetic **R**esonance **I**maging
NR	**N**ewton-**R**aphson
ODE	**O**rdinary **D**ifferential **E**quation
OO	**O**bject **O**riented
PB	**P**hysics **B**ased
PB-BDDC	**P**hysics **B**ased **B**alancing **D**omain **D**ecomposition by **C**onstraints
PDE	**P**artial **D**ifferential **E**quation
rPB-BDDC	**r**elaxed **P**hysics **B**ased **B**alancing **D**omain **D**ecomposition by **C**onstraints

rPB	**r**elaxed **P**hysics **B**ased
RK	**R**unge **K**utta
ST	**S**pace-**T**ime
STBDDC	**S**pace-**T**ime **B**alancing **D**omain **D**ecomposition by **C**onstraints
VEF	**V**ertex **E**dge **F**ace

To my parents

Chapter 1

Introduction

Computational tools provide the capability to simulate the behaviour of complex systems and natural phenomena. They allow us to explore fields that are either inaccessible to empirical discovery or where experimentation in design and optimization is prohibitively expensive. Indeed, computation is nowadays considered to be the third pillar of the scientific method, alongside with theory and experiments. This field of research and applications, which is known as Computational Science and Engineering (CSE), is often defined as the discipline found at (but not restricted to) the intersection among applied mathematics, computer science and specialized knowledge in engineering or science.

CSE relies on the fact that we can define mathematical models that describe the behaviour of a system in terms of partial differential equations (PDEs), for which analytical solutions are not available in the vast majority of cases. Fortunately, discretization techniques allow to approximate the solution of these equations. Actually, computational models are ultimately discrete approximations of continuous phenomena. There are several approaches to discretize the problem in order to obtain a solution described by a finite number of values. In this thesis, the finite element (FE) method, which has a strong mathematical foundation behind, is selected. The main idea behind FE method (FEM) is to look for the approximation of the solution in a finite dimensional vector space, which is built on top of a tessellation of the computational domain. These vector spaces are usually spanned by sets of (scalar or vector-valued) polynomial functions, which are denoted as the bases of shape functions. Then, the approximation of the solution is fully determined by a linear combination of the basis of shape functions with a set of coefficients, which are the unknowns of the problem. The FE method requires to define a tessellation of the computational domain into elements, the finite dimensional space and its spanning set of basis functions. The concept of FE is precisely the restriction of those ingredients to one element from the tessellation.

The best known and widespread FE method uses the Lagrangian FE, where unknowns represent discrete values of the approximation at some points (nodes) of the computational domain. Sometimes this element is viewed as the standard FE method, but in fact there is a whole family of different elements that allow us to properly exploit the mathematical particularities of different problems. While Lagrangian elements are the common choice in structural mechanics, the Raviart-Thomas FE is the preferred choice for some porous media flow and elasticity equations. In this sort of FEs, unknowns

represent fluxes across the faces of the grid covering the computational domain. In turn, edge (or Nédélec) elements are a natural choice in the simulation of electromagnetic problems due to their sound mathematical structure, and will have a key role in the present thesis. They allow to model vector fields with continuous tangential components and discontinuous normal components for the approximation of the solution. In fact, they receive its name because for first order approximations each degree of freedom (DOF) is associated to each edge of the mesh covering the computational domain, and DOFs can be understood as circulations along edges. Thus, FE methods are based on piecewise polynomial approximations of the solution of the problem. As mentioned, the method employs a tessellation of the domain into elements, which can be of variable size (h), and the definition of the polynomial functions, which can be of variable polynomial degree (p). The accuracy of the method, i.e., the error of the approximation with regard to the solution of the problem, is dependent on both h and p. It is well known that suitable combinations of h-refinements, i.e., dividing some of the elements into smaller ones, and p-refinements, i.e., increasing the polynomial degree of some of the elements, allow either to optimize the number of unknowns for a required goal precision or to increase the accuracy of the method for a fixed number of unknowns. The FE discretization of the (system of) PDEs that governs a given problem results in a system of algebraic equations, i.e. $Ax = b$, whose solution is the vector of unknowns (i.e., DOFs) that fully describes the approximation to the solution.

The need of High Performance Computing (HPC) appears when a particular application exceeds the memory capacity of standard computers or time-to-solution exceeds reasonable computing times. The main idea behind HPC is to use aggregated computing power from several different units, thus requiring the concept of algorithm parallelization. Unless an HPC tool is employed, no meaningful results can be obtained for realistic FEM models for the most CPU demanding applications, e.g., climate modelling, geophysics, plasma physics, turbulent flows, combustion or biomedical applications. In addition, more classical computational applications such as structural and fluid mechanics are always under constant evolution in order to face increasingly complex problems and can also leverage higher levels of parallelism.

At the beginning of next decade supercomputers are expected to reach a peak performance of one exaflop/s (10^{18} floating point operations per second), which implies a 100 times improvement with respect to current supercomputers. But since recent years, the clock speed has stagnated and the increase of performance is not based on faster processors, but on a much higher number of processors. This increasing growth in computational power should go hand in hand with parallel algorithms that exploit further levels of concurrency.

The solution of the global system arising from the FE discretization of the PDEs governing the problem is the main bottleneck of the simulation pipeline. The use of direct solvers, which usually rely on an efficient factorization of the matrix A, is prohibitively

expensive at a sufficiently fine scale due to their high computational and memory demands. Hard to parallelize by its nature, its complexity and scalability issues make the approach impractical for large scale simulations. On the other hand, iterative solvers do not rely on the expensive factorization of the matrix A but on evaluations of simple matrix-vector products Ax, which can be naturally implemented in distributed-memory environments with few communications among processors. In the simulation pipeline of parallel FE codes on memory-distributed platforms, every processor can receive a part of the mesh, which covers the entire computational domain. Then, every processor is in charge of discretizing the FE problem for its local counterpart, which is an embarrassingly parallel task (i.e., can be fully performed in parallel). Hence, the (non)linear system arising from the FE discretization of the corresponding PDE can be efficiently assembled among processors, where every one holds its local contribution to the global system. The solution of this global problem can be obtained with iterative solvers, where efficient preconditioning techniques are crucial in order to obtain good convergence rates.

In order to assess the parallel performance of these preconditioning techniques (and solvers in general), scalability is the key measurement tool. Scalability is the ability of an algorithm to efficiently exploit increasingly available computational resources. Depending on the interests of a given application, two basic definitions to measure the parallel efficiency of the implementation are distinguished. Strong scalability is the ability to reduce time-to-solution when increasing available computational resources for a fixed problem size. Strong scalability is of high interest in moderate core counts. However, in general, it is hard to achieve good strong scaling efficiency for large core counts due to the complexity of the parallelization of the algorithms. On the other hand, weak scalability is the ability to exploit increasing computational resources (i.e., solving larger problems) without increasing the metrics of the preconditioned system (e.g., number of iterations until convergence, time-to-solution), keeping constant the load per processor. It is typically the case for simulations requiring to solve large problems that do not fit in memory in a single node. Roughly speaking, with a weakly scalable implementation, increasing X times the number of parallel tasks one is able to solve X times larger problem in approximately the same amount of time. This fact is the key to exploit parallelism in future exascale architectures.

Weakly scalable solvers require complex mathematical approaches, like multigrid (MG) [141] or multilevel domain decomposition (DD) methods [138]. These methods can be formulated as solvers, but they are more conveniently used as preconditioners for iterative solvers such as Krylov subspace methods.

The original motivation behind the MG preconditioning approach is to accelerate the convergence of a basic iterative method, i.e., to optimize the number of arithmetic operations. They take the advantages of *smoothers*, i.e., cheap iterative methods that are capable of damping out oscillatory (high frequency) error components in few iterations (e.g., weighted Jacobi, Gauss-Seidel), and combine it with coarse grid problems to correct the limitations of the *smoother*, i.e., to handle the smooth (low frequency) error modes.

Projection and restriction operators allow to transfer the solution across the different levels of the grid hierarchy. The idea is to restrict the problem, with moderate coarsening ratios, as many times as needed in order to apply a direct solver at the coarsest level, and apply *smoothers* at the remaining levels.

On the other hand, DD is motivated as a technique for parallel computing from its inception. It is based on a divide and conquer approach, where the (global) problem is divided into sub-problems in order to have less load per processor, i.e., to divide the arithmetic load among processors. Thus, DD preconditioners involve small (local to each processor) problems. Usually, these local problems are such that the application of direct methods is available to obtain fast and accurate local solutions. Besides, one could consider using MG in local problems combined with the DD algorithm to exploit the optimal complexity of the first and the parallelism of the latter. Nevertheless, one-level (i.e., only local) preconditioners preclude weak scalability since the information (i.e., transmission conditions) is only transferred between nearest neighbours. This fact implies that the number of iterations that a preconditioned Krylov method will need to converge increases with the number of parts. To overcome this situation, a coarse problem, which couples all the subdomains, can be used to propagate information globally in each iteration. Therefore, in large scale applications, the ultimate objective of DD preconditioners is to obtain favourable bounds in the condition number of the preconditioned system, having the properties of optimality, (i.e., independent of the global problem size) and algorithmic weak scalability (i.e., can only increase with the local problem size).

The balancing domain decomposition by constraints (BDDC) preconditioner, introduced by C. Dohrmann in 2003 [57], is one of the most advanced non-overlapping DD methods (a.k.a. iterative substructuring methods), where the subdomains only intersect on their interfaces. It can be understood as an evolution of the earlier Balancing domain decomposition (BDD) method [95], which requires a characterization of the kernel of the local operators in order to build the coarse problem and make local (Neumann) problems solvable. All the weaknesses of the BDD preconditioner are conveniently addressed by the BDDC preconditioner, that has three outstanding properties that makes it an excellent candidate for extreme scale computing, namely: 1) local and coarse problems can be solved in parallel. In practical implementations, that means that the computing times for the coarse problem related tasks can be masked by local (Neumann) problems computing times as long as they are not exceeded. 2) it allows for an aggressive coarsening. The resulting coarse problem only involves few unknowns per subdomain. 3) it allows for multilevel extension. The size of the coarse problem scales with the number of processors and may become the bottleneck of the solver. Fortunately, a multilevel implementation of the algorithm is quite natural, where computations among different levels can also be performed in parallel.

Many complex problems are governed by time-dependent PDEs, e.g, among others, wave propagation, climate modelling, heat transfer, fluid flows or additive manufacturing processes. When dealing with transient problems, the natural choice is to exploit

time sequentiality since information always travels from earlier times to future times. Indeed, the sequential-in-time approach is used in the vast majority of CSE simulations. Although parallel-in-time solvers date back to approximately 50 years ago, it is not until recent times that time parallel integration methods have received an increasing interest. This fact is certainly motivated by the tremendous amount of computational resources that are and will be available. Spatial parallelism may saturate at some point (i.e., no more accuracy is traduced into more meaningful results) and the time direction offers itself an opportunity for further parallelization. In short, parallel-in-time methods aim to provide the solution in one shot aggregating multiple time steps (i.e., time windows) or even for the whole time interval, rather than the usual step-by-step solution process. Although many efforts have been done in the application of DD methods to space-time formulations, space-time parallel BDDC methods is still an unexplored path that the present thesis will open.

The scientific software project `FEMPAR`

The development and implementation of cutting-edge advanced numerical discretizations and solvers for the simulation of problems governed by PDEs is the main motivation of the open source, scientific software project `FEMPAR` [20].

The `FEMPAR` project, which stands for Finite Element Multiphysics PARallel solvers, was launched in 2011 as an in-house code by Santiago Badia, Alberto F. Martín, and Javier Principe, in the frame of the Starting Grant of Santiago Badia. It has been since then actively developed by the members of the Large Scale Scientific Computing (LSSC) group at the International Center for Numerical Methods in Engineering (CIMNE) and the Technical University of Catalonia (UPC), where the present thesis has been given birth. This section aims to provide a general presentation of `FEMPAR`. Nevertheless, the reader will discover throughout the thesis the particular software features involving the developed algorithms regarding the content of each chapter.

`FEMPAR` is a general purpose, parallel scientific software for the FE simulation of complex multiphysics problems governed by PDEs written in Fortran200X following object-oriented principles. It supports several computing and programming environments, such as, e.g., multi-threading via OpenMP for moderate scale problems and hybrid MPI/OpenMP for HPC clusters and massively parallel supercomputers. For each programming environment, it offers a set of flexible data structures and algorithms for each step in the simulation pipeline, which can be customized and/or combined in multiple ways in order to satisfy the particular application problem needs; see [19] for a deep coverage of the software architecture of `FEMPAR`. It is distributed as open source software under the terms of the GNU GPLv3 license. Among other features, `FEMPAR` provides a set of state-of-the-art adaptive mesh refinement techniques and numerical discretizations, including FE methods, discontinuous Galerkin methods, XFEM, and spline-based functional spaces. The library was originally designed to efficiently exploit distributed-memory supercomputers and easily handle multiphysics problems. In this sense, it also

provides a set of highly scalable numerical linear algebra solvers based on multilevel DD for the systems of equations that arise from PDE discretizations.

1.1 Motivation

The driving application of this work is the electromagnetic modelling of High Temperature Superconductors (HTS) at low frequencies. The work was initially motivated by the experiment *Simulation of HTS*, enclosed in the 7th Framework Programme (FP7) project FORTISSIMO. The objective of the experiment, which was funded by the European Commission, was the development of business relevant simulations of industrial processes. The experiment engaged CIMNE with other experts in different fields around superconductivity, namely: the Institut de Ciència de Materials de Barcelona (ICMAB-CSIC), as a superconductivity theoretical and experimental expert, the company Oxolutia as the end-user of the simulation software, and the Centro de Supercomputación de Galiza (CESGA) as the HPC service provider.

A HTS is a material that reaches the superconductor state at temperatures above the technological threshold defined by the liquid nitrogen boiling temperature of 77 K. These temperatures are far above the range of critical temperatures needed by conventional superconductors to reach the superconductor state (from around 20 K to less than 1 K), which must be cooled with liquid helium instead.

Its resistance-free characteristics at high temperatures make it an ideal choice for a wide range of applications in the industry of applied superconductivity, namely:

a) Electro-technical equipment for the efficient generation, transport and use of energy in the grid, for a safer grid and more efficient energetic system with increased quality, including: HTS AC/DC cables, fault current limiters, superconducting magnetic energy storage systems, passive levitating fly wheels, transformers, synchronous condensers, generators and motors.

b) Medical diagnose, mainly related to the generation of uniform high magnetic fields for magnetic resonance imaging (MRI), and magnetic biosensors for in-situ diagnose based on the reconstruction of the electrical currents from its magnetic field.

c) Scientific instrumentation, also as equipment for attaining higher magnetic fields compared to the existing low-temperature superconductors (LTS) in a cost-effective way, as helium as cryogen could be replaced by nitrogen, which is far more abundant (thus cheaper). The efficient generation of high magnetic fields used for confining magnetically the hot plasma in future nuclear fusion reactors (like ITER or DEMO) is an example.

In all the aforementioned applications, a quantitative knowledge of the magnetic field, current density, temperature, heat generation, and mechanical stresses is essential for the virtual prototyping of HTS devices. The knowledge of fields and currents in

every point within the superconductor is mandatory if one aims to accurately predict its behaviour. In order to have reliable yet fast solutions, several attempts have been made in the HTS modelling field by neglecting the time dependence and simplifying the current distributions (Bean and Kim models). In this work a more complete approach is followed, which includes a highly non-linear constitutive equation for HTS based on the Maxwell's set of equations for low frequency and averaging properties on the HTS volume. Although in 2D, simple problems can be solved in hours up to some weeks, the complexity and the time required for obtaining a 3D solution reduces the procedure to very specific symmetries. Consequently, the obtained results are not enough reliable for a good and efficient solution. In addition, local phenomena has a strong effect on the overall structure, thus the problem cannot be simplified in many realistic applications, where the knowledge of fields and currents in every point inside the superconductor is mandatory if one aims to accurately predict its behaviour.

The electromagnetic response of superconductors is an extremely challenging problem from a numerical point of view, since requires complex approaches for each one of the steps of a simulation process. The hard non-linearity, hysteretic behaviour and spatial and time scales of the HTS problem require a robust, fast and powerful computing tool for obtaining meaningful solutions for a productive design cycle. Existing commercial software tools are usually based on lowest order elements and explicit time domain integration (i.e., solutions are obtained in a purely sequential fashion) and do not provide tailored, scalable solvers for large scale applications of the problem at hand. The computational performance is therefore affected, making these tools virtually useless if our objective is to solve realistic 3D engineering problems in reasonable times. Therefore, the development of numerical models and HPC tools is essential to push forward the limits in the design and optimization of HTS devices.

The HTS problem serves as a driving application for the developments presented in this thesis. Localized phenomena, multi-scale and highly nonlinear behaviour, high contrast in material properties, large scale simulations for realistic 3D problems and heavy time stepping are problem characteristics that pose great challenge for almost every ingredient of a numerical FE framework. In order to meet the goal of enabling feasible computation times in the solution of the HTS problem, it is essential to consider parallel, scalable algorithms for each building block of a FE simulation process.

In Sect. 1.2, the followed approaches in order to build an efficient, fully-parallel FE framework for the solution of the HTS problem will be described, alongside with the precise objectives and expected contributions of the present thesis.

1.2 Thesis objectives

Keeping in mind the motivation of the thesis, we proceed to unwrap some specific goals that the present thesis aims to achieve. New contributions in numerical approaches are often developed as a step further of previously existing approaches, which must be

justly referenced. In like manner, ambitious large FE software projects are developed and evolve under a collaborative effort. In this sense, it would have been impossible to set the present objectives without the previous work developed by the LSSC group at CIMNE in the scientific software project `FEMPAR`. Throughout this section, the author aims to provide a frame for each objective to clarify the expected contributions in each one of the thesis goals.

- **Arbitrary order edge FEs.** In general, Lagrangian elements can be used for homogeneous electromagnetic problems, but these methods are not robust for heterogeneous problems, specially involving high contrast of coefficients between different materials like the ones targeted in this thesis, where the edge elements are the preferred choice. Edge elements are theoretically sound and widely used in the simulation of electromagnetic problems. However, its implementation, specially for high orders, is not trivial. In the present thesis, the first objective is to revisit the theoretical background of edge FEs and provide a clear understanding of them. Next, to implement arbitrary-order tetrahedral/hexahedral edge FE schemes (of the first kind) in the scientific software project `FEMPAR`, which contains an abstraction for the generation of FEs based on polynomial spaces, see [19, Sect. 3]. This first step will establish the basic framework for the solution of problems posed in curl-conforming spaces.

- **h-adaptive edge FE framework.** HTS problems exhibit localized phenomena and multi-scale features. In such simulations, one aims to achieve a high degree of accuracy in areas of the domain of particular interest while saving computational efforts in other areas. This is precisely what can be achieved with mesh adaptive methods. To this end, an objective of the thesis is to provide a comprehensive description and produce a novel implementation of curl-conforming spaces on *non-conforming*, hierarchically refined meshes with arbitrary order edge FEs. In this work, the adaptive mesh refinement (AMR) is based on octree-based meshes, which are efficiently handled by the `p4est` library [47] on distributed-memory processors. The implementation is grounded on the current interface of `FEMPAR` with the `p4est` library.

- **Highly scalable BDDC solvers for problems in $H(\mathbf{curl})$.** The first step towards a robust solver based on DD methods for the HTS problem is to implement the extension of the BDDC preconditioner for problems posed in $H(\mathrm{curl})$. `FEMPAR` contains an abstract implementation of the BDDC preconditioner [18], which can be customized for the problem needs. In particular, the user has the freedom to define different operators, the continuity among processors and the averaging operator. However, it is well known that iterative substructuring methods are not scalable with the standard basis of shape functions for edge FEs, due to the strong correlation between the energy of subdomain faces and edges, therefore a change of variables, which introduces many complexities, must be considered. The objective

will be to implement and test on HPC platforms a highly scalable implementation of the BDDC solver for problems in $H(\text{curl})$.

- **Robust BDDC solvers for heterogeneous problems in $H(\text{curl})$.** Many realistic simulations in electromagnetics involve multi-material problems, e.g. the HTS problem, which is composed by dielectric and conducting materials. In turn, the definition of the physical coefficients (for the PDE governing the problem) describing the behaviour of the materials may involve high jumps and/or variations. This fact poses great challenge for iterative solvers, specially when the jumps of coefficients are not aligned with the partition of the domain into subdomains. Modern robust BDDC methods propose *coarse* space enrichment techniques and optimized averaging operators that involve expensive eigenvalue and auxiliary local problems. The objective is to construct novel highly scalable BDDC preconditioners for problems posed in curl-conforming spaces that are robust with regard to the jump and heterogeneity of coefficients and keep the simplicity of the standard BDDC method. The approach follows the ideas of the physics-based (PB)-BDDC preconditioner for problems in grad-conforming spaces [16]. In this sense, we aim to design robust, scalable parallel BDDC preconditioners for the solution of large scale heterogeneous electromagnetic problems and test their implementation in HPC platforms.

- **Build a complete FE parallel framework for the simulation of realistic 3D HTS problems.** The achievement of the previous objectives will allow to compose a fully-parallel simulation framework for the solution of realistic 3D HTS problems. The widespread in the HTS modelling community H-formulation is selected. The program, which will constitute a new application of `FEMPAR`, will require the definition of all the basic building blocks in the simulation pipeline leveraging the previously developed tools for problems in $H(\text{curl})$. Nevertheless, the hard non-linearity of the laws describing the behaviour of HTS devices will require a complex linearization approach, where the application of the developed BDDC methods as preconditioners for linear systems can be effective. In this sense, the goal of the implementation is to significantly reduce time-to-solution for the proposed HTS problems. Alongside with providing efficient simulation codes, the thesis aims to validate the implementation of the proposed numerical tools against experimental data for HTS tapes, kindly provided by the group of X. Granados from ICMAB in the frame of collaboration under the experiment FORTISSIMO.

- **Design parallel-in-time nonlinear ordinary differential equation (ODE) solvers.**

  The acquired knowledge in DD preconditioners will be used to explore parallelism not only in spatial variables but also in the time direction. Nevertheless, the time direction is special, since there is a *causality* principle that is naturally sequential: the solution later in time is affected by the solution of the preceding time. In

a first stage, driven by DD ideas, the objective is to explore the application of concepts to nonlinear ODEs, where the complexity of the developments is clearly reduced by the fact of dealing with one-dimensional problems. However, it will allow to identify key ingredients that will be required for the following goals, e.g. the incidence of the time *causality* principle in the design of algorithms. Due to the nature of the objectives set in this direction, this preliminary exploration will be performed with a personal code built from scratch.

- **Develop a time-extension of the BDDC preconditioner.**

  Finally, we aim to provide a novel, scalable space-time preconditioner based on BDDC methods. When dealing with transient problems, at every time step one has to solve a spatial problem before proceeding to the next step, and thus parallelism can be exploited for the space variables. This is the common scenario in simulations in CSE. However, many realistic applications, (e.g. HTS, additive manufacturing, turbulent flows) involve a heavy time stepping, where exploiting parallelism would be highly beneficial in order to achieve further levels of concurrency. In opposition to the sequential approach, we aim to provide the solution for the transient problem in one shot. The objective is two-fold: First, to design a space-time (ST) method based on space-time BDDC concepts by combining parallel-in-time ideas with the space-only BDDC preconditioner. Secondly, to obtain a highly scalable implementation of the proposed preconditioner.

1.3 Structure of the document

The present thesis involves several contributions with a common objective: the design of highly scalable BDDC preconditioners for HPC applications. The development of the thesis led to different scientific articles, which are either published or submitted in international journals. Indeed, the referenced works constitute the backbone of the thesis such that the final goal of providing scalable space-time solvers for complex electromagnetic problems is addressed in a step-by-step procedure. Although all the chapters here enclosed are aligned with the common goal, two major lines of research can be clearly identified throughout the document. A first part, devoted to the design of robust, efficient and scalable parallel solvers for linear systems arising from realistic 3D electromagnetic applications, spans three chapters. Next, a new step is taken in the direction of parallel-in-time solvers, spanning two more chapters. The collection of all the ideas involved in both directions constitutes the body of the present thesis. This fact motivates the one-to-one correspondence between articles and chapters of the thesis. Indeed, the structure of the articles has been preserved and all of them are self-contained. That means that all chapters can be understood as independent research contributions and each one contains its specific notation. In like manner, some definitions can be repeated in several parts of

the document.

The first chapter provides a brief introduction to the field of research and describes the main motivations that have driven the current work. Next, it sets the specific objectives of the thesis.

In Chapter 2, the curl-conforming FE framework is presented. Firstly, an general introduction to the background of the edge FE is provided in Sect. 2.1. Sect. 2.3 is devoted to an exhaustive theoretical presentation of the curl-conforming tetrahedral/hexahedral edge FEs of arbitrary order, providing several examples of the involved polynomial spaces in Sect. 2.3.2 and thoroughly describing the construction of resulting shape functions in Sect. 2.3.4. In Sect. 2.5 a novel strategy to deal with h-adaptive implementations of edge FE discretizations on non-conforming meshes is described. Then, in Sect. 2.6, the implementation in `FEMPAR` of the edge FEs is validated through the comparison of theoretical and numerical convergence rates of the method for a wide range of different discretizations. To finalize the chapter, some conclusions are drawn in Sect. 2.7. This work corresponds to the paper submitted for consideration of publication

[111] M.O., S.Badia and A.F. Martín. On a general implementation of h and p-adaptive curl-conforming finite elements. *submitted*. 2018. Available at *ArXiv:1810.10314*.

In Chapter 3, scalable BDDC solvers for heterogeneous problems in electromagnetics with curl-conforming FE discretizations are developed. First, a state-of-the art exposition of BDDC approaches for curl-conforming spaces is presented in Sect. 3.1. The formulation is defined in Sect. 3.2, where key ingredients for the BDDC method are presented. Sect. 3.3 is devoted to the presentation of the extension of the BDDC preconditioner for complex heterogeneous 3D problems in curl-conforming spaces. Next, in Sect. 3.4, a comprehensive description of all implementation issues behind the method is provided. A detailed study of the weak scalability properties of the implementation is carried out in Sect. 3.5, where the method is tested with both general heterogeneous and the HTS problems. To finalize the chapter, some conclusions are provided in Sect. 3.6. This work corresponds to the paper in preparation

[17] S.Badia, A.F. Martín and M.O. Scalable solvers for complex electromagnetic problems. *In preparation*. 2018.

In Chapter 4, a complete parallel FE framework to simulate the low-frequency electromagnetic response of superconducting devices is presented. Firstly, the selected formulation is derived from Maxwell's equations in Sect. 4.2. Then, the FE approximation to the problem is described in Sect. 4.3, including the time integration method. In Sect. 4.3.3, an AMR technique is described, putting emphasis on the generation of a conforming global FE space. In Sect. 4.4 the composition of a tailored transient, nonlinear solver based on Newton Raphson and BDDC methods for curl-conforming spaces is defined. The method relies on the developed BDDC method for heterogeneous problems.

In Sect. 4.5 a detailed set of numerical experiments is presented. In order to show the availability of the approximation to model the phenomena, a validation exercise against experimental data is included. In order to study the parallel performance of the implementation, the reproduction of a selected 3D benchmark is performed. Finally, Sect. 4.6 is devoted to provide some conclusions. This work corresponds to the published paper

[112] M.O., S.Badia and A.F. Martín. Simulation of High Temperature Superconductors and experimental validation. *Computer Physics Communications*, *in press*, 2018.

In Chapter 5 a parallel-in-time solver for (non)linear ODEs based on multilevel DD concepts is proposed. In this thesis, this work opens the path to parallel-in-time methods, where the motivation is to exploit higher levels of parallelism not only in space but also in time to reduce time-to-solution. In Sect. 5.3 a time-parallel direct solver for linear ODEs is exposed, whereas in Sect. 5.4 an extension of the method to nonlinear ODEs is presented. Two different strategies for exploiting parallelism in nonlinear problems are proposed, namely: a Newton method over the global nonlinear ODE in Sect. 5.4.1 and a global nonlinear problem in terms of local nonlinear problems in Sect. 5.4.2. Finally, some conclusions regarding time-parallel methods for ODEs are drawn in Sect. 5.6. This work corresponds to the published paper

[25] S.Badia and M.O. Nonlinear parallel-in-time Schur complement solvers for ordinary differential equations. *Journal of Computational and Applied Mathematics*, 344 (2018), 794-806.

In Chapter 6 a time-extension of the BDDC preconditioner is presented and tested. In Sect. 6.2 the problem and some notation are introduced. In Sect. 6.3 the classical (only) space-parallel BDDC preconditioners are stated. Then, in Sect. 6.4 the ST-BDDC extension of the method is developed. The weak scalability properties of the implementation are examined in Sect. 6.5. Finally, we draw some conclusions out of the obtained results in Sect. 6.6. This work corresponds to the published paper

[24] S.Badia and M.O. Space-Time Balancing Domain Decomposition. *SIAM Journal on Scientific Computing*, 39,2 (2017), C194-C213.

Finally, in Chapter 7, some conclusions are drawn and the thesis contributions are synthesized with regard to the proposed objectives. Alongside with the conclusions, some current limitations of the developed methods are identified, which lead to some open lines of research and future work aligned with the research paths explored within the present thesis.

1.4 Conference and workshop presentations

The thesis also led to the following presentations in international conferences, which cover all the developments contained in the present thesis.

2018 *BDDC methods for the simulation of High Temperature Superconductors.* World Congress in Computational Mechanics XIII, New York, USA.

2017 *A highly scalable finite element framework for complex multiphysics.* High performance Computing and Open platforms for energy technology modelling. Napoli, Italy.

2017 *Nonlinear parallel-in-time multilevel Schur complement solvers for ordinary differential equations.* Parallel-in-time methods VI, Lugano, Switzerland.

2017 *On scalable space-time balancing domain decomposition solvers.* Domain Decomposition XXIII, Svalbard, Norway.

2016 *On scalable space-time balancing domain decomposition solvers.* Parallel-in-time methods workshop. University Paul Sabatier, Toulouse, France.

Chapter 2

Curl-conforming Finite Elements

In this chapter, we aim to provide a comprehensive description of a novel and general implementation of edge finite elements (FEs) of arbitrary order on tetrahedral and hexahedral non-conforming meshes. Edge (or Nédélec) finite elements are theoretically sound and widely used by the computational electromagnetics community. However, its implementation, specially for high order methods, is not trivial, since it involves many technicalities that are not properly described in the literature. To fill this gap, we provide in this chapter a comprehensive description of a general implementation of edge elements of the first kind within the scientific software project `FEMPAR`. We cover into detail how to implement arbitrary order (i.e., p-adaptive) elements on hexahedral and tetrahedral meshes. First, we set the three classical ingredients of the finite element definition by Ciarlet, both in the reference and the physical space: cell topologies, polynomial spaces and moments. With these ingredients, shape functions are automatically implemented by defining a judiciously chosen polynomial *pre-basis* that spans the local finite element space combined with a change of basis to automatically obtain a canonical basis with respect to the moments at hand. Next, we discuss global finite element spaces putting emphasis on the construction of global shape functions through oriented meshes, appropriate geometrical mappings, and equivalence classes of moments, in order to preserve the inter-element continuity of tangential components of the magnetic field. Finally, we extend the proposed methodology to generate global curl-conforming spaces on *non-conforming* hierarchically refined (i.e., h-adaptive) meshes with arbitrary order finite elements. Numerical results include experimental convergence rates to test the proposed implementation, which will define the curl-conforming framework employed in the forthcoming chapters of the thesis.

2.1 Introduction

Edge elements were originally proposed in the seminal work by Nédélec [104]. They are a natural choice in electromagnetic FE simulations due to their sound mathematical structure [101]. In short, edge FE spaces represent curl-conforming fields with continuous tangential components and discontinuous normal components. It is recognized to be their greatest advantage against Lagrangian FEs [102]. There are different approaches to design discretizations of the Maxwell equations that rely on Lagrangian FEs and can

theoretically converge to singular solutions (see, e.g., [14]). However, these methods are not robust for complex electromagnetic problems. In our experience (see Chapter 4), the usage of edge elements is of imperative importance in the modelling of fields near singularities by allowing normal components to jump across interfaces between two different media with highly contrasting properties. The curl operator has null space, the gradient of any scalar field. To remove its kernel, one needs to include a zero-order term, a transient problem or a divergence-free constraint over the magnetic field times resistivity[1] (also known as Coulomb gauge) [92].

Edge elements are widespread in the computational electromagnetics community, mainly for the lowest order case. In fact, they receive the name of edge elements because for first order approximations each degree of freedom (DOF) is associated with an edge of the mesh, and DOFs can be understood as circulations along edges. However, its implementation, specially for high order, is not a trivial task and involves many technicalities. On the other hand, there are few works addressing the implementation details of arbitrary-order edge FE schemes.

The construction of local bases for edge elements has been addressed by few authors. A basis for high order methods, expressed in terms of the (barycentric) affine coordinates of the simplex (i.e., tetrahedra) is described in [77, 37], including also implementation details in [36]. The definition of affine-related coordinates is used to define expressions for the bases in hexahedra, prisms and pyramids in [69]. On the other hand, basis functions can be obtained as the dual basis with respect to suitable edge degrees of freedom functionals acting on the FE, described for prisms and simplices in [11, 75].

Better documented is the construction of global curl-conforming spaces [126, 124], which relies on the Piola mapping to achieve the global continuity of the tangent components by using moments defined in the reference cell. In the case of unstructured hexahedral meshes, where non-affine geometrical mappings must be used in general, optimal convergence of *standard* edge elements in the $H(\mathrm{curl})$ norm cannot be achieved [66]. Complex geometries can still be considered using, e.g., unfitted FE techniques [26] on octree background meshes to avoid non-affine mappings.

Special care has to be taken in the definition of edge DOFs to enforce the right continuity across cells. To prevent the so-called sign conflict, different alternatives have been proposed. A simple sign flip [12] can assure consistency, i.e., all local DOFs shared by two or more tetrahedra represent the same global DOF, for first order 3D FEs only. For higher order methods, a remedy can be based on the construction of all possible shape functions combined with a permutation that depends on local edge/face orientations [69, 76]. Our preferred solution, for simplicity and ease of implementation, is to *orient* tetrahedral meshes, requiring that local nodes numbering within every element are always sorted based on their global indices [124]. On the other hand, although there

[1] Edge FEs provide solutions that are pointwise divergence-free in the element interiors. However, the normal component of the field can freely jump across inter-element faces. The zero inter-element jump constraint has to be enforced by a Lagrange multiplier [14] to eliminate the kernel.

are hexahedral meshes that cannot be oriented in 3D [5], we restrict ourselves to forest-of-octrees meshes [47] such that that the initial coarse mesh can be oriented, e.g., a uniform, structured mesh.

Few open source FE software projects include the implementation of edge FEs of arbitrary order. FEniCS supports arbitrary order FEs [7], but is restricted to tetrahedra and has its own domain specific language for weak forms to automatically generate the corresponding FE code, which may prevent *hp*-adaptivity. On the other hand, the deal.ii library also provides arbitrary order edge FEs [30, 29], even on *h*-adaptive meshes, but it is restricted to hexahedral meshes. Recently, a C++ plug-in that defines high order edge FE ingredients [36] has been developed for the FreeFEM++ library [83], but it is restricted to up to third order and tetrahedral meshes. The most complete approaches found in the literature are the implementations in MFEM [2], which provide arbitrary order tetrahedral FEs and *h*-adaptivity on hexahedral meshes, and the *hp*-adaptive approaches in hp2D/hp3D [55, 56] and Netgen/NGSolve [127, 3], which additionally employ arbitrary order edge prism and pyramid elements in 3D.

The usage of non-nodal based (compatible or structure-preserving) FEs, especially in multi-physics applications in combination with incompressible fluid and solid mechanics [128, 113, 144], is still scarce. It is motivated by the fact that most codes in computational mechanics are designed from inception for being used with nodal FEs, and the extension to other types of FEs seems to be a highly demanding task. It does not help the poor, fractioned and spread information on key implementation issues.

As mentioned before, the motivation of this chapter is to provide *a comprehensive description of a novel and general implementation of edge FEs of arbitrary order on tetrahedral and hexahedral*[2] *non-conforming meshes*, confronted with typical hard-coded implementations of the shape functions that preclude high order methods. With this aim, we will address three critical points that are specially complex in the construction of arbitrary order edge FEs: 1) an automatic manner of generating the associated polynomial spaces; 2) a general solution for the so-called sign conflict; and 3) a construction of the discrete $H(\text{curl})$-conforming FE spaces atop *non-conforming* meshes. With regard to 1), we generate arbitrary order edge FE shape functions in a simple but extensible manner. First, we propose a *pre-basis* of vector-valued polynomial functions spanning arbitrary order local edge FE spaces for both tetrahedral and hexahedral topologies; see [35] for a related approach on tetrahedra. Next, we compute a change of basis from the *pre-basis* to the canonical basis with respect to the edge FE DOFs, i.e., the basis of shape functions. To address 2) and properly determine the inter-element continuity, we rely on oriented meshes. Finally, in order to address 3), we combine a standard adaptive mesh refinement (AMR) nodal-based implementation of constraints on non-conforming geometrical objects (a.k.a. as hanging) with the relation between a Lagrangian *pre-basis*

[2] Although this thesis is restricted to hexahedra and tetrahedra, one can also find definitions for prism and pyramid edge elements [34] with optimal rates of convergence of the numerical solution towards the exact solution in the $H(\mathbf{curl}, \Omega)$ norm.

and the edge basis, to avoid the evaluation of edge moments on the refined cells. Besides, the same machinery can also be applied to other polynomial based FEs, like Raviart-Thomas FEs, Brezzi-Douglas-Marini FEs [44], second kind edge FEs [105], or recent divergence-free FEs [106].

The proposed approach is the result of the experience gained by the authors after the implementation in the scientific software `FEMPAR` (see Chapter 1 for an introduction). In any case, we present an automatic, comprehensive strategy rather than focusing on a particular software implementation. Consequently, thorough definitions and examples will be provided, but the exact software implementation will not be shown. The reader is directed to [19] for code details, where an exhaustive introduction to the software abstractions of `FEMPAR` is presented.

The chapter is organized as follows. In Sect. 2.2, we introduce some notation and a general problem with the curl-curl formulation. In Sect. 2.3, we will cover the definition and construction of the edge FE of arbitrary order, focusing on an straightforward manner of generating the involved polynomial spaces. Sect. 2.4 is devoted to the construction of global curl-conforming FE spaces. In Sect. 2.5, a strategy to deal with h-adaptive spaces with the edge element will be described. Finally, in Sect. 2.6, we will show some convergence results, which will serve to validate the implementation of the edge elements.

2.2 Problem statement

2.2.1 Notation

In this section, we introduce the model problem to be solved and some mathematical notation. Bold characters will be used to describe vector functions or tensors, while regular ones will determine scalar functions or values. No difference is intended by using upper-case or lower-case letters for functions.

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with $d = 2, 3$ the space dimension. In 3D, the rotation operator of a vector field $\boldsymbol{v}$ is defined as:

$$\boldsymbol{\nabla} \times \boldsymbol{v} = \begin{bmatrix} \partial_2 v_3 - \partial_3 v_2 \\ \partial_3 v_1 - \partial_1 v_3 \\ \partial_1 v_2 - \partial_2 v_1 \end{bmatrix}. \tag{2.1}$$

We use the standard multi-index notation for derivatives, with $\boldsymbol{\alpha} := (\alpha_1, \ldots, \alpha_d)^T \in \mathbb{Z}_+^d$ and $|\boldsymbol{\alpha}|_1 = \sum_{i=1}^d |\alpha_i|$. Let us define the spaces

$$H^r(\Omega) := \{v \in L^2(\Omega) \mid \partial^{\boldsymbol{\alpha}} v \in L^2(\Omega) \text{ for all } |\boldsymbol{\alpha}|_1 \leq r\}, \tag{2.2}$$

$$H^r(\mathbf{curl}, \Omega) := \{\boldsymbol{v} \in H^r(\Omega)^d \mid \boldsymbol{\nabla} \times \boldsymbol{v} \in H^r(\Omega)^d\}. \tag{2.3}$$

The space $H^0(\mathbf{curl}, \Omega)$ is represented with $H(\mathbf{curl}, \Omega)$. We also consider the subspaces

$$H_0^1(\Omega) := \{v \in H^1(\Omega) \mid v = 0 \text{ in } \partial\Omega\}, \tag{2.4}$$

$$H_0(\mathbf{curl}, \Omega) := \{\boldsymbol{v} \in H(\mathbf{curl}, \Omega) \mid \boldsymbol{n} \times \boldsymbol{v} = \mathbf{0} \text{ in } \partial\Omega\}, \tag{2.5}$$

where $\boldsymbol{n}$ denotes the outward unit normal to the boundary of the domain Ω. The space $H(\mathbf{curl}, \Omega)$ in Eq. (2.3) is equipped with the norm

$$\|\boldsymbol{u}\|_{H(\mathbf{curl},\Omega)} := \left(\|\boldsymbol{u}\|^2_{L^2(\Omega)^d} + \|\boldsymbol{\nabla} \times \boldsymbol{u}\|^2_{L^2(\Omega)^d}\right)^{\frac{1}{2}}.$$

In 2D, a scalar version of the curl operator can be defined as $\text{curl}(\boldsymbol{v}) \doteq \partial_1 v_2 - \partial_2 v_1$. We can similarly define $H^r(\text{curl}, \Omega)$ and its related subspaces. In the following, we consider the notation for the 3D case.

2.2.2 Formulation

The model problem consists in finding a vector field $\boldsymbol{u}$ (magnetic field) solution of

$$\boldsymbol{\nabla} \times \alpha \boldsymbol{\nabla} \times \boldsymbol{u} + \beta \boldsymbol{u} = \boldsymbol{f} \qquad \text{in } \Omega, \tag{2.6}$$

where $\boldsymbol{f}$ is a given source term. Problem parameters α, β can range from scalar, positive values for isotropic materials to positive definite tensors for anisotropic materials. Besides, (2.6) needs to be supplied with appropriate boundary conditions. The boundary of the domain $\partial\Omega$ is divided into its Dirichlet boundary part, i.e., $\partial\Omega_D$, and its Neumann boundary part, i.e., $\partial\Omega_N$, such that $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \cap \Omega_N = \emptyset$. Then, boundary conditions for the problem at hand read

$$\boldsymbol{u} \times \boldsymbol{n} = \boldsymbol{g}_D \qquad \text{on } \partial\Omega_D, \tag{2.7}$$

$$\boldsymbol{n} \times (\alpha \boldsymbol{\nabla} \times \boldsymbol{u}) = \boldsymbol{g}_N \qquad \text{on } \partial\Omega_N, \tag{2.8}$$

where $\boldsymbol{n}$ is a unit normal to the boundary. Dirichlet (essential) boundary conditions prescribe the tangent component of the field $\boldsymbol{u}$ on the boundary of the domain. On the other hand, Neumann (natural) boundary conditions arise in the integration by parts of the curl-curl term

$$\int_\Omega (\boldsymbol{\nabla} \times \alpha \boldsymbol{\nabla} \times \boldsymbol{u}) \cdot \boldsymbol{v} = \int_\Omega (\alpha \boldsymbol{\nabla} \times \boldsymbol{u}) \cdot (\boldsymbol{\nabla} \times \boldsymbol{v}) - \int_{\Omega_N} (\alpha \boldsymbol{\nabla} \times \boldsymbol{u}) \cdot (\boldsymbol{n} \times \boldsymbol{v}). \tag{2.9}$$

Consider now two different non-overlapping regions on the domain Ω corresponding to two different media (usual case in electromagnetic simulations), namely Ω_1 and Ω_2, such that $\Omega_1 \cup \Omega_2 = \Omega$, and let us define the *interface* as $\Gamma := \Omega_1 \cap \Omega_2$. Let us denote by $\boldsymbol{n}_1, \boldsymbol{n}_2$ the unit normal pointing outwards of Ω_1 and Ω_2 on Γ, resp. The *transmission*

conditions (in absence of other sources) for Eq. (2.6) are stated as follows:

$$\boldsymbol{n} \times (\alpha_1 \boldsymbol{\nabla} \times \boldsymbol{u}_1 - \alpha_2 \boldsymbol{\nabla} \times \boldsymbol{u}_2) = 0 \qquad \text{on } \Gamma,$$

where $\boldsymbol{n}$ can, e.g., be $\boldsymbol{n}_1 = -\boldsymbol{n}_2$. For the sake of simplicity in the presentation (not in the implementation), let us consider $\partial\Omega_D = \partial\Omega$, so that the Neumann boundary term Eq. (2.9) can be removed from the weak form, and homogeneous Dirichlet boundary conditions. Hence, the variational form of the double curl formulation Eq. (2.6) reads: find $\boldsymbol{u} \in H_0(\mathbf{curl}, \Omega)$ such that

$$(\alpha \boldsymbol{\nabla} \times \boldsymbol{u}, \boldsymbol{\nabla} \times \boldsymbol{v}) + (\beta \boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}), \qquad \forall \boldsymbol{v} \in H_0(\mathbf{curl}, \Omega). \tag{2.10}$$

2.3 Edge FEs

Let $\mathcal{T}_h$ be a quasi-uniform partition of Ω into a set of hexahedra (quadrilaterals in 2D) or tetrahedra (triangles in 2D) geometrical cells K. For every $K \in \mathcal{T}_h$, we denote by h_K its diameter and set the characteristic mesh size as $h = \max_{K \in \mathcal{T}_h} h_K$. Let us denote by $v \in \mathcal{N}$, $e \in \mathcal{E}$ and $f \in \mathcal{F}$ the components and global sets of vertices, edges and faces of $\mathcal{T}_h$, with cardinality $n_{\mathcal{N}}, n_{\mathcal{E}}$ and $n_{\mathcal{F}}$, resp. Using Ciarlet's definition, a FE is represented by the triplet $\{K, \mathcal{V}, \Sigma\}$, where $\mathcal{V}$ is the space of functions on K and Σ is a set of linear functionals on $\mathcal{V}$. The elements of Σ are called DOFs (moments) of the FE. We will denote the number of functionals on the cell as n_Σ, and $\Sigma \doteq \{\sigma_a\}_{a=1}^{n_\Sigma}$. Σ is a basis for $\mathcal{V}'$, which is dual to the so-called basis of shape functions $\{\phi^a\}_{a=1}^{n_\Sigma}$ for $\mathcal{V}$, i.e., $\sigma_a(\phi^b) = \delta_{ab}$, $\forall a, b \in \{1, \ldots, n_\Sigma\}$.

At this point, we must distinguish between the *reference* FE, built on a reference cell, and the FE in the physical space. Our implementation of the space of functions and moments is based on a unique reference FE $\{\hat{K}, \hat{\mathcal{V}}, \hat{\Sigma}\}$. Then, in the physical space, the FE triplet on a cell K relies on its reference FE, a geometrical mapping $\boldsymbol{\Phi}_K$ such that $K = \boldsymbol{\Phi}(\hat{K})$ and a linear bijective function mapping $\hat{\Psi}_K : \hat{\mathcal{V}} \to \hat{\mathcal{V}}$. It is well known that quadrilateral FEs may result in a loss of accuracy on general meshes, e.g., when elements are given as images of hexahedra under invertible bilinear maps, in comparison with the accuracy achieved with squares or cubes [13]. In this text, hexahedra in the physical space are obtained with affine transformations from the reference square or cube, thus optimal convergence properties are guaranteed. Nevertheless, this fact does not restrict the simulations to simple geometries, since unfitted approaches [26] may be employed to model complex geometries with structured background meshes. Let us denote by $\boldsymbol{J}_K$ the Jacobian of the geometrical mapping, i.e., $\boldsymbol{J}_K \doteq \frac{\partial \boldsymbol{\Phi}_K}{\partial \boldsymbol{x}}$. The functional space is defined as $\mathcal{V} \doteq \{\hat{\Psi}_K(\hat{v}) \circ \boldsymbol{\Phi}_K^{-1} : \hat{v} \in \hat{\mathcal{V}}\}$; we will also use the mapping $\Psi_K : \hat{\mathcal{V}} \to \mathcal{V}$ defined as $\Psi_K(\hat{v}) \doteq \hat{\Psi}_K(\hat{v}) \circ \boldsymbol{\Phi}_K^{-1}$. Finally, the set of DOFs on the physical FE is defined as $\Sigma \doteq \{\hat{\sigma} \circ \Psi_K^{-1} : \hat{\sigma} \in \hat{\Sigma}\}$ from the set of the reference FE linear functionals. In the following subsections, we go into detail into these concepts and provide some practical examples.

2.3.1 Reference cell

A *polytope* is mathematically defined as the convex hull of a finite set of points (vertices). The concept of polytope may be of practical importance, because it allows one to develop codes that can be applied to any topology of arbitrary dimension that fits into the framework, see [19] for a thorough exposition. For the sake of ease, we restrict ourselves to two possible polytopes: d-cubes and d-simplices (with $d = 2, 3$), defined as the convex hull of a set of $n_v = 2^d$ and $n_v = d + 1$ geometrically distinct points, respectively. An *ordered* set of vertices $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n_v}\}$ not only defines the topology of the cell $\hat{K}$ but an *orientation*. Edges and faces are polytopes of lower dimension, and an ordered set of their vertices also defines their orientation. For the sake of illustration, the local indexing at the cell level is depicted in Figs. 2.1a and 2.2a for a 3-cube and a 3-simplex, resp. The 2D cases follow by simply considering the restriction of the 3D cell to one of its faces.

2.3.2 Polynomial spaces

Local FE spaces are usually spanned by polynomial functions. Let us start by defining basic polynomial spaces that will be needed in the forthcoming definitions. We define here all polynomial spaces in an arbitrary polytope K, but they will only be used for the reference cell in the next sections. The space of polynomials of degree less than or equal to $k > 0$ in all the variables $\{x_i\}_{i=1}^d$ is denoted by $\mathcal{Q}_k(K)$. Analogously, we can define the space of polynomials of degree less than or equal to $\{k_i\}_{i=1}^d$ for the variable $\{x_i\}_{i=1}^d$, denoted by $\mathcal{Q}_{\boldsymbol{k}}(K)$, with $\boldsymbol{k} = [k_1, \ldots, k_d]$. Clearly, the dimension of this space, denoted by $\dim(\mathcal{Q}_{\boldsymbol{k}}(K))$, is $\prod_{i=1}^d (k_i + 1)$. Let us also define the corresponding truncated polynomial space $\mathcal{P}_k(K)$ as the span of the monomials with degree less than or equal to k. To determine the dimension of the truncated space, we note that the number of components can be expressed with the triangular or tetrahedral number T_n^d, $d = \{2, 3\}$, such that

$$\dim(\mathcal{P}_k(K)) = T_{k+1}^d, \qquad T_n^d = \frac{\prod_{i=1}^d (n + i - 1)}{d!}. \tag{2.11}$$

Construction of polynomial spaces

Given an order k, it is trivial to form the set of monomials $q^k = \{x^i\}_{i=0}^k$, that spans the 1D space $\mathcal{Q}_k(K)$. We construct a basis spanning the multi-dimensional space $\mathcal{Q}_{\boldsymbol{k}}(K)$ with a Cartesian product of the monomials for each dimension, i.e., $\{q_1^{k_1} \times \ldots \times q_d^{k_d}\}$, thus we have $\mathcal{Q}_{\boldsymbol{k}}(K) = \text{span}\{\prod_{i=1}^d x_i^{\alpha_i} \text{ s.t. } \alpha_i \leq k_i\}$. Let us denote by $|\alpha|$ the summation of the exponents for a given monomial. Then, the multi-dimensional truncated space is defined as $\mathcal{P}_k(K) = \text{span}\{\prod_{i=1}^d x_i^{\alpha_i} \text{ s.t. } |\alpha| \leq k\}$.

Let us also define Lagrangian polynomials spanning $\mathcal{Q}_{\boldsymbol{k}}(K)$, which will be used in the definition of moments in Sect. 2.3.3. Given an order k and a set $\mathcal{N}_k$ of different nodes in $\mathbb{R}$, usually equidistant in the interval $[x_0, x_k]$, we can define the set of Lagrangian

polynomials $\{\ell_i\}_{i=0}^k$ as,

$$\ell_i := \frac{\prod_{n\in\mathcal{N}_k\setminus i}(x-x_n)}{\prod_{n\in\mathcal{N}_k\setminus i}(x_i-x_n)}, \tag{2.12}$$

where we indistinctly represent nodes by their index i or its position x_i. The set of all polynomials $\{\ell_i^k\}_{i=0}^k$ defines the Lagrangian basis L^k. In the grad-conforming (Lagrangian) FE, the definition of moments $\{\sigma_a\}_{a=1}^k$ simply consists in the evaluation of the functions on the given points $x \in \mathcal{N}_k$. Clearly, polynomials in Eq. (2.12) evaluated at points x_a satisfy the duality $\sigma_a(\ell_i) = \ell_i(x_a) = \delta_{ai}$ for every $a, i \in \{0, \dots k\}$, i.e., they are shape functions. For multi-dimensional spaces, we define the set of nodes as the Cartesian product of 1D nodes. Given a d-dimensional space of order $\boldsymbol{k} = [k_1, \dots, k_d]$, the set of nodes is defined as $\mathcal{N}^{\boldsymbol{k}} = \mathcal{N}^{k_1} \times \dots \times \mathcal{N}^{k_d}$.

Hexahedra

The space of functions on the cell $\mathcal{V}_k(K)$ for this sort of elements is defined as the space of gradients for the scalar polynomial space $\mathcal{Q}_k(K)$, i.e.,

$$\mathcal{V}_k(K) := \{\mathcal{Q}_{k-1,k}(K) \times \mathcal{Q}_{k,k-1}(K)\}, \tag{2.13}$$
$$\mathcal{V}_k(K) := \{\mathcal{Q}_{k-1,k,k}(K) \times \mathcal{Q}_{k,k-1,k}(K) \times \mathcal{Q}_{k,k,k-1}(K)\}, \tag{2.14}$$

in 2D and 3D, resp. Let us illustrate the polynomial space with a couple of examples.

Example 2.3.1. *The polynomial space for the lowest order ($k = 1$) edge hexahedral element is defined as*

$$\mathcal{V}_k(K) = \{\mathcal{Q}_{0,1,1}(K) \times \mathcal{Q}_{1,0,1}(K) \times \mathcal{Q}_{1,1,0}(K)\}$$

which can be represented as the span of the set of vector-valued functions

$$\left\{ \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}x_2\\0\\0\end{bmatrix}, \begin{bmatrix}x_3\\0\\0\end{bmatrix}, \begin{bmatrix}x_2x_3\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\x_1\\0\end{bmatrix}, \begin{bmatrix}0\\x_3\\0\end{bmatrix}, \begin{bmatrix}0\\x_1x_3\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\x_1\end{bmatrix}, \begin{bmatrix}0\\0\\x_2\end{bmatrix}, \begin{bmatrix}0\\0\\x_1x_2\end{bmatrix} \right\}.$$

Example 2.3.2. *The polynomial space for the second order quadrilateral element is defined as*

$$\mathcal{V}_k(K) = \{\mathcal{Q}_{1,2}(K) \times \mathcal{Q}_{2,1}(K)\},$$

which can be represented by the spanning set

$$\left\{ \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}x_1\\0\end{bmatrix}, \begin{bmatrix}x_2\\0\end{bmatrix}, \begin{bmatrix}x_1x_2\\0\end{bmatrix}, \begin{bmatrix}x_2^2\\0\end{bmatrix}, \begin{bmatrix}x_1x_2^2\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\x_1\end{bmatrix}, \begin{bmatrix}0\\x_1^2\end{bmatrix}, \begin{bmatrix}0\\x_2\end{bmatrix}, \begin{bmatrix}0\\x_1x_2\end{bmatrix}, \begin{bmatrix}0\\x_1^2x_2\end{bmatrix} \right\}.$$

Tetrahedra

The polynomial space for tetrahedral elements is slightly more involved. For the sake of brevity, let us omit the cell (i.e., K) in the notation for the local FE spaces. Let us start by defining the homogeneous polynomial space $[\tilde{\mathcal{P}}_k]^d := [\mathcal{P}_k]^d \setminus [\mathcal{P}_{k-1}]^d$, where $[\mathcal{P}_k]^2 = \mathcal{P}_k \times \mathcal{P}_k$ and $[\mathcal{P}_k]^3 = \mathcal{P}_k \times \mathcal{P}_k \times \mathcal{P}_k$. The space $[\tilde{\mathcal{P}}_k]^d$ has dimension $d \cdot \dim(\tilde{\mathcal{P}}_k)$, and using Eq. (2.11) we have:

$$\dim(\tilde{\mathcal{P}}_k) = \dim(\mathcal{P}_k) - \dim(\mathcal{P}_{k-1}) = T^d_{k+1} - T^d_k = T^{d-1}_{k+1}.$$

The function space of order k on a tetrahedron is then defined as

$$\mathcal{V}_k(K) = [\mathcal{P}_{k-1}]^d \oplus \mathcal{S}_k, \tag{2.15}$$

where $\mathcal{S}_k$ is the space of polynomials

$$\mathcal{S}_k := \{p(\boldsymbol{x}) \in [\tilde{\mathcal{P}}_k]^d \text{ such that } p(\boldsymbol{x}) \cdot \boldsymbol{x} = 0\}.$$

Note that if $p(\boldsymbol{x}) \in [\tilde{\mathcal{P}}_k]^d$, then $p \cdot \boldsymbol{x} \in \tilde{\mathcal{P}}_{k+1}$ and any polynomial in $\tilde{\mathcal{P}}_{k+1}$ may be written in this way. The dimension of the space $\mathcal{S}_k$ is

$$\dim(\mathcal{S}_k) = d \cdot \dim(\tilde{\mathcal{P}}_k) - \dim(\tilde{\mathcal{P}}_{k+1}), \tag{2.16}$$

which leads to $(k+2)k$ in 3D and k in 2D, resp. Among all the possible forms of representing the space $\mathcal{S}_k$ in 3D, we consider the following spanning set

$$\mathcal{S}_k = \text{span}\left\{ \bigcup_{\beta=1}^{k} \bigcup_{\alpha=1}^{k+1-\beta} \left(\begin{bmatrix} -x_1^{\alpha-1} x_2^{k-\alpha-\beta+2} x_3^{\beta-1} \\ x_1^{\alpha} x_2^{k-\alpha-\beta+1} x_3^{\beta-1} \\ 0 \end{bmatrix}, \begin{bmatrix} -x_1^{k-\alpha-\beta+1} x_2^{\beta-1} x_3^{\alpha} \\ 0 \\ x_1^{k-\alpha-\beta+2} x_2^{\beta-1} x_3^{\alpha-1} \end{bmatrix} \right), \tag{2.17a}$$

$$\bigcup_{\alpha=1}^{k} \begin{bmatrix} 0 \\ -x_1^0 x_2^{\alpha-1} x_3^{k-\alpha+1} \\ x_1^0 x_2^{\alpha} x_3^{k-\alpha} \end{bmatrix} \Bigg\}. \tag{2.17b}$$

Proposition 2.3.1. *The set of vector functions defined in Eq. (2.17) forms a basis of the space $\mathcal{S}_k$.*

Proof. First, we note that the proposed basis contains $\{p_i(\boldsymbol{x})\}_{i=1}^{(k+2)k}$ vector functions. Clearly, the total degree of the monomials found on each component for all functions is k, thus $S \subset \tilde{\mathcal{P}}^d_k$. Further, it is easy to check that $p_i(\boldsymbol{x}) \cdot \boldsymbol{x} = 0$ for any $p_i(\boldsymbol{x}) \in S$. The proof is completed by showing that all the functions are linearly independent. It is trivial to see that all the functions in the first set of functions are indeed linearly independent; the two sets have different non-zero components. Finally, vector functions from the last subset Eq. (2.17b) are independent of x_1, whereas all functions in the previous subset in Eq. (2.17a) do depend on x_1 and are linearly independent among them. □

In two dimensions, the analytical expression of the spanning set simply reduces to

$$S_k = \text{span}\left\{ \bigcup_{\alpha=1}^{k} \begin{bmatrix} -x_1^{\alpha-1} x_2^{k-\alpha+1} \\ x_1^{\alpha} x_2^{k-\alpha} \end{bmatrix} \right\}. \tag{2.18}$$

The dimension of the space $\mathcal{V}_k(K)$ for tetrahedra can be obtained by adding (2.11) and (2.16),

$$\dim(\mathcal{V}_k(K)) = \frac{k \prod_{i=2}^{d}(k+i)}{(d-1)!}. \tag{2.19}$$

Let us give some examples of polynomial bases for tetrahedral edge elements spanning the requested spaces.

Example 2.3.3. *A set of polynomial spanning $\mathcal{V}_1(K) = [P_0]^3 \oplus \mathcal{S}_1$ (i.e., lowest order tetrahedral element) is*

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -x_2 \\ x_1 \\ 0 \end{bmatrix}, \begin{bmatrix} -x_3 \\ 0 \\ x_1 \end{bmatrix}, \begin{bmatrix} 0 \\ -x_3 \\ x_2 \end{bmatrix} \right\}.$$

Example 2.3.4. *A polynomial base spanning $\mathcal{V}_2(K) = [P_1]^2 \oplus \mathcal{S}_2$ (i.e., second-order triangular element) is*

$$\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} x_1 \\ 0 \end{bmatrix}, \begin{bmatrix} x_2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ x_1 \end{bmatrix}, \begin{bmatrix} 0 \\ x_2 \end{bmatrix}, \begin{bmatrix} -x_2^2 \\ x_1 x_2 \end{bmatrix}, \begin{bmatrix} -x_1 x_2 \\ x_1^2 \end{bmatrix} \right\}.$$

In both cases, vector functions that span the subspaces $[\mathcal{P}_{k-1}]^d \subset \mathcal{V}_k(K)$ and $\mathcal{S}_k \subset \mathcal{V}_k(K)$ can easily be identified from the full sets of vector-valued functions in Exs. 2.3.3 and 2.3.4.

Note that local spaces $\mathcal{V}_k(K)$ lie between the full polynomial spaces of order $k-1$ and k, i.e., $[\mathcal{Q}_{k-1}(K)]^d \subset \mathcal{V}_k(K) \subset [\mathcal{Q}_k(K)]^d$, $[\mathcal{P}_{k-1}(K)]^d \subset \mathcal{V}_k(K) \subset [\mathcal{P}_k(K)]^d$ for hexahedra and tetrahedra, resp. This kind of elements are called edge FEs of the first kind [104]. Another edge FE, the so-called second kind, was introduced also by Nédélec in [105]. It follows similar ideas but considers full polynomial spaces, i.e., $[\mathcal{Q}_k(K)]^d$ or $[\mathcal{P}_k(K)]^d$, instead of anisotropic polynomial spaces (see Eq. (2.14)) or incomplete polynomial spaces (see Eq. (2.15)) for hexahedra and tetrahedra, resp., which clearly simplifies the implementation of the polynomial spaces. Edge elements of the second kind offer better constants in the error estimates at the cost of increasing the number of DOFs (see detailed definitions in [53, Ch. 2]).

2.3.3 Edge moments in the reference element

In order to complete the definition of edge FEs, it remains to define a set of (linearly independent) functionals to be applied on $\mathcal{V}_k(\hat{K})$. Edge moments are integral quantities

over geometrical sub-entities of the cell against some test functions. These moments involve directions in 1D entities (edges) and orientations in 2D (faces) or 3D (volumes). Thus, we need to define local orientations for the lower dimension geometrical entities of the reference cell, see Figs. 2.1a and 2.2a. Let us distinguish between three different subsets of moments $\hat{\Sigma}$, namely DOFs related to edges $\sigma_{\hat{e}}$, to faces $\sigma_{\hat{f}}$ and volume $\sigma_{\hat{K}}$ such that $\hat{\Sigma} = \sigma_{\hat{e}} \cup \sigma_{\hat{f}} \cup \sigma_{\hat{K}}$.

Hexahedra

The set of functionals, for the reference element, that form the basis in two dimensions reads ([101, Ch. 6]):

$$\sigma_{\hat{e}}(\hat{\boldsymbol{u}}_h) := \int_{\hat{e}} (\hat{\boldsymbol{u}}_h \cdot \hat{\boldsymbol{\tau}}) \hat{q} \quad \forall \hat{q} \in \mathcal{P}_{k-1}(\hat{e}), \quad \forall \hat{e} \in \hat{K}, \tag{2.20a}$$

$$\sigma_{\hat{K}}(\hat{\boldsymbol{u}}_h) := \int_{\hat{K}} \hat{\boldsymbol{u}}_h \cdot \hat{\boldsymbol{q}} \quad \forall \hat{\boldsymbol{q}} \in \mathcal{Q}_{k-1,k-2}(\hat{K}) \times \mathcal{Q}_{k-2,k-1}(\hat{K}). \tag{2.20b}$$

We will have $4k$ DOFs associated to edges and $2k(k-1)$ inner DOFs. Therefore, the complete set of functionals has cardinality $n_{\hat{\Sigma}} = 2k(k+1)$. In the case of $k = 1$, only edge DOFs $\sigma_{\hat{e}}$ appear. In three dimensions the set is defined as

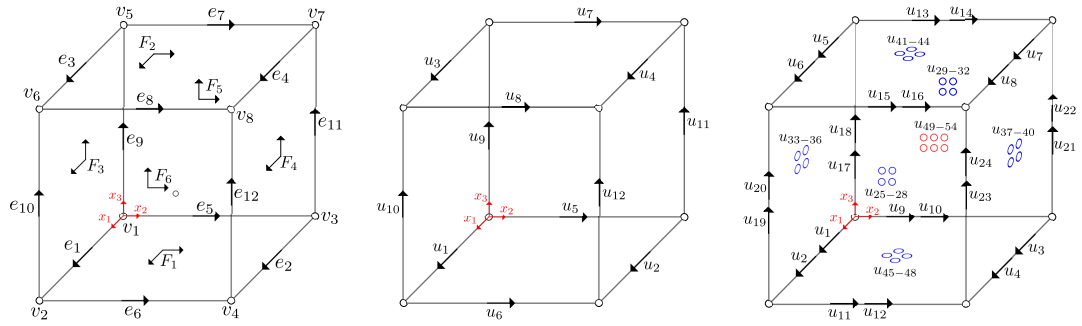
$$\sigma_{\hat{e}}(\hat{\boldsymbol{u}}_h) := \int_{\hat{e}} (\hat{\boldsymbol{u}}_h \cdot \hat{\boldsymbol{\tau}}) \hat{q} \qquad \forall \hat{q} \in \mathcal{P}_{k-1}(\hat{e}), \quad \forall \hat{e} \in \hat{K} \tag{2.21a}$$

$$\sigma_{\hat{f}}(\hat{\boldsymbol{u}}_h) := \int_{\hat{\mathcal{F}}} (\hat{\boldsymbol{u}}_h \times \hat{\boldsymbol{n}}) \cdot \hat{\boldsymbol{q}} \qquad \forall \hat{\boldsymbol{q}} \in \mathcal{Q}_{k-2,k-1}(\hat{\mathcal{F}}) \times \mathcal{Q}_{k-1,k-2}(\hat{\mathcal{F}}), \quad \forall \hat{\mathcal{F}} \in \hat{K} \tag{2.21b}$$

$$\begin{aligned} \sigma_{\hat{K}}(\hat{\boldsymbol{u}}_h) := \int_{\hat{K}} \hat{\boldsymbol{u}}_h \cdot \hat{\boldsymbol{q}} \qquad & \forall \hat{\boldsymbol{q}} \in \mathcal{Q}_{k-1,k-2,k-2}(\hat{K}) \times \mathcal{Q}_{k-2,k-1,k-2}(\hat{K}) \times \\ & \mathcal{Q}_{k-2,k-2,k-1}(\hat{K}), \end{aligned} \tag{2.21c}$$

where $\hat{\boldsymbol{\tau}}$ is the unit vector along the edge and $\hat{\boldsymbol{n}}$ the unit normal to the face. In this case, we have $12k$ DOFs associated to edges, $6 \cdot 2k(k-1)$ DOFs associated to the 6 faces of the cell and $3k(k-1)^2$ inner DOFs. We have a total number of $n_{\hat{\Sigma}} = 3k(k+1)^2$ DOFs. Note that in the case of the lowest order elements, i.e., $k = 1$, only DOFs associated to edges appear. For higher order elements, i.e., $k \geq 2$, all kinds of DOFs occur in both dimensions.

DOFs are labelled in the reference cell as follows. First, for every DOF, we can determine the geometrical entity that *owns* it, e.g., an edge of the reference FE in (2.21a). Within every geometrical entity, there is a one-to-one map between DOFs and test functions. Thus, we can number DOFs by the numbering of the test functions in the test space, e.g., the test functions in $\mathcal{P}_{k-1}(\hat{e})$ for the DOFs in (2.21a). We note that all the test spaces in the DOF definitions can be built using a nodal-based (Lagrangian) basis, and thus, every DOF in a geometrical object can be conceptually associated to one and only one node. The numbering of the nodes in the geometrical object is determined by its orientation in the reference FE. The composition of a geometrical object numbering



(A) Oriented cell geometry. (B) Lowest order DOFs. (C) Second order DOFs. Face DOFs depicted in blue, inner DOFs depicted in red.

FIGURE 2.1: 3D hexahedral reference FE.

(see Figs. 2.1–2.2) and the object-local node numbering provides the local numbering of DOFs within the reference cell.

Tetrahedra

The set of functionals described in this section follows [101, Ch. 5]. In the 2D case, the set of moments for the reference FE is defined as

$$\sigma_{\hat{e}}(\hat{\boldsymbol{u}}_h) := \int_{\hat{e}} (\hat{\boldsymbol{u}}_h \cdot \hat{\boldsymbol{\tau}})\hat{q} \quad \forall \hat{q} \in \mathcal{P}_{k-1}(\hat{e}), \quad \forall \hat{e} \in \hat{K} \tag{2.22a}$$

$$\sigma_{\hat{K}}(\hat{\boldsymbol{u}}_h) := \int_{\hat{K}} \hat{\boldsymbol{u}}_h \cdot \hat{\boldsymbol{q}} \quad \forall \hat{\boldsymbol{q}} \in [\mathcal{P}_{k-2}(\hat{K})]^2. \tag{2.22b}$$

Clearly, we have $3k$ edge DOFs and $k(k-1)$ inner DOFs, which lead to a total number of $n_{\hat{\Sigma}} = k(k+2)$ DOFs. In three dimensions, the set $\hat{\Sigma}$ is defined as

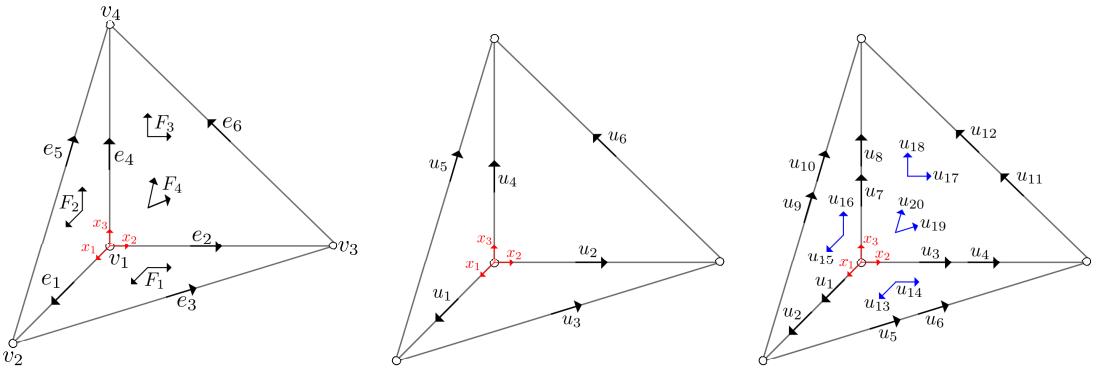
$$\sigma_{\hat{e}}(\hat{\boldsymbol{u}}_h) := \int_{\hat{e}} (\hat{\boldsymbol{u}}_h \cdot \hat{\boldsymbol{\tau}})\hat{q} \qquad \forall \hat{q} \in \mathcal{P}_{k-1}(\hat{e}), \quad \forall \hat{e} \in \hat{K} \tag{2.23a}$$

$$\sigma_{\hat{f}}(\hat{\boldsymbol{u}}_h) := \frac{1}{\|\hat{\mathcal{F}}\|} \int_{\hat{\mathcal{F}}} \hat{\boldsymbol{u}}_h \cdot \hat{\boldsymbol{q}} \qquad \forall \hat{\boldsymbol{q}} \in [\mathcal{P}_{k-2}(\hat{K})]^3 \text{ s.t. } \hat{\boldsymbol{q}} \cdot \hat{\boldsymbol{n}} = 0, \quad \forall \hat{f} \in \hat{K} \tag{2.23b}$$

$$\sigma_{\hat{K}}(\hat{\boldsymbol{u}}_h) := \int_{\hat{K}} \hat{\boldsymbol{u}}_h \cdot \hat{\boldsymbol{q}} \qquad \forall \hat{\boldsymbol{q}} \in [\mathcal{P}_{k-3}(\hat{K})]^3, \tag{2.23c}$$

where $\hat{\boldsymbol{n}}$ in Eq. (2.23b) is the unit normal to the face. The set of moments Eq. (2.23) contains $6k$ edge DOFs, $4k(k-1)$ face DOFs and $\frac{k(k-1)(k-2)}{2}$ inner DOFs. Therefore, the tetrahedral edge element has a total number of $n_{\hat{\Sigma}} = \frac{k(k^2+5k+6)}{2}$ DOFs. The local numbering of DOFs is analogous as for the hexahedral case.

Remark 1. *The face moments in the definition Eq. (2.23b) seem to differ from the rest of definitions in Eq. (2.23), which are not scaled with a geometrical entity measure. We follow here [101, Ch. 5], where this expression of face moments is used to prove affine equivalence, i.e., proving that DOFs are affine invariant under the transformation from the reference to the physical element.*



(A) Oriented cell geometry. (B) Lowest order DOFs. (C) Second order DOFs. Face DOFs depicted in blue.

FIGURE 2.2: 3D tetrahedral reference FE.

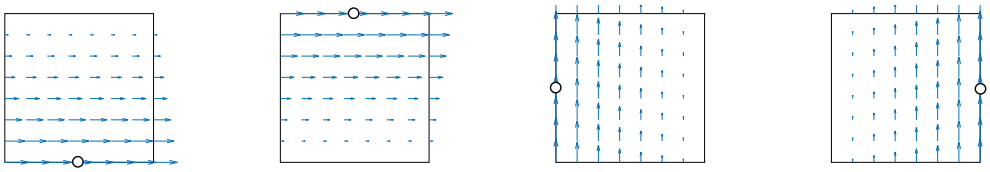
Note that in the case of the lowest order elements, i.e., $k = 1$, only DOFs associated to edges occur. For second order elements, i.e., $k = 2$, we also find DOFs related to faces (inner DOFs in 2D), and it is in orders higher than two where all kinds of DOFs occur. Note that vector-valued test functions $\hat{\boldsymbol{q}}$ involved in (2.22b), (2.23b) and (2.23c) can be understood as products between linearly independent vectors $\{\hat{\boldsymbol{\tau}}_i\}_1^d$ that form a basis in the geometrical entity of dimension d and scalar polynomials $\hat{q} \in \mathcal{P}_{k-d}$.

2.3.4 Construction of edge shape functions

Usually, low order edge elements are implemented via hard-coded expressions of their respective shape functions. However, such an approach is not suitable for high order edge FEs, which involve complex analytical expressions of the shape functions. In this section, we provide an automatic generator of arbitrary order shape functions.

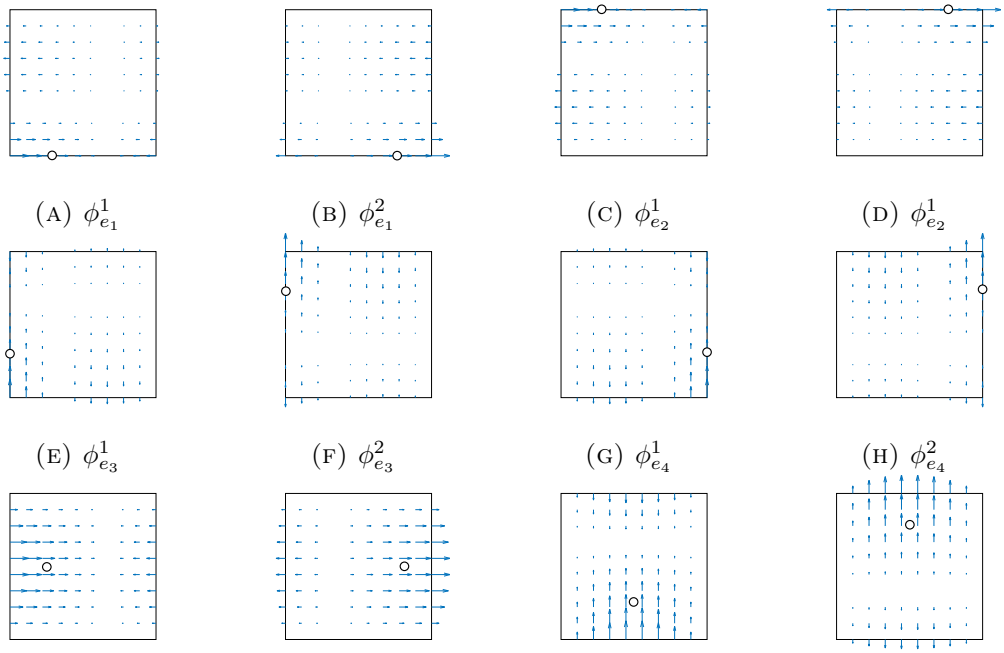
The definition of the moments for the edge FE (in (2.20a,2.20b), (2.21a-2.21c) or (2.22a,2.22b), (2.23a-2.23c) for hexahedra and tetrahedra, resp.) requires the selection of functions spanning the requested polynomial space in each case. First, we generate a *pre-basis* $\{\varphi^a\}_{a=1}^{n_\Sigma}$ that spans the local FE space $\mathcal{V}_k(\hat{K})$. To this end, we consider the tensor product of Lagrangian polynomial basis (see Sect. 2.3.2) for hexahedra or the combination of a Lagrangian basis of one order less in (2.15) plus the bases of monomials in (2.17) for tetrahedra. Our goal is to build another (canonical) basis $\{\phi^a\}_{a=1}^{n_\Sigma}$ that spans the same space and additionally satisfies $\sigma_a(\phi^b) = \delta_{ab}$ for $a, b \in \{1, \dots, n_\Sigma\}$, i.e., the basis of shape functions for the edge element. Thus, we are interested in obtaining a linear combination of the functions of the *pre-basis* such that the duality between moments and functions is satisfied. As a result, an edge shape function ϕ^a can be written as $\phi^a = \sum_{b=1}^{n_\Sigma} Q_{ab}\varphi^b$. Let us make use of Einstein's notation to provide the definition of the change of basis between the two of them. We have:

$$\phi^b = Q_{bc}\varphi^c, \qquad \sigma_a(\phi^b) = \sigma_a(Q_{bc}\varphi^c), \qquad \delta_{ab} = \sigma_a(\varphi^c)Q_{bc}, \tag{2.24}$$



(A) ϕ_{e_1} (B) ϕ_{e_2} (C) ϕ_{e_3} (D) ϕ_{e_4}

FIGURE 2.3: Vector-field plots of the shape functions in the lowest order 2D hexahedra. The indices of edges follow the geometrical entities indices for the reference hexahedron in Fig. 2.1a restricted to the plane $z = 0$. Auxiliary circles denote the geometrical entity where the moment is defined.



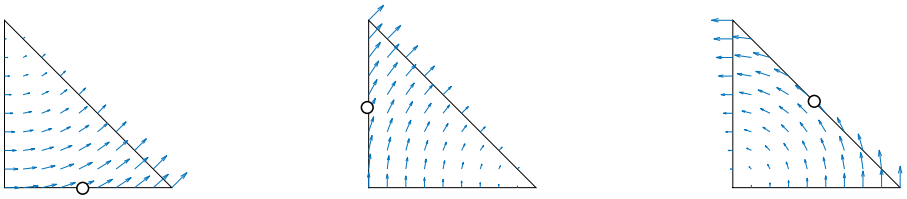
(A) $\phi^1_{e_1}$ (B) $\phi^2_{e_1}$ (C) $\phi^1_{e_2}$ (D) $\phi^1_{e_2}$

(E) $\phi^1_{e_3}$ (F) $\phi^2_{e_3}$ (G) $\phi^1_{e_4}$ (H) $\phi^2_{e_4}$

(I) ϕ^1_K (J) ϕ^2_K (K) ϕ^3_K (L) ϕ^4_K

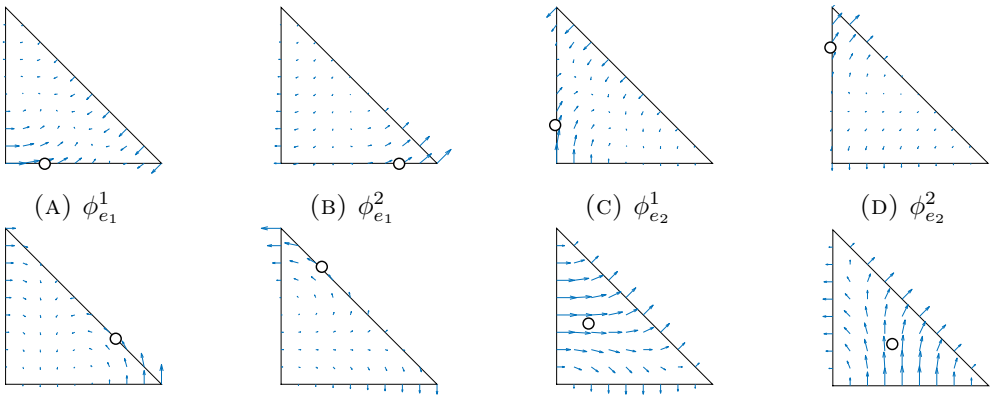
FIGURE 2.4: Vector-field plots of the shape functions in the second order hexahedral edge element. The indices of edges follow the geometrical entities indices for the reference hexahedron in Fig. 2.1a restricted to the plane $z = 0$. Auxiliary circles denote the geometrical entity where the moment is defined.

or in compact form $\boldsymbol{I} = \boldsymbol{C}\boldsymbol{\mathcal{Q}}^T$, thus $\boldsymbol{\mathcal{Q}}^T = \boldsymbol{C}^{-1}$. As a result, the edge shape functions are obtained as $\phi^a = Q_{ab}\varphi^b = C^{-1}_{ba}\varphi^b$. For the sake of illustration, we provide some examples of full sets of edge shape functions for different orders. To make the visualization easy, we provide only 2D examples for first and second order square (Figs. 2.3 and 2.4) and triangular elements (Figs. 2.5 and 2.6). In these figures, we put a circle on top of the node corresponding to the DOF dual to the shape function. The superscript indicates the local moment numbering on the particular geometrical entity.



(A) ϕ_{e_1} (B) ϕ_{e_2} (C) ϕ_{e_3}

FIGURE 2.5: Vector-field plots of the shape functions in the lowest order tetrahedral element. The indices of edges follow the geometrical entities indices for the reference tetrahedron in Fig. 2.2a restricted to the plane $z = 0$. Auxiliary circles denote the geometrical entity where the moment is defined.



(A) $\phi^1_{e_1}$ (B) $\phi^2_{e_1}$ (C) $\phi^1_{e_2}$ (D) $\phi^2_{e_2}$

(E) $\phi^1_{e_3}$ (F) $\phi^2_{e_3}$ (G) ϕ^1_K (H) ϕ^2_K

FIGURE 2.6: Vector-field plots of the shape functions in the second order edge element. The indices of edges follow the geometrical entities indices for the reference tetrahedron in Fig. 2.2a restricted to the plane $z = 0$. Auxiliary circles denote the geometrical entity where the moment is defined.

2.4 Global FE spaces and conformity

A FE space is $H(\text{curl})$-conforming if the tangential components at the interface between elements are continuous, i.e., they do not have to satisfy normal continuity over element faces. The discrete global FE space where the magnetic field solution $\boldsymbol{u}_h$ lies is defined as

$$\mathcal{ND}_k(\Omega) = \{\boldsymbol{v}_h \in H(\mathbf{curl}, \Omega) \text{ such that } \boldsymbol{v}_h|_K \in \mathcal{V}_k(K) \,\forall\, K \in \mathcal{T}_h\}, \tag{2.25}$$

where $\mathcal{V}_k$ has been defined for the hexahedral and tetrahedral edge FEs in Sect. 2.3.2.

2.4.1 Moments in the physical space and the Piola map

In this section we define a set of DOFs (moments) in the physical space that allow one to interpolate analytical functions onto the edge FE space, e.g., to enforce Dirichlet data.

One can check that the continuity of those DOFs which lay at the boundary between cells provides the desired continuity of the tangential component. On top of that, using the so-called covariant Piola mapping $\Psi_K(\hat{\boldsymbol{v}}) \doteq \hat{\Psi}_K(\hat{\boldsymbol{v}}) \circ \boldsymbol{\Phi}_K^{-1}$, where

$$\hat{\Psi}_K(\hat{\boldsymbol{v}}) \doteq \boldsymbol{J}_K^{-T}\hat{\boldsymbol{v}}, \tag{2.26}$$

it can be checked that a DOF in the reference space of a function $\hat{\boldsymbol{v}}$ on $\hat{K}$ is equal to the corresponding DOF in the physical space for $\Psi_K(\hat{\boldsymbol{v}})$ on K (see [101, Ch. 5] for details). Thus, the Piola mapping, which preserves tangential traces of vector fields [124], is the key to achieve a curl-conforming global space by using a FE space relying on a reference FE definition.

Hexahedra

Let us now define the moments for a curl-conforming FE defined on a general hexahedron. Given an integer $k \geq 1$, the set of functionals that forms the basis in two dimensions reads:

$$\sigma_e(\boldsymbol{u}_h) := \int_e (\boldsymbol{u}_h \cdot \boldsymbol{\tau}) q \quad \forall q \in \mathcal{P}_{k-1}(e), \quad \forall e \in K, \tag{2.27a}$$

$$\begin{aligned}\sigma_K(\boldsymbol{u}_h) := \int_K \boldsymbol{u}_h \cdot \boldsymbol{q}, \quad \forall \boldsymbol{q} \quad \text{obtained by mapping} \quad \boldsymbol{q} = \left(\frac{1}{\det(\boldsymbol{J}_K)}\right) \boldsymbol{J}_K \hat{\boldsymbol{q}}, \\ \hat{\boldsymbol{q}} \in \mathcal{Q}_{k-1,k-2}(\hat{K}) \times \mathcal{Q}_{k-2,k-1}(\hat{K}),\end{aligned} \tag{2.27b}$$

where $\boldsymbol{\tau}$ is the unit vector along the edge. In 3D, the set of functionals reads

$$\sigma_e(\boldsymbol{u}_h) := \int_e (\boldsymbol{u}_h \cdot \boldsymbol{\tau}) q \qquad \forall q \in \mathcal{P}_{k-1}(e), \quad \forall e \in K \tag{2.28a}$$

$$\begin{aligned}\sigma_f(\boldsymbol{u}_h) := \int_{\mathcal{F}} (\boldsymbol{u}_h \times \boldsymbol{n}) \cdot \boldsymbol{q} \qquad \forall \boldsymbol{q} \quad \text{obtained by mapping} \quad \boldsymbol{q} = \boldsymbol{J}_K^{-T}(\boldsymbol{J}_{\hat{f}} \hat{\boldsymbol{q}}), \\ \hat{\boldsymbol{q}} \in \mathcal{Q}_{k-2,k-1}(\hat{\mathcal{F}}) \times \mathcal{Q}_{k-1,k-2}(\hat{\mathcal{F}}), \quad \forall \mathcal{F} \in K\end{aligned} \tag{2.28b}$$

$$\begin{aligned}\sigma_K(\boldsymbol{u}_h) := \int_K \boldsymbol{u}_h \cdot \boldsymbol{q} \qquad \forall \boldsymbol{q} \quad \text{obtained by mapping} \quad \boldsymbol{q} = \left(\frac{1}{\det(\boldsymbol{J}_K)}\right) \boldsymbol{J}_K \hat{\boldsymbol{q}}, \\ \hat{\boldsymbol{q}} \in \mathcal{Q}_{k-1,k-2,k-2}(\hat{K}) \times \mathcal{Q}_{k-2,k-1,k-2}(\hat{K}) \times \mathcal{Q}_{k-2,k-2,k-1}(\hat{K})\end{aligned} \tag{2.28c}$$

where $\boldsymbol{n}$ is the unit normal to the face. The definition of the face moments in Eq. (2.28b) requires to transfer either the 3D vector $(\boldsymbol{u}_h \times \boldsymbol{n})$ to the face $\mathcal{F}$ or the vector $\hat{\boldsymbol{q}}$, contained in a reference face, to the 3D physical cell. We choose this latter option, which implies the transformation of the vector $\hat{\boldsymbol{q}}$ to the reference cell through the face Jacobian $\boldsymbol{J}_{\hat{f}} = \frac{\partial \boldsymbol{\Phi}_{\hat{K}}}{\partial \boldsymbol{x}_{\hat{f}}}$.

Tetrahedra

On a general tetrahedron K, we define the moments for a curl-conforming FE as follows. Given an integer $k \geq 1$, the set of functionals that forms the basis in two dimensions

reads

$$\sigma_e(\boldsymbol{u}_h) := \int_e (\boldsymbol{u}_h \cdot \boldsymbol{\tau}) q \quad \forall q \in \mathcal{P}_{k-1}(e), \quad \forall e \in K \tag{2.29a}$$

$$\begin{aligned} \sigma_K(\boldsymbol{u}_h) := \int_K \boldsymbol{u}_h \cdot \boldsymbol{q} \quad \forall \boldsymbol{q} \quad \text{obtained by mapping} \\ \boldsymbol{q} = \left(\frac{1}{\det(\boldsymbol{J}_K)} \right) \boldsymbol{J}_K \hat{\boldsymbol{q}}, \quad \hat{\boldsymbol{q}} \in [\mathcal{P}_{k-2}(\hat{K})]^2. \end{aligned} \tag{2.29b}$$

Clearly, we have $3k$ edge DOFs and $k(k-1)$ inner DOFs, which lead to a total number of $n_\Sigma = k(k+2)$ DOFs. In three dimensions the set Σ is defined as

$$\sigma_e(\boldsymbol{u}_h) := \int_e (\boldsymbol{u}_h \cdot \boldsymbol{\tau}) q \qquad \forall q \in \mathcal{P}_{k-1}(e), \quad \forall e \in K \tag{2.30a}$$

$$\sigma_f(\boldsymbol{u}_h) := \frac{1}{\|\mathcal{F}\|} \int_{\mathcal{F}} \boldsymbol{u}_h \cdot \boldsymbol{q} \qquad \forall \boldsymbol{q} = \boldsymbol{J}_K \hat{\boldsymbol{q}} \quad \text{s.t. } \hat{\boldsymbol{q}} \cdot \hat{\boldsymbol{n}} = 0, \tag{2.30b}$$

$$\hat{\boldsymbol{q}} \in [\mathcal{P}_{k-2}(\hat{f})]^3 \quad \forall f \in K$$

$$\begin{aligned} \sigma_K(\boldsymbol{u}_h) := \int_K \boldsymbol{u}_h \cdot \boldsymbol{q} \qquad \forall \boldsymbol{q} \quad \text{obtained by mapping} \\ \boldsymbol{q} = \left(\frac{1}{\det(\boldsymbol{J}_K)} \right) \boldsymbol{J}_K \hat{\boldsymbol{q}}, \quad \hat{\boldsymbol{q}} \in [\mathcal{P}_{k-3}(\hat{K})]^3. \end{aligned} \tag{2.30c}$$

2.4.2 Nédélec interpolator

The space of edge FE functions can be represented as the range of an interpolation operator π^h that is well defined for sufficiently smooth functions $\boldsymbol{u} \in H(\mathbf{curl}, \Omega)$ by

$$\pi^h(\boldsymbol{u}) := \sum_a u^a(\boldsymbol{u}) \phi^a \tag{2.31}$$

where $u^a(\boldsymbol{u}) = \sigma^a(\boldsymbol{u})$ are the evaluation of the moments for the function $\boldsymbol{u}$ described for the hexahedra and tetrahedra cases in Sect. 2.4.1 for all $e \in \mathcal{E}$, $F \in \mathcal{F}$ and $K \in \mathcal{T}_h$. Note that Dirichlet boundary conditions can be strongly imposed in the resulting system (usual implementation in FE codes) by evaluating the moments corresponding to edges and/or faces on the Dirichlet boundary given the analytical expression of the function to be imposed.

2.4.3 Global DOFs

In order to guarantee global inter-element continuity with Piola-mapped elements, special care has to be taken with regard to the orientation of edges and faces at the cell level. Let us consider a global numbering for the vertices in a mesh and a local numbering at the cell level. Given an edge/face, sorting its vertices with respect to the local (resp. global) index of their vertices, one determines the local (resp. global) orientation of the edge/face. A mesh in which the local and global orientation of all its edges and faces

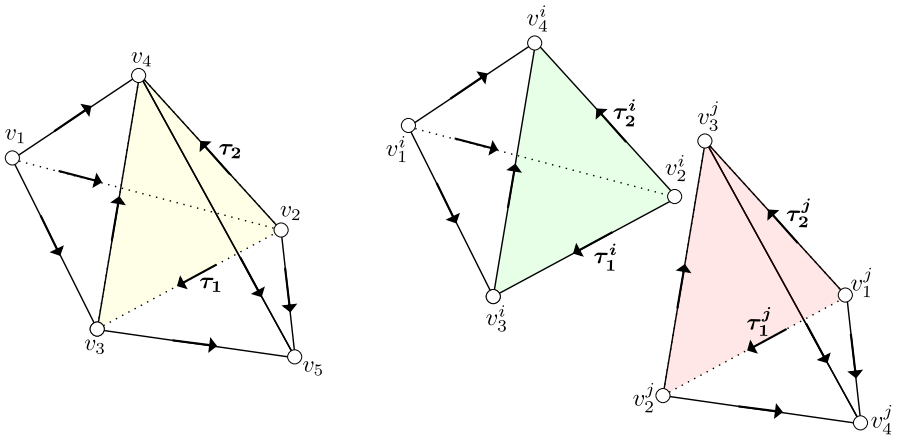


FIGURE 2.7: Oriented mesh. Global information (left) and its local, oriented, counterpart (right) for two elements K_i and K_j. Common edges for two adjacent tetrahedra will always agree in its direction, as well as chosen tangent vectors $\boldsymbol{\tau_1}$ and $\boldsymbol{\tau_2}$ to the common face.

coincide is called an *oriented mesh*. For oriented meshes, common edges or faces for two adjacent tetrahedra will always agree in its orientation, thus represent the same global DOF. Tetrahedral meshes are oriented with a simple local renumbering [124], and we restrict ourselves to hexahedral octree meshes that are oriented by construction.

Local DOFs are uniquely determined by the cell in which they are defined, the geometrical entity within the reference cell that owns them and the local numbering of DOFs within the geometrical entity (see Sect. 2.3.3). The local numbering only depends on the orientation of the geometrical entity through the ordering of its vertices.

Global DOFs are defined as an equivalence class over the union of the local DOFs for all cells. Two local DOFs are the same global DOF if and only if they belong to the same geometrical object and the same local numbering within the geometrical object in their respective cells. The previous equivalence class leads to a curl-conforming FE space if local orientations of geometrical objects coincide with a global orientation, i.e., on oriented meshes only.

For the sake of illustration, see the simple mesh depicted in Fig. 2.7, composed by two tetrahedra defined by the global vertices v_1, v_2, v_3, v_4 and v_5. The two elements share a common face, defined by 3 vertices that have different local indices for all $v_i \in K_i$ and $v_j \in K_j$. The ordering convention, local indexing according to ascending global indices, ensures that both triangles agree on the direction of the common edges and the common face. Note that the Jacobian of the transformation may become negative with the orientation procedure. We only need to make sure that the absolute value of the Jacobian is taken whenever the measure of the change of basis is applied. The situation is much more involved for general hexahedral meshes. In any case, our implementation of hexahedral meshes relies on octree meshes, where consistency is ensured. It is easy to check that an octree mesh in which a cell inherits the orientation of its parent is *oriented* by construction.

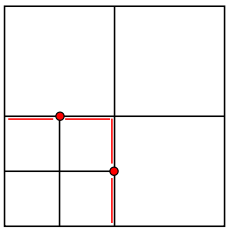
2.5 Edge FEs in h-adaptive meshes

In this section, we expose an implementation procedure for *non-conforming* meshes and edge FEs of arbitrary order. In the following, we restrict ourselves to hexahedral meshes, even though the extension to tetrahedral meshes is straightforward. Our procedure can also be readily extended to any FE space that relies on a *pre-basis* of Lagrangian polynomial spaces plus a change of basis, e.g., Raviart-Thomas FEs.

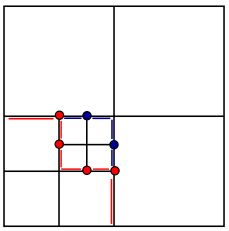
2.5.1 Hierarchical AMR on octree-based meshes

The AMR generation relies on hierarchically refined octree-based hexahedral meshes [139] and the p4est [47] library is used for such purpose. In this method, one must enforce constraints to ensure the conformity of the global FE space, which amounts to compute constraints between coarser and refined geometrical entities shared by two cells with different level of refinement [55]. For Lagrangian elements, restriction operators in geometrical sub-entities are generally obtained by evaluating the shape functions associated with the coarse side of the face at the interpolation points of the shape functions on the refined side of the face [31]. The idea is conceptually equivalent for edge elements but considerably more complex to implement because DOFs do not represent nodal values but moments on top of geometrical entities. In this thesis, we follow an approach based on the relation between a Lagrangian *pre-basis*, where moments are trivial to evaluate, and the edge basis, so we can avoid the evaluation of edge moments on the refined cell to compute constraints.

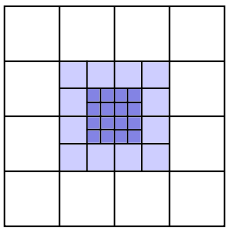
Let $\mathcal{T}_h^{'}$ be a conforming partition of Ω into a set of hexahedra (quadrilateral in 2D) geometrical cells K. $\mathcal{T}_h^{'}$ can, e.g., be as simple as a single quadrilateral or hexahedron. Starting from $\mathcal{T}_h^{'}$, hierarchical AMR is a multi-step process in which at each step, some cells of the input mesh are marked for refinement. A cell marked for refinement is partitioned into four (2D) or eight (3D) children cells by bisecting all cell edges $\mathcal{E}_K$. Let us denote by $\mathcal{T}_h$ the resulting partition of Ω. $\mathcal{T}_h$ can be thought as a collection of quads (2D) or octrees (3D) where the cells of $\mathcal{T}_h^{'}$ are the roots of these trees, and children cells branch off their parent cells. The leaf cells in this hierarchy form the mesh in the usual meaning, i.e., $\mathcal{T}_h$. Thus, for every cell $K \in \mathcal{T}_h$ we can define $\ell(K)$ as the level of K in the aforementioned hierarchy, where $\ell(K) = 0$ for the root cells, and $\ell(K) = \ell(\text{parent}(K)) + 1$ for any other cell. Clearly, the cells in $\mathcal{T}_h$ can be at different levels of refinement. Thus, these meshes are *non-conforming*. In order to complete the definition of $\mathcal{T}_h$, let us introduce the concept of *hanging* geometrical entities. For every cell $K \in \mathcal{T}_h$, consider its set of vertices $\mathcal{N}_K$, edges $\mathcal{E}_K$ and faces $\mathcal{F}_K$. We can represent the set of geometrical entities that have lower dimension than the cell by $\mathcal{G}_K = \mathcal{N}_K \cup \mathcal{E}_K \cup \mathcal{F}_K$. Its global counterpart is defined as $\mathcal{G}_\mathcal{T} = \cup_{K \in \mathcal{T}_h} \mathcal{G}_K$. For every geometrical entity $s \in \mathcal{G}_\mathcal{T}$, we can represent by $\mathcal{T}_s$ the set of cells $K \in \mathcal{T}_h$ such that $s \in \mathcal{G}_K$. Additionally, $\tilde{\mathcal{T}}_s$ is defined as the set of cells $K \in \mathcal{T}_h$ such that $s \subsetneq s'$ for some $s' \subset \mathcal{G}_K$. Roughly speaking, one set contains all the cells where the geometrical



(A) *Hanging* entities marked in red.



(B) Not permitted *hanging* entities marked in blue.



(C) Centered refinement pattern. $\ell \in \{2, 3, 4\}$ from clearest to darkest.

FIGURE 2.8: h-adaptive refined (single octree) *non-conforming* meshes.

entity is found while the other set contains all the cells where the geometrical entity is a strict subset of a coarser geometrical entity. A geometrical entity $s \in \mathcal{G}_{\mathcal{T}}$ is *hanging* (or *improper*) if and only if there exists at least one neighbouring cell in $\tilde{\mathcal{T}}_s$. We represent the set of hanging geometrical entities with $\mathcal{G}_{\mathcal{T}}^{\text{hg}}$. The definition of the proposed AMR approach is completed by imposing the so-called 2:1 balance restriction, which is used in mesh adaptive methods by a majority of authors as a reasonable trade-off between performance gain and complexity of implementation [139, 47, 55].

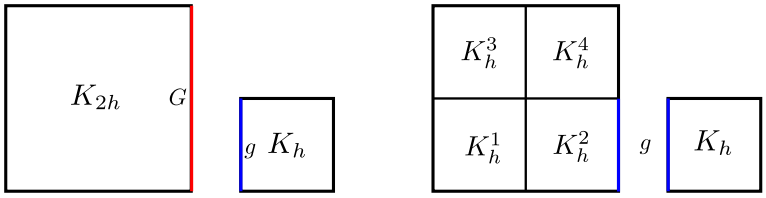
Definition 1. *A d-tree ($d = \{2, 3\}$) is 2:1 k-balanced if and only if, for any cell $K \in \mathcal{T}_h$ there is no $s \in \mathcal{G}_K$ of dimension $m \in [k, d)$ having non-empty intersection with the closed domain of another finer cell $K' \in \tilde{\mathcal{T}}_h$ such that $\ell(K') - \ell(K) > 1$.*

For edge FEs it is enough to consider 1-balance since vertices do not contain any associated DOF. For the sake of clarity, in Figs. 2.8a and 2.8b, allowed *hanging* geometrical entities are depicted in red, whereas in Fig. 2.8b, not allowed ones are shown in blue. Clearly, the latter mesh is the result of a refinement process that does not accomplish the 2:1 balance, thus not permitted in our AMR approach. Note that in order to enforce the 2:1 balance in the situation depicted in Fig. 2.8b, one would need to apply additional refinement to some cells with lower values for $\ell(K)$ until the restriction (1) is satisfied.

2.5.2 Conformity of the global FE space

In order to preserve the conformity of the FE space $\mathcal{ND}_k(\Omega)$, we cannot allow an arbitrary value for DOFs placed on top of *hanging* geometrical entities, which will be denoted by *hanging* DOFs. Our approach is to eliminate the *hanging* DOFs of the global system by defining a set of constraints such that curl-conformity is preserved. We propose an algorithm that computes the edge FE constraints by relying on the ones of Lagrangian FEs and the change of basis in Sect. 2.3.4. This way, one can reuse existing ingredients in a nodal-based AMR code and work already required to define the edge FE shape functions.

The computation of constraints requires some preliminary work at the geometrical level, i.e., independent of the FE space being used. Let us first compute the set of hanging



(A) Fine cell $K_h \equiv K(g)$ and coarse cell $K_{2h} \equiv K(G(g))$.

(B) Coarse cell refined into children cells $\{K_h^s\}_{s=1}^4$. In this example, $s(g) = 2$.

FIGURE 2.9: The coarse cell K_{2h} is refined in order to meet the highest level of refinement. Then, the index $s(g)$ can be determined.

geometrical entities $\mathcal{G}_{\mathcal{T}}^{\text{hg}}$. For every $g \in \mathcal{G}_{\mathcal{T}}^{\text{hg}}$, let us compute the coarser geometrical entity $G(g) \in \mathcal{G} \setminus \mathcal{G}_{\mathcal{T}}^{\text{hg}}$ that contains it. Let us assign to every $g \in \mathcal{G}_{\mathcal{T}}$ one cell such that $K(g) \in \mathcal{T}_g$. With this information, for every $g \in \mathcal{G}_{\mathcal{T}}^{\text{hg}}$, we can extract the fine cell $K_h \equiv K(g)$ and the coarse cell $K_{2h} \equiv K(G(g))$ (see Fig. 2.9a).

The coarse cell K_{2h} can be refined (once) to meet the level of refinement of K_h. (The coarse cell is only refined for the computation of constraints so the original mesh is not affected.) Let us consider its children cells $\{K_h^s\}_{s=1}^{2^d}$ obtained after isotropic refinement by the procedure exposed in Sect. 2.5.1, see Fig. 2.9b. We represent the patch of subcells with $\tilde{K}_h = \bigcup_{i=1}^{2^d} K_h^s$. We can determine a subcell index $s(g)$ such that $g \in \mathcal{G}_{K_h^{s(g)}}$; $s(g)$ is not unique in general, see Fig. 2.9b.

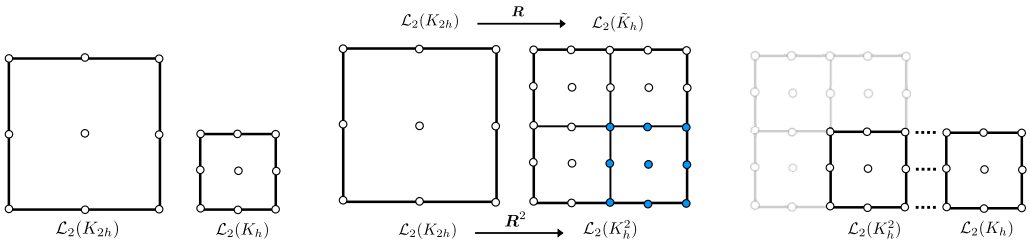
Constraints for Lagrangian FEs

In this section, we compute the constraints for Lagrangian FE spaces. Using the geometrical information above, given a hanging geometrical entity g, let us consider the Lagrangian FE spaces $\mathcal{L}_{\boldsymbol{k}}(K_{2h})$, $\mathcal{L}_{\boldsymbol{k}}(\tilde{K}_h)$ and $\mathcal{L}_{\boldsymbol{k}}(K_h^s)$ (see Sect. 2.4 for the definitions, Figs. 2.10a and 2.10b for an illustration). The objective is to compute the constraints over DOFs on $g \in \mathcal{G}_{\mathcal{T}}^{\text{hg}}$ that enforce global continuity. This continuity is enforced by evaluating the coarse cell K_{2h} shape functions on the fine cell K_h nodes (DOFs) on g. Since both cells share the FE order, such set of constraints leads to full continuity across g [130, Ch. 3]. Let us explain the steps followed to compute these constraints.

Clearly, $\mathcal{L}_{\boldsymbol{k}}(K_{2h}) \subset \mathcal{L}_{\boldsymbol{k}}(\tilde{K}_h)$, so we can apply the Lagrangian interpolant to inject $\boldsymbol{u}^{2h} \in \mathcal{L}_{\boldsymbol{k}}(K_{2h})$ into $\mathcal{L}_{\boldsymbol{k}}(\tilde{K}_h)$. For this purpose, consider the set of original Lagrangian basis functions $\{\varphi^j\}_{j=1}^{n_k}$ spanning $\mathcal{L}_{\boldsymbol{k}}(K_{2h})$, which has cardinality $n_k = \prod_i^d (k_i + 1)$ (see Sect. 2.3.2 for details). Further, consider the set of moments $\{\sigma_i\}_{i=1}^{\tilde{n}_k}$ uniquely defining a solution in $\mathcal{L}_{\boldsymbol{k}}(\tilde{K}_h)$, which has cardinality $\tilde{n}_k = \prod_i^d (2k_i + 1)$ (see Sect. 2.3.2). Given the vector of DOF values of $\boldsymbol{u}^{2h}$, the restriction operator $\boldsymbol{R} : \mathcal{L}_{\boldsymbol{k}}(K_{2h})' \to \mathcal{L}_{\boldsymbol{k}}(\tilde{K}_h)'$

$$R_{ij} \doteq \sigma_i(\varphi^j), \qquad i = 1, \dots, \tilde{n}_k, \quad j = 1, \dots, n_k, \tag{2.32}$$

provides the DOFs of the interpolated function.



(A) FE spaces representation.

(B) A function in the coarse cell injected in the refined space or its restriction to a subcell (denoted in blue).

(C) DOFs identification as conforming meshes.

FIGURE 2.10: Scheme for enforcing continuity at common entities for two cells with different level of refinement with second order Lagrangian FEs.

Further, note that given a *hanging* DOF $u^{h,i}$ on top of g, it is easy to check that $R_{ij} = 0$ if DOF $u^{2h,j}$ is not on $G(g)$.

For every subcell K_h^s one can readily determine the map $w_s(\cdot)$ such that, given the local cell index $i \in \{1, \dots, n_k\}$ for moments defined on $\mathcal{L}_{\boldsymbol{k}}(K_h^s)'$ ($s \in \{1, \dots, 2^d\}$), returns a global moment index $i' \in \{1, \dots, \tilde{n}_k\}$ in the space $\mathcal{L}_{\boldsymbol{k}}(\tilde{K}_h)'$. It leads to the subcell restriction operator (of DOF values) $\boldsymbol{R}^s : \mathcal{L}_{\boldsymbol{k}}(K_{2h})' \to \mathcal{L}_{\boldsymbol{k}}(K_h^s)'$ defined as

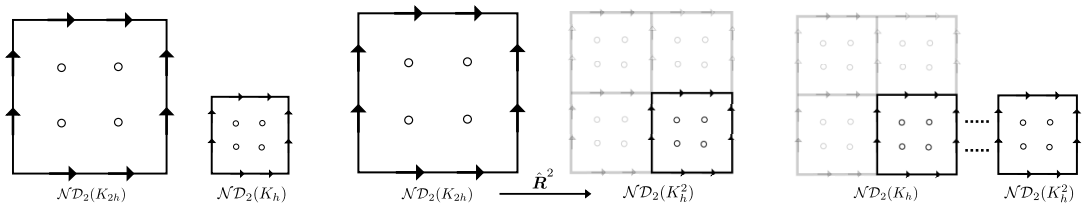
$$R_{ij}^s \doteq R_{w_s(i)j} \qquad i = 1, \dots, n_k, \quad j = 1, \dots, n_k. \tag{2.33}$$

A representation of the action of $\boldsymbol{R}$ and $\boldsymbol{R}^s$ can be seen in Fig. 2.10b. Note that they are independent of the level of refinement and the cell in the physical space; they are computed only once at the reference cell.

We can identify every DOF $i \in \mathcal{L}_{\boldsymbol{k}}(K_h)'|_g$ with a DOF $i' \in \mathcal{L}_{\boldsymbol{k}}(K_h^{s(g)})'|_g$ as for conforming Lagrangian meshes; two local DOF values must be identical if their corresponding nodes are located at the same position (see Fig. 2.10c). Finally, *the DOF value* $i \in \mathcal{L}_{\boldsymbol{k}}(K_h)'|_g$ *is constrained by the DOF values of the coarse cell through the row* i' *of* $\boldsymbol{R}^{s(g)}$.

Constraints for edge FEs

In order to compute the constraints for edge FEs one could follow a similar procedure as the one above, see Fig. 2.11 for an illustration. Nevertheless, we propose a different approach for the implementation of restriction operator in edge FEs (see Fig. 2.11b), which allows us to reuse the restriction operator defined for nodal FEs and avoids the evaluation of edge moments in subcells. Besides, our approach for computing the edge FE constraints is applicable to *any* FE space that relies on a pre-basis of Lagrangian polynomials plus a change of basis (e.g., Raviart-Thomas FEs) without *any* additional implementation effort.



(A) FE spaces representation.

(B) Restriction of a function injected in the refined space to a subcell.

(C) DOFs identification as conforming meshes.

FIGURE 2.11: Scheme for enforcing continuity at common entities for two cells with different level of refinement with second order edge FEs.

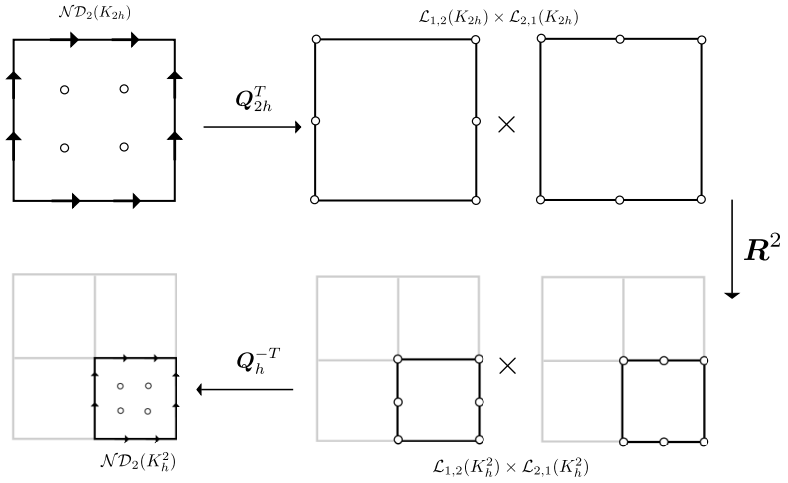


FIGURE 2.12: Sequence of spaces in order to compute the restriction operator entries for K_h^2 with second order edge FEs. Representation of edge and inner DOFs by arrows and nodes, resp.

The idea is to build the restriction operator for edge FEs as a composition of operators. We recall the change of basis matrix $\boldsymbol{Q} : \mathcal{L}_{\boldsymbol{k}}(K) \to \mathcal{ND}_k(K)$ between Lagrangian and edge FE basis functions (see Sect. 2.3.4). It leads to the adjoint operator $\boldsymbol{Q}^T : \mathcal{ND}_k(K)' \to \mathcal{L}_{\boldsymbol{k}}(K)'$ and its inverse $\boldsymbol{Q}^{-T} : \mathcal{L}_{\boldsymbol{k}}(K)' \to \mathcal{ND}_k(K)'$. As a result, given a subcell K_h^s, we can define the restriction operator $\hat{\boldsymbol{R}}^s \doteq \boldsymbol{Q}^{-T}\boldsymbol{R}^s\boldsymbol{Q}^T : \mathcal{ND}_k(K_{2h})' \to \mathcal{ND}_k(K_h^s)'$ that takes the edge FE DOF values in the coarse cell K_{2h} and provides the ones of the interpolated function (see (2.31)) in K_h^s. Again, $\hat{\boldsymbol{R}}^s$ can be computed once at the reference cell. An illustration of the sequence of spaces is shown in Fig. 2.12.

Given a hanging edge/face $g \in \mathcal{G}_{\mathcal{T}}^{\text{hg}}$ (vertices do not have associated DOFs for edge FEs), the constraints of its DOFs are computed as follows. We can identify every DOF $i \in \mathcal{ND}_k(K_h)'|_g$ with a DOF $i' \in \mathcal{ND}_k(K_h^{s(g)})'|_g$ as for conforming meshes (see the equivalence class in Sect. 2.4.3 and Fig. 2.11c). As a result, *the DOF value* $i \in \mathcal{ND}_k(K_h)'|_g$ *is constrained by the DOF values of the coarse cell through the row* i' *of* $\hat{\boldsymbol{R}}^{s(g)}$.

Global assembly for non-conforming meshes

The definition of the constraints is used in the assembly of the linear system as follows. First of all, let us write the problem Eq. (2.10) in algebraic form. The solution $\boldsymbol{u} \in \mathcal{ND}_k(\Omega)$ is expanded by $\{\phi^a\}_{a=1}^n$. The elemental matrices are defined as $\boldsymbol{\mathcal{M}}_{ij} = \int_K \beta\phi^j \cdot \phi^i$ and $\boldsymbol{\mathcal{K}}_{ij} = \int_K (\alpha\boldsymbol{\nabla}\times\phi^j)\cdot(\boldsymbol{\nabla}\times\phi^i)$, whereas the elemental right-hand side is the discrete vector $\boldsymbol{f}_i = \int_K \boldsymbol{f}_K \cdot \phi^i$ for $i, j \in \{1, \ldots, n\}$. The usual assembly for every $K \in \mathcal{T}_h$ is performed to obtain the global matrix and array, hence the algebraic form reads $\boldsymbol{\mathcal{A}}\boldsymbol{u} = \boldsymbol{f}$, where $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{K}} + \boldsymbol{\mathcal{M}}$. Let us now distinguish between the set of *conforming* DOFs $\boldsymbol{u}_c$ and the set of *non-conforming* DOFs $\boldsymbol{u}_{nc}$, whose cardinalities are $n_c + n_{nc} = n$. One can write the constraints that $\boldsymbol{u}_{nc}$ must satisfy in compact form, $\boldsymbol{u}_{nc} = \boldsymbol{G}\boldsymbol{u}_c$, where the entries of $\boldsymbol{G}$ (of dimension $n_{nc} \times n_c$) are given by the described procedure for obtaining the *constraining* factors between the *constrained* and the *constraining* part. Then, the solution $\boldsymbol{u}_h$, expressed in terms of both types of DOFs can be written as

$$\boldsymbol{u}_h = \begin{bmatrix} \boldsymbol{u}_c \\ \boldsymbol{u}_{nc} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{G} \end{bmatrix} \boldsymbol{u}_c = \boldsymbol{P}\boldsymbol{u}_c, \tag{2.34}$$

and we can write the constrained problem only in terms of the *conforming* DOFs $\boldsymbol{u}_c$ as

$$\bar{\boldsymbol{\mathcal{A}}}\boldsymbol{u}_c = \bar{\boldsymbol{f}}, \tag{2.35}$$

where $\bar{\boldsymbol{\mathcal{A}}} = \boldsymbol{P}^T\boldsymbol{\mathcal{A}}\boldsymbol{P}$ and $\bar{\boldsymbol{f}} = \boldsymbol{P}^T\boldsymbol{f}$. Our implementation directly builds the constrained operator $\bar{\boldsymbol{\mathcal{A}}}$ by *locally* applying the constraints to eliminate $\boldsymbol{u}_{nc}$ DOFs using the constraints given by $\boldsymbol{G}$ in the assembly process (see [15] for further details). Once the solution for *conforming* DOFs is obtained, *hanging* DOF values are recovered by Eq. (2.34) as a postprocess.

2.6 Numerical experiments

In this section, we test the implementation of edge elements and $H(\text{curl})$-conforming spaces in the scientific software `FEMPAR` (see Chapter 1 for an introduction). Regarding the content of this chapter, `FEMPAR` supports arbitrary order edge FEs on both hexahedra and tetrahedra, on either structured and unstructured conforming meshes, and also mesh generation by adaptation using hierarchically refined octree-based meshes. In order to test our implementation, we will compare both theoretical and experimental convergence rates. Generally, log-log plots of the computed error in the considered norms (L_2-norm or $H(\text{curl})$-norm) against different values of h or number of DOFs will be shown. Let us first cite some *a priori* error estimates for edge FEs of the first kind. In the $H(\text{curl})$-norm, we find the following optimal estimate, presented in [8]:

Theorem 1. *If $\mathcal{T}_h$ is a regular family of triangulations on Ω for $h > 0$, then there exists a constant $C > 0$ such that*

$$\|\boldsymbol{v} - \pi_h^k \boldsymbol{v}\|_{H(\mathrm{curl})} \leq C h^{\min\{r,k\}} \|\boldsymbol{v}\|_{H^r(\mathrm{curl})}, \tag{2.36}$$

for all $\boldsymbol{v} \in H^r(curl)$, where $r > \frac{1}{2}$ determines the regularity of the function $\boldsymbol{v}$, which is valid for H(curl)-conforming elements. If the function $\boldsymbol{v}$ is smooth enough to have bounded derivatives such that $s > k$, then the estimate states that superlinear convergence can be achieved as we increase the polynomial order k.

For the $L_2(\Omega)$ approximation, the following convergence rate can be expected [104].

Theorem 2. *If $\mathcal{T}_h$ is a regular family of triangulations on Ω for $h > 0$, then there exists a constant $C > 0$ such that*

$$\|\boldsymbol{v} - \pi_h^k \boldsymbol{v}\|_{L_2(\Omega)} \leq C h^k \|\boldsymbol{v}\|_{H^k(\Omega)}. \tag{2.37}$$

In general, we choose analytical functions that do not belong to the FE space. We show convergence plots with respect to the mesh size for uniform mesh refinement or the number of DOFs for h-adaptive mesh refinement. In all cases, we will solve the reference problem Eq. (2.6) with homogeneous, scalar parameters $\alpha = \beta = 1$. Unless otherwise stated, the results are computed in the unit box domain $\Omega := [0, 1]^d$. We utilize the method of *manufactured* solutions, i.e., to plug an analytical function $\boldsymbol{u}^*$ in the exact form of the problem and obtain the corresponding source term that verifies the equation. Then, one can solve the problem for the unknown $\boldsymbol{u}$ so that the computed solution must converge to the exact solution with mesh refinement.

2.6.1 Uniform mesh refinement

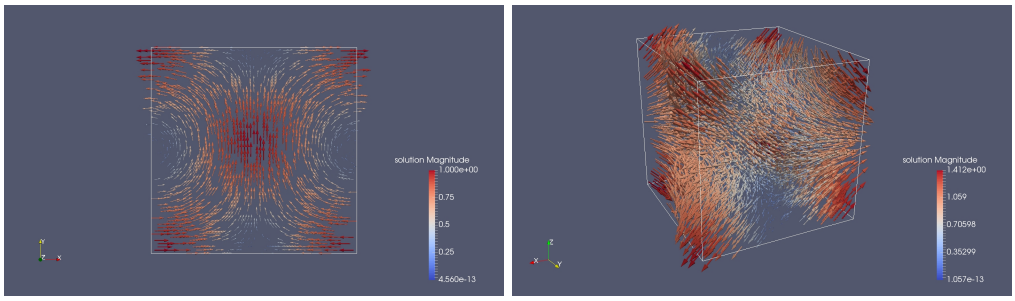
In this section, the experimental rate of convergence will be numerically included in the legend as the value for the slope computed with the two last available data points in each plot. We will make use of the following analytical function and source term for all the 2D cases presented in this section:

$$\boldsymbol{u}^* = \begin{bmatrix} \cos(\pi x_1)\cos(\pi x_2) \\ \sin(\pi x_1)\sin(\pi x_2) \end{bmatrix}, \qquad \boldsymbol{f} = (2\pi^2 + 1)\boldsymbol{u}^*,$$

whereas in 3D cases the analytical function and corresponding source term are given by

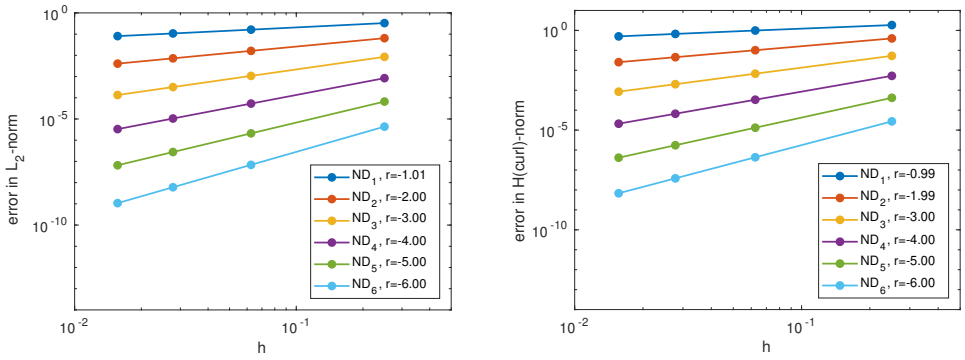
$$\boldsymbol{u}^* = \begin{bmatrix} \cos(\pi x_1)\cos(\pi x_2) \\ \sin(\pi x_2)\sin(\pi x_3) \\ \cos(\pi x_1)\cos(\pi x_3) \end{bmatrix}, \qquad \boldsymbol{f} = (\pi^2 + 1)\boldsymbol{u}^* + \pi^2 \begin{bmatrix} \sin(\pi x_1)\sin(\pi x_3) \\ \sin(\pi x_1)\sin(\pi x_2) \\ \cos(\pi x_2)\cos(\pi x_3) \end{bmatrix}.$$

Dirichlet boundary conditions are strongly imposed over the entire boundary, i.e., $\partial\Omega_D := \partial\Omega$, where we enforce the tangent component of the analytical function $\boldsymbol{u}_\tau^*$.



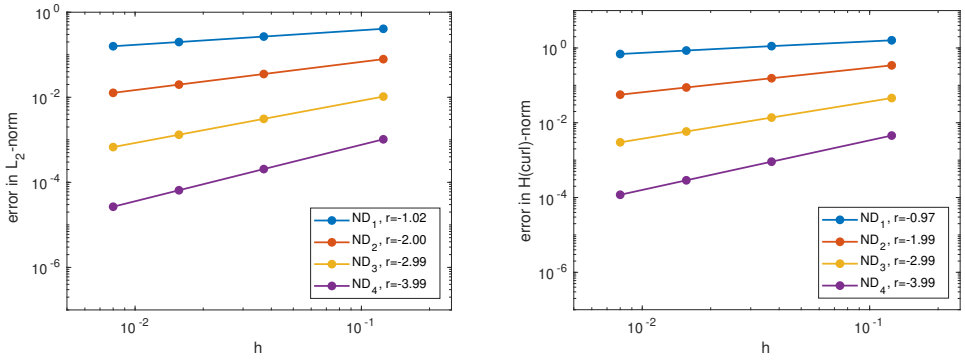
(A) Analytical function $\boldsymbol{u}^*$ in the 2D case. (B) Analytical function $\boldsymbol{u}^*$ in the 3D case.

FIGURE 2.13: Analytical functions.



(A) L_2-norm of the error. (B) $H(\text{curl})$-norm of the error.

FIGURE 2.14: Error norms for different orders 2D hexahedral edge FEs.

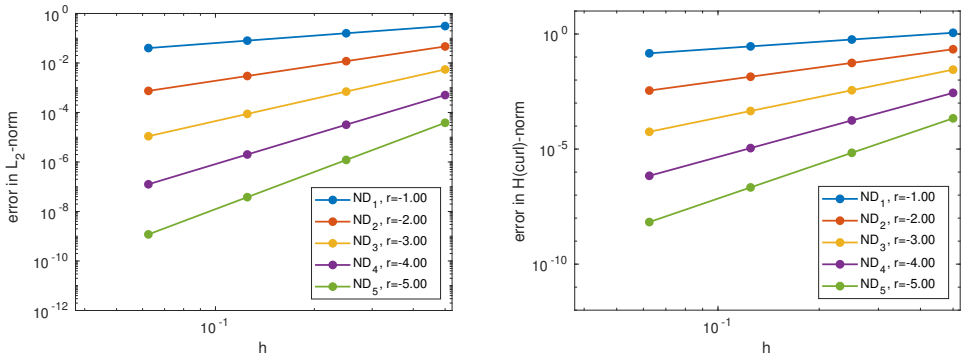


(A) L_2-norm of the error. (B) $H(\text{curl})$-norm of the error.

FIGURE 2.15: Error norms for different orders 3D hexahedral edge FEs.

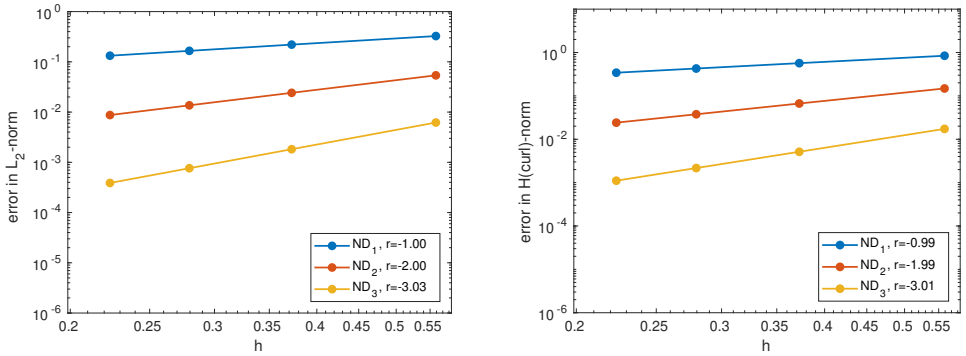
Hexahedral meshes

Structured simulations are performed with a structured mesh on the unit box domain with the same number of elements n_K in each direction. Figs. 2.14 and 2.15 show the convergence rates with the order of the element k for 2D and 3D cases, resp. In all cases, the expected convergence ratio (see (2.36) and (2.37)) is achieved, presented up to $k = 6$ in 2D and $k = 4$ in 3D.



(A) L_2-norm of the error. (B) $H(\text{curl})$-norm of the error.

FIGURE 2.16: Error norms for different orders 2D tetrahedral edge FEs.



(A) L_2-norm of the error. (B) $H(\text{curl})$-norm of the error.

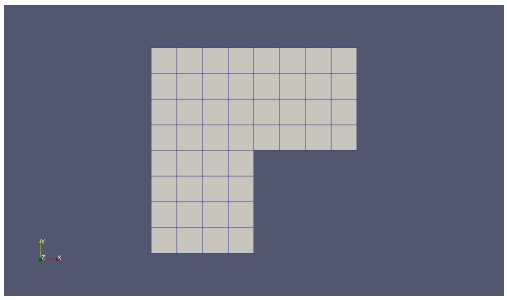
FIGURE 2.17: Error norms for different orders 3D tetrahedral edge FEs.

Tetrahedral meshes

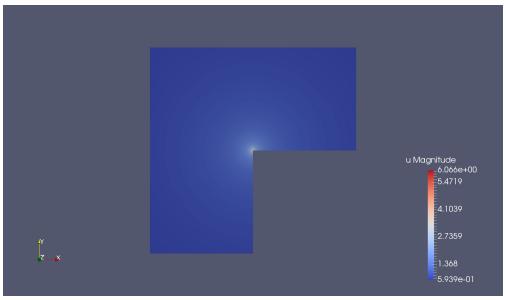
Edge FEs are tested in this section with tetrahedral mesh partitions of the domain Ω. Consider a family of tetrahedral meshes $\{\mathcal{T}_m\}_{m=1}^{M}$, obtained by structured, hexahedral meshes plus triangulation of hexahedral cells. Here the element size denotes the usual definition $h = \max_{K \in T_m} h_K$, being h_K the diameter of the largest circumference or sphere containing K for 2D and 3D, resp. Convergence results in Figs. 2.16 and 2.17 are computed with this family of meshes. In all cases, computed convergence ratios are consistent with the estimates in (2.36) and (2.37).

2.6.2 *h*-adaptive mesh refinement

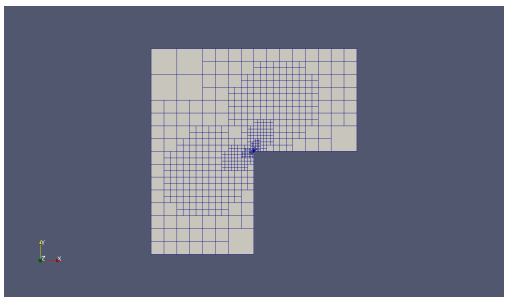
In this section, we present results for meshes obtained by adaptive refinement from an initial structured, hexahedral *conforming* mesh. The refinement process follows the usual steps: 1) solve the problem on a given mesh, 2) compute an estimation of the local error contribution at every cell using the solution computed at the previous step, 3) mark the cells with more error for refinement, and 4) refine the mesh and restart the process if the stopping criterion is not fulfilled. Since the analytical solution is available, we compute



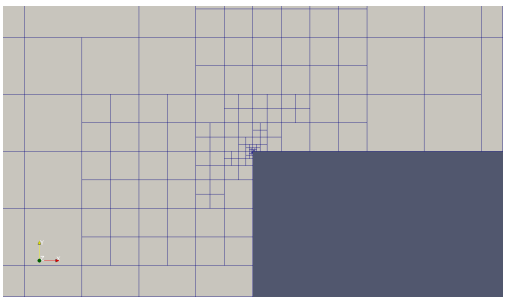
(A) L-shaped domain with initial mesh.



(B) Analytical solution for the 2D L-shape domain with a singularity at the corner.



(C) L-shaped domain with refined mesh after 18 iterative mesh refinements.



(D) Corner zoom of the refined mesh.

FIGURE 2.18: fichera 2D problem. At each mesh refinement step the 5% of cells with highest local cell contribution to the L_2-norm of the error are refined.

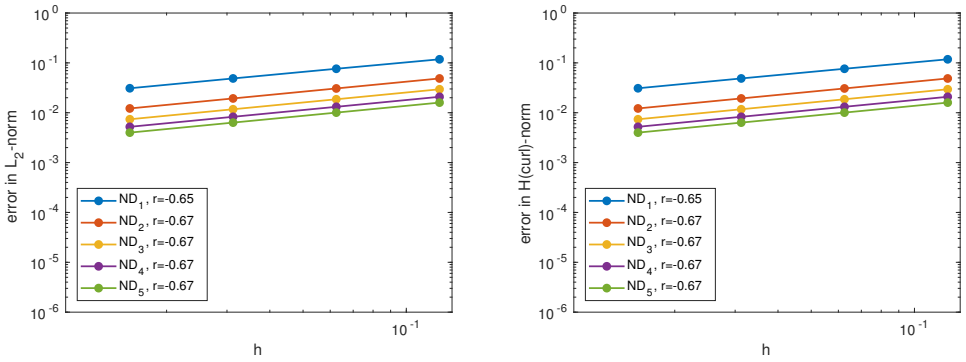
directly the cell contribution to the true error in step 2). For comparison purposes, we will also consider a uniform mesh refinement. Then, log-log convergence plots for the (L_2 or $H(\text{curl})$-) error against the number of free DOFs involved in the simulation is presented for analytical solutions that contain a singularity in a re-entrant corner.

Let us consider a L-shaped domain, $\Omega = [-1,1]^2 \setminus ([0,1] \times [-1,0])$ with Dirichlet boundary conditions imposed over the entire domain boundary. The source term $\boldsymbol{f}$ is such that the solution in polar coordinates (r,θ) is

$$\boldsymbol{u} = \boldsymbol{\nabla}\left(r^{\frac{2n}{3}}\sin\left(\frac{2n}{3}\theta\right)\right). \tag{2.38}$$

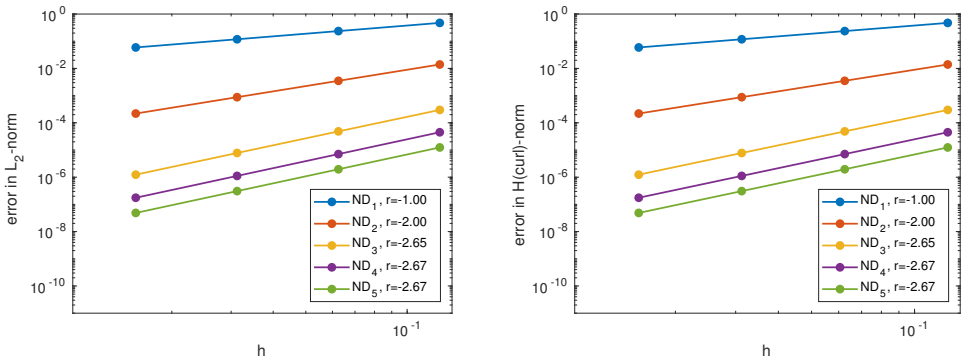
The chosen analytical function $\boldsymbol{u}$ has a singular behaviour at the origin of coordinates for $n = 1$, which prevents the function to be in $H^1(\Omega)$. Larger values of n lead to smoother solutions. The regularity of the solution is well studied [108], and theoretical convergence rates Eq. (2.36) are bounded by $\min(2n/3, k)$.

In Figs. 2.19 and 2.20, the theoretical convergence rates are achieved for every order of converge. In the case $n = 1$, all FE orders lead to the same convergence rate since $k > 2/3$. For a smoother solution, corresponding to $n = 4$, solutions converge to the expected order $\min(k, 8/3)$. Next, we analyze the error with adaptive refinement. The refinement process is such that, at every iterate, the 5% of cells with higher local contribution to the L_2-error are marked for refinement. The fraction of cells to be refined is intentionally



(A) L_2-norm of the error. (B) $H(\text{curl})$-norm of the error.

FIGURE 2.19: Error norms for the fichera 2D problem with uniform refinement for $n = 1$.



(A) L_2-norm of the error. (B) $H(\text{curl})$-norm of the error.

FIGURE 2.20: Error norms for the fichera 2D problem with uniform refinement for $n = 4$.

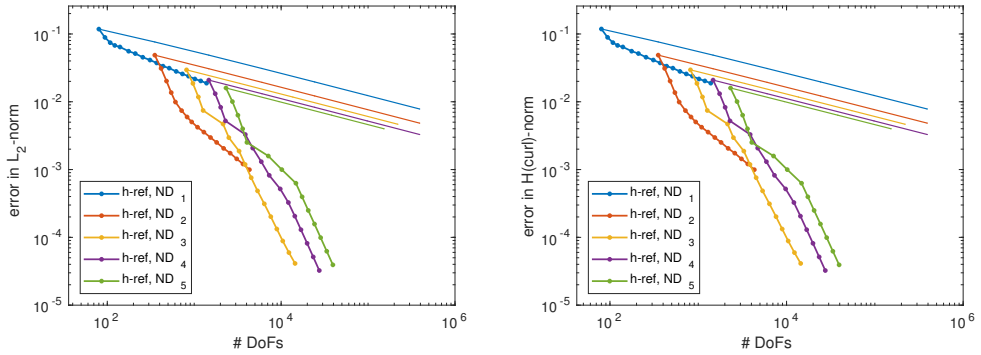
chosen to be small since we aim to obtain a localized refinement around the singularity. Figs. 2.18a and 2.18c show the initial mesh and the final mesh after 18 refinement steps, resp. In Fig. 2.21, we plot the (L_2- or $H(\text{curl})-$) error against the number of DOFs. We show two plots for each order, namely uniform refinement (solid line) and adaptive refinement (solid line with circles). In all cases, better efficiency is achieved by adaptive meshes, i.e., less error for a given number of DOFs.

Let us now consider the Fichera domain $\Omega = [-1, 1]^3 \setminus [-1, 0]^3$. The source term $\boldsymbol{f}$ is such that the solution is

$$\boldsymbol{u} = \boldsymbol{\nabla}\left(r^{\frac{2}{3}} \sin\left(\frac{2t}{3}\right)\right) \qquad t = \arccos\left(\frac{xyz}{r}\right), \tag{2.39}$$

where r is the radius in 3D polar coordinates. The analytical solution has a singular behaviour near the origin and again $\boldsymbol{u} \notin H^1(\Omega)$. We follow the same analysis to determine the efficiency of the h-adaptive scheme.

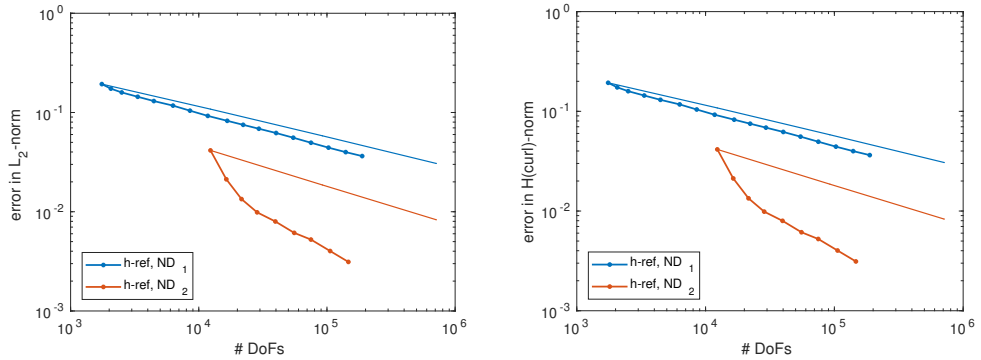
Fig. 2.22 shows the error vs. the number of DOFs of the h-adaptive refinement process. In Fig. 2.23, we illustrate the refinement process that takes place from an initial



(A) L_2-norm of the error with number of DOFs.

(B) $H(\text{curl})$-norm of the error with number of DOFs.

FIGURE 2.21: Error norms for the fichera 2D with $n = 1$ and adaptive refinement, which at every iterate marks for refinement the 5% of cells that show the highest local cell L_2-norm of the error. Lines without markers show the error convergence with uniform refinement process for every FE order.

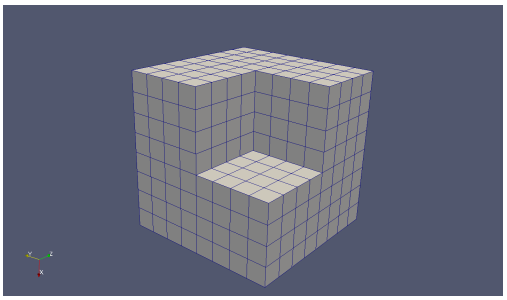


(A) L_2-norm of the error with number of DOFs.

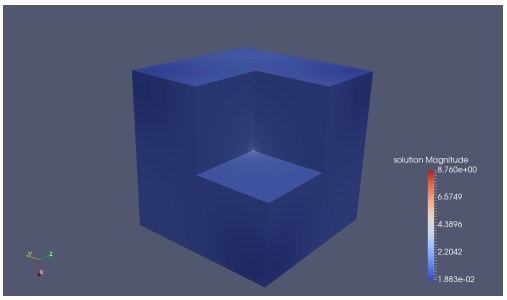
(B) $H(\text{curl})$-norm of the error with number of DOFs.

FIGURE 2.22: Error norms for the Fichera 3D problem with uniform (straight line) and adaptive refinement. 5% of cells that have the highest local L_2-norm of the error are marked for refinement at every refinement step.

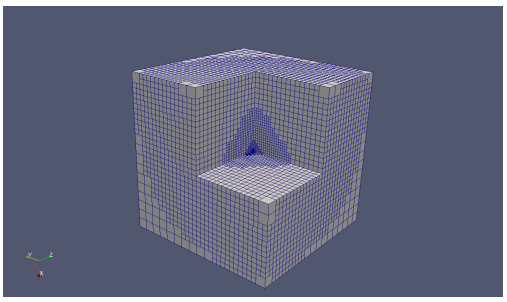
structured mesh composed of 8^3 elements (see Fig. 2.23a). Clearly, the closer a cell K is to the corner (see Fig. 2.23b), the higher the final $\ell(K)$. However, Fig. 2.23b also shows that the refinement is not as localized as in the 2D case, with a greater portion of the domain with higher levels of refinement. This fact has a clear impact on the efficiency achieved by the adaptive refinement for first order edge FEs, where the efficiency gain is mild. On the other hand, the gain for second order FEs is noticeably higher. Finally, Figs. 2.23c and 2.23d show the refined mesh after 12 refinement iterates and a zoom at the corner with the singularity.



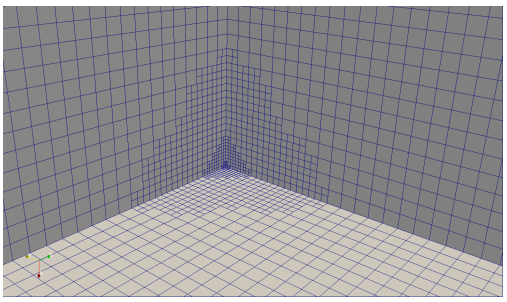
(A) Initial mesh for the Fichera cube domain.



(B) Analytical solution for the 3D Fichera cube.



(C) Mesh after 12 iterative refinement steps for the Fichera cube domain.



(D) Corner zoom for the 3D Fichera cube.

FIGURE 2.23: Adaptive meshes for the Fichera 3D problem. Cells with highest 5% Local L_2-error(K) are refined at each refinement step.

2.7 Conclusions

In this chapter, we have covered in detail a general implementation of p-adaptive and h-adaptive tetrahedral and hexahedral edge FE methods. We have implemented pre-bases that span the local FE spaces (anisotropic polynomials) which combined with a change of basis automatically provide the shape functions bases. It leads to a general arbitrary order implementation, confronted with hard-coded implementations that preclude high order methods. In order to guarantee the tangent continuity of Piola-mapped elements, special care must be taken with the orientation of the cell geometrical entities. In the implementation, we require the local numbering of nodes within every element to rely on sorted global indices, i.e., oriented meshes. This manner, we automatically satisfy consistency in every geometrical entity shared by two or more FEs. Finally, we propose an original approach to implement global curl-conforming FE space on hierarchically refined octree-based *non-conforming* meshes. The strategy, which is straightforwardly extensible to any FE based on polynomial spaces, is based on the original Lagrangian constraints and the interplay between the sets of Lagrangian and edge basis functions. To obtain every constraint, a sequence of spaces can be built so as we can avoid the evaluation of moments in the edge FE space. A detailed set of numerical experiments served to test the implementation, where we show agreement between theoretical and numerical rates of convergence. The proposed approach has been implemented (for first kind edge $H(\text{curl})$-conforming FE) in `FEMPAR`.

We note that this framework can be extended to other polynomial-based FEs. Customizable ingredients are the original pre-basis of polynomials, the moments, the geometrical mapping, and the equivalence class of DOFs. However, the change of basis approach to obtain the corresponding shape functions and the enforcement of continuity for non-conforming meshes is identical. In fact, the same machinery has already been used in `FEMPAR` to implement Raviart-Thomas FEs and can straightforwardly be used to implement Brezzi-Douglas-Marini FEs [44], second kind edge FEs [105], or recent divergence-free FEs [106].

We believe that the comprehensive description of all the implementation issues behind edge FE method provided herein will be of high value for other researchers and developers that have to purport similar developments and increase their penetration in the computational mechanics community.

Chapter 3

Balancing Domain Decomposition by Constraints solvers for heterogeneous problems in $H(\text{curl})$

In this chapter, we present scalable balancing domain decomposition by constraints (BDDC) methods for linear systems arising from arbitrary order edge finite element (FE) discretizations of multi-material and heterogeneous 3D problems. These methods rely on the definition of a FE space with relaxed continuity, which is defined by choosing only some quantities of the solution (e.g., averages, moments) to be continuous across the interface between subdomains rather than the solution itself. In order to enforce this continuity, we use a partition of the interface objects (edges and faces) into sub-objects determined by the variation of the physical coefficients of the problem. For multi-material problems, a constant coefficient condition is enough to define this sub-partition of the objects. For arbitrarily heterogeneous problems, a relaxed version of the method is defined, where we only require that the maximal contrast of the physical coefficient in each object is smaller than a predefined threshold. Besides, the addition of perturbation terms to the preconditioner is empirically shown to be effective in order to deal with the case where the two coefficients of the model problem jump simultaneously across the interface. The new method, in contrast to existing approaches for problems in curl-conforming spaces, preserves the simplicity of the original preconditioner whilst providing robustness with regard to coefficient jumps and heterogeneous materials. A detailed set of numerical experiments, which includes the application of the preconditioner to 3D realistic cases, shows excellent weak scalability properties of the implementation of the proposed algorithms.

3.1 Introduction

Realistic simulations in electromagnetic problems often involve multiple materials (e.g., dielectric and conducting materials), which may imply high contrasts in the coefficients describing the physical properties of the different materials. Besides, the behaviour of conducting materials may be modelled by highly variable, heterogeneous coefficients.

This problem definition inevitably leads to high condition numbers for the resulting linear systems arising from curl-conforming FE discretizations of the corresponding partial differential equation (PDE), which pose great challenge for solvers. Furthermore, the design of solvers for $H(\text{curl})$-conforming approximations poses additional difficulties, since the kernel of the curl operator is non-trivial. Consequently, for realistic electromagnetic simulations in 3D, the use of robust iterative solvers is imperative in terms of complexity and scalability. In this chapter, we will focus on the development of robust BDDC preconditioners for problems posed in $H(\text{curl})$ involving high variation of the coefficients for the corresponding PDE.

BDDC preconditioners [57] belong to the family of non-overlapping domain decomposition (DD) methods [138]. They can be understood as an evolution of the earlier Balancing DD method [95]. These methods rely on the definition of a FE space with relaxed inter-element continuity, which is defined by choosing some quantities to be continuous across subdomain interfaces, i.e., the *coarse* or *primal* degrees of freedom (DOFs). Then, the continuity of the solution at the interface between subdomains is restored with an averaging operator. The method has two properties that make it an outstanding candidate for extreme scale computing, namely it allows for aggressive coarsening and computations among the different levels can be performed in parallel. Outstanding scalability results have been achieved by an implementation in the scientific computing software `FEMPAR` (see Chapter 1 for an introduction), that exploits these two properties in up to almost half a million cores and two million subdomains (MPI tasks) [23]. Another work showing excellent scalability properties up to two hundred thousand cores is [146], which is implemented in the software project PETSc [28].

The main purpose of this chapter is to construct BDDC methods for the linear systems arising from arbitrary order edge (Nédélec) FE discretizations of heterogeneous electromagnetic problems. An analysis for 3D FETI-DP[1] algorithms with the lowest order Nédélec elements of the first kind was given by Toselli in [137], who argued that the difficulty of iterative substructuring methods for edge element approximations mainly lies in the strong coupling between the energy of subdomain faces and edges. In short, no efficient and robust iterative substructuring strategy is possible with the standard basis of shape functions for the edge FE (see Chapter 2). A suitable change of basis was introduced in [137] for lowest order edge elements and box-subdomains. Besides, an extension to arbitrary order edge FEs and subdomain geometrical shapes is presented in [149]. In this work, we will offer some new insights in the definition and construction of the change of basis for the latter general case. As pointed out in [58], the change of variables can be implemented in practice with just a few simple modifications to the standard BDDC algorithm [57].

Modern BDDC methods [120] propose coarse space enrichment techniques that adapt

[1] FETI-DP algorithms [67] are closely related to BDDC methods. In fact, it can be shown that the eigenvalues of the preconditioned operators associated with BDDC and FETI-DP are almost identical [97, 91, 43].

to the variation of coefficients of the problem [99, 131, 50, 133, 134, 88], where coarse DOFs are adaptively selected by solving generalized eigenvalue problems. This approach is backed up by rigorous mathematical theory and has been numerically shown to be robust for general heterogeneous problems. On the other hand, several different scalings have been proposed for the averaging operator in the literature to improve the lack of robustness of the *cardinality* (i.e., arithmetic mean) scaling for coefficient jumps. The *stiffness*[2] scaling takes more information into account but can lead to poor preconditioner performance with mildly varying coefficients [121]. The most robust approach up-to-date is the *deluxe* scaling, first introduced in [58] for 3D problems in curl-conforming spaces. It is based on the solution of local auxiliary Dirichlet problems to compute efficient averaging operators [59, 120, 110, 146, 147, 148], involving dense matrices per subdomain vertex/edge/face. However, to solve eigenvalue and auxiliary problems is expensive and extra implementation effort is required as coarse spaces in DD methods are not naturally formulated as eigenfunctions.

The main motivation of this chapter is to construct robust BDDC preconditioners for problems in curl-conforming spaces that keep the simplicity of the standard BDDC method, i.e., to avoid the spectral solvers of adaptive versions, whereas keeping robustness and low computational cost. In order to do so, we follow the idea of the physics-based BDDC (PB-BDDC) preconditioner, presented in [16] for problems in grad-conforming spaces. Based on the fact that BDDC methods (and DD methods in general) are robust with regard to jumps in the material coefficients when these jumps are aligned with the partition [138, 89], one can use a physics-based (PB)-partition obtained by aggregating elements of the same (or similar) coefficient value. However, using this type of partition can lead to a poor load balancing among subdomains and large interfaces. To overcome this situation, the PB-BDDC respects the original partition (well-balanced) but considers a sub-partition of every subdomain based on the physical coefficients, leading to a partition of the objects into sub-objects defined according to the variation of the coefficients. Consequently, the method is also based on an enrichment of the coarse space but with the great advantage of not requiring to solve eigenvalue or auxiliary problems, i.e., the simplicity of the original BDDC preconditioner is maintained. The PB-BDDC preconditioner turned out to be one order of magnitude faster than the BDDC method with deluxe scaling in [146] for linear elasticity and thermal conductivity problems with high contrast.

Our problem formulation arises from the time-domain quasi-static approximation to the Maxwell's Equations for the magnetic field (see Chapter 4), which involves two different operators, the mass and double curl terms. This fact certainly poses more complexities than the ones faced in [16] for the PB-BDDC solver, since it has to deal with the interplay of both (simultaneous) coefficient jumps. Our solution is to propose a simple technique to recover the scenario where only one coefficient has a jump across interfaces: we will add a perturbation at the preconditioner level so that the perturbed

[2] weighted averages with the diagonal entries of the operator for every DOF

formulation does not involve a jump for the mass-matrix terms across interfaces. The effectiveness of the technique will be empirically shown. In order to extend the PB-BDDC algorithm to heterogeneous materials, a relaxed definition of the PB-partition will be stated where we only require that the maximal contrast of the physical coefficient in each PB-subdomain is smaller than a predefined threshold. The threshold can be chosen so that the condition number is reasonably small while the size of the coarse problem is not *too large*.

The chapter outline is as follows. The problem is defined in Sect. 3.2, where basic definitions are introduced. Sect. 3.3 is devoted to the presentation of the PB-BDDC preconditioner for heterogeneous 3D problems in $H(\mathrm{curl})$. In Sect. 3.4 we will give some implementation insights, based on our experience through the implementation of the algorithms in the scientific software project `FEMPAR`. In Sect. 3.5, we present a detailed set of numerical experiments, covering a wide range of cases and applications for the PB-BDDC preconditioner. Finally, some conclusions are drawn in Sect. 3.6.

3.2 Problem setting

Let us consider the boundary value Maxwell problem on a physical domain $\Omega \subset \mathbb{R}^3$:

$$\boldsymbol{\nabla} \times (\alpha \boldsymbol{\nabla} \times \boldsymbol{u}) + \beta \boldsymbol{u} = \boldsymbol{f} \quad \text{in}\, \Omega, \tag{3.1}$$

$$\boldsymbol{n} \times (\boldsymbol{u} \times \boldsymbol{n}) = 0 \quad \text{on}\, \partial\Omega, \tag{3.2}$$

where $\alpha \geq 0$, $\beta > 0$ are the resistivity and the magnetic permeability of the materials, respectively, $\boldsymbol{n}$ is an unit normal to the boundary and $\boldsymbol{\nabla}\times$ is the 3D curl operator, see [101]. For the sake of simplicity, we consider homogeneous Dirichlet conditions, i.e., zero tangential traces on $\partial\Omega$. Nevertheless, all the developments in this work can readily be applied to Neumann and/or inhomogeneous conditions, see Chapter 2 for proper definitions. In order to pose the weak form of the problem, let us define the functional space $H(\boldsymbol{\nabla}\times;\Omega)$ as follows:

$$H(\boldsymbol{\nabla}\times;\Omega) \doteq \{\boldsymbol{v} \in L^2(\Omega)^3 \,:\, \boldsymbol{\nabla} \times \boldsymbol{v} \in L^2(\Omega)^3\}, \tag{3.3}$$

and its subspace that satisfies homogeneous Dirichlet boundary conditions,

$$H_0(\boldsymbol{\nabla}\times;\Omega) \doteq \{\boldsymbol{v} \in H(\boldsymbol{\nabla}\times;\Omega) \,:\, \boldsymbol{n} \times (\boldsymbol{u} \times \boldsymbol{n}) = 0 \,\text{on}\, \partial\Omega\}. \tag{3.4}$$

Besides, we will also make use of the space

$$H^1(\Omega) \doteq \{v \in L^2(\Omega) \,:\, \boldsymbol{\nabla} v \in L^2(\Omega)^3\}. \tag{3.5}$$

Functions in $H(\boldsymbol{\nabla}\times;\Omega)$ are approximated by edge FE methods of arbitrary order, which we represent by $X_h \subset H(\boldsymbol{\nabla}\times;\Omega)$. In addition, functions in $H^1(\Omega)$ are approximated

by standard scalar, continuous Lagrangian FE methods, which we represent by $V_h \subset H^1(\Omega)$. The weak form of the boundary value Maxwell problem in Eq. (3.1) reads: find $\boldsymbol{u} \in H_0(\boldsymbol{\nabla}\times;\Omega)$ such that

$$\mathcal{A}_h(\boldsymbol{u},\boldsymbol{v}) = (\boldsymbol{f},\boldsymbol{v}), \qquad \forall \boldsymbol{v} \in H_0(\boldsymbol{\nabla}\times;\Omega), \tag{3.6}$$

where

$$\mathcal{A}_h(\boldsymbol{u},\boldsymbol{v}) = \int_\Omega \left[(\alpha\boldsymbol{\nabla}\times\boldsymbol{u})\cdot(\boldsymbol{\nabla}\times\boldsymbol{v}) + \beta\boldsymbol{u}\cdot\boldsymbol{v}\right] dx, \qquad (\boldsymbol{f},\boldsymbol{v}) = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}dx. \tag{3.7}$$

3.2.1 Domain partition

Let us consider a bounded polyhedral domain $\Omega \subset \mathbb{R}^3$. Let $\mathcal{T}_h$ be a partition of Ω into a set of tetrahedral or hexahedral cells K. For every cell $K \in \mathcal{T}_h$ consider its set of vertices $\mathcal{V}_K$, edges $\mathcal{E}_K$, or faces $\mathcal{F}_K$. They constitute the set of geometrical entities of the cell (excluding itself) as $\mathcal{G}_K = \mathcal{V}_K \cup \mathcal{E}_K \cup \mathcal{F}_K$. The union of these sets for all cells is represented with $\mathcal{G} \doteq \cup_{K\in\mathcal{T}_h}\mathcal{G}_K$. We consider a partition Θ of the domain Ω into non-overlapping subdomains $\tilde{\Omega}_i$, $i = 1,\dots,\tilde{N}$ obtained by aggregation of elements $K \in \mathcal{T}_h$. These subdomains are assumed to be such that the computational cost of solving the discrete Maxwell problem in the different subdomains leads to a well-balanced distribution of computational loads among processors in memory distributed platforms. We denote by $\Gamma(\Theta)$ the interface of the partition Θ, i.e., $\Gamma(\Theta) = \cup_{\tilde{\Omega}_i\subset\Omega}\, \partial\tilde{\Omega}_i \setminus \partial\Omega$. Every subdomain $\tilde{\Omega}_i \subset \Omega$ can be also partitioned into the smallest set of subdomains Ω_{ij}, $j = 1,\dots,N_i$, such that the material properties (α,β) in Eq. (3.1) are constant at every Ω_{ij}. For obvious reasons, we call this sub-partition a PB-partition and will be denoted by Θ_{pb}. Clearly, the resulting global Θ_{pb} is also a partition of Ω, and there is a unique $\mathcal{D} \in \Theta$ for every $\mathcal{D}' \in \Theta_{\text{pb}}$ such that $\mathcal{D}' \subset \mathcal{D}$. We consider a global numbering for the PB-subdomains, i.e., Ω_k, $k = 1,\dots,N$, having a one-to-one mapping between the two indices labels. Analogously, we define the interface of the PB-partition as $\Gamma(\Theta_{\text{pb}}) = \cup_{\Omega_k\subset\Omega}\, \partial\Omega_k \setminus \partial\Omega$.

3.2.2 Finite Element spaces

Let us define the FE spaces $X_h^i \doteq X_h(\tilde{\Omega}_i) \subset H(\boldsymbol{\nabla}\times;\tilde{\Omega}_i)$ for every subdomain $\mathcal{D} \in \Theta$, and the corresponding Cartesian product space $\widehat{X}_h = \Pi_{i=1}^{\tilde{N}} X_h^i$. Note that functions belonging to this space are allowed to have discontinuous tangent traces across the interface $\Gamma(\Theta)$. The global space in which the global problem is sought, i.e., X_h, can be understood as the subspace of functions in $\widehat{X}_h$ that have continuous tangent traces across $\Gamma(\Theta)$. We can now define the subdomain FE operator $\mathcal{A}_h^i : X_h^i \to X_h^{i\prime}$, $i = 1,\dots,\tilde{N}$, as $\mathcal{A}_h^i(\boldsymbol{u}_i,\boldsymbol{v}_i) = \int_{\tilde{\Omega}_i}\left[(\alpha\boldsymbol{\nabla}\times\boldsymbol{u}_i)\cdot(\boldsymbol{\nabla}\times\boldsymbol{v}_i) + \beta\boldsymbol{u}_i\cdot\boldsymbol{v}_i\right] dx$ for all $\boldsymbol{u}_i,\boldsymbol{v}_i \in X_h^i$. Then, the sub-assembled operator $\widehat{\mathcal{A}}_h : \widehat{X}_h \to \widehat{X}_h'$ is defined as $\widehat{\mathcal{A}}_h(\boldsymbol{u},\boldsymbol{v}) = \prod_i^{\tilde{N}} \mathcal{A}_h^i(\boldsymbol{u}_i,\boldsymbol{v}_i)$, in which contributions between subdomains have not been assembled. The assembled operator $\mathcal{A}_h : X_h \to X_h{}'$ (see Eq. (3.7)) is the Galerkin projection of the operator $\widehat{\mathcal{A}}_h$ onto X_h.

The space of edge FE functions can be represented as the range of an interpolation operator π^h, which is well-defined for sufficiently smooth functions $\boldsymbol{u} \in H(\boldsymbol{\nabla}\times;\Omega)$, by

$$\pi^h(\boldsymbol{u}) := \sum_a \sigma^a(\boldsymbol{u})\boldsymbol{\varphi}^a \tag{3.8}$$

where $\sigma^a(\boldsymbol{u})$ are the evaluation of the moments, i.e., the DOF values, and $\boldsymbol{\varphi}^a$ are the elements of the unique basis of functions that satisfies $\sigma_a(\boldsymbol{\varphi}^b) = \delta_{ab}$, i.e., the shape functions. The reader is referred to Chapter 2 for a comprehensive definition of edge moments and the construction of polynomial spaces and basis of shape functions for the tetrahedral/hexahedral edge FE of arbitrary order.

3.2.3 Objects

In this section we introduce the definitions of global objects, or simply *globs*, which are heavily used in DD preconditioners (see, e.g., [138]). Given a geometrical entity $s \subset \Gamma(\Theta)$ and a subdomain partition Θ, we denote by $\text{neigh}_\Theta(s)$ the set of subdomains in Θ that contain s. Then, we define a geometrical object as the maximal set λ of geometrical entities in $\Gamma(\Theta)$ with the same $\text{neigh}_\Theta(s)$ subdomain set. We denote by $\text{neigh}_\Theta(\lambda)$ the set of subdomains in Θ containing λ and by $\text{ndof}(\lambda)$ the total number of DOFs placed on top of all $s \in \lambda$. An object λ such that $\text{ndof}(\lambda)>0$ is a face F if $|\text{neigh}_\Theta(\lambda)| = 2$ or an edge E if $|\text{neigh}_\Theta(\lambda)| > 2$. In addition, an object such that $\text{ndof}(\lambda)=0$ is a corner. Grouping together the objects of the same type, we obtain the set of corners Λ_C, edges Λ_E and the set of faces Λ_F. Therefore, the set of *globs* is defined as $\Lambda(\Theta) = \Lambda_C \cup \Lambda_E \cup \Lambda_F$.

Remark 2. *This definition differs from the standard one (see, e.g., [21]). It is intentionally done in order to isolate* globs *that do not contain DOFs, i.e., Λ_C, which can be omitted in the rest of our exposition.*

Once *globs* are defined, let us also introduce the set of PB-*globs*, denoted by $\Lambda_{\text{pb}}(\Theta)$, as classification of all $s \subset \Gamma(\Theta)$ into Λ_C, Λ_E or Λ_F by considering the previous definitions based on $\text{neigh}_{\Theta_{\text{pb}}}(s)$ rather than $\text{neigh}_\Theta(s)$. $\Lambda_{\text{pb}}(\Theta)$ is a sub-partition of $\Lambda(\Theta)$ where coefficients are constant within each $\lambda \in \Lambda_{\text{pb}}(\Theta)$.

3.3 Physics-Based BDDC

3.3.1 Change of basis

Any BDDC method that employs a standard 3D edge FE basis of shape functions is bound to show a factor dependent on the element size h^{-2} in the condition number [137], which precludes scalability. A key aspect of the curl-conforming edge FE spaces is the fact that $\boldsymbol{\nabla}V_h^i \subset X_h^i$. One of the main ingredients of any BDDC method are the averaging operators $\mathcal{W}_h : \widehat{X}_h \to X_h$ (see detailed exposition in Sect. 3.3.2) that restore the continuity of the solution at the interface among subdomains. Since the averaging

operators are usually based on some algebraic operations over DOF values, they are, more precisely, scaling matrices that depend on the basis being used to describe $\widehat{X}_h$ (and X_h^i, by restriction to every subdomain). A key property that must hold such operator to end up with a stable decomposition is the following: Given a function $\boldsymbol{u} \in \widehat{X}_h$ such that its local component in every processor belongs to $\boldsymbol{\nabla} V_h^i$, the restriction of the resulting function $\mathcal{W}_h \boldsymbol{u} \in X_h$ to every subdomain must belong to $\boldsymbol{\nabla} V_h^i$ too. Otherwise, the energy of such functions is much increased after the averaging operation, and thus, the decomposition is not *scalable*. A key result in this direction is the decomposition proposed in [137] in the frame of FETI-DP methods for problems in $H(\boldsymbol{\nabla}\times;\Omega)$.

Edge FE space moments can be assigned to edges/faces of the mesh (see Chapter 2). Let us denote by $X_h^i(I)$ the subspace of functions of X_h^i such that their DOF values are not *located* on some $E \in \Lambda_E$ or $F \in \Lambda_F$, i.e., the DOFs are interior. Clearly, $X_h^i = \{X_h^i(I)\} \oplus \{X_h^i(F)\}_{F\subset\partial\tilde{\Omega}_i} \oplus \{X_h^i(E)\}_{E\subset\partial\tilde{\Omega}_i}$. On the other hand, a function $\boldsymbol{v}^i \in X_h^i(E)$ for a coarse edge $E \subset \partial\tilde{\Omega}_i$ admits a unique decomposition as follows (see [137, 59] for more details):

$$\boldsymbol{v}^i \cdot \boldsymbol{t}_E = s_{0,E}(\boldsymbol{v}^i)\Phi_E \cdot \boldsymbol{t}_E + \sum_{j=1}^{n_E-1} w_{jE}(\boldsymbol{v}^i)\boldsymbol{\nabla}\phi_E^j \cdot \boldsymbol{t}_E, \qquad \forall E \in \Lambda_E, \tag{3.9}$$

with $\phi_E^j \in V_h^i(E)$ being the Lagrangian shape functions related to the internal nodes of E and n_E their cardinality, whereas $s_{0,E}(\boldsymbol{v}^i) \doteq \int_E \boldsymbol{v}^i \cdot \boldsymbol{t}_E ds$. It is clear from Eq. (3.9) that two kind of DOFs arise in the new basis for each subdomain edge $E \in \Lambda_E$: a DOF associated with the basis function Φ_E, which represents the average tangent value over the coarse edge E, and DOFs associated with gradients of scalar, Lagrangian shape functions placed at the internal nodes (i.e., nodes $\xi \in E$ such that $\xi \notin \partial E$). An illustration for the variables in the old (original) and new basis for a given E is presented in Fig. 3.1. Thus, we have that $X_h^i(E) \doteq \boldsymbol{\nabla} V_h^i(E) \oplus \Phi_E$. As a result, X_h^i admits the unique decomposition:

$$X_h^i = \{X_h^i(I)\} \oplus \{X_h^i(F)\}_{F\subset\partial\tilde{\Omega}_i} \oplus \{\boldsymbol{\nabla} V_h^i(E)\}_{E\subset\partial\tilde{\Omega}_i} \oplus \{\Phi_E\}_{E\subset\partial\tilde{\Omega}_i}, \tag{3.10}$$

where Φ_E is the tangential vector such that $\Phi_E \cdot \boldsymbol{t}_E = 1$, being $\boldsymbol{t}_E$ the unit tangent to $E \in \Lambda_E$.

Let us now describe the relation between the original set of DOFs (old basis in the global space) and the one that arises from Eq. (3.10) (new basis) for X_h. A function $\boldsymbol{u} \in X_h$ can be written in the old basis as $\boldsymbol{u} = \sum_a u^a \boldsymbol{\varphi}^a$, where $\boldsymbol{\varphi} = \{\boldsymbol{\varphi}^1, \ldots, \boldsymbol{\varphi}^n\}$ is the set of global edge shape functions. Furthermore, consider the set of new basis functions $\boldsymbol{\psi} = \{\boldsymbol{\psi}^1, \ldots, \boldsymbol{\psi}^n\}$, where old basis elements $\boldsymbol{\varphi}^a$ associated to $E \in \Lambda_E$ are replaced by its corresponding functions in Eq. (3.9) (i.e., interior and face edge functions, Lagrangian shape functions gradients, and the coarse edge functions). The interpolation operator π^h (see Eq. (3.8)) induces the change of basis matrix, whose entries are computed by evaluating the original edge moments σ^a for the introduced set of new basis functions $\boldsymbol{\psi}$

as (Einstein notation)

$$u^a_{\text{old}} = \sigma^a(\boldsymbol{u}_{\text{new}}) = \sigma^a(\boldsymbol{\psi}^b)u^b_{\text{new}} = \mathcal{Q}_{ab}u^b_{\text{new}}, \tag{3.11}$$

or in compact form, $\boldsymbol{u}_{\text{old}} = \mathcal{Q}\boldsymbol{u}_{\text{new}}$. Furthermore, we can readily define the inverse change of basis as $\boldsymbol{u}_{\text{new}} = \mathcal{Q}^{-1}\boldsymbol{u}_{\text{old}}$. The usual restriction operator $R_i : X_h \to X_h^i$ is used to obtain local restrictions of the global change of basis as $\mathcal{Q}_i = R_i\mathcal{Q}R_i^T$. Finally, local restrictions lead to the change of basis $\widehat{\mathcal{Q}} = \prod_i \mathcal{Q}_i$, which will be applied for functions defined on $\widehat{X}_h$. A detailed exposition of an implementation strategy for the change of basis is found in Sect. 3.4.2.

3.3.2 Preconditioner

Similarly to other BDDC methods, we associate coarse DOFs to some of the *globs* in $\Lambda_{\text{pb}}(\Theta)$. In particular, BDDC methods for 3D curl-conforming spaces associate two coarse DOFs to every $E \in \Lambda_E$, defined as

$$s_{0,E}(\boldsymbol{v}^i) \doteq \int_E \boldsymbol{v}^i \cdot \boldsymbol{t}_E ds \tag{3.12a}$$

$$s_{1,E}(\boldsymbol{v}^i) \doteq \int_E s\boldsymbol{v}^i \cdot \boldsymbol{t}_E ds, \tag{3.12b}$$

where s is an arc-length parameter $s \in [-|E|/2, |E|/2]$. Thus, the expression Eq. (3.12b) refers to the first order moment of the tangent component of the solution on the edge E, in contrast to the zero-order moment in Eq. (3.12a). In the new basis (see Eq. (3.9)), it is easy to check that $s_{0,E}(\boldsymbol{v}^i) = s_{0,E}(\Phi_E)$ and $s_{1,E}(\boldsymbol{v}^i) = s_{1,E}(\sum_{j=1}^{n_e-1} w_{jE}\boldsymbol{\nabla}\phi^j_E)$ [137]. Let us define the subspace $\widetilde{X}_h$ as

$$\widetilde{X}_h \doteq \{\boldsymbol{w} \in \widehat{X}_h : s^{\mathcal{D}}_{0,E} = s^{\mathcal{D}'}_{0,E},\ s^{\mathcal{D}}_{1,E} = s^{\mathcal{D}'}_{1,E}\ \forall E \in \Lambda_E, \forall \mathcal{D}, \mathcal{D}' \in \text{neigh}_{\Theta_{\text{pb}}}(E)\} \tag{3.13}$$

i.e., the subspace $\widetilde{X}_h \subset \widehat{X}_h$ such that for all $\boldsymbol{w} \in \widehat{X}_h$, coarse DOFs (3.12a) and (3.12b) are continuous across subdomain interfaces $\Gamma(\Theta)$ for all $E \in \Lambda_E$. Clearly, $X_h \subset \widetilde{X}_h \subset \widehat{X}_h$.

The following key ingredient in the BDDC method is the averaging operator $\mathcal{W}_h : \widehat{X}_h \to X_h$, defined as some weighted average of the DOF values at the interface. This operator is in practice defined as a matrix for a particular choice of the basis functions for X_h^i (and by extension X_h). Let us consider the new basis functions in $\boldsymbol{\psi}$. Given a fine edge/face $f \subset \Gamma(\Theta)$, we define a weight for each $\mathcal{D} \in \text{neigh}_\Theta(f)$ as

$$\delta^\dagger_{\mathcal{D}}(f) = \frac{\sum_{\mathcal{D}' \in \text{neigh}_{\Theta_{\text{pb}}}(f) \cap \mathcal{D}} \chi_{\mathcal{D}'}}{\sum_{\mathcal{D}' \in \text{neigh}_{\Theta_{\text{pb}}}(f)} \chi_{\mathcal{D}'}}, \tag{3.14}$$

where the choice of χ defines the scaling: the *cardinality* scaling with $\chi = 1.0$ or the α-based, β-based scaling with $\chi = \alpha$ or $\chi = \beta$, respectively. Besides, one can consider a

weighted coefficient for χ, $\omega = \alpha + \beta h^2$, which is also constant within all $E \in \Lambda_E$ in our definitions if regular structured meshes are considered. We note that all the expressions for the scalings are constant on *globs* by construction, due to objects generation based on Θ_{pb} with constant coefficients. Then, we define the weighted function $\mathcal{W}_h \boldsymbol{v} \in X_h$ as follows. First, we compute for every subdomain the weighted local functions as

$$\boldsymbol{w}^i = \boldsymbol{v}_I^i + \sum_{F \subset \Lambda_F} \delta_{\mathcal{D}}^{\dagger}(F) \boldsymbol{v}_F^i + \sum_{E \subset \Lambda_E} \delta_{\mathcal{D}}^{\dagger}(E) \boldsymbol{v}_E^i, \tag{3.15}$$

where $\boldsymbol{v}^I$, $\boldsymbol{v}_F^i$, and $\boldsymbol{v}_E^i$ include the components related to interior, face, and edge DOFs in 3.10, respectively. Next, we sum the values of DOFs on different subdomains that represent the same DOFs in $\widehat{X}_h$, i.e., assemble the DOFs as

$$\boldsymbol{v} = \sum_i R_i^T \boldsymbol{w}^i. \tag{3.16}$$

Next, we recover the sub-assembled bilinear form $\widehat{\mathcal{A}}_h$, whereas $\mathcal{A}_h$ and $\widetilde{\mathcal{A}}_h$ are the Galerkin projection of $\widehat{\mathcal{A}}_h$ onto X_h and $\widetilde{X}_h$, respectively. We additionally define the harmonic extension operator $\mathcal{E}$, that, given $\boldsymbol{u} \in X_h$, provides $\boldsymbol{u} + \delta \boldsymbol{u}_I$, where $\delta \boldsymbol{u}_I \in X_h^i(I)$ is a bubble function that vanishes on the interface $\Gamma(\Theta)$ and holds:

$$\langle \mathcal{A}_h^i \delta \boldsymbol{u}_I^i, \boldsymbol{v}_I^i \rangle = -\langle \mathcal{A}_h^i \boldsymbol{u}^i, \boldsymbol{v}_I^i \rangle, \quad \forall \boldsymbol{v}_I^i \in X_h^i(I). \tag{3.17}$$

Let us denote the Galerkin projection of $\mathcal{A}_h$ onto the global bubble space $X_h(I) \doteq \{\boldsymbol{v} \in X_h \text{ such that } \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma(\Theta)\}$ by $\mathcal{A}_{h,0}$. Thus, the action of the harmonic extension operator can be written as

$$\mathcal{E}\boldsymbol{u} \doteq (1 - \mathcal{A}_{h,0}^{-1} \mathcal{A}_h) \boldsymbol{u}. \tag{3.18}$$

We finally define the operator $\mathcal{H} = \mathcal{E}\mathcal{W}_h$. We can now state the BDDC preconditioner as

$$\mathcal{P} \doteq \mathcal{A}_{h,0}^{-1} + \mathcal{H}(\widetilde{\mathcal{A}}_h)^{-1} \mathcal{H}^T. \tag{3.19}$$

Note that having the expression of the operators associated with the new basis is essential in order to apply the averaging operator $\mathcal{W}_h$. Nevertheless, it is possible to employ the original operators in the standard basis and work with the change of basis matrix $\mathcal{Q}$ [58]. In this case, the only difference with regard to Eq. (3.19) is the application of the averaging operator as

$$\mathcal{H} = \mathcal{E}\mathcal{Q}\mathcal{W}_h \widehat{\mathcal{Q}}^{-1} \qquad \text{or} \qquad \mathcal{H}^T = \widehat{\mathcal{Q}}^{-T} {\mathcal{W}_h}^T \mathcal{Q}^T \mathcal{E}^T. \tag{3.20}$$

Application details for the change of basis are detailed in Sect. 3.4.2. Therefore, the definition of the preconditioner is the one of the standard BDDC [57] with a set of

globs generated by a partition based on coefficients and a modification of the averaging operator to take into account this fact. Besides, one can work with the standard basis of edge FEs and use strategically the change of basis required to attain a scalable algorithm in the application of the weighting operator.

3.3.3 Perturbed PB-BDDC preconditioner

The presented PB-BDDC preconditioner has been shown to be robust with the jump of coefficients in the steady Poisson equation [16]. However, the problem in Eq. (3.1) adds the complexity of the interplay between the two different parameters α and β across the interface. Following the robust approach in [16], our idea is to get rid of the jump of one coefficient across the interface so the preconditioner has not to deal with the interplay between the two of them and the scenario where the method is successful is recovered. In order to decide which coefficient is affected, we consider the locality of the mass matrix operator in front of the double curl terms. The main idea is to add a perturbation in the original formulation of the preconditioner so we end up with common information for the mass matrix operator for DOFs that are replicated among different subdomains, i.e., located on top of the interface $\Gamma(\Theta)$. Therefore, the problem posed in $\widetilde{X}_h$ will only contain a jump in the double curl term across the interface.

Given a function $\boldsymbol{u}^i \in X_h^i$, we can define its extension as a global function $\bar{\boldsymbol{u}}^i \in X_h$ such that all DOFs belonging to $\tilde{\Omega}_i$ are identical to the ones of $\boldsymbol{u}^i$ and the rest are zero. The extended function has support on $\tilde{\Omega}_i$ and its neighbours, denoted by $\bar{\Omega}_i$. The perturbed preconditioner for a local subdomain $\tilde{\Omega}_i$ is expressed as:

$$\tilde{\mathcal{A}}_h^i(\boldsymbol{u}^i, \boldsymbol{v}^i) = \int_{\bar{\Omega}_i} (\alpha \boldsymbol{\nabla} \times \boldsymbol{u}^i) \cdot (\boldsymbol{\nabla} \times \boldsymbol{v}^i) dx + \int_{\bar{\Omega}_i} \beta \bar{\boldsymbol{u}}^i \cdot \bar{\boldsymbol{v}}^i dx. \tag{3.21}$$

Therefore, entries for interface DOFs in the local mass matrix will be *fully-assembled* instead of *partially* assembled, leading to common information at the interface across all subdomains. In the situation where no jump occurs for the mass matrix coefficients at the interface among subdomains, we consider the original preconditioner presented in Sect. 3.3.2, avoiding the perturbed formulation for obvious reasons.

Remark 3. *The definition of the original problem is not modified, we only consider the perturbed local operator $\tilde{\mathcal{A}}_h^i$ in the formulation for the preconditioner.*

3.3.4 Relaxed PB-BDDC

In previous sections, the definition of Θ_{pb} (and consequently the definition of PB-*globs*) is based on the requirement that coefficients are constant in each PB-subdomain, i.e., different subparts with constant coefficients can be identified in a subdomain, e.g. a problem composed by different homogeneous materials. However, physical coefficients may vary across a wide spectrum of values, even in a small spatial scale. Besides, the requirement that coefficients have to be constant in each PB-subdomain may result in

an over-partitioned domain where coefficient jumps are not significant among different PB-subdomains. In order to address these situations and to deal with a more general applicability of the preconditioner, we introduce the relaxed PB-BDDC (rPB-BDDC) extension of the preconditioner. In short, relaxed PB (rPB)-subdomains are not determined by constant coefficients within the original partition but we only require that the maximal contrast in each PB-subdomain is less than some predefined tolerance r. We define the maximal contrast independently for each coefficient present in the problem Eq. (3.1), thus defining two (different) thresholds. Then, one can find a rPB-partition, which we denote by Θ^r_{pb}, such that

$$\frac{\alpha_{\max}(\mathcal{D})}{\alpha_{\min}(\mathcal{D})} < r_\alpha \quad \text{and} \quad \frac{\beta_{\max}(\mathcal{D})}{\beta_{\min}(\mathcal{D})} < r_\beta \qquad \forall \mathcal{D} \in \Theta^r_{\text{pb}} \tag{3.22}$$

where $\{r_\alpha, r_\beta\} \geq 1$. Hence, the choice of both thresholds will determine the partition Θ^r_{pb} as a sub-partition of the original partition Θ. Note that if we consider $r_\alpha = r_\beta = \infty$, we recover the original partition Θ, while lower values for the thresholds lead to an increasing number of subparts, consequently *globs*, and thus richer coarse spaces. The rPB-BDDC preconditioner can be defined for any value of the threshold $r > 1$. By tuning r one can obtain the right balance between computational time and robustness.

As coefficients α, β are no longer constant in each rPB-subdomain, we propose to use averaged coefficients in Eq. (3.14) in order to define the averaging operator. The averaged coefficients, denoted as $\bar{\alpha}$ and $\bar{\beta}$, are computed in the rPB-subdomain Ω_k simply as

$$\bar{\alpha} = \frac{1}{\|\Omega_k\|} \int_{\Omega_k} \alpha dx, \qquad \bar{\beta} = \frac{1}{\|\Omega_k\|} \int_{\Omega_k} \beta dx. \tag{3.23}$$

Hence, the definition of the averaging operator Eq. (3.14) is not modified and all DOFs on top of the same coarse geometrical entity are weighted by the same constant value. Thus, under the Θ^r_{pb}, the preconditioner expression is written exactly in the same form as in Sect. 3.3.2.

3.4 Implementation aspects

In this section, we expose implementation strategies for some key points that the authors find of interest for potential users/developers of similar methods, namely: an edge partition algorithm to avoid problematic cases in (unstructured) PB-partitions, the construction of the change of basis, the implementation of the original BDDC constraints and the aggregation of cells into rPB-subdomains based on heterogeneous coefficients α, β and the thresholds r_α, r_β. For a comprehensive implementation strategy of arbitrary order curl-conforming tetrahedral/hexahedral FEs, the reader is referred to Chapter 2.

3.4.1 Coarse Edge partition

Special care has to be taken with the general definition for subdomain edges presented in Sect. 3.2.3. In particular, when *globs* are generated based on Θ_{pb} or a partition obtained with graph partitioners, e.g. METIS, the presented definition of E in Sect. 3.2.3 may not be sufficient for expressing the function and the coarse DOFs in the new basis. We detail the pathological cases identified in [59] plus an additional case, for which we provide examples. We propose a unique cure, based on the partition of problematic coarse edges E into coarse sub-edges E_j such that the problematic cases are solved.

[1] *Disconnected components.* We say that fine edges $e \in E$ are connected if they have an endpoint in common. Consequently, if a coarse edge E has m disconnected components, it has $2m$ endpoints. Note that, while this fact does not preclude the invertibility of the change of basis, if each of the components is treated as a coarse edge we recover original meaningful definitions for continuity constraints across subdomains. Furthermore, if the disconnected components are due to disconnected subdomains, each one must be treated separately in order to ensure the well-posedness of local Neumann problems.

[2] *Interior node in touch with another subdomain.* This case occurs when an internal node v to E does not have the same set of subdomains as $\text{neigh}_{\Theta_{\text{pb}}}(E)$, i.e., is shared by $\text{neigh}_{\Theta_{\text{pb}}}(E)$ plus additional subdomains. In fact, v is then an element of Λ_C in the classification provided in Sect. 3.2.3. We recall that the change of variables is made for gradients of scalar, Lagrangian functions $\boldsymbol{\nabla}\phi_E$ defined on all internal nodes of E. However, if we consider a nodal shape function associated to v, it will be coupled with other internal nodal DOFs for E, thus introducing a coupling between an external subdomain to $\text{neigh}_{\Theta_{\text{pb}}}(E)$ and itself, which is clearly not present in the original basis. A remedy for it consists on simply splitting the coarse edge E into two sub-edges at the problematic node v. Let us denote by V_p the subset of this kind of nodes for all $e \in E$.

[3] *Edge n-furcation.* This situation occurs when a coarse edge E that does not have disconnected components has more than two endpoints. At some internal node the coarse edge is n-furcated into n edges, so the definition of the shape function Φ_E in the new basis loses its original meaning. Furthermore, this fact precludes the locality of the change of basis for every edge E. In this case, a simple remedy is again to split the edge into sub-edges at any node shared by more than two edges.

[4] *Closed loop.* In this case we cannot identify endpoints for a coarse edge and therefore define a unique orientation for it. Furthermore, the new set of basis functions is not well defined since the definition in Sect. 3.3.1 relies on the fact that every edge has 2 end points, thus not being applicable in this case. In this situation, an internal node for the coarse edge E must be chosen as start/end point (common

in all subdomains) to assign an orientation to the edge and be treated as an edge endpoint in the change of basis definition.

In order to address all the presented problematic cases we propose a simple algorithm based on a classification $\forall e \in E$ into sub-edges. Our goal is to find a partition of $e \in E \in \Lambda_E$ into E_j such that every E_j is constructed connecting fine edges that share (only) one vertex with the following edge. Therefore, every coarse sub-edge $E_j \subset E$ has a unique starting point, a chain of connected *fine* edges sharing only one node and a unique end-point, which defines its unique orientation across all subdomains. Let us consider the set of nodes $V = \cup_{e \in E}(v \in \partial e)$, where the number of occurrences for each node $v \in V$ is denoted by count(v). First, we can identify the set of nodes where E is n-furcated as

$$n\text{-furcation nodes} \quad V_N \doteq \{v \in V \setminus V_p | \quad \text{count}(v) > 2\} \tag{3.24}$$

We note that V_p is already identified in the *glob* generation algorithm. Then, we can find a partition of the set of nodes into the two following subsets:

$$\text{Edge boundary nodes} \quad V_B \doteq \{v \in V | \quad \text{count}(v) = 1\} \cup V_p \cup V_N \tag{3.25a}$$

$$\text{Interior nodes} \quad V_I \doteq \{v \in V \setminus V_B\}. \tag{3.25b}$$

Note that by definition of interior nodes count$(v) = 2$ holds. Such classification is performed by simply counting the number of appearances of nodes plus setting problematic nodes belonging to other objects as edge boundary nodes. Then, the coarse edge partitioning Alg. 1 finds paths from one edge boundary node (with a global criteria to select it) till the following edge boundary node. Furthermore, in this procedure we identify the direction of every fine edge with regard to its container coarse edge.

From this point onwards, we consider that each $E \in \Lambda_E$ is a (sub-)edge of the original coarse edges such that they do not present problematic cases.

3.4.2 Change of basis

In this section we provide some implementation details of the change of basis described in Sect. 3.3.1. In the application of the averaging operator in Eq. (3.20), we note that one must apply the global change of basis and its restriction to subdomains. A practical implementation of their application in both cases can be performed with the local restriction of the change of basis to the subdomains, i.e., $\mathcal{Q}_i = R_i \mathcal{Q} R_i^T$, thus it can be performed in parallel in distributed memory environments. The parallel application of the inverse of the restriction of the change of basis to subdomains $\widehat{\mathcal{Q}}^{-1}$ or $\widehat{\mathcal{Q}}^{-T}$ (Eq. (3.20)) is immediate given its definition, see Sect. 3.3.1. On the other hand, the sparsity pattern of of the global change of basis $\mathcal{Q}$ can be exploited in order to achieve a parallel implementation of the application of $\mathcal{Q}$ and $\mathcal{Q}^T$ to a function $\boldsymbol{u} \in X_h$ that only relies on restricted (to the subdomains) information.

Data: $V = V_B \sqcup V_I$, $e \in E$
Result: E_j s.t. $E = \sqcup E_j$
$j \leftarrow 0$
while card$(V) > 0$ **do**
  if card$(V_B) > 0$ **then**
    Find $v^s \in V_B$ with minimum global id
  else
    Find $v^s \in V_I$ with minimum global id
    $V_I \leftarrow V_I \setminus v^s$ and $V_B \leftarrow V_B \cup v^s$
  end
  $j \leftarrow j + 1$
  Find v^e s.t. $\{v^s, v^e\} \in \partial e$ with minimum global id $E_j \leftarrow \{e\}$
  update counters and subsets (Alg. 2)
  while $v^e \in V_I$ **do**
    $v^s \leftarrow v^e$
    Find v^e s.t. $\{v^s, v^e\} \in \partial e$ $E_j \leftarrow E_j \cup e$
    update counters and subsets (Alg. 2)
  end
end

Algorithm 1: Edge partition algorithm

Data: v^s, v^e, V_B, V_I
Result: V_B, V_I
for k *in* s, e **do**
  count$(v^k) \leftarrow$ count$(v^k) - 1$
  if *count*$(v^k) = 0$ *.and.* $v^k \in V_B$ **then**
    $V_B \leftarrow V_B \setminus v^k$
  else if *count*$(v^k) = 0$ *.and.* $v^k \in V_I$ **then**
    $V_I \leftarrow V_I \setminus v^k$
  end
end

Algorithm 2: Update counters and subsets

Proposition 3.4.1. *The expression* $R_i\mathcal{Q}\boldsymbol{u} = \mathcal{Q}_i\boldsymbol{u}^i$ *holds, where* $\boldsymbol{u} \in X_h$ *and* $\boldsymbol{u}^i \in X_h^i$.

Proof. Let us denote by $X_h^{i\,c}$ the subspace of functions in X_h such that functions in X_h^i vanish, thus $X_h = X_h^i \oplus X_h^{i\,c}$. Now, we can define $R_{c_i} : X_h \to X_h^{i\,c}$, i.e., the complementary operator to any R_i such that $\boldsymbol{u} = (R_i^T R_i + R_{c_i}^T R_{c_i})\boldsymbol{u}$. Then, we can make use of this identity to write

$$R_i\mathcal{Q}\boldsymbol{u} = R_i\mathcal{Q}R_i^T R_i\boldsymbol{u} + R_i\mathcal{Q}R_{c_i}^T R_{c_i}\boldsymbol{u}. \tag{3.26}$$

If $\boldsymbol{v} = R_{c_i}^T R_{c_i}\boldsymbol{u} \in X_h^{i\,c}$, the application of $R_i\mathcal{Q}\boldsymbol{v}$ results in a null local vector, since the evaluation of (standard) edge local moments in $X_h^{i\,\prime}$ of the function $\boldsymbol{v} \in X_h^{i\,c}$ (expressed in the new basis) vanishes, thus we can write

$$R_i\mathcal{Q}\boldsymbol{u} = R_i\mathcal{Q}R_i^T R_i\boldsymbol{u} = \mathcal{Q}_i\boldsymbol{u}^i. \tag{3.27}$$

□

Unfortunately, this reasoning cannot be applied to the transpose of the change of basis since $R_{c_i}\mathcal{Q}R_i^T R_i\boldsymbol{u}$ evaluates the edge moments in $X_h^{i\,c\prime}$ of a function $R_i^T R_i\boldsymbol{u} \in X_h^i$, which can be nonzero.

Proposition 3.4.2. *Consider arbitrary local weighting diagonal matrices* W_i *for every subdomain such that* $\boldsymbol{u} = \sum_i R_i^T W_i R_i\boldsymbol{u}$*, i.e., they form a partition of the unity. Then, the expression* $R_i\mathcal{Q}^T\boldsymbol{u} = R_i\sum_i R_i^T\mathcal{Q}_i^T W_i\boldsymbol{u}^i$ *holds, where* $\boldsymbol{u} \in X_h$ *and* $\boldsymbol{u}^i \in X_h^i$.

Proof. We can make use of the restriction R_i and its complementary operator R_{c_i} for every subdomain to write

$$\begin{aligned} R_i\mathcal{Q}^T\boldsymbol{u} &= R_i\sum_i \mathcal{Q}^T R_i^T W_i R_i\boldsymbol{u} \\ &= R_i\sum_i R_i^T R_i\mathcal{Q}^T R_i^T W_i R_i\boldsymbol{u} + R_i\sum_i R_{c_i}^T R_{c_i}\mathcal{Q}^T R_i^T W_i R_i\boldsymbol{u}. \end{aligned} \tag{3.28}$$

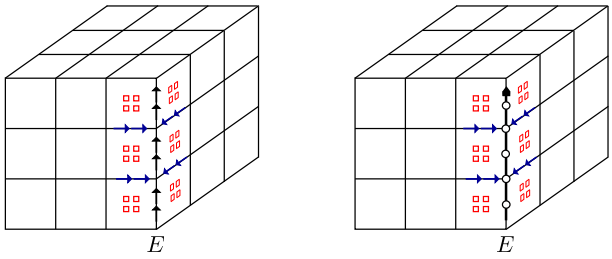
Again, if $\boldsymbol{v} = R_i^T W_i R_i\boldsymbol{u} \in X_h^i$, an analogous reasoning to that in proof of Prop. 3.4.1 leads to $R_{c_i}\mathcal{Q}^T\boldsymbol{v} = \boldsymbol{0}$, thus we can state

$$R_i\mathcal{Q}^T\boldsymbol{u} = R_i\sum_i R_i^T R_i\mathcal{Q}^T R_i^T W_i R_i\boldsymbol{u} = R_i\sum_i R_i^T\mathcal{Q}_i^T W_i\boldsymbol{u}^i. \tag{3.29}$$

□

Therefore, the application of the change of basis can rely only on restrictions of the same to subdomains whereas the application of the transpose change of basis can be performed in parallel with subdomain restrictions plus nearest neighbour communications.

With this purpose in mind, we detail here how to implement the restriction of the change of basis local to subdomains. Let us define a partition of the DOFs in $X_h^{i\,\prime}$ into three subsets of DOFs, namely: the DOFs placed on top of $e \in E$, the DOFs placed on



(A) Black arrows represent standard basis DOFs over E.

(B) Nodes represent Lagrangian-like new basis DOFs. Long arrow represents the DOF for function Φ_E.

FIGURE 3.1: Standard (old) and new basis DOFs for 3D hexahedra second order edge FE over E. Additional depicted DOFs values are affected by the change of basis for E, while the remaining DOFs are invariant under the change of basis.

top of interface edges/faces $f \notin E$ such that $\partial f \cap E \neq \emptyset$, and the remaining DOFs in $X_h^{i\prime}$, denoted by $\boldsymbol{u}_E$, $\boldsymbol{u}_f$ and $\boldsymbol{u}_r$, respectively. Furthermore, let us consider that DOFs in $\boldsymbol{u}_E$ are sorted such that DOFs belonging to the same coarse E are found in consecutive positions. Note that shape functions associated to $\boldsymbol{u}_f$, $\boldsymbol{u}_r$ are common in both (i.e., old and new) bases. For edge FEs of order k, k moments (i.e., DOFs) are defined on top of each $e \in \mathcal{T}_h$. Let us denote by n_e the number of fine edges $e \in E$. Then, the total number of DOFs on top of a coarse edge E is kn_e. On the other hand, the number of Lagrangian-like DOFs interior to E (i.e., excluding ∂E) is $(kn_e - 1)$, i.e., the number of shape functions of the type $\boldsymbol{\nabla}\phi_E$. The change of basis is completed with the addition of the function Φ_E to the new basis so that the dimension of both bases coincides. For the sake of illustration, both sets of basis functions restricted to a coarse edge E are depicted in Fig. 3.1.

Let us denote by n_E the number of coarse edges $E \in \Lambda_E$ for a given subdomain. Then, we define the change of basis $\mathcal{Q}^{E_j}$, $j = \{1, \ldots, n_E\}$ local to every $E_j \in \Lambda_E$ as

$$\mathcal{Q}^{E_j}_{ab} = \sigma_a(\boldsymbol{\nabla}\phi^b_E), \quad \text{for} \quad b = 1, \ldots, kn_e^{E_j} - 1 \qquad \mathcal{Q}^{E_j}_{a,kn_e^{E_j}} = \sigma_a(\Phi_{E_j}), \tag{3.30}$$

where σ_a, $a = 1, \ldots, kn_e^{E_j}$, are the (original basis) edge moments defined on top of E_j (the superscript in $n_e^{E_j}$ has been introduced to show that it depends on the coarse edge). We can now define the change of basis $\mathcal{Q}^E = \text{diag}(\mathcal{Q}^{E_1}, \ldots, \mathcal{Q}^{E_{n_E}})$ local to coarse edges. We remark that the same orientation for every coarse edge $E \in \Lambda_E$ must be defined on the set of subdomains $\mathcal{D} \in \text{neigh}_\Theta(E)$. Otherwise, the definition of the new basis function Φ_E is not consistent across subdomains. In addition, the change of basis must take into account the effect of the new DOF values $\boldsymbol{u}_E$ associated to $\boldsymbol{\nabla}\phi^j_E$ and Φ_E for E in the old values $\boldsymbol{u}_f$. Thus, we evaluate for all indices b of shape functions associated

to $\boldsymbol{u}_E$

$$\mathcal{Q}^f_{cb} = \sigma_c(\boldsymbol{\nabla}\phi^b_E), \tag{3.31}$$

where c corresponds to the index of all moments associated to $\boldsymbol{u}_f$. We recall that, by definition, $\sigma_c(\Phi_E) = 0$. The application of the moments σ_c to the (original) shape functions associated to $\boldsymbol{u}_f$ results in $\sigma_c(\boldsymbol{\varphi}^b) = \delta_{cb}$. Finally, DOFs in $\boldsymbol{u}_r$ are invariant under the change of basis. We can now state the (local) change of basis for the subdomain as

$$\boldsymbol{u}_{\text{old}} = \begin{bmatrix} \boldsymbol{u}_E \\ \boldsymbol{u}_f \\ \boldsymbol{u}_r \end{bmatrix}_{\text{old}} = \begin{bmatrix} \mathcal{Q}^E & 0 & 0 \\ \mathcal{Q}^f & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_E \\ \boldsymbol{u}_f \\ \boldsymbol{u}_r \end{bmatrix}_{\text{new}} = \mathcal{Q}_i \boldsymbol{u}_{\text{new}}, \tag{3.32}$$

where becomes clear the fact that the inverse of the change of basis is well defined if and only if $\mathcal{Q}^E$ is invertible. In turn, $\mathcal{Q}^E$ will be invertible if and only if every change of basis local to $E \in \Lambda_E$ (Eq. (3.30)) is invertible.

Remark 4. *Although it is used in this exposition for the sake of clarity, we do not require any particular ordering of DOFs in $\boldsymbol{u} \in X_h^{i\,\prime}$ in a practical implementation of $\mathcal{Q}_i$.*

3.4.3 BDDC constraints

In this subsection we propose a practical manner of computing the BDDC constraints Eqs. (3.12a) and (3.12b) for local problems. In our implementation, constraints over local problems are strongly imposed through the usage of Lagrange multipliers on the original basis. Therefore, the local matrix $\mathcal{A}^i_h$ is extended with the discrete version of the constraints C_i in order to obtain constrained (Neumann) local problems.

The computation of constraints requires to integrate zero and first order moments for the solution over all coarse edges E. We note that the first constraint Eq. (3.12a) can be easily implemented for k-order edge FEs as

$$\begin{aligned} s_{0,E}(\boldsymbol{u}) \doteq \int_E \boldsymbol{u}\cdot\boldsymbol{t}_E ds = \int_E (\sum_{a=1}^{kn_e} u^a \boldsymbol{\varphi}^a)\cdot\boldsymbol{t}_E ds = \sum_{a=1}^{kn_e} u^a \int_E \boldsymbol{\varphi}^a\cdot\boldsymbol{t}_E ds \\ = \sum_{a=1}^{kn_e} u^a \int_E \boldsymbol{\varphi}^a\cdot\boldsymbol{t}_E(\sum_{b=1}^{k} p_b) ds = \sum_{a=1}^{kn_e}(\boldsymbol{t}_e\cdot\boldsymbol{t}_E)u^a = \sum_{a=1}^{kn_e} C^a u^a, \end{aligned} \tag{3.33}$$

where we used the partition of the scalar, unit function into the set of Lagrangian test functions p_b belonging to the polynomial space $V_h(E)$ of order $k-1$. These functions are used for defining the k local moments on every e as $\sigma^b(\boldsymbol{u}) = \int_e \boldsymbol{u}\cdot\boldsymbol{t}_e p_b$, $b = \{1,\ldots,k\}$ (see Chapter 2 for details). Its duality with basis shape functions, i.e., $\sigma^b(\boldsymbol{\varphi}^a) = \delta_{ba}$, has been used in Eq. (3.33). Thus, to compute the first constraint one only needs to add ± 1 at the corresponding entry in C for each DOF, where the sign is determined by the agreement between fine and coarse edge orientations, i.e., $\boldsymbol{t}_e\cdot\boldsymbol{t}_E$. On the other hand,

the computation of the second constraint Eq. (3.12b) requires to define an arc-length parameter over E. A practical implementation of the constraint Eq. (3.12b) can avoid it by considering the constraint in the new basis. Since ϕ_E^j vanish at ∂E, integration by parts yields

$$s_{1,E}(\boldsymbol{w}^i) \doteq \int_E s\boldsymbol{w}^i \cdot \boldsymbol{t}_E ds = \int_E s(u_E\Phi_E + \sum_{a=1}^{kn_e-1} u^a \boldsymbol{\nabla}\phi_E^a) \cdot \boldsymbol{t}_E ds = \tag{3.34}$$
$$- \sum_{a=1}^{kn_e-1} u^a \int_E \phi_E^a ds = - \sum_{a=1}^{kn_e-1} C_{\text{new}}^a u^a,$$

where the contribution of Φ_E is null due to the antisymmetry of the product $s\Phi_E$ (we recall that $s \in [-|E|/2, |E|/2]$) over E. Then, we can apply the change of basis to obtain the expression in the original basis, i.e., $C = C_{\text{new}}\mathcal{Q}^{-1}$.

3.4.4 Building Θ_{pb}^r

In the rPB-BDDC method, a Θ_{pb}^r partition is used such that the maximal contrast, for each one of the coefficients, is lower than a predefined tolerance r in each subdomain. Our goal is to identify a partition of every subdomain into $\mathcal{D}' \in \Theta_{\text{pb}}^r$ subdomains where the thresholds $\frac{\alpha_{\max}(\mathcal{D}')}{\alpha_{\min}(\mathcal{D}')} < r_\alpha$ and $\frac{\beta_{\max}(\mathcal{D}')}{\beta_{\min}(\mathcal{D}')} < r_\beta$ are respected. It can be accomplished using different algorithms. One approach is to consider a seed cell and aggregate the surrounding cells such that the contrast(s) are below the give threshold(s), proceeding recursively till no neighbouring cells can be aggregated. We take another seed among the non-aggregated cells and proceed again till all cells have been processed.

Alternatively, one can first determine the maximum and minimum values for α and β in a given subdomain $\mathcal{D} \in \Theta$. With this information and the thresholds r_α, r_β, we can determine the number of sub-intervals for every subdomain and coefficient as follows. First, we compute $\ell_\alpha(\mathcal{D})$ and $\ell_\beta(\mathcal{D})$ as the smallest positive integers for which

$$\frac{\alpha_{\max}(\mathcal{D})}{\alpha_{\min}(\mathcal{D})} < r_\alpha^{\ell_\alpha(\mathcal{D})}, \qquad \frac{\beta_{\max}(\mathcal{D})}{\beta_{\min}(\mathcal{D})} < r_\beta^{\ell_\beta(\mathcal{D})}, \tag{3.35}$$

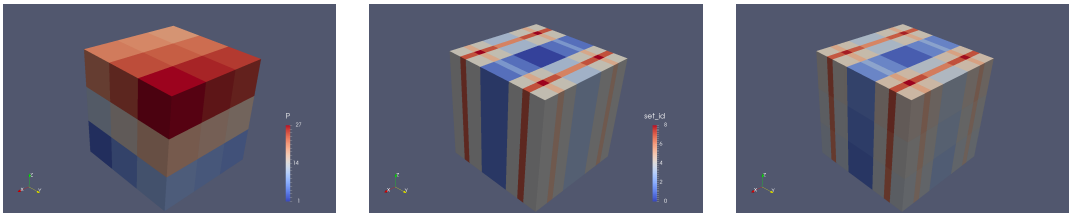
respectively. One can now define the intervals

$$I_{i,j} \doteq [r_\alpha^{i-1}\alpha_{\min}(\mathcal{D}), r_\alpha^i \alpha_{\min}(\mathcal{D})] \times [r_\beta^{j-1}\beta_{\min}(\mathcal{D}), r_\beta^j \beta_{\min}(\mathcal{D})],$$

for $i \in [1, \ell_\alpha(\mathcal{D})]$, $j \in [1, \ell_\beta(\mathcal{D})]$. Cells with their coefficients on the same interval $I_{i,j}$ are aggregated in a PB-subdomain. Those cells that have coefficients across multiple sub-intervals are treated as additional PB-subdomains.

For the sake of illustration, we include an example where all the different subset indices are presented for a unit cube domain: the original partition into $P = 3 \times 3 \times 3$ subdomains Fig. 3.2a, the aggregation of cells into subsets based on $\log(\beta) = 3\sin(3\pi y)$ (see Fig. 3.3a) for $r = 10^3$ in Fig. 3.3b, combined with an analogous partition for $\log(\alpha) =$

$3\sin(3\pi x)$ leading to a coefficient-based partition in Fig. 3.2b and the final rPB-partition Θ^r_{pb} in Fig. 3.2c.

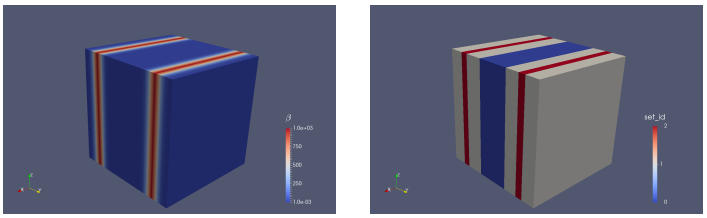


(A) Original geometrical partition Θ

(B) Aggregation of cells into subsets based on coefficients

(C) Resulting rPB-partition Θ^r_{pb}.

FIGURE 3.2: Partitions for scalar coefficients described by $\log(\alpha) = 3\sin(3\pi x)$ and $\log(\beta) = 3\sin(3\pi y)$ with an initial $3 \times 3 \times 3$ partition of the unit cube.



(A) Analytical function.

(B) Aggregation of cells into subsets.

FIGURE 3.3: Aggregation of cells into subsets based on $log(\beta) = 3\sin(3\pi(y))$ with threshold $r = 10^3$.

Note that the PB-BDDC approach requires an interface between the problem discretization, i.e., access to the physical properties, and the preconditioner set-up. This is in the core of `FEMPAR`, which provides a tight interaction among building blocks to fully exploit the mathematical structure of the PDE operator.

3.5 Numerical results

In this section we evaluate the weak scalability of the proposed preconditioner for the problem in Eq. (3.1), within the preconditioned conjugate gradient (CG) Krylov iterative solver. The robustness of the rPB-BDDC-CG solver is tested in 3D simple domains, which are discretized either with structured and unstructured meshes. As performance metrics, we focus on the number of rPB-BDDC preconditioned CG iterations required to attain the convergence criteria, which is defined as the reduction of the initial residual algebraic L_2-norm by a factor 10^{-6}. On the other hand, the total computation time will be presented, which will include both preconditioner set-up and the preconditioned iterative solution of the linear system in all the experiments reported. The particular definition of coefficients α and β and its distribution will be specified throughout the section for each case.

3.5.1 Experimental framework

The rPB-BDDC methods have been implemented in the scientific software project `FEMPAR` (see Chapter 1). Regarding the content of this chapter, `FEMPAR` provides the basic tools for the efficient parallel distributed-memory implementation of substructuring DD solvers [21, 23], based on a *fully-distributed* implementation of data structures involved in the parallel simulation. The parallel codes in `FEMPAR` heavily use standard computational kernels provided by BLAS and LAPACK. Besides, through proper interfaces to several third party libraries, the local constrained Neumann problems and the global coarse-grid problem can be solved via sparse direct solvers. In this thesis, we use the overlapped BDDC implementation proposed in [18], with excellent scalability properties. It is based on the overlapped computation of *coarse* and *fine* duties. As long as coarse duties can be fully overlapped with fine duties, perfect weak scalability can be attained. We refer to [23] for more details.

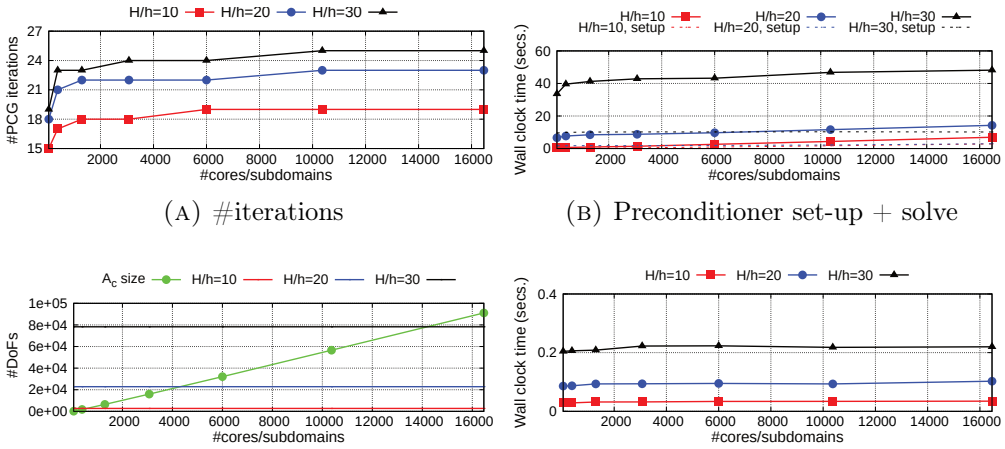
The experiments in this section have been performed on the MareNostrum-IV [4] (MN-IV) supercomputer, hosted by the Barcelona Supercomputing Center (BSC). In all cases, we consider a one-to-one mapping among subdomains, cores and MPI tasks. Provided that the algorithm allows for a high degree of overlapping between fine and coarse duties, an additional MPI task is spawn into a full node (i.e., 48 cores) in order to perform the coarse problem related tasks. The multi-threaded PARDISO solver in Intel MKL is used to solve the coarse-grid problem within its computing node.

Unless otherwise stated, the problem Eq. (3.1) will be solved in the unit cubic domain $\Omega = [0,1]^3$ with Dirichlet homogeneous boundary conditions on the whole boundary and the forcing term $\boldsymbol{f} = 1$. Let us denote by h the usual mesh element size, and by H the size of the subdomain. Then, local problem sizes can be characterized in a structured mesh and partition by $\frac{H}{h}$. In order to perform a weak scalability analysis, we build a set of structured meshes consisting on $(4\frac{H}{h}k \times 4\frac{H}{h}k \times 3\frac{H}{h}k)$ hexahedra. A uniform partition of the meshes into $P = (4k \times 4k \times 3k) = 48k^3$ subdomains is considered, where local problem sizes are $(\frac{H}{h})^3$.

3.5.2 Homogeneous problem

Let us first consider homogeneous coefficients $\alpha = \beta = 1.0$ for the whole domain Ω. We test the problem with different local problem sizes $\frac{H}{h}$ and FE orders. In this case, the PB-BDDC preconditioner reduces to the standard BDDC preconditioner since $\Theta = \Theta_{\text{pb}}$. In Fig. 3.4, we present weak scalability results for the homogeneous problem up to 16464 subdomains with different local problem sizes $\frac{H}{h} = \{10, 20, 30\}$, where the largest case has more than 10^3 M DOFs. We present the number of solver iterations until convergence in Fig. 3.4a, and employed wall clock times in Fig. 3.4b, which are composed by the preconditioner set-up time and the solution time with the BDDC preconditioned solver. The plots indicate that both the algorithm and its implementation in `FEMPAR` have excellent weak scalability properties. Provided that the algorithm overlaps fine

and coarse tasks, coarse tasks computing times are masked as long as they not exceed computing times for local problems, which allow us to observe excellent weak scalability times for the largest case $\frac{H}{h} = 30$ in Fig. 3.4b. The size of both local and coarse problems is presented in Fig. 3.4c. Finally, we add a plot (Fig. 3.4d) of the time needed to set-up the change of basis, which includes the edge partition algorithm (Alg. 1) to detect problematic cases, to show that consumed time is only dependent on the local problem size and not significant compared to the one spent in the solver run.



(A) #iterations

(B) Preconditioner set-up + solve

(C) Coarse size problem.

(D) Change of basis set-up times

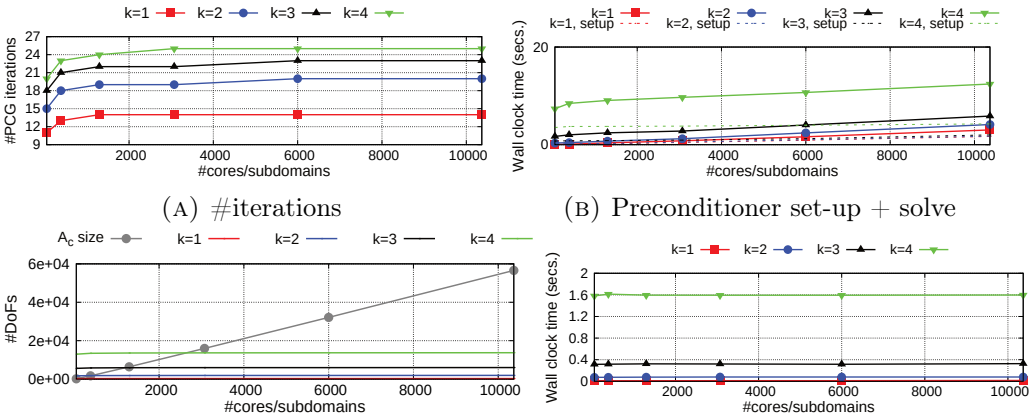
FIGURE 3.4: Weak scalability results for first order edge FE with a constant distribution of materials for different local problem $\frac{H}{h}$ sizes.

Fig. 3.5 shows the weak scalability results for an homogeneous problem with constant coefficients and different FE orders up to 4. First, the number of iterations is (asymptotically) constant, thus the method is scalable. In this case, local problems sizes are such that the coarse problem is larger from an early number or subdomains (see Fig. 3.5c), thus it is reflected in the solver times plot in Fig. 3.5b. In Fig. 3.5d, the time spent in the set-up of the change of basis is presented. Out of the presented results for the homogeneous case, a clear conclusion can be drawn for the standard BDDC: the algorithm and its implementation have excellent weak scalability properties.

3.5.3 Multi-material problem

Checkerboard distribution

The checkerboard arrangement of coefficients is a widely used distribution of materials to test the robustness of the BDDC algorithms for problems in $H(\text{curl})$ against the jump of coefficients across the interface [137, 59]. In short, it is a bi-material distribution of subdomain-constant coefficients such that every subdomain presents a jump of coefficients through the faces to all its neighbours. For the sake of ease, let us distinguish between black and white subdomain-constant materials, see Fig. 3.6a. Note that in the checkerboard distribution case, the jumps of coefficients are aligned with the partition, thus $\Theta_{\text{pb}} = \Theta$.



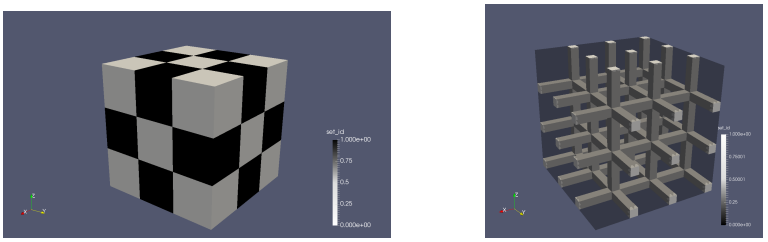
(A) #iterations

(B) Preconditioner set-up + solve

(C) Coarse size problem.

(D) Change of basis set-up times

FIGURE 3.5: Weak scalability results for different order edge FE with a constant distribution of materials and a local problem $\frac{H}{h} = 10$.



(A) Checkerboard distribution of coefficients, aligned with the partition and thus $\Theta_{\text{pb}} = \Theta$.

(B) Channel distribution of coefficients with $N = 3$ and $\gamma = 0.2$.

FIGURE 3.6: Bi-material distribution cases in a structured $3 \times 3 \times 3$ partition of the unit cube $[0, 1]^3$.

We first test the robustness of the algorithm against the contrast of the coefficients. Consider a $3 \times 3 \times 3$ partition of a unit cube domain with a checkerboard distribution of coefficients such that $\alpha_{\text{white}} = \beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^i$, $\beta_{\text{black}} = 10^{-i}$. The contrast is defined here as $\frac{\alpha_{\text{black}}}{\beta_{\text{black}}}$. With the variation of the value for i in the range $[-5, 5]$ we test all the possible scenarios, namely the mass dominated problem ($i < 0$) and the curl dominated problem ($i > 0$). The number of iterations with the contrast of the coefficients for different configurations of the preconditioner is presented in Fig. 3.7. Out of the plot, the most salient property is the robustness of the perturbed preconditioner (see Sect. 3.3.3) with the contrast of the coefficients. In fact, the original formulation of the preconditioner suffers from a large number of iterations when the contrast between the two coefficients is large, specially in the curl-dominated case. Therefore, the proposed perturbation of the preconditioner is essential to achieve a robust preconditioner, in the case where both coefficients α and β jump across the interface. Clearly, the perturbed formulation only has a (negligible) negative impact in the case $i = 0$, since actually no jump occurs across the interface. In the curl-dominated limit, α and ω-based scalings show the same behaviour, as it is suggested by the definition of ω when $\alpha >> \beta$. On

$\frac{H}{h}$ / P	2^3	3^3	4^3	5^3	6^3	7^3	8^3
4	14	24	35	38	40	40	41
8	26	37	61	65	70	69	70
12	31	52	72	78	82	82	84

(A) Standard BDDC preconditioner

$\frac{H}{h}$ / P	2^3	3^3	4^3	5^3	6^3	7^3	8^3
4	8	9	10	10	11	12	12
8	12	14	16	16	17	17	17
12	15	22	21	21	21	21	21

(B) Perturbed BDDC preconditioner

TABLE 3.1: Weak scalability in terms of number of iterations for both preconditioners. Checkerboard distribution of materials with $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$.

the other hand, when the coefficient β becomes dominant, the choice of *cardinal* and ω-based scalings also leads to good scalability results in the limit. In sum, the combination of the perturbed formulation and α-based scaling is the most robust approach. Unless otherwise stated, this combination will be used throughout the section.

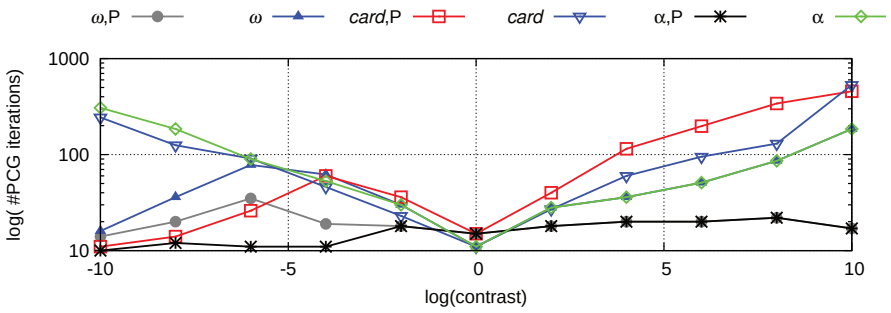
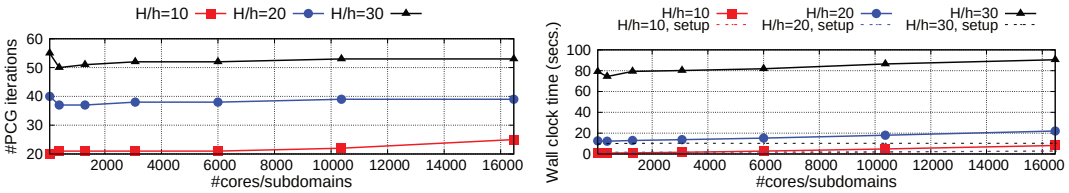


FIGURE 3.7: Number of iterations for first order edge FE with a $3\times3\times3$ partition of the unit cube and $H/h = 8$. A checkerboard arrangement of materials is defined: $\alpha_{\text{white}}1.0$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^i$ and $\beta_{\text{black}} = 10^{-i}$, leading to a contrast $= \frac{\alpha_{\text{black}}}{\beta_{\text{black}}} = \frac{10^i}{10^{-i}} = 10^{2i}$. Labels include scaling information, where P denotes perturbation of the preconditioner.

Let us now consider a checkerboard arrangement of coefficients such that $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$. In order to show the importance of the perturbed formulation of the preconditioner for jumps of both coefficients across interfaces, we collect the number of iterations for the original and perturbed preconditioner in Tabs. 3.1a and 3.1b, respectively. The problem is solved with a $P = N\times N\times N$ partition of the unit cube and the ω-based scaling is employed in both cases. Iteration counts for the perturbed preconditioner are noticeably lower in all cases without exception.

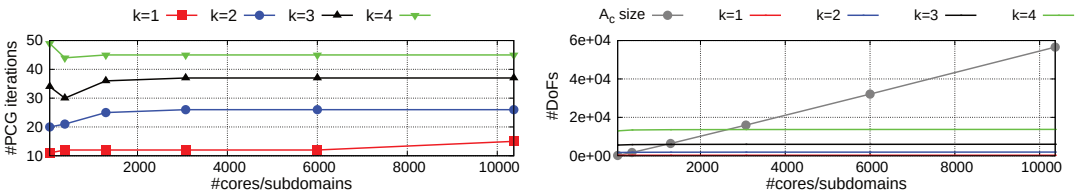
Once we have shown the importance of the perturbed formulation, we present a weak scalability analysis up to 16464 subdomains and the checkerboard arrangement of materials with the perturbed preconditioner. Problem sizes in this experiment coincide to the ones presented for the homogeneous problem in Fig. 3.4c. As expected, plots in Fig. 3.8 show excellent scalability properties of the preconditioner in this case, i.e., the preconditioner is robust with jumps of coefficients across the interface. Although higher values of $\frac{H}{h}$ lead to a significantly higher number or iterations, these ones are (asymptotically) constant and remain in a reasonable range, see Fig. 3.8a. On the other hand, Fig. 3.9 presents the number of iterations for different order FEs and problem

size $H/h = 4$, which also is shown to be scalable. Out of the contrast and scalability results, we would like to remark the following issues. First, the perturbed formulation of the preconditioner is essential to achieve a robust preconditioner. Second, the method is weakly scalable for problems with high coefficient jumps across interfaces for different local sizes and FE orders.



(A) #iterations (B) Preconditioner set-up + solve

FIGURE 3.8: Weak scalability results for first order edge FE with a checkerboard distribution of materials: $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$.



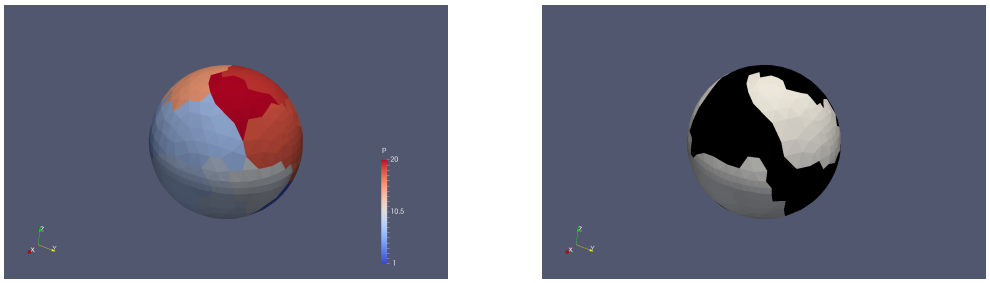
(A) #iterations (B) Problem sizes

FIGURE 3.9: Weak scalability results for different order edge FE and $H/h = 4$ with a checkerboard distribution of materials: $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$.

In order to show the robustness of the method not only with structured, regular hexahedral meshes we solve the problem for a spherical domain and partition with a graph partitioner METIS. Let us consider a spherical domain with $R = 0.5$, discretized with an unstructured tetrahedral mesh containing around 50.000 cells. In order to achieve high contrast of coefficients across interfaces, a bi-material distribution of coefficients is assigned such white or black subdomain-constant materials are randomly assigned, see Fig. 3.10b for an illustration. The definition of the sets of coefficients is $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$. Fig. 3.11 shows the number of iterations with the original and the perturbed formulation of the preconditioner. The perturbed preconditioner, combined with a α-based scaling, is the unique method shown to be robust with regard to the coefficients contrast, reproducing the behaviour observed in the structured case.

Multiple channels

In this arrangement of materials we have a background domain, denoted by black region, and a set of inclusions, denoted by white regions, that cross the domain from one boundary to the opposite one, in parallel to the axis directions. We include one channel



(A) Partition of the sphere into 20 subdomains.

(B) Bi-material random distribution of materials with constant properties on each subdomain.

FIGURE 3.10: Sphere partition and distribution of materials.

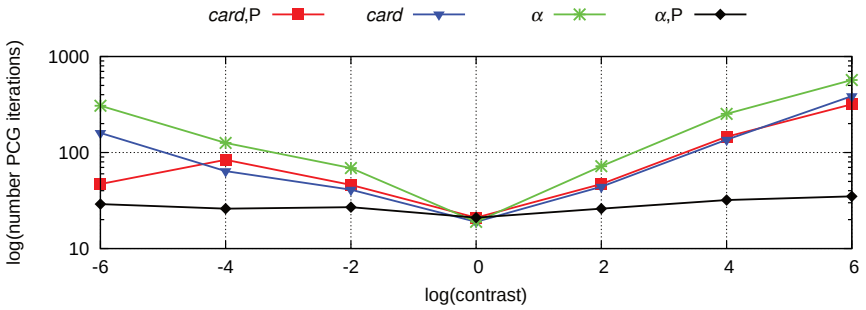


FIGURE 3.11: Robustness for a tetrahedral mesh for different scalings. Material parameters defined as $\alpha_{\text{white}} = 10^i$, $\alpha_{\text{black}} = 1.0$, $\beta_{\text{white}} = 10^{-i}$ and $\beta_{\text{black}} = 1.0$, contrast defined $\frac{\alpha_{\text{white}}}{\beta_{\text{white}}}$. Mesh partitioned into 20 subdomains and random assignment of materials. Labels include scaling information, where P denotes perturbation of the preconditioner.

per direction per subdomain so that with an increasing number of P subdomains we are solving a harder problem with P channels. Channels are parallel to the axis and are positioned in the lowest corners. They and have a squared cross-section of size γH, thus occupying a $\gamma^2|\tilde{\Omega}_i|$ volume in every subdomain, see Fig. 3.6b for an illustration of a $3 \times 3 \times 3$ partition of the unit cube with the described channel inclusions. The distribution of coefficients is such that contains coefficient jumps within each subdomain and also across all interfaces, thus the perturbed formulation of the PB-BDDC preconditioner will be employed.

Let us first compare the number of iterations for the PB-BDDC preconditioner against the ones that one would have with the standard BDDC, where the definition of *globs* is generated with the original partition Θ. In Sect. 3.5.3, the effectiveness of the perturbation formulation has been empirically shown. Consequently, in order to provide a fair comparison among them, the perturbed formulation is considered for both preconditioners. On the other hand, while α-based scaling is shown to be the most robust approach for PB-BDDC preconditioner, it miserably fails when considered with the standard *globs*, i.e., given by Θ. In this case, a better result is obtained with *cardinality* scaling. Tabs. 3.2a and 3.2b show iteration counts for the described BDDC and PB-BDDC preconditioners, respectively, for the solution of the channels problem with

$\frac{H}{h}$/c	10^{-4}	10^{-2}	1.0	10^{2}	10^{4}
4	36	29	13	31	74
8	67	36	16	38	104

(A) BDDC preconditioner

$\frac{H}{h}$/c	10^{-4}	10^{-2}	1.0	10^{2}	10^{4}
4	14	14	11	13	14
8	18	19	16	17	20

(B) PB-BDDC preconditioner

TABLE 3.2: Comparison in number of iterations for both preconditioners in a $3 \times 3 \times 3$ partition. Channel distribution of materials with $\gamma = 0.5$ and $\alpha_{\text{black}} = \beta_{\text{black}} = 1.0$, $\alpha_{\text{white}} = 10^{i}$ and $\beta_{\text{white}} = 10^{-i}$. Contrast defined as $c = \frac{\alpha_{\text{white}}}{\beta_{\text{white}}}$.

$\gamma = 0.5$ and a partition of the unit cube into $P = 3 \times 3 \times 3$ subdomains with local size $H/h = 8$. We define the coefficients $\alpha_{\text{black}} = \beta_{\text{black}} = 1.0$, while we distinguish between $\alpha_{\text{white}} = 10^{i}$ and $\beta_{\text{white}} = 10^{-i}$, which allow us to define the contrast as $\frac{\alpha_{\text{white}}}{\beta_{\text{white}}}$. As expected, the PB-BDDC preconditioner is robust with the contrast of coefficients. On the other hand, the number of iterations is notably higher for the BDDC preconditioner in the curl-dominated case.

The following experiment evaluates the weak scalability properties for a channel-type distribution of materials. In Figs. 3.12 and 3.13 we present weak scalability results for different problem sizes and FE orders. The most salient property out of this plots is that the number of iterations is asymptotically constant for all cases. However, coarse problem sizes become larger as the partition into PB-subdomains generates a higher number of coarse DOFs, see Fig. 3.12c. In this context, the coarse problem is larger than local problems size from an early number of subdomains, thus coarse tasks will predominate computing times precluding wall clock time scalability, as it is shown in Fig. 3.12b. In Figs. 3.12d and 3.13b we present scalable wall clock times for the change of basis set-up, for different local problem sizes and FE orders.

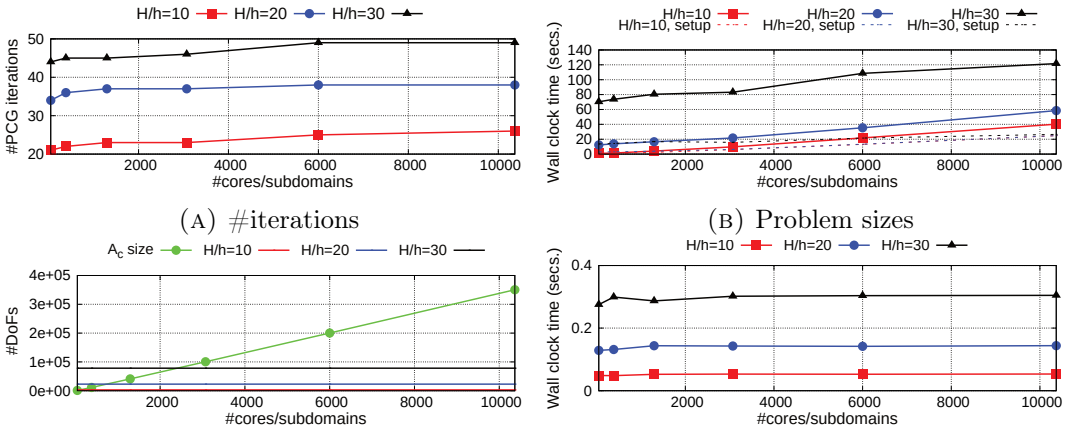
We would like to remark that the proposed PB-BDDC preconditioner is weakly scalable for the number of iterations until convergence not only with regard to the jump of coefficients across interfaces but also for distributions of different materials within each subdomain. A multilevel version of the preconditioner for curl-conforming spaces [149], not addressed in this thesis, is expected to push forward the limits of the computing times scalability results.

3.5.4 Heterogeneous problems

In this section we study the scalability of the rPB-BDDC method for problems where the coefficients α, β are described by continuous (at least element-wise) functions, which contain high contrasts for their maximum and minimum values.

Periodic analytical functions

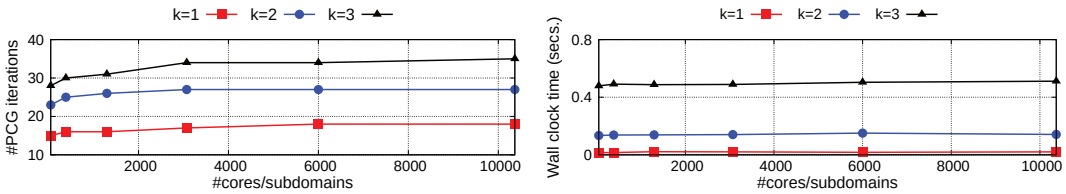
In this case, α and β are defined as exponential functions with a sinusoidal exponent such that the function is periodic on the domain and the number of peaks scales with the number of original subdomains in Θ, thus solving a harder problem as we increase



(A) #iterations (B) Problem sizes

(C) Problem sizes (D) #iterations

FIGURE 3.12: Weak scalability results for first order edge FE with a channel distribution of materials: $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$.

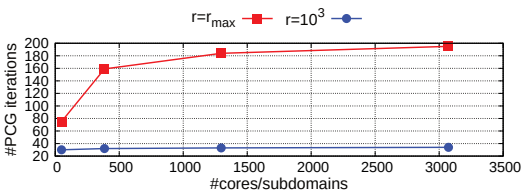


(A) #iterations (B) Change basis set-up time

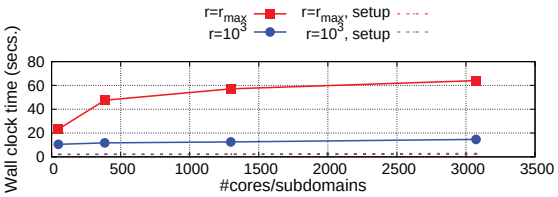
FIGURE 3.13: Weak scalability results for different order edge FE and $H/h = 4$ with a channel distribution of materials: $\alpha_{\text{white}} = 10^2$, $\beta_{\text{white}} = 1.0$ and $\alpha_{\text{black}} = 10^4$ and $\beta_{\text{black}} = 10^{-2}$.

the number of processors. In particular, let us consider $\log(\alpha) = \frac{c_{\max}}{2}\sin(N_x\pi x)$ and $\log(\beta) = \frac{c_{\max}}{2}\sin(N_y\pi y)$, where N_x, N_y denote the number of subdomains in x, y directions, respectively, in a $P = N_x \times N_y \times N_z$ structured partition, (see β depicted in Fig. 3.3a for the case $c_{\max} = 6$, $N_x = N_y = 3$). Clearly, the maximum contrast within each coefficient is given by $r_{\max} = 10^{c_{\max}}$. We present weak scalability results up to 3072 subdomains for two different thresholds $r = \{r_{\max}, 10^3\}$, and $\frac{H}{h} = 20$ local problem size in three different scenarios: coefficients α (Fig. 3.14), β (Fig. 3.15) or both are heterogeneous (Fig. 3.16), being set to $\alpha = 1.0$, $\beta = 1.0$ otherwise. Out of these plots, we can draw some conclusions. First, the case where only β is heterogeneous converges in a lower number of iterations compared to problems with heterogeneous α. Secondly, the consideration of lower values for r consequently results in larger coarse problems, but its size is only (approximately) doubled when only one coefficient is heterogeneous or (approximately) quadrupled when both are defined heterogeneous. In fact, in the range of subdomains considered in this experiment, the coarse problem computational times in all cases can be masked by local problem ones (Figs. 3.14b, 3.15b and 3.16b). Finally, and most salient, the rPB-BDDC method with $r = 10^3$ is weakly scalable in all cases with an excellent reduction in the number of iterations and computing times compared

to the case where $r = r_{\max}$.

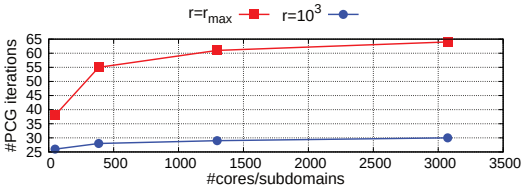


(A) Number of iterations

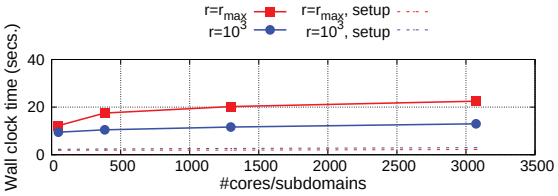


(B) Set-up and total solver time.

FIGURE 3.14: Weak scalability for the rPB-BDDC when only an heterogeneous α is considered, $\beta = 1.0$.

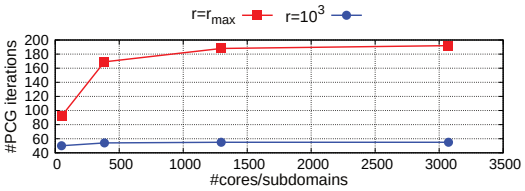


(A) Number of iterations

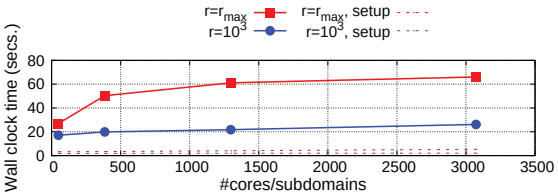


(B) Set-up and total solver time.

FIGURE 3.15: Weak scalability for the rPB-BDDC when only an heterogeneous β is considered, $\alpha = 1.0$.



(A) Number of iterations



(B) Set-up and total solver time.

FIGURE 3.16: Weak scalability for the rPB-BDDC when both coefficients are heterogeneous.

High Temperature Superconductors

Next, we study the scalability of the algorithm with a practical application, the modelling of High Temperature Superconductors (HTS). The problem consists in the magnetization of a superconducting cube completely surrounded by a dielectric material (see Fig. 3.18a), subjected to an external AC magnetic field. The formulation Eq. (3.1) arises in the time-domain quasi-static approximation of the Maxwell's equations for solving the magnetic field, see Chapter 4 for details. Furthermore, the standard Backward Euler method is used to perform time integration over a time interval $[0, T]$, so let us define a time partition $\{0 = t^0, t^1, \ldots, t^N = T\}$ into N time elements. Then, the form Eq. (3.1) can be used to compute the magnetic field for a particular time t^n, provided the solution on the previous time $\boldsymbol{u}^{n-1}$. The coefficient β is affected by the current time step size $\Delta t = (t^n - t^{n-1})$ as $\beta = \frac{\mu_0}{\Delta t}$, where $\mu_0 = 4\pi \cdot 10^{-7}$ is the magnetic permeability of the

vacuum. While the dielectric material is modelled with a constant value for $\alpha = 10^{-3}$, the superconducting material behaviour is modelled with a stiff nonlinear dependence of the resistivity α with the solution as $\alpha = \alpha_0 \left(\frac{\|\boldsymbol{\nabla}\times\boldsymbol{u}\|}{J_c}\right)^m$, with $m = 100$, $J_c = 10^{-8}$ and $\alpha_0 = 10^{-12}$. The equivalence with Eq. (3.1) is completed by considering the source term $\boldsymbol{f} = \beta \boldsymbol{u}^{n-1}$ and the strong imposition of an external magnetic field $\boldsymbol{u}^n \times \boldsymbol{n} = \boldsymbol{u}_0^n$ over the whole boundary. For the time step t^n, the weak form of the nonlinear problem reads: find $\boldsymbol{u}^n \in X_h$ such that

$$(\alpha(\boldsymbol{u}^n)\boldsymbol{\nabla}\times\boldsymbol{u}^n, \boldsymbol{\nabla}\times\boldsymbol{v}) + \beta(\boldsymbol{u}^n, \boldsymbol{v}) = \beta(\boldsymbol{u}^{n-1}, \boldsymbol{v}) \quad \forall \boldsymbol{v} \in X_h. \tag{3.36}$$

In order to derive the linearized form with Newton's method we consider the current approximation $\boldsymbol{u}^{n,k}$ and a (small) correction $\delta\boldsymbol{u}^{n,k}$ for the iterate k such that $\boldsymbol{u}^{n,k+1} = \boldsymbol{u}^{n,k} + \delta\boldsymbol{u}^{n,k}$. We plug the expression in Eq. (3.36), consider a first order Taylor expansion of $\alpha(\boldsymbol{u}^{n,k+1})$ around $\boldsymbol{u}^{n,k}$ and neglect the quadratic terms with respect to $\delta\boldsymbol{u}^{n,k}$, which yields the linearized problem: find $\delta\boldsymbol{u}^{n,k} \in X_h$ such that

$$\mathcal{J}(\boldsymbol{u}^{n,k}, \delta\boldsymbol{u}^{n,k}, \boldsymbol{v}) = -\mathcal{R}(\boldsymbol{u}^{n-1,k}, \boldsymbol{u}^{n,k}, \boldsymbol{v}) \qquad \forall \boldsymbol{v} \in X_h, \tag{3.37}$$

where

$$\begin{aligned}\mathcal{J}(\boldsymbol{u}^{n,k}, \delta\boldsymbol{u}^{n,k}, \boldsymbol{v}) &= (\alpha(\boldsymbol{u}^{n,k})\boldsymbol{\nabla}\times\delta\boldsymbol{u}^{n,k}, \boldsymbol{\nabla}\times\boldsymbol{v}) + \beta(\delta\boldsymbol{u}^{n,k}, \boldsymbol{v}) + \\ &\quad + (\alpha'(\boldsymbol{u}^{n,k})\delta\boldsymbol{u}^{n,k}\boldsymbol{\nabla}\times\boldsymbol{u}^{n,k}, \boldsymbol{\nabla}\times\boldsymbol{v}), \qquad (3.38a)\\ \mathcal{R}(\boldsymbol{u}^{n-1,k}, \boldsymbol{u}^{n,k}, \boldsymbol{v}) &= -\beta(\boldsymbol{u}^{n-1}, \boldsymbol{v}) + \beta(\boldsymbol{u}^{n,k}, \boldsymbol{v}) + (\alpha(\boldsymbol{u}^{n,k})\boldsymbol{\nabla}\times\boldsymbol{u}^{n,k}, \boldsymbol{\nabla}\times\boldsymbol{v}). \qquad (3.38b)\end{aligned}$$

The reader is directed to Chapter 4 for further details regarding the linearization procedure. Therefore, the rPB-BDDC preconditioner is applied to the linearized problem Eq. (3.37) at every nonlinear iteration of every time step. In this section, we will focus on the performance of the linear solver.

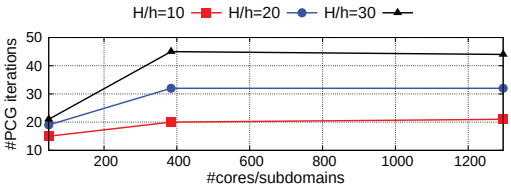
The problem is solved in $\Omega = [0, 40]$ mm^3, composed by an outer dielectric Ω_{air} material which includes a concentric superconducting cube Ω_{hts} of size 10 mm such that $\Omega = \Omega_{\text{hts}} \cup \Omega_{\text{air}}$. There is no source term and Dirichlet-type boundary conditions are imposed over the entire boundary as the time-dependent magnetic field $\boldsymbol{u}_{ext} = \frac{B_0}{\mu_0}[0, 0, \sin(2\pi\omega t)]$, where $B_0 = 200$ mT and $\omega = 50$ Hz. We solve the problem in the time interval $[0, 5]$ ms, which corresponds to a quarter of a full cycle in the applied $\boldsymbol{u}_{ext}$. Initial conditions are simply $\boldsymbol{u}^0 = \boldsymbol{0}$. The partition Θ^r_{pb} is obtained in all simulations for $r = 10^2$. The nonlinear scheme is stopped when the L_2-norm of the nonlinear residual (Eq. (3.38b)) is below 10^{-4}, while the convergence criteria for the rPB-BDDC preconditioned linear solver is the reduction of the initial L_2-norm of the residual of the linearized system by 10^{-8}.

We first present weak scalability results for the first set-up and solve with the rPB-BDDC preconditioner in Fig. 3.17, i.e., the first linearized problem (Eq. (3.37))

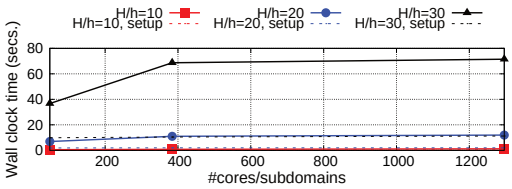
P	# Average iters.	Average size(A_c)	size(A_c) ratio
49	16.8	196.7	$1.31c_0$
385	18.9	2424.5	$2.02c_0$
1297	21.7	7728.7	$1.91c_0$

TABLE 3.3: Average metrics for the simulation of the time interval $T = [0, 5]$ ms. c_0 denotes the num. of coarse DOFs of the original, geometrical partition.

for the first time step. We include results for $\frac{H}{h} = \{10, 20, 30\}$. As expected, the method shows good weak scalability properties in number of iterations (see Fig. 3.17a) and computing times (see Fig. 3.17b).



(A) Number of iterations.



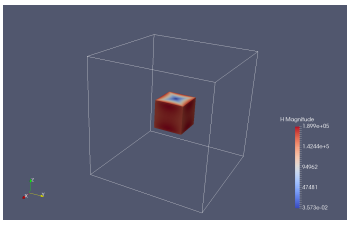
(B) Wall clock time for the total solver and only the set-up phase.

FIGURE 3.17: Weak scalability for the first linear solver in the HTS problem with $r = 10^2$.

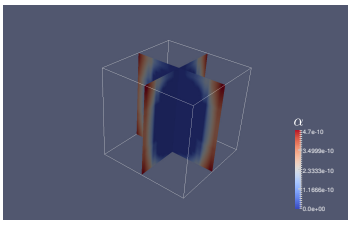
Next, we present average counters for the total number of linear solver applications for the simulation of the whole time interval $[0, 5]$ ms in Table 3.3. The resulting aggregation of cells into subsets based on their physical coefficient α (see Sect. 3.4.4) for $t = 4$ ms is depicted in Fig. 3.18c. We can identify two main regions in the distribution of α (see Fig. 3.18b): an inner region that is still not magnetized (i.e., with null resistivity) and a surrounding region, separated by a thin layer. Therefore, the selected value for r allows us to capture the behaviour of the different regions in Ω_{hts}. Out of the results in Tab. 3.3, scalability in the average number of iterations is observed. Besides, we show how the coarse problem size is only (approximately) doubled regarding to the size that would be obtained with the partition Θ instead of the Θ_{pb}.

3.6 Conclusions

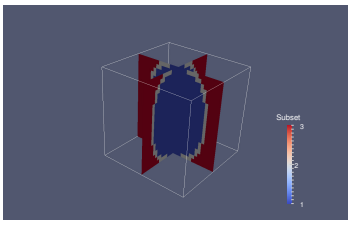
In this chapter, we have proposed an extension of the BDDC preconditioners for arbitrary order curl-conforming spaces that are robust for heterogeneous problems with high contrast of coefficients. The main idea is to enrich the continuity constraints enforced among subdomains (i.e., coarse DOFs) for those which contain high contrast of coefficients. The approach, which is shown to be robust for the grad-conforming case in [16], makes use of the knowledge about the physical coefficients to define a sub-partition of the original edge FE-based definition of coarse objects (edges and faces). The motivation for



(A) Magnetic field strength in the HTS device surrounded by a dielectric box, for which only the outline is depicted.



(B) Distribution of α in the HTS device.



(C) Subsets arising by the rPB-partition with $r = 10^2$ in the HTS device.

FIGURE 3.18: Domain and HTS device for $t = 4$ ms.

that is the well-known robustness of DD methods when there are only jumps of physical coefficients across the interface between subdomains. However, our case is more complex than the one in [16] for Poisson and elasticity problems, since two different coefficients are involved in the time-domain quasi-static approximation to the Maxwell's equations. Our solution is to add a perturbation term to the preconditioner so as we recover a scenario similar to the one in which only one coefficient jumps across the interface. A relaxed definition of the PB subdomains, where we only require that the maximal contrast of the two physical coefficients is smaller than a predefined thresholds, allows one to extend the range of applicability of the preconditioner to truly heterogeneous materials. Our preconditioners, which use the crucial change of variables in [137] to obtain weakly scalable algorithms for problems in $H(\text{curl})$ with few modifications to the standard BDDC algorithm in [57], are empirically shown to be robust with the contrast of coefficients. We would like to remark that our preconditioners maintain the simplicity of the standard BDDC and do not require to solve any eigenvalue or auxiliary problem.

We devoted a section to describe all the non-trivial implementation issues behind the method based on our experience through the implementation of the preconditioners in `FEMPAR`. Its task-overlapping implementation of the PB-BDDC preconditioner allows one to mask the computing times for the coarse problem, as long as they do not exceed local solvers time. With such implementation, we have been able to provide notable weak scalability results in the application of our new preconditioners to a wide range of multi-material and heterogeneous electromagnetics problems, including realistic 3D problems where coefficients can be defined by arbitrary functions, even dependent on the solution itself. In the future, the multilevel extension of the algorithm is expected to push forward the limits of its scalability properties.

Chapter 4

Simulation of High Temperature Superconductors and experimental validation

In this chapter, we present a parallel, fully-distributed finite element numerical framework to simulate the low-frequency electromagnetic behaviour of superconducting devices, which efficiently exploits high performance computing platforms. We select the so-called H-formulation, which uses the magnetic field as a state variable. Nédélec elements (of arbitrary order) are required for an accurate approximation of the H-formulation for modelling electromagnetic fields along interfaces between regions with high contrast medium properties. An h-adaptive mesh refinement technique customized for Nédélec elements leads to a structured fine mesh in areas of interest whereas a smart coarsening is obtained in other regions. The composition of a tailored, robust, parallel nonlinear solver completes the exposition of the developed tools to tackle the problem. First, a comparison against experimental data is performed to show the availability of the finite element approximation to model the physical phenomena. Then, a selected state-of-the-art 3D benchmark is reproduced, focusing on the parallel performance of the algorithms.

4.1 Introduction

High Temperature Superconductors (HTS) devices possess a number of unique properties that make them attractive for its use in a wide range of engineering applications. In order to design and optimize devices using superconducting tapes or bulks, computational tools are a powerful technique to simulate its electromagnetic behaviour by solving the system of partial differential equations (PDEs) that governs the problem. In this context, finite element (FE) methods are commonly used because they can handle complicated geometries whilst providing a rigorous mathematical framework. However, the electromagnetic modelling of superconductors at low frequencies is an extremely challenging simulation process that stresses many aspects of a numerical code such as multiphysics modelling, multiscale modelling, highly nonlinear behaviour, and a large number of time steps involved. Hence an appropriate definition of the formulation, the FE method, and

the solver will play a crucial role in order to obtain meaningful results in a reasonable amount of time.

Many formulations exist for the eddy-current problem [48]. These formulations can be mainly classified into three kinds, named after the variables used in the system of PDEs that one aims to solve: the A-V formulation [79, 94, 51], which is based on the magnetic vector potential, the T-Φ formulation [82, 10, 136], which is based on the current vector potential, and the H-formulation [119, 64, 80, 150], which is based directly on the magnetic field. Additionally, the mixed H-φ-Ψ formulation [135, 90] of the FE method uses cohomology basis functions in the dielectric region and allows to treat the air as an exact zero conductivity region. An alternative approach to the FE method is the variational method, which is valid for any electric field-current density relation and exists for several formulations: the H-formulation [38, 62, 27], the effective magnetization T-formulation [116] and the J-ϕ formulation [122, 118, 123, 103]. Other accurate results have been achieved with circuit models, such as those in [142] for calculating alternating currrents (AC) losses. In this chapter, we have selected the most common and widespread formulation, which is the H-formulation. The H-formulation provides the direct solution to the magnetic vector field, and has the advantage of dealing with boundary conditions in the model in a simple way. External magnetic fields can be applied directly by setting boundary values of the magnetic field, while currents in the superconductor device can be injected through Ampère's law. We stress, however, that the techniques presented here are also applicable with minor adaptations to other formulations as well.

In this chapter, the FE discretization of the magnetic field relies on the curl-conforming edge (or Nédélec) element of arbitrary order (see Chapter 2). Edge elements are preferred over grad-conforming Lagrangian ones, since they facilitate the modelling of the field near singularities by allowing normal fields components to jump across interfaces between two different media with highly contrasting properties [102]. In general, Lagrangian (nodal) elements with a weak imposition of the divergence constraint can converge to singular solutions for homogeneous problems (see, e.g., [14]), but these methods are not robust for heterogeneous problems like the ones in HTS modelling.

The high complexity of the problem at hand (i.e., modelling of superconducting devices with surrounding air or dielectric material regions) certainly requires customized solutions for every step in the simulation pipeline, namely mesh generation, discretization of the PDE system at hand and solution of the nonlinear system arising from discretization. With regard to mesh generation, computational cost may become rapidly expensive as we have to mesh not only the region of interest (superconducting device and immediate surrounding) but also the entire dielectric region. Common practice in the H-formulation FE modelling is to choose the latter region to be large enough such that interferences between the external applied magnetic field and the magnetic field generated by the superconducting device are avoided. This may imply that a large number of degrees of freedom (DOFs) are used on the mesh cells covering the dielectric region, while only a small portion of these DOFs might be actually needed for the accurate approximation of

the magnetic field on this region. The most immediate approach found in the literature to tackle this issue consists in the usage of *conforming* unstructured meshes with variable size cells, where the mesh cells are coarsened as the boundary of the dielectric domain is approached. The usage of a more fitted (to the superconductor) dielectric domain has been used as well in [78]. In such a case, one has to take into account self-generated magnetic fields while imposing external magnetic field boundary conditions. Finally, a more involved approach is the employment of cohomology basis functions, which allows to significantly reduce the number of DOFs in the dielectric region [135, 90]. Alternatively, methods taking the current density as state variable may reduce the number of DOFs, since only the sample volume is taken into account. For 2D problems, this has been done by the variational method in J formulation [122], integral methods [40, 39] and circuit methods [143]. In this thesis, a adaptive mesh refinement (AMR) strategy for Nédélec elements has been implemented, see Chapter 2. Using AMR, we can introduce an aggressive coarsening of the mesh in the dielectric region, whereas a fine mesh is achieved in the superconducting device. On the other hand, one can start with a very coarse (possibly unstructured) *conforming* mesh that represents the geometry at hand, drastically reducing the mesh generation computational cost. The AMR strategy is based on octree-based meshes, which can be compactly represented and efficiently manipulated in high-end distributed-memory computers. In this work, the `p4est` [47] MPI library is used for such purpose. For the solution of the nonlinear problem at every time step, we use the Newton-Raphson solver. At each nonlinear iteration, the resulting linear system is solved by means of preconditioned Krylov subspace iterative solvers [125]. An efficient preconditioner is crucial for their robustness and (parallel/algorithmic) scalability. In Chapter 3, we introduced the so-called balancing domain decomposition by constraints (BDDC) preconditioning approach [57, 18, 23]. For the problem at hand, we propose a curl-conforming BDDC preconditioner equipped with the coarse space presented in [137] for finite element tearing and interconnect (FETI)-fual-primal (DP) methods, and the approach in [16] to deal with high jumps of material properties.

Research on the simulation of the HTS problem has typically focused on the application side, and has considered moderate scale test cases with commercial software implementations of linear (at most quadratic) edge elements; see, e.g., [150, 84, 6]. On the other hand, 3D problems are of high interest in HTS modelling [94, 82, 150, 117, 68], but far from being at the maturity level as one can find in 2D, due to their high computational complexity and the poor (parallel) scalability of commercial software. Indeed, for large-scale FE 3D simulations, the efficient exploitation of High Performance Computing (HPC) resources becomes a must for providing a reasonable time-to-solution. In this context, the current work goes one step beyond by proposing a *parallel*, *fully-distributed* simulation software pipeline for the electromagnetic behaviour of HTS based on state-of-the-art numerical techniques for every building block.

The proposed algorithms in this chapter are available in the scientific software `FEMPAR`

(see Chapter 1). Therefore, this work also aims to introduce `FEMPAR` to the HTS modelling community as a new and powerful HPC tool for their simulations. In order to show its applicability, a validation with the Hall probe mapping experiment [78] is performed, obtaining a good agreement between the simulation results and the experimental data. In order to show the benefit of the proposed *fully-parallel* simulation software, a selected state-of-the-art 3D benchmark [85] is reproduced with excellent time-to-solution reductions on a massively parallel supercomputer.

The outline of the chapter is as follows. The problem is defined in Sect. 4.2 and some notation is introduced. The FE approximation of the problem is developed in Sect. 4.3. In Sect. 4.3.3, we present advanced mesh generation techniques customized for our model problem. In Sect. 4.4, we present a customized nonlinear parallel solver suitable for the problem at hand. We present a detailed set of numerical experiments in Sect. 4.5, which include a validation phase against experimental data and the reproduction of a selected benchmark, together with a strong scaling analysis. Finally, some conclusions are drawn in Sect. 4.6.

4.2 The system of equations

4.2.1 Notation

In this section, we introduce the problem to be solved and its particularities. Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with $d = 2, 3$ the space dimension. Let us denote by $L^2(\Omega)$ the space of square integrable functions. Furthermore, we will make use of the space

$$\mathcal{H}(\mathbf{curl}, \Omega) := \{\boldsymbol{v} \in L^2(\Omega)^d \mid \nabla \times \boldsymbol{v} \in L^2(\Omega)^d\}, \tag{4.1}$$

and its subspace

$$\mathcal{H}_0(\mathbf{curl}, \Omega) := \{\boldsymbol{v} \in \mathcal{H}(\mathbf{curl}, \Omega) \mid \boldsymbol{n} \times \boldsymbol{v} = \mathbf{0} \text{ in } \partial\Omega\}, \tag{4.2}$$

where $\boldsymbol{n}$ denotes the outward unit normal to the boundary of the domain Ω. In the sequel, bold characters will be used to describe vector functions or tensors, while regular ones will determine scalar functions or values. No difference is intended by using upper-case or lower-case letters for functions. Calligraphic letters are used to describe functional spaces and bold calligraphic letters will denote bilinear operators.

4.2.2 Maxwell equations in electromagnetics

Let us first state the Maxwell equations, which physically describe magnetostatics. Let us consider $\Omega \subset \mathbb{R}^d$ to be a simply connected nonconvex polyhedral domain with a

connected Lipschitz continuous boundary $\partial\Omega$. The differential Maxwell equations read

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t}, \qquad \text{Maxwell-Faraday equation} \tag{4.3}$$

$$\nabla \times \boldsymbol{B} = \mu_0 \boldsymbol{J}, \qquad \text{Ampère's circuital law} \tag{4.4}$$

$$\nabla \cdot \boldsymbol{B} = 0, \qquad \text{Gauss's law for magnetism} \tag{4.5}$$

$$\nabla \cdot \boldsymbol{E} = \frac{\varrho}{\varepsilon_0}, \qquad \text{Gauss's law} \tag{4.6}$$

in $\Omega \times (0, T]$, where $\boldsymbol{E}$ is the electric field, $\boldsymbol{B}$ is the magnetic field, $\boldsymbol{J}$ is the electric current density, μ_0 is the magnetic permeability of the vacuum, ε_0 is the vacuum permittivity and ϱ is the electric charge density. This form of the equations is valid for negligible displacement current. Furthermore, we add the constitutive law that specifies the (possibly nonlinear) relationship between the electric field and the current density in a material by

$$\boldsymbol{E} = \rho \boldsymbol{J}, \qquad \text{Ohm's law} \tag{4.7}$$

$\rho > 0$ being the material resistivity tensor (inverse of conductivity). In this chapter, we restrict ourselves to non-magnetic media since we consider the constitutive law $\boldsymbol{B} = \mu_0 \boldsymbol{H}$ for magnetic fields.

4.2.3 The H-formulation

After some trivial manipulation of Eqs. (4.3)-(4.4) and the substitution of the constitutive law in Eq. (4.7), one can obtain the so-called H-formulation for the magnetic field $\boldsymbol{H}$. The proposed formulation reads: seek a magnetic field $\boldsymbol{H}$ solution of

$$\frac{\partial \mu_0 \boldsymbol{H}}{\partial t} + \nabla \times \rho \nabla \times \boldsymbol{H} = \boldsymbol{f} \qquad \text{in } \Omega \times (0, T], \tag{4.8}$$

where $\boldsymbol{f}$ is a solenoidal given source term. Taking the divergence of Eq. (4.8), and given that $\boldsymbol{H}$ is solenoidal at the initial time, follows that $\boldsymbol{H}$ is solenoidal for every time. Besides, Eq. (4.8) needs to be supplied with appropriate boundary and initial conditions. The boundary of the domain $\partial\Omega$ is divided into its Dirichlet boundary part, i.e., $\partial\Omega_D$, and its Neumann boundary part, i.e., $\partial\Omega_N$, such that $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \cap \Omega_N = \emptyset$. Then, boundary and initial conditions for the problem at hand read

$$\boldsymbol{H} \times \boldsymbol{n} = \boldsymbol{g} \qquad \text{on } \partial\Omega_D \times (0, T], \tag{4.9}$$

$$\boldsymbol{n} \times (\rho \nabla \times \boldsymbol{H}) = 0 \qquad \text{on } \partial\Omega_N \times (0, T], \tag{4.10}$$

$$\boldsymbol{H}(\boldsymbol{x}, t = 0) = \boldsymbol{0} \qquad \text{in } \Omega. \tag{4.11}$$

Note that Dirichlet boundary conditions prescribe the tangent component of the magnetic field on the boundary of the domain, while Neumann boundary conditions prescribe the tangent component of the electric field $\boldsymbol{E}$ (see Eq. (4.7)). Finally, the variational

form of the H-formulation reads as follows: find $\boldsymbol{H} \in \mathcal{H}(\mathbf{curl}, \Omega)$ such that

$$(\frac{\partial \mu_0 \boldsymbol{H}}{\partial t}, \boldsymbol{v}) + (\rho \nabla \times \boldsymbol{H}, \nabla \times \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}), \qquad \forall \boldsymbol{v} \in \mathcal{H}_0(\mathbf{curl}, \Omega). \tag{4.12}$$

4.2.4 Transmission conditions

Natural boundary conditions appear on the formulation after integrating by parts Eq. (4.12):

$$\int_\Omega (\nabla \times \rho \nabla \times \boldsymbol{H}) \cdot \boldsymbol{v} = \int_\Omega (\rho \nabla \times \boldsymbol{H}) \cdot (\nabla \times \boldsymbol{v}) - \int_{\partial \Omega_N} (\rho \nabla \times \boldsymbol{H}) \cdot (\boldsymbol{n} \times \boldsymbol{v}), \tag{4.13}$$

where we can identify the condition $\boldsymbol{n} \times (\rho \nabla \times \boldsymbol{H})$ to be introduced in the Neumann boundary $\partial \Omega_N$. Consider now two different non-overlapping regions on the domain Ω corresponding to two different media, namely Ω_1 and Ω_2 such that $\Omega = \Omega_1 \cup \Omega_2$ and $\Gamma = \Omega_1 \cap \Omega_2$. Let us denote by $\{\boldsymbol{n}_{\Gamma_1}, \boldsymbol{n}_{\Gamma_2}\}$ the unit normal pointing outwards of $\{\Omega_1, \Omega_2\}$. Clearly, $\boldsymbol{n}_{\Gamma_1} = -\boldsymbol{n}_{\Gamma_2}$ and we state the natural interface conditions (*transmission conditions*) for Eq. (4.8) as

$$\boldsymbol{n} \times (\rho_1 \nabla \times \boldsymbol{H}_1 - \rho_2 \nabla \times \boldsymbol{H}_2) = 0 \qquad \text{on } \Gamma, \tag{4.14}$$

where $\boldsymbol{n}$ holds in this case for $\boldsymbol{n} = \boldsymbol{n}_{\Gamma_1}$. Note that, with no other sources, Eq. (4.14) enforces the continuity of the tangent component of the electric field $\boldsymbol{E}$ over the interface, i.e., $\boldsymbol{n} \times (\boldsymbol{E}_1 - \boldsymbol{E}_2) = 0$. For the problem at hand, the domain will be composed of an HTS device Ω_{hts} and a surrounding dielectric region Ω_{air}.

On the other hand, currents in the superconductor device are injected through Ampère's circuital law, Eq. (4.4), in a closed surface S as (by the Stokes theorem)

$$\int_S (\nabla \times \boldsymbol{H}) \cdot \boldsymbol{n} = \oint_{\partial S} \boldsymbol{H} \cdot \boldsymbol{\tau} = I_{\text{app}}, \tag{4.15}$$

where now $\boldsymbol{n}$ denotes the unit normal pointing outwards to the surface S defined by a section of Ω_{hts} (the domain itself in a 2D case). On the other hand, $\boldsymbol{\tau}$ is the unit tangent to the surface boundary. The scalar value I_{app} is the net current enforced in the superconductor in the perpendicular direction to the surface.

The final HTS problem will read as follows: find $\boldsymbol{H} \in \mathcal{H}(\mathbf{curl}, \Omega)$ such that Eq. (4.12) holds in Ω_{hts} and Ω_{air}, together with the transmission condition (4.14) and the constraint (4.15).

4.2.5 Material modelling

In previous sections, a suitable H-formulation (applicable to general electromagnetics) has been presented. In this section, the general formulation is extended to superconductivity by means of the constitutive law definition that relates current densities and

electric fields. We consider a dielectric domain Ω_{air} large enough for neglecting boundary effects associated to the magnetization of the superconductor. In order to model its non-conducting behaviour, we consider a conductivity (inverse of resistivity) value that ideally tends to 0. However, the dramatic jump of resistivity on the interface introduces a boundary layer on the interface that would require huge computational resources to be captured, whereas we are mainly interested in the superconductor behaviour. Thus, it is common practice to consider a fixed value for the resistivity in Ω_{air} that, whilst maintaining a large magnitude ratio with regard to the superconducting material, allows the computation to take place with a desired level of precision. On the other hand, a nonlinear electric field-current density relation is used in Ω_{hts} to describe the penetration of the magnetic flux and induced currents. For describing $\boldsymbol{E}(\boldsymbol{J})$, we will use the power law

$$\boldsymbol{E} = \frac{E_c}{J_c} \left(\frac{\|\boldsymbol{J}\|}{J_c} \right)^n \boldsymbol{J}, \tag{4.16}$$

where E_c is the critical electric field, J_c is the critical current density, and n is the exponent of the power law. This kind of expression tends to the analytical Bean's model [32] when n tends to infinity. One can identify Ohm's law (Eq. (4.7)) in this $\boldsymbol{E}$-$\boldsymbol{J}$ relation, with the following expression for the resistivity parameter in the superconducting region:

$$\rho_{\text{hts}}(\boldsymbol{H}) = \frac{E_c}{J_c} \left(\frac{\|\nabla \times \boldsymbol{H}\|}{J_c} \right)^n. \tag{4.17}$$

In turn, J_c may be considered a fixed current density value, or a value dependent on the magnetic field (magnitude and direction). For instance, for the magnetization of type-II superconductors, the Kim's model [86] introduces a dependence on the magnetic field strength,

$$J_c(\boldsymbol{B}) = \frac{J_{c0} B_0}{B_0 + \mu_0 \|\boldsymbol{H}\|}, \tag{4.18}$$

where J_{c0} and B_0 are parameters determined by the physical properties of the superconducting material.

4.3 Numerical approximation

For the FE approximation of the problem, we select the edge (Nédélec) hexahedral element of arbitrary order. Nevertheless, we do not go into detail in this section for conforming FE spaces with respect to $\mathcal{H}(\mathbf{curl}, \Omega)$, since it has been deeply covered in Chapter 2. Therefore, we present the remaining ingredients of the numerical approximation in order to discretize, in space and time, the problem at hand.

4.3.1 Time discretization

Let us consider a partition of the time interval $[0, T]$ into N time slabs. We denote the n-th time slab by $\Delta t^n = (t^{n-1}, t^n]$, for $n = 1, \ldots, N$. We also denote each time slab size by $|\Delta t^n|$. Time integration is performed with a θ-method, even though the use of other time integrators is straightforward. For the sake of clarity, we use the Backward-Euler (BE) time integration in the presentation of the method. The already discretized in time problem reads: Given $\boldsymbol{H}(t^0) = \boldsymbol{0}$, find at every time step $n = 1, \ldots, N$ the solution $\boldsymbol{H}_h^n \in \mathcal{ND}_k$ such that

$$\frac{\mu_0}{|\Delta t^n|}(\boldsymbol{H}_h^n, \boldsymbol{v}_h) + (\rho(\boldsymbol{H}_h^n)\nabla \times \boldsymbol{H}_h^n, \nabla \times \boldsymbol{v}_h) \tag{4.19}$$

$$= (\boldsymbol{f}_h^n, \boldsymbol{v}_h) + \frac{\mu_0}{|\Delta t^n|}(\boldsymbol{H}_h^{n-1}, \boldsymbol{v}_h), \qquad \forall \boldsymbol{v}_h \in \mathcal{ND}_k \tag{4.20}$$

where $\boldsymbol{f}_h^n$ is the discrete version of the source term $\boldsymbol{f}$ evaluated at time t^n. Note that non-homogeneous initial conditions are readily imposed by considering $\boldsymbol{H}_h^0 = \boldsymbol{H}_h(t^0)$, where $\boldsymbol{H}_h(t^0)$ is the interpolation of the initial value onto the FE space $\mathcal{ND}_k$.

4.3.2 Dirichlet Boundary Conditions

The H-formulation is preferred over other formulations, among other reasons, due to its straightforward manner of handling magnetic fields and currents. External magnetic fields can be applied directly by setting boundary values of the magnetic field on Dirichlet boundaries, while currents in the superconductor device can be injected through Ampère's law (see Eq. (4.4)) through constraints. Dirichlet boundary conditions will be strongly imposed on the resulting system (usual implementation in FE codes), hence we need the DOF values over the boundary to be fixed. $\boldsymbol{H}_h$ DOFs are obtained by means of moments defined over edges, faces, and cells. As a result, one has to use the corresponding Nédélec FE interpolator, which consists in evaluating the moments (see Chapter 2) for the continuous boundary function.

4.3.3 Mesh generation by hierarchical AMR

Uniform refinement is not a choice for problems that exhibit localized phenomena and multiscale features, since the size of the resulting system may rapidly become prohibitive. The purpose of any mesh adaptive method is to achieve a high degree of accuracy in areas of the domain of particular interest while saving computational efforts in other areas. To this end, the mesh is refined in regions of the domain that present a complex behaviour of the solution. AMR is mostly used with *a posteriori* error estimate that determines the accuracy of the solution in every mesh cell and requires to know the solution of the PDE at every iteration (dynamic AMR). However, the areas of interest in the problem at hand are located in an *a priori* known region of the domain, the superconductor region,

therefore we can leverage the AMR technique presented in Chapter 2 with a refined static mesh approach. Let us clarify this approach with the problem at hand.

Superconducting regions of the domain and immediate surroundings show high values for the gradients of the solution (i.e., high variation of the solution in small regions of the domain), while a smoother solution is expected in a vast majority area of the dielectric region. Besides, quantities of interest (e.g., AC losses) are computed on Ω_{hts}, hence we are mainly interested in taking control over the degree of accuracy that can be achieved there. On the other hand, the ratio between the volume of the superconductor material and the dielectric region is very low and thus saving efforts in Ω_{air} is crucial to obtain results in a reasonable amount of time. At this point, we should stress that we restrict ourselves to h-adaptivity, i.e., only the mesh is adapted, in contrast to the so-called hp-adaptivity, where the polynomial order p of the FEs (see Sect. 4.3) may also be adapted, and thus vary among mesh cells. In this chapter, we explore the usage of hierarchically refined octree-based hexahedral meshes in the discretization of the HTS problem.

4.4 Nonlinear transient solver

HTS modelling requires not only a reliable constitutive model but also a robust and efficient nonlinear solver. Superconductor phenomena may occur in a very short time period and thus abrupt changes of behaviour are found in a very small time scale. Furthermore, the nonlinearity associated to the constitutive law $\boldsymbol{E} - \boldsymbol{J}$ presents extreme parameters ($n \sim 30 - 100$). Spatial scales, time scales, and nonlinearity make superconductivity modelling a very challenging task from a computational point of view. The nature of the nonlinearity is very stiff, since the exponent in (4.16) usually takes high values, which makes the Lipschitz continuity constant of the nonlinear PDE operator at hand large. For this purpose, a specific nonlinear transient solver is proposed in this section.

4.4.1 Algebraic form

For the sake of clarity in forthcoming sections, let us write the problem in algebraic form. The magnetic vector field $\boldsymbol{H}_h$ is expanded by means of the vector shape functions $\{\boldsymbol{\phi}^i\}_{i=1}^{N_H}$ related to the curl-conforming edge element. Let us define the following element-wise matrices in the element K:

$$\boldsymbol{\mathcal{M}}_K := \sum_{i=1}^{N_H}\sum_{j=1}^{N_H}\int_K \boldsymbol{\phi}^i \cdot \boldsymbol{\phi}^j \tag{4.21}$$

$$\boldsymbol{\mathcal{K}}_K := \sum_{i=1}^{N_H}\sum_{j=1}^{N_H}\int_K \rho(\boldsymbol{H}_h)(\nabla \times \boldsymbol{\phi}^j) \cdot (\nabla \times \boldsymbol{\phi}^i) \tag{4.22}$$

Consider also the right-hand side discrete vector $\boldsymbol{F}^i_K := \int_K \boldsymbol{f}_K \boldsymbol{\phi}^i$. Then, the usual assembly is performed to obtain global matrices and arrays.

Once the operators have been introduced in algebraic form, the problem for a single time step t^n $(n > 0)$ in algebraic form reads:

$$\begin{bmatrix} \frac{\mu_0}{|\Delta t^n|}\boldsymbol{\mathcal{M}} + \boldsymbol{\mathcal{K}}(\boldsymbol{H}_h^n) & \boldsymbol{\mathcal{C}}^t \\ \boldsymbol{\mathcal{C}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{H}_h^n \\ \lambda^n \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_h(t^n) + \frac{\mu_0}{|\Delta t^k|}\boldsymbol{\mathcal{M}}\boldsymbol{H}_h^{n-1} \\ I_{\text{app}} \end{bmatrix}, \tag{4.23}$$

where $\boldsymbol{H}_h$ is the discrete function containing the DOF values for the magnetic field. λ^n is the Lagrange multiplier introduced to enforce the current constraint, and $\boldsymbol{\mathcal{C}}$ is the one-row matrix that enforces the current value I_{app} over a given closed surface S through Eq. (4.15), i.e., the application of a constraint over the solution

$$\boldsymbol{\mathcal{C}}\boldsymbol{H}_h^n = \sum_{i=1}^{N_H} \int_S \left(\nabla \times \boldsymbol{\phi}^i\right) \cdot \boldsymbol{n}\boldsymbol{H}_h^{n_i} = I_{\text{app}}.$$

4.4.2 Adaptive time stepping

Time scales may be very small in this problem due to the applied fields frequency. However, the process of magnetization of the superconductor allows to identify different needs in different periods of the process. Although restrictions in the time step size are severe in some periods of time (when $\|\boldsymbol{J}\|$ becomes larger than J_c in some region), the time step can be relaxed in monotone magnetization curves. The same effect occurs for the validation model that will be considered in Sect. 4.5.3, where an injected current is kept constant for a period of time before proceeding to the following current load increment. A simple adaptive time stepping will be used to accelerate convergence. The time step size is updated with

$$|\Delta t^n| = \frac{\kappa}{\#\text{iters}} |\Delta t^{n-1}|, \tag{4.24}$$

where κ stands for a selected growing ratio and #iters denotes the number of iterations to attain nonlinear convergence for the last converged time step. Usually, one may select κ as the "ideal" number of iterations to convergence sought in the nonlinear algorithm. Finally, the trial time step size is restricted to upper and lower bound values

$$|\Delta t^n| = \begin{cases} \Delta t_{\min} & \text{if } |\Delta t^n| \leq \Delta t_{\min}, \\ |\Delta t^n| & \text{if } \Delta t_{\min} \leq |\Delta t^n| \leq \Delta t_{\max}, \\ \Delta t_{\max} & \text{if } \Delta t_{\max} \leq |\Delta t^n|. \end{cases} \tag{4.25}$$

4.4.3 Linearization

The problem and its residual are stated in an algebraic form as

$$\boldsymbol{\mathcal{A}}(\boldsymbol{x})\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{R} = \boldsymbol{\mathcal{A}}(\boldsymbol{x})\boldsymbol{x} - \boldsymbol{b} = \mathbf{0}, \tag{4.26}$$

where the full vector of unknowns $\boldsymbol{x}$ and the right-hand side $\boldsymbol{b}$ have been presented in (4.23). It is essential to build a robust nonlinear solver together with an effective adaptive time stepping technique. Note that the resistivity takes a constant value ρ_{air} in the air region Ω_{air} hence the problem is linear in this part of the domain. However, a highly nonlinear problem is found in the superconductor region Ω_{hts}. Therefore, our strategies focus on the linearization of the problem associated to the extreme nonlinearity given by the resistivity $\rho_{\text{hts}}(\boldsymbol{H})$. For that purpose, we will make use of a composition of nonlinear solvers. Our nonlinear solver is the composition of a Newton-Raphson (NR) method with an exact derivation of the Jacobian and a Cubic Backtracking (CB) line search algorithm (see [46]).

By means of the NR method, we obtain (for the time step t^n) the direction of the solution update at the iterate k, i.e., $\delta\boldsymbol{x}^{n,k} = \boldsymbol{x}^{n,k+1} - \boldsymbol{x}^{n,k}$, solving the linearized problem for the current linearization point $(\boldsymbol{H}_h^{n,k}, \lambda^{n,k})$

$$\boldsymbol{\mathcal{J}}(\boldsymbol{x}^{n,k})\delta\boldsymbol{x}^{n,k} = -\boldsymbol{R}(\boldsymbol{x}^{n,k}). \tag{4.27}$$

Later, the BT technique tries to minimize the residual of the iterate $\boldsymbol{x}^{n,k+1} = \boldsymbol{x}^{n,k} + \beta\delta\boldsymbol{x}^{n,k}$ with the found direction $\delta\boldsymbol{x}^{n,k}$ by means of the step length β, i.e.,

$$\beta = \underset{0<\tilde{\beta}\leq 1}{\operatorname{argmin}} \|\boldsymbol{R}(\boldsymbol{x}^{n,k} + \tilde{\beta}\delta\boldsymbol{x}^{n,k})\|^2, \tag{4.28}$$

and the process is repeated until a convergence criterion is attained. It is not our intention to define neither the CB technique nor the basic NR algorithm, which can be found in [46], but we will introduce the expression of the application of the Jacobian operator $\boldsymbol{\mathcal{J}}$ that is specific to our formulation. To this end, let us first define the discrete residual of the resulting algebraic system evaluated at the point $(\boldsymbol{H}_h^{n,k}, \lambda^{n,k})$. For the sake of simplicity, time step and iterate indices $\{n, k\}$ will be omitted in the rest of the section, where expressions always refer to a concrete evaluation point. The component-wise definition of the residual $\boldsymbol{R}$, of the form $\boldsymbol{R}(\boldsymbol{H}_h, \lambda) = [\boldsymbol{R}^{\boldsymbol{H}_h}, R^{\lambda}]$, follows

$$\begin{aligned}\boldsymbol{R}^{H_h^i}(\boldsymbol{H}_h, \lambda) = &\sum_{j=1}^{N_H} \int_{\Omega_{\text{hts}}} \frac{\mu_0}{|\Delta t|}\boldsymbol{\phi}^j \cdot \boldsymbol{\phi}^i H_h^j + \sum_{j=1}^{N_H} \int_{\Omega_{\text{hts}}} \rho_{\text{hts}}(\boldsymbol{H}_h)(\nabla \times \boldsymbol{\phi}^j) \cdot (\nabla \times \boldsymbol{\phi}^i) H_h^j \\ &+ \int_S (\nabla \times \boldsymbol{\phi}^i) \cdot \boldsymbol{n}\lambda - \int_{\Omega_{\text{hts}}} \boldsymbol{f} \cdot \boldsymbol{\phi}^i,\end{aligned} \tag{4.29}$$

$$R^{\lambda}(\boldsymbol{H}_h) = \sum_{j=1}^{N_H} \int_S (\nabla \times \boldsymbol{\phi}^j) \cdot \boldsymbol{n} H_h^j - I_{\text{app}} \tag{4.30}$$

for the magnetic field DOFs $\{H_h^i\}_{i=1}^{N_H}$ and the Lagrange multiplier λ used to enforce the applied current I_{app} in the closed surface S. In this case, $\boldsymbol{n}$ denotes the unit normal to the surface Ω_{hts}. The application of the Jacobian to a given direction $\boldsymbol{z}$, i.e., $\boldsymbol{\mathcal{J}}(\boldsymbol{x}, \boldsymbol{z})$,

given the linearization point $\boldsymbol{x}$, i.e., $\boldsymbol{\mathcal{J}}(\boldsymbol{x})\boldsymbol{z}$, reads:

$$\boldsymbol{\mathcal{J}}(\boldsymbol{x})\boldsymbol{z} = D\boldsymbol{R}(\boldsymbol{x},\boldsymbol{z}) = \boldsymbol{\mathcal{A}}(\boldsymbol{x})\boldsymbol{z} + D\boldsymbol{\mathcal{A}}(\boldsymbol{x},\boldsymbol{z})\boldsymbol{x}. \tag{4.31}$$

where, e.g., $D\boldsymbol{R}(\boldsymbol{x},\boldsymbol{z})$ is the Gâteaux derivate of $\boldsymbol{R}$ at $\boldsymbol{x}$ in the direction of $\boldsymbol{z}$. Following the notation proposed in Sect. 4.4.1, the linearized $\boldsymbol{\mathcal{J}}(\boldsymbol{x})$ at each iterate $\{n,k\}$ can be stated as

$$\boldsymbol{\mathcal{J}}(\boldsymbol{x}) = \begin{bmatrix} \frac{\mu_0}{|\Delta t|}\boldsymbol{\mathcal{M}} + \boldsymbol{\mathcal{K}}(\boldsymbol{H}_h) + \dfrac{\partial \boldsymbol{\mathcal{K}}(\boldsymbol{H}_h)}{\partial \boldsymbol{H}_h}\boldsymbol{H}_h & \boldsymbol{\mathcal{C}}^t \\ \boldsymbol{\mathcal{C}} & \boldsymbol{0} \end{bmatrix}, \tag{4.32}$$

where the original operator $\boldsymbol{\mathcal{A}}$ can be directly identified. The entries for the block corresponding to the magnetic field read:

$$[\boldsymbol{\mathcal{J}}(\boldsymbol{H}_h)]_{ij} = \frac{\partial \boldsymbol{R}_i}{\partial \boldsymbol{H}_h^j} = [\boldsymbol{\mathcal{A}}(\boldsymbol{H}_h)]_{ij} + \int_{\Omega_{\text{hts}}} \frac{\partial \rho_{\text{hts}}(\boldsymbol{H}_h)}{\partial H_h^j}(\nabla \times \boldsymbol{H}_h)\cdot(\nabla \times \boldsymbol{\phi}^i), \tag{4.33}$$

where $\{i,j\} = 1,\cdots,N_H$, i.e., the magnetic field number of unknowns. It becomes clear in this expression that the nonlinearity is given by the discrete magnetic field $\boldsymbol{H}_h$ whereas the Lagrange multiplier enforcing the current is a linear relation. If we go one step further, for the constitutive law presented in Eq. (4.7), and considering J_c independent of the magnetic field, we obtain the expression for the *tangent* resistivity with respect to the magnetic field at the evaluation point $\boldsymbol{H}_h$:

$$\frac{\partial \rho_{\text{hts}}(\boldsymbol{H}_h)}{\partial H_h^j} = \frac{E_c}{J_c} n \left(\frac{\|\nabla \times \boldsymbol{H}_h\|}{J_c}\right)^{n-2} \frac{(\nabla \times \boldsymbol{H}_h)}{J_c} \cdot \frac{(\nabla \times \boldsymbol{\phi}^j)}{J_c}. \tag{4.34}$$

4.4.4 Parallel linear solver

The presented nonlinear solver (Sect. 4.4.3) can rely on either parallel sparse direct or iterative solvers to solve the linearized problems that arise at every nonlinear iteration. In contrast to sparse direct solvers, iterative solvers can be efficiently implemented in parallel computer codes for distributed-memory computers. However, they have to be equipped with an efficient preconditioner, which is crucial for their robustness and (parallel/algorithmic) scalability. In this chapter, we ground on the so-called BDDC preconditioning approach [57, 18, 23], which has been presented in Chapter 3 for curl-conforming problems, as a preconditioner for Krylov subspace iterative solvers [125]. In this section, we are interested in strong scaling[1], since we aim at reducing time-to-solution for a fixed problem size. For the sake of robustness, we have implemented the most conservative coarse space proposed in [137], called Alg. C in this reference, and the approach in [16] to deal with high jumps of material properties.

[1] Strong scaling is the ability of a parallel algorithm to reduce time-to-solution with increasing number of processors in the solution of a fixed problem size.

4.5 Numerical experiments

In this section, we test the h-adaptive FE approximation of the H-formulation and the parallel nonlinear solver presented in the previous sections. In general, the physical domains in our simulations consist of a superconducting bulk completely surrounded by a dielectric box (see Fig. 4.3 for an illustrative example). Dirichlet boundary conditions are applied on the boundary of the outer domain. Thus, we make sure that this external boundary is far enough from the superconducting device to avoid interior magnetic fields generated by itself reaching the boundary and interfere external applied fields. Unless otherwise stated, the stopping criterion for the nonlinear solver is the reduction of the ratio between the discrete L_2-norm of the nonlinear residual and right-hand side below 10^{-10}. At the same time, the stopping criterion for the iterative linear solver applied to each linearized time step is the reduction of the discrete L_2-norm of the relative residual below 10^{-12}. The adaptive time stepping algorithm in Sect. 4.4.2 will be used with $\kappa = 5$.

4.5.1 Software

All the algorithms described in this work are available in the scientific software `FEMPAR` (see Chapter 1). Regarding the content of this chapter, `FEMPAR` supports arbitrary order edge FEs on both hexahedra and tetrahedra, on either structured or unstructured *conforming* meshes (i.e., typically generated by an external mesh generator), and mesh generation and adaptation using hierarchically refined octree-based meshes. The serial and MPI-parallel versions of the process described in Sect. 4.3.3 are grounded on `p4est` [47]. `p4est` is an MPI library for efficiently handling (forest of) octrees on distributed-memory processors. Among others, it provides a set of octree manipulation primitives which are essential for our approach: (1) to adapt an octree by refining (or coarsening) its cells; (2) to redistribute the octree cells among the available processors for dynamic load-balancing (by means of space-filling curves [47]); and (3) to enforce the 2:1 balance ratio. Using a compact representation, `p4est` provides memory-efficient and scalable algorithms for all the aforementioned manipulation primitives [47]. On top of `p4est`, `FEMPAR` builds a richer representation of the mesh (essentially mesh cells and lower dimensional geometrical entities connectivity information) to support the implementation of adaptive FE methods using *hanging* DOFs constraints (see Sect. 4.3.3). Both the mesh, and the rest of data structures used in the simulation are *fully-distributed* among the processors involved in the parallel simulation. This implies, e.g., that each processor holds a partial portion of the global mesh cells, and a subset of the DOFs of the global FE space. It is essential to scale FE simulations to large core counts.

At the linear solver kernel, it offers several alternatives depending on the programming environment at hand. In this chapter, we used the ones described in the sequel. For small scale problems on, e.g., a Desktop computer, we rely on a parallel multi-threaded sparse direct solver available at Intel MKL PARDISO [1]. On the other hand, for a

hybrid OpenMP/MPI environment, linear solvers are based on preconditioned Krylov subspace solvers. `FEMPAR` hallmark is an abstract object-oriented (OO) framework for the implementation of widely applicable highly scalable multilevel domain decomposition (DD) solvers. The preconditioners which are accommodated within this framework require the solution of linear systems which are local to each subdomain, and the so-called coarse-grid problem, that is crucial for preconditioner efficiency and scalability. These problems are solved using the aforementioned sparse direct solver. Each MPI task in the parallel computation handles the computations to be performed at a single subdomain. Provided that the algorithm lets a high degree of overlapping to be achieved among fine and coarse-grid tasks, an additional MPI task is spawn in order to carry out coarse-grid-related computations; see, e.g., [18] for additional details. Our BDDC preconditioner implementation (see Chapter 3) can deal with curl-conforming spaces of arbitrary order, tetrahedral/hexahedral meshes and structured/unstructured partitions.

4.5.2 Experimental framework

The experiments in this section have been performed on the MareNostrum-IV [4] (MN-IV) supercomputer, hosted by the Barcelona Supercomputing Center (BSC). MN-IV is equipped with 3456 compute nodes connected together with the Intel OPA HPC network. Each node is equipped with 2x Intel Intel Xeon Platinum 8160 multi-core CPUs, with 24 cores each (i.e., 48 cores per node), and 96 GBytes of RAM. `FEMPAR` was compiled with the Intel Fortran compiler (v18.0.1) using system-recommended optimization flags, and linked against the Intel MPI Library (v2018.1.163) for message-passing, and the BLAS/LAPACK and PARDISO available on the Intel MKL library for optimized dense linear algebra kernels, and sparse direct solvers, respectively. All floating-point calculations were performed in IEEE double precision.

4.5.3 Comparison against experimental data

In this section, the proposed FE model is validated against experimental data, obtained by means of the Hall scanning magnetometer experiment, exposed in detail in [78]. Our goal is to compare the experimental data for a 2G tape sample (see Tab. 4.1 for properties) with the numerical results obtained with `FEMPAR`. The problem of a HTS tape magnetized by a current flowing through it has been solved by several authors (see, e.g., [109, 107, 114]). Besides, several comparisons between experimental and numerical results can be found, e.g., in [61, 70, 129].

In the experiment we consider that the current is applied in a HTS tape by a sequence of step functions, with time intervals of 100 seconds each before proceeding to the following increment. The current applied is gradually increased up to $I_{\text{app}} = 460$ A, which corresponds to $1.03 \cdot I_c$ (i.e., $I_c = 446.16$). See Fig. 4.1b for a clear exposition of the injected current. The applied current remains constant during short periods of

Parameter	Comparison to experimental data	3D benchmark	Units
air domain size	100×100	$100 \times 100 \times 100$	mm^d
HTS width	12	10	mm
HTS thickness	110	1000	μmm
n power law exponent	32	24	
μ_0	$4\pi \cdot 10^{-7}$	$4\pi \cdot 10^{-7}$	H/m
μ_r	1.0	1.0	H/m
E_0	10^{-4}	10^{-4}	V/m
J_c	$3.38 \cdot 10^8$	$1.0 \cdot 10^8$	A/m^2
I_c	446.16	$1.0 \cdot 10^3$	A
ρ_{air}	1.0	10^{-2}	Ω·m

TABLE 4.1: Geometric and electrical parameters of the HTS tape used in different problems. Domain units depend on domain dimension d.

time so it allows the flux creep effects to pass, and therefore the current distribution is stabilized along the superconductor specimen.

We will compare the experimental and numerical profiles of the vertical component of the magnetic field (i.e., $B_y = \mu_0 H_y$) 400 μm above the HTS tape surface (where the active part of the sensor is located). For the computational model, we simplify the superconducting region as a homogenization of the multiple layers typically found in a 2G tape (e.g., the substrate, the silver, and the copper covering), where only a layer of 1-2 μm corresponds to superconducting material. As it is shown in this section, such approximation is accurate in the magnetic field computation at 400 μm above the tape surface. Following [78], a critical current dependence with the magnetic field, i.e., $J_c(\boldsymbol{B})$, is introduced through tabulated values (see Fig. 4.1a), where effective J_c values can be obtained at each nonlinear iteration from the corresponding linearization point $\boldsymbol{B}$. A Lift Factor (LF=J_c/J_{c0}) is obtained for each component B_i of $\boldsymbol{B}$, i.e., $LF_i(B_i)$, where $i \in \{1, 2\}$ in this 2D simulation (see Fig. 4.1a). Then, the resulting J_c is obtained as $J_{c0} \cdot \|LF\|$, where J_{c0} is the value of J_c in absence of magnetic field, i.e., $J_c(\mathbf{0})$, detailed in Tab. 4.1.

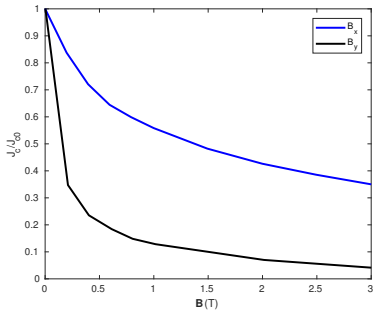
Fig. 4.2 shows the comparison between the experimental and numerical vertical magnetic field $\mu_0 H_y$. The numerical results are obtained for first order edge FEs. All plots in Fig. 4.2 show profiles along the x-axis section for the full HTS tape length and 6 additional mm to each side. Therefore, with the reference domain $\Omega = [0, 100] \times [0, 100]$ mm^2, these profiles correspond to the line $y = 50.455,\ 38 < x < 62$ mm. Out of these results, some conclusions can be drawn. First and most important, the experimental data is in good agreement with the performed simulations and therefore, the proposed formulation is able to reproduce the physical phenomena. Second, the magnetic field peak values are, in all cases, in excellent agreement. It is important to note that the experimental data does not possess symmetry, while the computed data respects such expected symmetry.

This fact can be attributed to small imperfections in the experiment. Out of the validation test, we can identify three phases. A first one, where the computed data reproduces with a high accuracy the experimental data (see Figs. 4.2a, 4.2b, and 4.2c). A second stage, in which, even though the peak values are in good agreement, the experimental data presents small variations in the superconductor area (see Figs. 4.2d-4.2g). Finally, a third stage, coinciding with the unloading of the sample, where a slightly higher discrepancy is observed. However, peak values are correctly captured and the model predicts the physical phenomenon throughout the entire simulation.

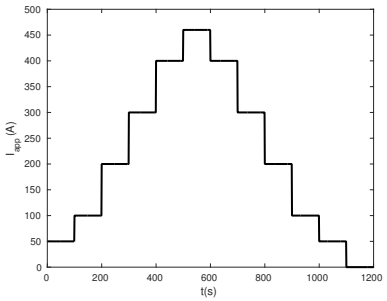
4.5.4 3D benchmark

In this section, our algorithms are applied to a recently proposed benchmark in [85], suggested as a stepping stone for future investigations in the 3D modelling of the electromagnetic behaviour of superconductors. The benchmark consists in the magnetization of a superconducting parallelepiped subjected to an AC magnetic field making an angle with the normal to the larger surface. Many authors have previously analysed the case of a tape under an external field by several methods [122, 107, 41, 9, 115]. In our case, the scenario is valid to model two different cases: the magnetization of an isotropic superconducting bulk and the one of stacks of HTS coated conductors. The stack can be modelled as a bulk [151] in which the current cannot flow along one direction due to the presence of high-resistivity layers.

Consider a box domain $\Omega = [0, 100] \times [0, 100] \times [0, 100]$ mm^3 where the superconductor fills the volume $\Omega_{\text{hts}} = [45, 55] \times [45, 55] \times [49.5, 50.5]$ mm^3. The air domain is defined as $\Omega_{\text{air}} = \Omega \setminus \Omega_{\text{hts}}$. There is no source term, i.e., $\boldsymbol{f} = \boldsymbol{0}$, and Dirichlet-type boundary conditions are imposed over the entire boundary as a time-dependent magnetic field. Initial conditions are simply $\boldsymbol{H}(t = 0) = \boldsymbol{0}$ T. On the other hand, no net current flow condition is imposed on the superconductor. The parallelepiped is subjected to a uniform external sinusoidal magnetic field in the xz plane at an angle $\alpha = \pi/6$ with respect to the x-axis, an amplitude of $\boldsymbol{B}_{ext} = 200$ mT, and frequency ω =50 Hz. The

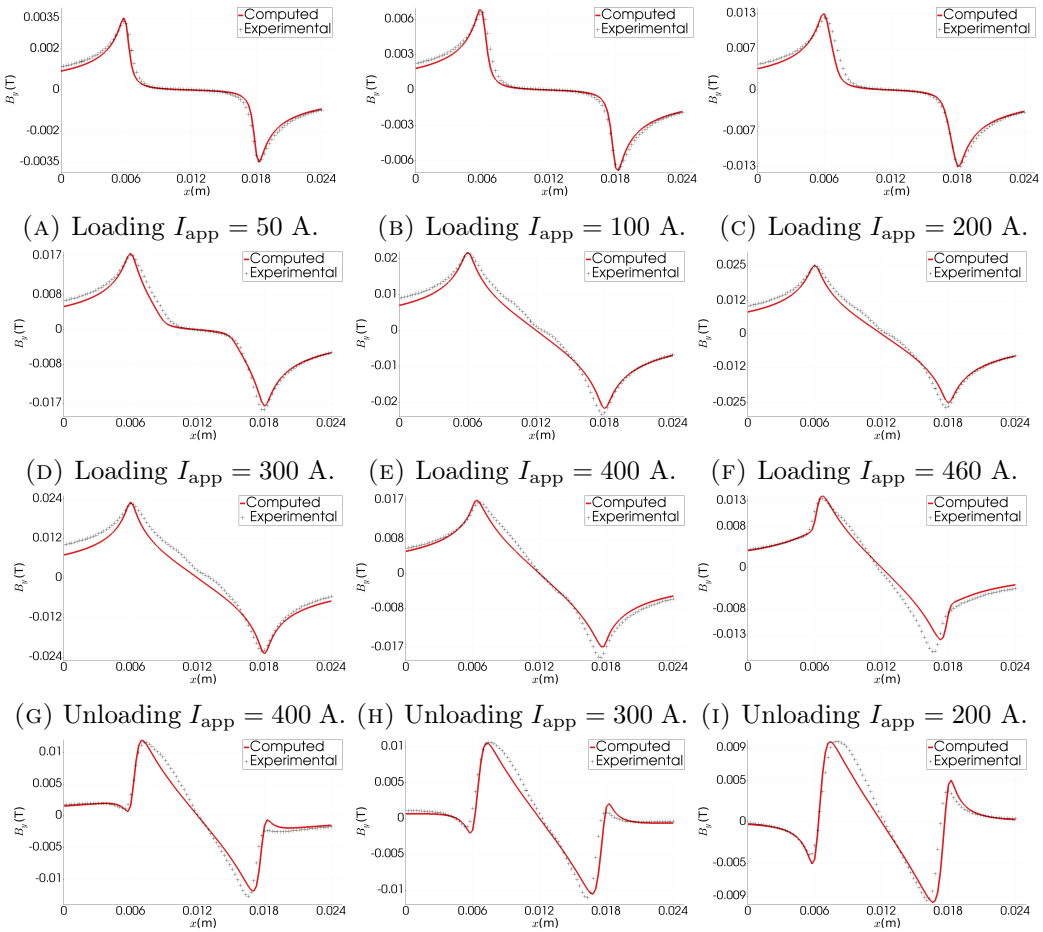


(A) Dependence of J_c on the magnetic field $\boldsymbol{B}$. Normalized with J_{c0}.



(B) Injected current in the superconductor as a step function. Peak value of $I = 460$ A, $I = 1.03 I_c$.

FIGURE 4.1: Validation problem inputs definition.



(A) Loading $I_{\text{app}} = 50$ A. (B) Loading $I_{\text{app}} = 100$ A. (C) Loading $I_{\text{app}} = 200$ A.

(D) Loading $I_{\text{app}} = 300$ A. (E) Loading $I_{\text{app}} = 400$ A. (F) Loading $I_{\text{app}} = 460$ A.

(G) Unloading $I_{\text{app}} = 400$ A. (H) Unloading $I_{\text{app}} = 300$ A. (I) Unloading $I_{\text{app}} = 200$ A.

(J) Unloading $I_{\text{app}} = 100$ A. (K) Unloading $I_{\text{app}} = 50$ A. (L) Unloading $I_{\text{app}} = 0$ A.

FIGURE 4.2: Magnetic field $B_y = \mu_0 H_y$ profiles for experimental (Hall probe mapping) and computed data in a full load-unload cycle for the validation test.

superconductor behaviour is modelled with the resistivity nonlinear law (see Eq. (4.16)) with an exponent $n = 24$. The first situation, where an isotropic resistivity tensor is considered, i.e., $\boldsymbol{\rho}_{\text{hts}} = \rho_{\text{hts}} \boldsymbol{I}_{3\times 3}$, models a bulk behaviour. On the other hand, the situation where current cannot flow along the z-direction models a stack. In this case, a high value is assigned to the z-component of a diagonal anisotropic resistivity tensor, $\rho_z = \rho_{\text{air}}$, whereas the resistivity is set to ρ_{hts} in the remaining directions. For the sake of brevity, the two situations are referred to as bulk and stack, respectively. A detailed exposition of the parameters being used can be found in Tab. 4.1. Note the slight reduction of the resistivity parameter in the dielectric region, which allows to enhance convergence performance without any impact in the computed quantities, as will be shown by results. Simulations are performed for a full cycle and an additional quarter of a cycle to take into account the initial magnetization of the HTS device. Maximum and minimum allowed time step sizes are set to $\frac{5T/4}{2{\cdot}10^2}$ and $\frac{5T/4}{2{\cdot}10^5}$, with $T = \omega^{-1}$. Therefore, the full simulation is performed with the lower bound of 200 time steps. Through this section, we will make use of several different meshes, described in Tab. 4.2. First, aiming to show

	#cells on Ω_{hts}		#DOFs on Ω_{hts}	#cells on Ω	#DOFs on Ω
Mesh 1	$12 \times 12 \times 2$	288	1,254	1,464	6,386
Mesh $1^{r=1}$	$24 \times 24 \times 4$	2,304	8,500	11,712	42,948
Mesh $1^{r=2}$	$48 \times 48 \times 8$	18,432	61,544	93,696	312,008
Mesh 2	$52 \times 52 \times 6$	16,224	55,438	28,624	101,368
Mesh 3	$102 \times 102 \times 10$	104,040	337,222	147,904	495,190

TABLE 4.2: Summary of meshes used to reproduce the benchmark. The superscript r denotes the number of uniform refinements applied to every cell of the original mesh.

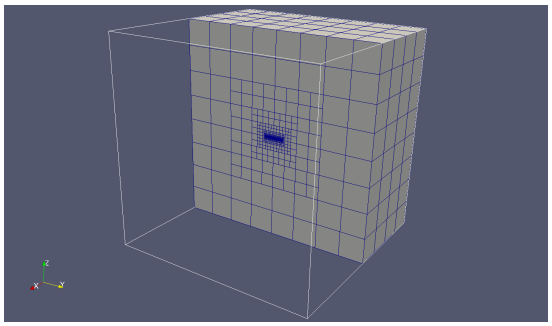
the accuracy of high-order Nédélec FEs, we will employ a very coarse mesh consisting of $12 \times 12 \times 2$ cells on the superconducting domain. Finer meshes are obtained through r isotropic refinements (i.e., dividing every single cell into 8 cells). Thus, these meshes are defined by the original mesh and the levels of refinement r in Tab. 4.2. Besides, we will make use of two different meshes for the study of the computational times: a coarser mesh, consisting of $52 \times 52 \times 6$ cells in the superconducting region and 28624 cells in the whole computational domain, and a finer mesh consisting of $102 \times 102 \times 10$ cells in the superconducting region and 147904 in total. This study is performed for first order Nédélec FEs. Nonlinear iterations are stopped when the Euclidean norm of the residual is below 10^{-4}. Further details of the benchmark can be found in [85].

Fig. 4.4 shows the pattern of computed current density distributions within the superconducting device for the bulk and the stack. All the current density plots are taken at $t = 5$ ms, when the sinusoidal function reaches its first peak value. On the other hand, Fig. 4.5 shows the computed magnetization loops in the bulk and the stack, respectively. Up to three different curves are shown for each magnetization plot: the projection of the magnetization on directions x, z and α (i.e., the direction of the applied magnetic field). Shown data is normalized with the magnitude $J_c \cdot b$, with b the length of the side of the base of the parallelepiped. Finally, Fig. 4.6 shows the computed instantaneous power loss in both cases.

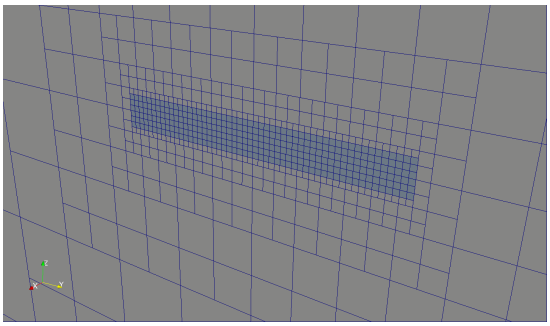
The estimation of AC losses is essential to assess the performance of superconducting devices (see [81], where a review of the field up to 2013 is presented). Therefore, any numerical tool modelling HTS devices must estimate AC losses accurately. Tab. 4.3 shows a direct comparison between computed AC losses from the benchmark and our numerical results, computed with Mesh 2 (see Tab. 4.2) and lowest order Nédélec elements. In particular, presented AC losses are computed via two different methods. The magnitude, for a full cycle, can be calculated by integrating the instantaneous power dissipation $\boldsymbol{J} \cdot \boldsymbol{E}$ in the superconductor, i.e.,

$$Q_{\text{JE}} = 2 \int_{\frac{T}{2}}^{T} \int_{\Omega_{\text{hts}}} \boldsymbol{J} \cdot \boldsymbol{E} = 2 \int_{\frac{T}{2}}^{T} \int_{\Omega_{\text{hts}}} \rho_{\text{hts}} \|\boldsymbol{J}\|^2, \tag{4.35}$$

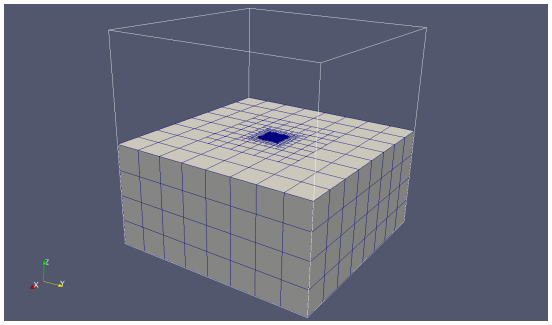
where the AC loss for half cycle is computed and doubled. Alternatively, the magnitude



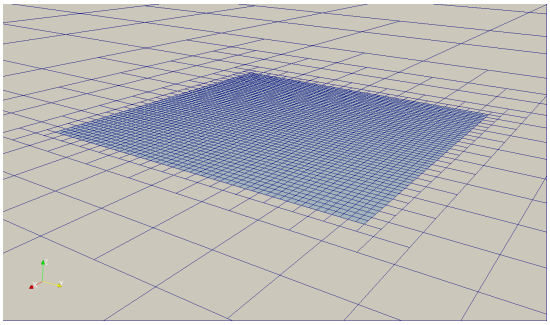
(A) Half domain in cross-section perpendicular to x-axis.



(B) Zoom to centered region. HTS domain depicted in blue.



(C) Half domain in cross-section perpendicular to z-axis.



(D) Zoom to centered region. HTS domain depicted in blue.

FIGURE 4.3: Illustration of Mesh 2 (see Tab. 4.2). Refinement pattern is common to all meshes. Adaptive refinement technique results in a structured mesh for the HTS device, while a smart coarsening following the 2:1 ratio can be observed in the dielectric domain surrounding the superconducting region.

	Q_{JE} bulk	Q_{MH} bulk	Q_{JE} stack	Q_{MH} stack
Reference	4.59	4.62	3.47	3.45
Computed	4.64	4.62	3.48	3.46

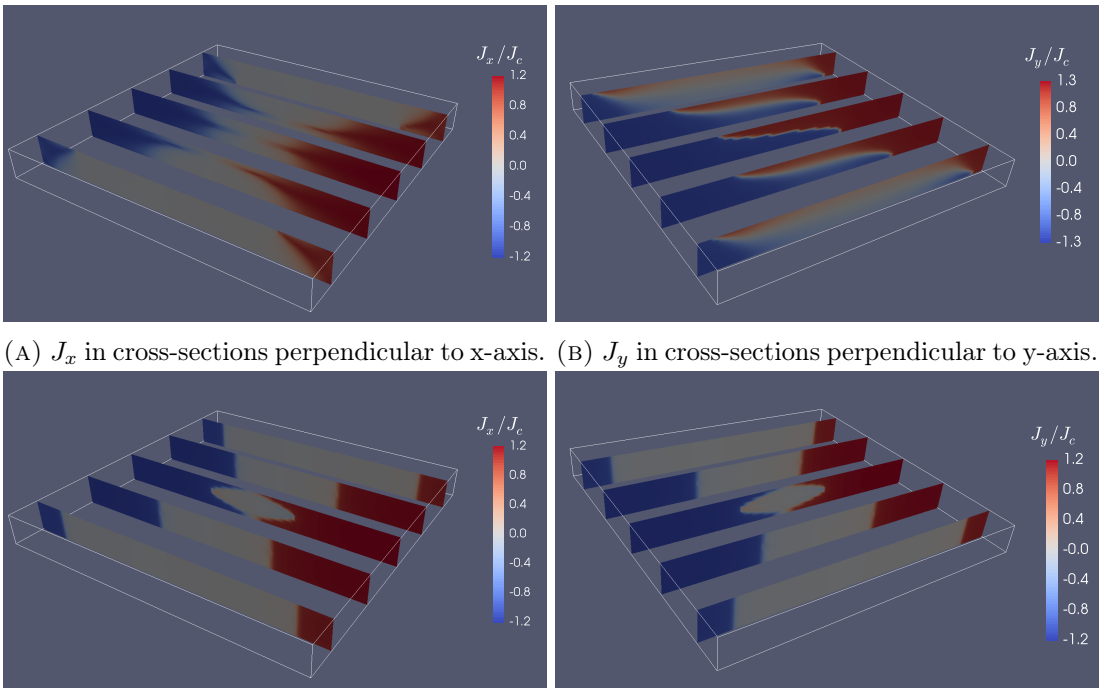
TABLE 4.3: Comparison of AC losses (in mJ) in the bulk and in the stack calculated with two different methods against benchmark results.

can be obtained with the magnetization loop in the direction of the applied external field $\boldsymbol{H}_\alpha$ as

$$Q_{\mathrm{MH}} = -\mu_0 \oint_{\boldsymbol{H}_\alpha} M_\alpha \, \|\Omega_{\mathrm{hts}}\| \tag{4.36}$$

for the full cycle, calculated in the time interval from peak to peak of the applied field. The numerical results are in excellent agreement with those presented in the benchmark, demonstrated by the integral quantities (4.35) and (4.36), see Tab. 4.3.

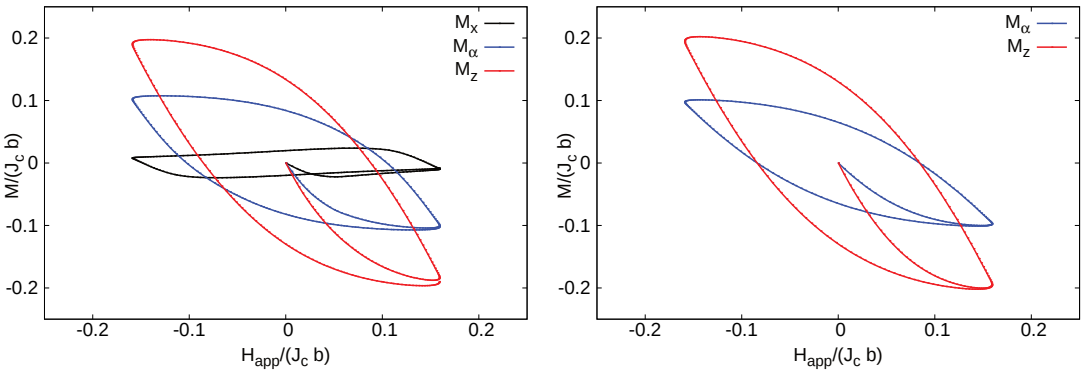
In order to show the accuracy of high order FE schemes, we intentionally choose a coarse mesh with $12 \times 12 \times 2$ elements in the HTS domain (Mesh 1). Better approximations can be obtained by either increasing the order of the FEs or reducing uniformly the mesh size h, known in the literature as p-refinement and h-refinement,



(A) J_x in cross-sections perpendicular to x-axis. (B) J_y in cross-sections perpendicular to y-axis.

(C) J_x in cross-sections perpendicular to x-axis. (D) J_y in cross section perpendicular to y-axis.

FIGURE 4.4: Distribution of normalized current densities for the bulk (top) and the stack (bottom).



(A) Bulk. (B) Stack.

FIGURE 4.5: Magnetization loops for the models. Magnitudes are normalized with critical current density and HTS device size.

respectively. The effect of applying r consecutive uniform refinements in all cells results in a mesh with $8^r N_c$ elements (i.e., every cell is partitioned into 8 children cells), where N_c is the number of elements in the coarsest mesh. We recall that $\mathcal{ND}_k$, where the discrete solution $\boldsymbol{H}$ lies, is the discrete curl-conforming space based on the polynomial space $\mathcal{Q}_{k-1,k,k} \times \mathcal{Q}_{k,k-1,k} \times \mathcal{Q}_{k,k,k-1}$ (see Sect. 4.3). The current density is obtained as $\boldsymbol{J} = \nabla \times \boldsymbol{H}$, thus belonging to $\mathcal{D}_k := \{\mathcal{Q}_{k,k-1,k-1} \times \mathcal{Q}_{k-1,k,k-1} \times \mathcal{Q}_{k-1,k-1,k}\}$. Fig. 4.7 depicts how the solution is converging with the order of the elements and a fixed mesh (Mesh 1), whereas Fig. 4.8 shows the behaviour of the solution for a uniformly size-refined

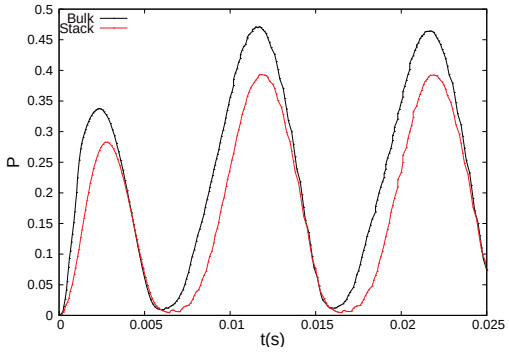


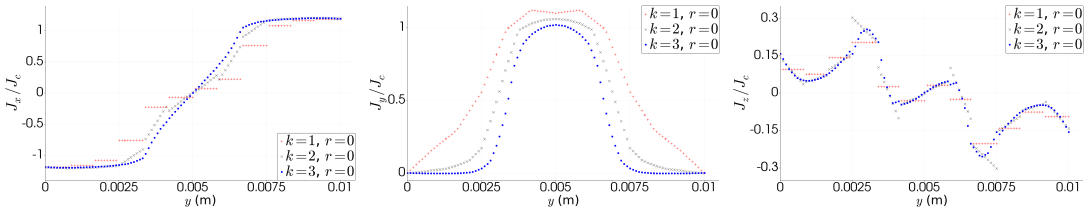
FIGURE 4.6: Instantaneous power dissipation in the bulk and the stack.

mesh and lowest order elements; current densities are normalized with the critical current density in all cases. Plots are taken for $t = 5$ ms, i.e., first peak value of the applied external field, and over the line $\{x = 0,\ z = 0\}$. The bottom axis represents the size of the parallelepiped, its centre being located at $y = 5$ mm in the plots. Out of this length, the normalized current density is negligible since it is out of the limits of the superconducting domain. Note that current density profiles are in general discontinuous across elements. Furthermore, as $\boldsymbol{J} \in \mathcal{D}_k$, its i-th component, for $i \in \{1, 2, 3\}$, is a polynomial of degree k with respect to x_i, and $k - 1$ otherwise.

Convergence to a solution is shown with respect to p-refinement (Fig. 4.7) and h-refinement (Fig. 4.8). It is clear from the plots that, for coarse meshes, the current density is much better captured with high order elements. Fig. 4.9a shows a direct comparison between quadratic FEs in the coarsest mesh and linear FEs in the mesh after one level of refinement, i.e., $r = 1$. Note that both discretizations involve the same number of DOFs. More accurate solutions, i.e., closer to the solution for the most accurate simulation in Fig. 4.9b, are obtained for quadratic FEs. It is expected, since, for smooth solutions, which is the case for the current profiles, p-refinement leads to exponential convergence rates.

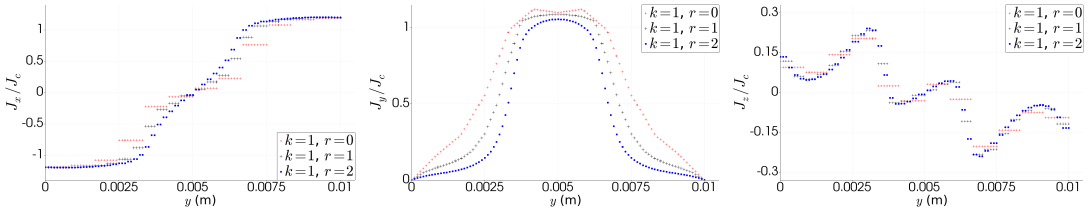
Fig. 4.9b shows a comparison between the coarsest mesh with third order FEs and first order elements in the mesh after two levels of refinement. The simulation with third order FEs involves a lower number of DOFs (136,038 vs 312,008). Again, p-refinement is more effective than h-refinement, achieving better results with a substantially lower number of DOFs. Summarizing, p and h-refinement converge to the same solution, but p-refinement is more effective for the smooth solutions at hand (see Figs. 4.9a and 4.9b).

Tabs. 4.4 and 4.5 show total parallel execution times for the 3D benchmark with Meshes 2 and 3, respectively (see Tab. 4.2). This total time includes the time spent in *every* single step of the simulation pipeline, including mesh generation. The BDDC-preconditioned conjugate gradient (CG) iterative solver (see Sect. 4.4.4) was set up such that the coarse-grid problem is mapped and solved on a single node of the MN-IV supercomputer by means of Intel MKL PARDISO on 48 threads (cores). Hence, the simulations have actually been run in $48 + P$ cores, P being the number of subdomains



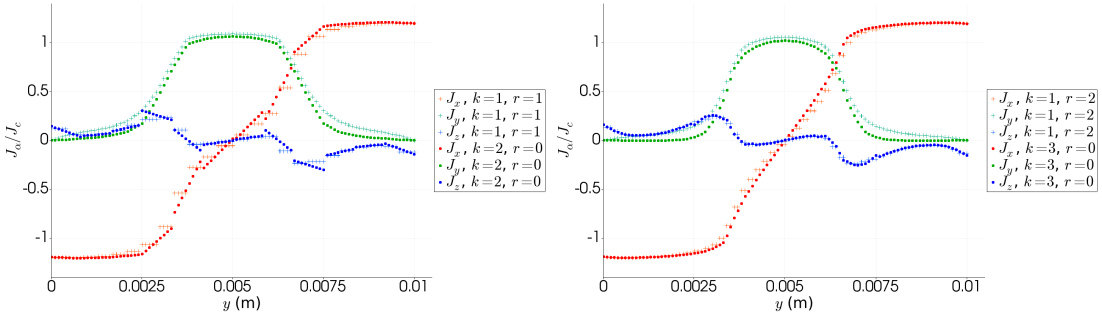
(A) J_x profiles. (B) J_y profiles. (C) J_z profiles.

FIGURE 4.7: Normalized $\boldsymbol{J}$ components over a line in the y-axis direction that passes through $z = 0$ with a fixed mesh and variable FE orders, i.e., p-refinement.



(A) J_x profiles. (B) J_y profiles. (C) J_z profiles.

FIGURE 4.8: Normalized $\boldsymbol{J}$ components over a line in the y-axis direction that passes through $z = 0$ with first order FEs and different uniformly refined meshes, i.e., h-refinement.



(A) Comparison with same #DOFs. (B) Higher order contains less #DOFs.

FIGURE 4.9: $\boldsymbol{J}$ components over a line in the y-axis direction that passes through $z = 0$ for different FE orders (p-refinement) and meshes (h-refinement).

in which the global domain is partitioned. These tables show two key parameters for measuring strong scalability: the parallel speed-up, defined as the ratio between the parallel execution time t_P when using P processors and the sequential execution time t_1, i.e., $S_p = t_P/t_1$, and the parallel efficiency, defined as $E_p = S_p/p$. (Thus, the closer S_p and E_p are to P and 1, resp., the better.) Note that the nonlinear convergence rate for a given problem does not depend on the linear solver being used (up to linear solver tolerance). Therefore, all the runs that involve the same mesh share the same the nonlinear convergence history (hence time stepping). The difference is found in the solution of the linearized problems, since the performance of the preconditioner (Sect. 4.4.4) does depend on the partition being used. Aiming to show the impact of the

P	**Wall clock time**	$\mathbf{S_p}$	$\mathbf{E_p}$	**#DOFs per part**	**#DOFs coarse solver**	**#Linear solver iterations**
Serial run	1d 19h 23'	1.00	1.00	101,428	-	-
6	8h 29'	5.23	0.87	19,091	286	22
12	4h 28'	9.95	0.82	9,854	520	29
24	2h 25'	18.26	0.76	5,143	990	30
48	1h 32'	28.81	0.60	2,775	1,852	34
72	1h 14'	35.79	0.49	1,928	2,507	35
96	1h 47'	22.87	0.23	1,489	3,037	42

TABLE 4.4: Computing times for problem solved using Mesh 2. Iteration and free DOF counters show average values. Simulation ends after 246 converged time steps involving 976 linearized problem solves.

P	**Wall clock time**	$\mathbf{S_p}$	$\mathbf{E_p}$	**#DOFs per part**	**#DOFs coarse solver**	**#Linear solver iterations**
Serial run*	23d 06h 14'	1.00	1.00	495,190	-	-
6	4d 22h 30'	4.71	0.78	88,807	474	28
12	2d 08h 23'	9.90	0.82	45,043	834	29
24	1d 08h 31'	17.18	0.71	23,001	1,550	29
48	17h 16'	32.64	0.68	11,938	2,846	34
96	12h 05'	46.23	0.48	6,214	4,924	42
120	10h 46'	51.86	0.43	5,087	6,253	44
144	10h 38'	52.53	0.36	4,316	7,446	47

TABLE 4.5: Computing times for problem solved using Mesh 3. Iteration and free DOF counters show average values. Simulation ends after 459 converged time steps involving 2374 linearized problem solves. *Serial run time is computed with an extrapolation with the number of linearized problem solves after 3 days of computation due to limited computing time in the access to MN-IV.

preconditioner for each partition, Tabs. 4.4 and 4.5 show the number of preconditioned Krylov iterations needed to attain convergence; the presented number of iterations is an arithmetic mean value of the number of iterations until convergence of all linearized problems taking place during the simulation, i.e., for every time step and nonlinear iteration.

Let us comment on the results in Tabs. 4.4 and 4.5. Clearly, the most salient property of the algorithms at hand is the remarkable reduction in time-to-solution in both cases. By exploiting parallel resources, the computational time is reduced by a factor of 35.8 and 52.5 for Mesh 2 (Tab. 4.4) and Mesh 3 (Tab. 4.5), respectively. As far as we know, such speed-ups have not been presented so far for FE HTS modelling. In practice, these speed-ups allow us to reduce the time-to-solution for a practical HTS simulation from days to hours. For the largest problem size, time-to-solution is reduced up to 144 parts. Above this core count, time-to-solution increases due to parallelism related

overheads; more computationally intensive simulations (i.e., larger loads per processor) would be required to exploit additional computational resources efficiently. Although average iteration counts increase mildly with the number of processors, the size of the coarse problem keeps growing while the local problems become smaller. Therefore, at some point, the coarse solver, which only exploits a bounded number of cores (i.e., 48), dominates computing times, loosing parallel efficiency. Fortunately, there is a large room for improvement in this direction, e.g., a multilevel version of the preconditioner is expected to push forward the limits of the presented strong scalability results (see, e.g., [23]).

4.6 Conclusions

In this chapter, we presented a parallel, fully-distributed FE framework suitable for the solution of nonlinear problems modelling the electromagnetic behaviour of HTS devices. We have selected the widespread H-formulation as a demonstrator of the potential of the presented advanced numerical algorithms, which have been tailored for the problem at hand and combined through the simulation pipeline, even though the ingredients presented in this chapter could be applied to other formulations. For the mesh generation, we have considered the advanced AMR technique presented in Chapter 2, which provides an aggressive coarsening in dielectric regions. The smart coarsening is restricted by the 2:1 balance, which allows for efficient parallel implementations. For the FE approximation, we choose edge (or Nédélec) elements of arbitrary order, which have been introduced in Chapter 2. These elements are favoured in electromagnetics simulations due to their sound mathematical structure. For the solution of the arising algebraic systems, we have presented the design of a tailored nonlinear parallel solver, which includes a linearization with a Newton-Raphson method (with exact Jacobian derivation) and advanced DD preconditioners for $\mathcal{H}(\mathbf{curl})$ spaces on adaptive meshes and heterogeneous problems, which have been defined in Chapter 3. Time integration was performed with the Backward Euler integrator and a variable time step, taking advantage of the convergence history of the nonlinear solver, and thus reducing time-to-solution. Finally, we have provided a detailed set of numerical experiments. First, a comparison with experimental data has shown an excellent agreement between experimental and numerical data. Second, a time-to-solution study reproducing a 3D benchmark has shown a remarkable reduction of computing times when exploiting parallel resources, and thus the capability of our algorithms to efficiently exploit HPC platforms. The work here presented has been implemented in the open source simulation software `FEMPAR`, which can become a powerful tool for the HTS modelling community.

Chapter 5

Nonlinear parallel-in-time solvers for ordinary differential equations

In this chapter, we propose a parallel-in-time solver for linear and nonlinear ordinary differential equations, which will allow us to identify the key ingredients towards a full space-time approach. The time parallel method is based on an efficient multilevel solver of the Schur complement related to a multilevel time partition. For linear problems, the scheme leads to a fast direct method. Next, two different strategies for solving nonlinear ordinary differential equations (ODEs) are proposed. First, we consider a Newton method over the global nonlinear ODE, using the multilevel Schur complement solver at every nonlinear iteration. Second, we state the global nonlinear problem in terms of the nonlinear Schur complement (at an arbitrary level), and perform nonlinear iterations over it. Numerical experiments show that the proposed schemes are weakly scalable, i.e., we can efficiently exploit increasing computational resources to solve for more time steps the same problem.

5.1 Introduction

At the beginning of the next decade supercomputers are expected to reach a peak performance of one exaflop/s, which implies a 100 times improvement with respect to current supercomputers. This improvement will not be based on faster processors, but on a much larger number of processors (in a broad sense). This situation will certainly have an impact in large scale Computational Science and Engineering (CSE). Parallel algorithms will be required to exhibit much higher levels of concurrency, keeping good scalability properties.

When dealing with transient problems, since information always moves forward in time, one can exploit sequentiality. However, the tremendous amounts of parallelism to be exploited in the near future certainly motivates to change this paradigm. One of the motivations to exploit higher levels of parallelism will be to reduce the time-to-solution. In the simulation of ODEs, the way to go is to exploit concurrency in time. The idea is to develop parallel-in-time solvers that provide the solution at all time values in one shot, instead of the traditional sequential approach that exploits the arrow of time. If

scalable parallel-in-time solvers are available, the use of higher levels of parallelism will certainly reduce the time-to-solution.

Parallel-in-time solvers are receiving rapidly increasing attention. Different iterative methods have been considered so far, e.g., the *parareal* method [93] or spectral deferred-correction time integrators [63]. With regard to direct methods, time-parallel methods can be found in [71]. In general these methods can exploit low levels of concurrency [52] or are tailored for particular types of equations [72]. We refer to [71] for an excellent up-to-date review of time parallelism.

In this chapter, we propose a parallel-in-time solver for ODEs that relies on the well-known Schur complement method in linear algebra. For linear (systems of) ODEs, the approach can be understood as a Schur complement solver in time.

When the coarse problem is too large compared to the local problems, due to the structure of the coarse problem, we can consider recursively the Schur complement strategy, leading to multilevel implementations, in order to push forward scalability limits. The method can be applied to θ-methods, discontinuous Galerkin (DG) methods, Runge-Kutta (RK) methods, and Backward differentiation formula (BDF) methods. We also note that the proposed method can also be understood as a parareal scheme in which the coarse solver is automatically computed in such a way that the scheme is a direct method (convergence in one iteration is assured). (The interpretation of the parareal method as an approximation of the Schur complement problem has already been pointed out in [65].) As a result, the proposed method solves the drawback of the parareal scheme, i.e., its poor parallel efficiency, inversely proportional to the number of iterations being required by the iterative algorithm. One of the messages of this work is to show that the approximation of the Schur complement in parareal methods does not really pay the price when (just by roughly multiplying by two the number of operations) one can have a highly scalable direct parallel-in-time solver.

In order to extend these ideas to nonlinear partial differential equations (PDEs), we consider two different strategies. First, we consider a global linearization of the problem in time using, e.g., Newton's method. We note that the idea of a global linearization of nonlinear ODEs to exploit time-parallelism is not new. It was already considered by Bellen and Zenaro in 1989 [33]. (In any case, the solvers proposed in [33] are different from the ones presented herein. They are restricted to the Steffensen's linearization method, which leads to a diagonal problem per nonlinear iterations, where parallelization can obviously be used.) After the linearization of the problem, we consider the Schur complement solver commented above at every nonlinear iteration. A second strategy consists in applying the nonlinear Schur complement strategy first, and next to consider the linearization of such operator, leading to nested nonlinear iterations.

The parallel-in-time ideas in this chapter can naturally be blended with domain decomposition (DD) ideas to design highly scalable space-time parallel solvers. Indeed, we have combined these ideas with a multilevel balancing domain decomposition by constraints (BDDC) preconditioner (see [98, 140]) in Chapter 6.

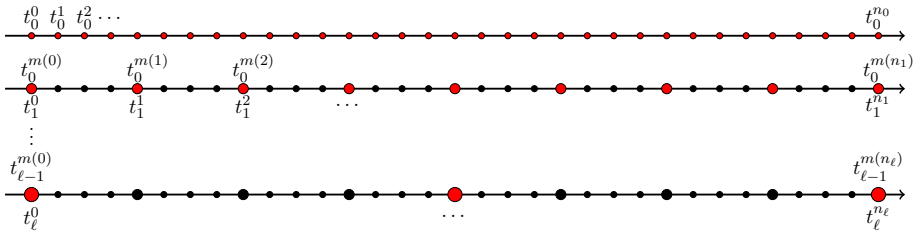


FIGURE 5.1: Multilevel time partition of $[0,T]$. The subindex in t_α^β denotes the partition level, whereas the superindex denotes the time step in such partition. In order to relate time values of two constitutive levels, we use the notation $t_\alpha^\beta = t_{\alpha-1}^{m(\beta)}$, i.e., the time value t_α^β corresponds to the time step $m(\beta)$ at the previous level.

The outline of the chapter is as follows. In Sect. 5.2, we state the problem. We introduce a time-parallel direct method for linear ODEs based on the computation of a multilevel time Schur complement in Sect. 5.3. In Sect. 5.4, we extend the method to nonlinear ODEs, by combining first a Newton linearization step with a Schur complement linear solver and next considering nonlinear Schur complement problems. We present a detailed set of numerical experiments in Sect. 5.5, showing the excellent scalability properties of the proposed methods. Finally, we draw some conclusions in Sect. 5.6.

5.2 Statement of the problem

In this section, we develop a parallel direct solver for the numerical approximation of ODEs. We consider a system of ODEs of size m_{unk}:

$$\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} + \boldsymbol{\kappa}(t, \boldsymbol{u}(t)) = \boldsymbol{0}, \qquad \boldsymbol{u}(t^0) = \boldsymbol{u}_0, \tag{5.1}$$

for $t \in (t^0 = 0, T]$. Let us assume that $\boldsymbol{\kappa}(\cdot,\cdot)$ is continuous with respect to the first argument and Lipschitz continuous with respect to the second argument, and that existence and uniqueness holds.

For the time interval $[0,T]$, we define a hierarhical multilevel partition as follows (see Fig. 5.1 for a detailed illustration). We define a (level-0) time partition $\{0 = t_0^0, t_0^1, \ldots, t_0^{n_0} = T\}$ into n_0 time elements. Next, we consider a (level-1) coarser time partition $\{0 = t_1^0, \ldots, t_1^{n_1} = T\}$ into n_1 time subdomains (or level-1 elements), defined by aggregation of elements at the previous level, i.e., for every $i \in \{0, \ldots, n_1\}$ there exists an $m(i) \in \{0, \ldots, n_0\}$ such that $t_1^i = t_0^{m(i)}$. We proceed recursively, creating coarser partitions for higher levels. We define the time element i at level-k as the time interval (t_k^i, t_k^{i+1}).

We will present the method in a general way that is independent of the time integration scheme being used. We define the nonlinear operators $\boldsymbol{\mathcal{A}}_0^{i+1} : \boldsymbol{u}^i \mapsto \boldsymbol{u}^{i+1}$ for $i \in \{0, \ldots, n_0 - 1\}$, such that, given the initial value $\boldsymbol{u}^i$, solves (5.1) in (t_0^i, t_0^{i+1}), and provides $\boldsymbol{u}^{i+1} = \boldsymbol{u}(t_0^{i+1})$. We conceptually state the solver at the continuous level, even

though one can consider different time integration schemes instead, e.g., θ-methods, DG methods, or RK methods. The use of BDF-type schemes requires some further ellaboration. We refer to Sect. 5.3.1 for more details.

5.3 An ODE direct solver

In this section, we assume that $\boldsymbol{\kappa}(t,\cdot)$ is a linear operator. (The nonlinear extension is described in Sect. 5.4.) In this case, it is easy to check that $\boldsymbol{\mathcal{A}}_0^{i+1}(\cdot)$ is an affine mapping, and we have $\boldsymbol{\mathcal{A}}_0^{i+1}(\boldsymbol{u}^i) = \boldsymbol{\Phi}_0^{i+1}\boldsymbol{u}^i + \boldsymbol{g}_0^{i+1}$, where $\boldsymbol{\Phi}_0^{i+1}$ is a linear operator (an $m_{\text{unk}} \times m_{\text{unk}}$ matrix). Both $\boldsymbol{\Phi}_0^{i+1}$ and $\boldsymbol{g}_0^{i+1}$ can be explicitly computed from $\boldsymbol{\kappa}(\cdot,\cdot)$ and $\boldsymbol{f}$ for a given time integration scheme. Thus, the global problem (5.1) for linear ODEs can be stated in algebraic form as:

$$\begin{pmatrix} \boldsymbol{\mathcal{I}} & & & \\ -\boldsymbol{\Phi}_0^1 & \boldsymbol{\mathcal{I}} & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{\Phi}_0^{n_0} & \boldsymbol{\mathcal{I}} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_0^0 \\ \boldsymbol{u}_0^1 \\ \vdots \\ \boldsymbol{u}_0^{n_0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{g}_0^1 \\ \vdots \\ \boldsymbol{g}_0^{n_0} \end{pmatrix}, \tag{5.2}$$

where $\boldsymbol{\mathcal{I}}$ is the identity matrix (of size $m_{\text{unk}} \times m_{\text{unk}}$). We can also represent the global system (5.2) in compact notation with

$$\mathbf{K}_0\mathbf{u}_0 = \mathbf{g}_0, \quad \text{or equivalently} \quad \begin{pmatrix} \mathbf{K}_0^{II} & \mathbf{K}_0^{I\Gamma} \\ \mathbf{K}_0^{\Gamma I} & \mathbf{K}_0^{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \mathbf{u}_0^I \\ \mathbf{u}_0^\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{g}_0^I \\ \mathbf{g}_0^\Gamma \end{pmatrix}, \tag{5.3}$$

where we have considered a segregation of degrees of freedom (DOFs) at level-0 $\mathbf{u}_0$ into the *interface* DOFs $\mathbf{u}_0^\Gamma$, namely the time step values that are also in the level-1 partition, and the *interior* DOFs $\mathbf{u}_0^I$. $\mathbf{K}_0$ is a 2-banded lower block-triangular matrix. In order to define the Schur complement problem, we first consider an *interior correction* of the problem at hand,

$$\mathbf{K}_0^{II}\mathbf{v}_0^I = \mathbf{g}_0^I, \tag{5.4}$$

i.e., we solve the system (5.2) in the subspace of vectors that vanish in the level-1 time values $t_1^0, \ldots, t_1^{n_1}$. The solution of (5.4) involves n_1 independent local ODE problems: solve for $i = 0, \ldots, n_1 - 1$

$$\begin{pmatrix} \boldsymbol{\mathcal{I}} & & & \\ -\boldsymbol{\Phi}_0^{m(i)+1} & \boldsymbol{\mathcal{I}} & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{\Phi}_0^{m(i+1)-1} & \boldsymbol{\mathcal{I}} \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_0^{m(i)} \\ \boldsymbol{v}_0^{m(i)+1} \\ \vdots \\ \boldsymbol{v}_0^{m(i+1)-1} \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{g}_0^{m(i)+1} \\ \vdots \\ \boldsymbol{g}_0^{m(i+1)-1} \end{pmatrix}. \tag{5.5}$$

After the interior correction, we must solve the problem

$$\begin{pmatrix} \mathbf{K}_0^{II} & \mathbf{K}_0^{I\Gamma} \\ \mathbf{K}_0^{\Gamma I} & \mathbf{K}_0^{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \delta\mathbf{u}_0^I \\ \mathbf{u}_0^\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{g}_0^\Gamma \end{pmatrix}, \quad \text{and compute} \quad \mathbf{u}_0^I = \mathbf{v}_0^I + \delta\mathbf{u}_0^I. \tag{5.6}$$

In order to solve (5.6), we define the following extension operator (usually denoted as the harmonic extension operator in the frame of domain decomposition solvers for PDEs):

$$\begin{pmatrix} \mathbf{K}_0^{II} & \mathbf{K}_0^{I\Gamma} \\ \mathbf{0} & \mathbf{I}_\Gamma \end{pmatrix} \begin{pmatrix} \mathbf{E}_0^I \\ \mathbf{E}_0^\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{I}_\Gamma \end{pmatrix}, \quad \text{thus} \quad \mathbf{E}_0 = \begin{pmatrix} -(\mathbf{K}_0^{II})^{-1}\mathbf{K}_0^{I\Gamma} \\ \mathbf{I}_\Gamma \end{pmatrix}. \tag{5.7}$$

Thus, using the fact that $\mathbf{u}_0^\Gamma = \mathbf{u}_1$, the Schur complement of level-0 reads:

$$(\mathbf{K}_0^{\Gamma\Gamma} + \mathbf{K}_0^{I\Gamma}\mathbf{E}_0^I)\mathbf{u}_1 = \mathbf{g}_0^\Gamma - \mathbf{K}_0^{I\Gamma}\mathbf{v}_0^I, \quad \text{represented by} \quad \mathbf{K}_1\mathbf{u}_1 = \mathbf{g}_1. \tag{5.8}$$

The Schur complement of the level-0 system is the level-1 problem. By construction, the extension operator solution of (5.7) can be written as a block-diagonal matrix $\mathbf{E}_0 = \mathrm{diag}(\mathbf{e}_{(0)}, \mathbf{e}_{(1)}, \cdots, \mathbf{e}_{(n_1-1)}, \boldsymbol{\mathcal{I}})$, which involves $n_1 \times m_{\text{unk}}$ independent local ODE problems: solve for $i = 0, \ldots, n_1 - 1$

$$\begin{pmatrix} \boldsymbol{\mathcal{I}} & & & \\ -\boldsymbol{\Phi}_0^{m(i)+1} & \boldsymbol{\mathcal{I}} & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{\Phi}_0^{m(i+1)-1} & \boldsymbol{\mathcal{I}} \end{pmatrix} \begin{pmatrix} (\boldsymbol{e}_{(i)})_0^{m(i)} \\ (\boldsymbol{e}_{(i)})_0^{m(i)+1} \\ \vdots \\ (\boldsymbol{e}_{(i)})_0^{m(i+1)-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mathcal{I}} \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{5.9}$$

After some manipulation, the Schur complement problem (5.8) can be written as:

$$\begin{pmatrix} \boldsymbol{\mathcal{I}} & & & \\ -\boldsymbol{\Phi}_1^1 & \boldsymbol{\mathcal{I}} & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{\Phi}_1^{n_1} & \boldsymbol{\mathcal{I}} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1^0 \\ \boldsymbol{u}_1^1 \\ \vdots \\ \boldsymbol{u}_1^{n_1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{g}_1^{m(1)} \\ \vdots \\ \boldsymbol{g}_1^{m(n_1)} \end{pmatrix}, \tag{5.10}$$

where $\boldsymbol{\Phi}_1^i \doteq \boldsymbol{\Phi}_0^{m(i)} \boldsymbol{e}_{(i-1)}^{m(i)-1}$ and $\boldsymbol{g}_1^i \doteq \boldsymbol{g}_0^{m(i)} + \boldsymbol{\Phi}_0^{m(i)-1} \mathbf{v}_0^{m(i)-1}$. Thus, the Schur complement matrix $\mathbf{K}_1$ at the next level has also the same structure as the original problem, i.e., it is a ODE-type solver over the coarser level-1 time partition. The computation of the interior correction, the extension operator $\mathbf{E}_0$, the Schur complement matrix $\mathbf{K}_1$, and the Schur complement right-hand side $\mathbf{g}_1$ can readily be computed in parallel. In Alg. 3 (for $k = 0$), all the steps to assemble the Schur complement problem are listed, indicating on the left-hand side the line task corresponding level.

In a two-level implementation, i.e., $\ell = 1$, the Schur complement problem (5.6) would be computed in serial. It would finally lead to $\mathbf{u}_0 = \mathbf{v}_0 + \mathbf{E}_0\mathbf{u}_1$. This approach leads to n_1 indepedent level-0 problems of size $\frac{n_0}{n_1}$ and one coarse level-1 problem of size n_1. Clearly, when n_1 increases, the coarse problem becomes the bottleneck of the

Algorithm 3: Schur complement set-up (level-k)

Data: $\mathbf{K}_k$, $\mathbf{g}_k$
Result: $\mathbf{v}_k$, $\mathbf{E}_k$, $\mathbf{K}_{k+1}$,$\mathbf{g}_{k+1}$

1: Compute the interior correction $\mathbf{v}_k$ solution of (5.4), by solving the n_{k+1} local ODE problems (5.5) for $i = 0, \dots, n_{k+1} - 1$ — $k-1$
2: Compute the extension operator $\mathbf{E}_k$ solution of (5.7), by solving the $n_{k+1} \times m_{\text{unk}}$ problems (5.9) for $i = 0, \dots, n_{k+1} - 1$, and compute $\mathbf{K}_k^{I\Gamma}\mathbf{E}_0^I$ and $-\mathbf{K}_k^{I\Gamma}\mathbf{u}_k^I$ in (5.8) — $k-1$
3: Assemble the Schur complement system (5.10), i.e., $\mathbf{K}_{k+1}$ and $\mathbf{g}_{k+1}$ (see (5.8)) — $k-1 \to k$

simulations. In order to push forward the scalability limits of this approach, we can consider a multilevel Schur complement technique. When n_1 exceeds $\frac{n_0}{n_1}$, since (5.10) has the same structure as the original system (5.2), one can consider the same Schur complement approach for the level-1 system, leading to a three-level algorithm. We can proceed recursively to include an arbitrary number of levels. In Alg. 4, we state the multilevel Schur complement ODE solver. We comment on the parallel efficiency and computational cost of this algorithm in Sect. 5.3.3.

Algorithm 4: Multilevel Schur complement ODE solver

Data: $\mathbf{K}_0$, $\mathbf{g}_0$
Result: $\mathbf{u}_0 = \mathbf{K}_0^{-1}\mathbf{g}_0$

1: **for** $k = 0, \dots, \ell - 1$ **do**
2: | Call Alg. 3 with $(\mathbf{K}_k, \mathbf{g}_k)$ to get $(\mathbf{v}_k, \mathbf{E}_k, \mathbf{K}_{k+1}, \mathbf{g}_{k+1})$ — $k, k+1$
3: **end**
4: Solve $\mathbf{K}_\ell \mathbf{u}_\ell = \mathbf{g}_\ell$ — ℓ
5: **for** $k = \ell - 1, \dots, 0$ **do**
6: | Compute $\mathbf{u}_k = \mathbf{v}_k + \mathbf{E}_k \mathbf{u}_{k+1}$ — k
7: **end**

5.3.1 Application to different time integrators

The previous approach can straightforwardly be used for θ-methods. For DG and RK methods, we can require multiple intermediate stages to move from t_0^i to t_0^{i+1}. In the DG case, we have some additional time values per time element. We can consider that all the element values but the last one are eliminated at the element level, using the so-called static condensation technique. In this case, the resulting discrete problem can be stated as in 5.2. The DG method being used only affects the expression of $\mathbf{\Phi}_0^{i+1}$ and $\boldsymbol{g}_0^{i+1}$. (We note that DG methods do not satisfy time causality at the element level, but it does not affect the lower 2-banded block-triangular structure after the elimination of "interior" element values.) We proceed analogously for multi-stage RK methods.

BDF schemes (of second and higher order) slightly differ from the fact that the computation of the solution at a given time step not only requires value from the previous time step, but some additional stages. For a BDF(X) scheme, the system matrix in (5.2) is a (X+1)-banded lower block-triangular matrix. It affects the concept of interface nodes Γ; to decouple the global problems into local problems, we require to increase the size of

the interface X times. As a result, the coarse-scale space dimension is X times larger, as well as the number of coarse space basis functions being computed. In this sense, high order BDF schemes are be a bad choice when dealing with parallel computations, since the interface among subdomains increases, with the corresponding computational cost, due to a loss of locality with respect to the continuous problem. High order RK or DG methods are better suited for time-parallel computations. In any case, after considering the modification described above, the techniques proposed in this chapter can be applied to BDF methods.

5.3.2 Parareal interpretation

The multilevel Schur complement solver defined in Alg. 4 can also be understood as a (multilevel) parareal scheme. In the parareal scheme, we consider a coarse solver to provide initial conditions to local fine solvers. Instead, in the Schur complement method, one first computes the (fine) interior correction, which is required to the assembly of the right-hand side in the coarse solver. The method above can be stated in a different way, by defining the restriction operator $\mathbf{F}_0$ as follows:

$$\begin{pmatrix} \mathbf{F}_0^I & \mathbf{F}_0^\Gamma \end{pmatrix} \begin{pmatrix} \mathbf{K}_0^{II} & 0 \\ \mathbf{K}_0^{\Gamma I} & \mathbf{I}_\Gamma \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{I}_\Gamma \end{pmatrix}, \quad \text{thus} \quad \mathbf{F}_0 = \begin{pmatrix} -\mathbf{K}_0^{\Gamma I}(\mathbf{K}_0^{II})^{-1} & \mathbf{I}_\Gamma \end{pmatrix}. \tag{5.11}$$

Analogously to the extension operator $\mathbf{E}_0$, the computation of the restriction operator is a block-diagonal matrix $\mathbf{F}_0 = \mathrm{diag}(\boldsymbol{\mathcal{I}}, \mathbf{f}_{(0)}^T, \mathbf{f}_{(1)}^T, \cdots, \mathbf{f}_{(n_1-1)}^T)$, which involves $n_1 \times m_{\text{unk}}$ independent local (backwards) ODE problems: solve for $i = 0, \ldots, n_1 - 1$

$$\begin{pmatrix} \boldsymbol{\mathcal{I}} & -(\boldsymbol{\Phi}_0^{m(i)+2})^T & & \\ & \boldsymbol{\mathcal{I}} & \ddots & \\ & & \ddots & -(\boldsymbol{\Phi}_0^{m(i+1)})^T \\ & & & \boldsymbol{\mathcal{I}} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_{(i)}^{m(i)+1} \\ \boldsymbol{f}_{(i)}^{m(i)+2} \\ \vdots \\ \boldsymbol{f}_{(i)}^{m(i+1)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \boldsymbol{\mathcal{I}} \end{pmatrix}. \tag{5.12}$$

The well-posedness of the backward problem is a direct consequence of the well-posedness of its transpose, the forward problem. Thus, the Schur complement problem reads:

$$\mathbf{K}_1 \mathbf{u}_1 = \mathbf{g}_0^\Gamma + \mathbf{F}_0^I \mathbf{g}_0^I = \mathbf{g}_1.$$

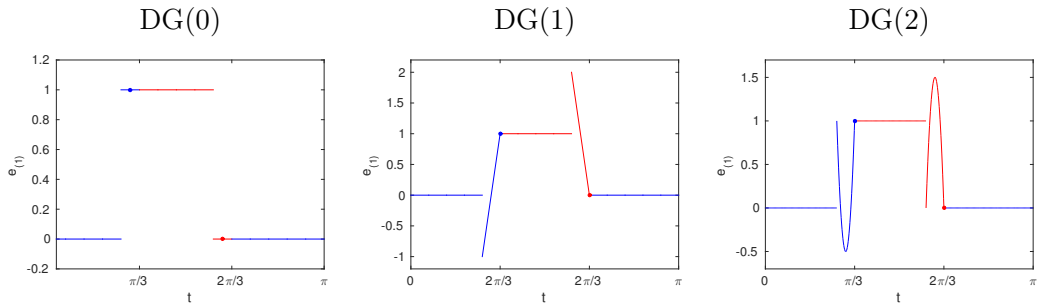
The coarse parareal problem in the Schur complement method is of Petrov-Galerkin type, using as coarse trial space the range of $\mathbf{E}_0$ and as coarse test space the range of $\mathbf{F}_0^T$, i.e,

$$\mathbf{K}_1 = \mathbf{F}_0 \mathbf{K}_0 \mathbf{E}_0, \quad \mathbf{g}_1 = \mathbf{F}_0 \mathbf{g}_0. \tag{5.13}$$

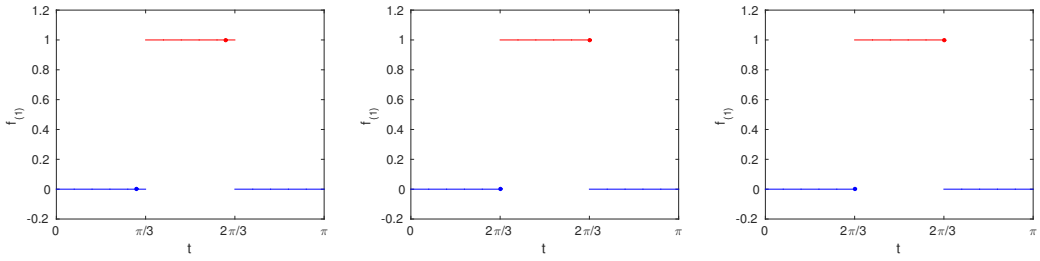
It is easy to check that (5.8) and (5.13) are equivalent. The fine solver is simply the interior correction (5.4). (We note that fine and coarse corrections are independent,

since they are $\mathbf{K}_0$-orthogonal.). The definition of the coarse space is automatic and the method is not an iterative but a direct solver. As a result, the method does not suffer from the low parallel efficiency of parareal methods, which is proportional to the inverse of parareal iterations [93]. Even though the implementation that involves the computation of the trial and test coarse spaces is the one being used in non-symmetric PDE solvers with inexact Schur complement preconditioning (see, e.g., [21]), it is not convenient for ODE solvers, since it involves an additional fine solver.

Fig. 5.2 depicts forward and backward coarse functions for a test problem. The local oscillations in the DG(1) and DG(2) schemes are due to the fact that time causality does not hold inside the element. (We note that the previous developments have been considered after eliminating (using element-wise solvers) all the element values but the last one (in time).) Fig. 5.3 shows coarse level-1, fine level-0, and full solution for a selected simple problem with different coarse DOFs position consideration.



(a) Coarse shape function $\mathbf{e}_{(1)}$ for $t_1^1 = \pi/3$



(b) Coarse shape function $\mathbf{f}_{(1)}$ for $t_1^1 = \pi/3$

FIGURE 5.2: Coarse shape functions $\mathbf{e}_{(1)}$ and $\mathbf{f}_{(1)}$ ($t_1^1 = \pi/3$) for the simple transport operator $\frac{du}{dt}$. A partition of an interval $[0, \pi]$ into three subdomains is considered. Each subdomain is partitioned into 5 time elements. Sub-intervals are depicted in consecutive different colour in order to aid visualization, and dots aid to identify coarse DOFs position (in the center of the element for DG(0)).

5.3.3 Parallel efficiency

Let us consider the multilevel Schur complement solver in Alg. 4 for a linear ODE. Using the same approach as in multigrid (MG) methods, we can consider a fixed coarsening ratio θ between subdomains, and leave free the number of levels ℓ required for a particular

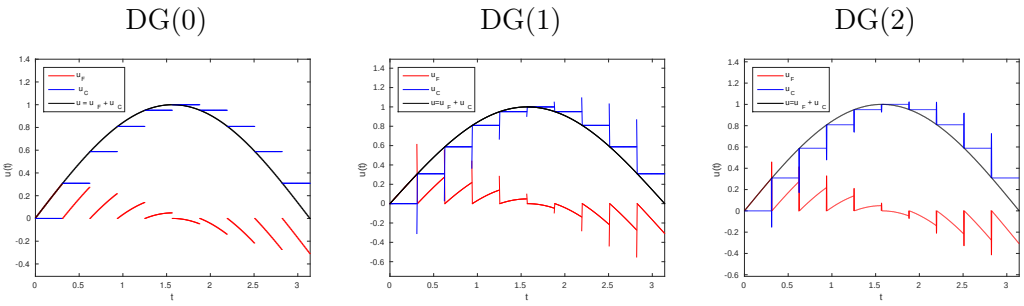


FIGURE 5.3: Decomposition of $\mathbf{u}_0$ into fine $\mathbf{E}_0\mathbf{u}_0$ and coarse $\mathbf{u}_1$ component for the simple problem $\partial_t u = \cos(t)$, on $[0, \pi]$. The time domain is partitioned into 10 subdomains and each subdomain is an aggregation of 50 elements.

simulation with n_0 time step values. The number of processors being used is assumed to be equal to $n_1 = n_0/\theta$, i.e., the number of subdomains at level-1, and the number of time steps per processor at all levels is θ (at the last level it can be smaller). Thus, we define $\ell + 1 = \text{ceiling}(\frac{\log n_0}{\log \theta})$, where ceiling$(A)$ returns the least integer greater than or equal to A.

Alg. 4 requires at levels $0, \ldots, \ell - 1$ to solve $1 + m_{\text{unk}}$ local linear ODE problems with θ time steps per processor (see Alg. 3), one to compute the interior correction and m_{unk} to compute the extension operator. One linear ODE with at most θ time steps must be solved at the last level. The different levels must be computed in a sequential way.

Based on the solver described above, we can estimate the order of floating point operations (FLOPs) required to solve the linear ODE. Sequentially, it is of the order of $\text{FLOP}_0 \approx n_0(m_{\text{unk}}^2 + m_{\text{unk}})$, since we require $m_{\text{unk}}^2 + m_{\text{unk}}$ operations per time step. On the other hand, the number of FLOPs required to compute the same problem in parallel using Alg. 4 is the one needed to solve $(1 + m_{\text{unk}})$ problems of size n_0, n_0/θ, $\ldots$, n_0/θ^ℓ. Using the geometric sequence sum formula, we get $\text{FLOP}_p \approx \text{FLOP}_0(1 + m_{\text{unk}})(1 - \frac{1}{\theta^{\ell+1}})(1 - \frac{1}{\theta})^{-1} < \text{FLOP}_0(1 + m_{\text{unk}})(1 - \frac{1}{\theta})^{-1}$, but these operations can be performed in parallel exploiting distributed memory machines. With regard to time, the total central processing unit (CPU) time of the multilevel Schur complement in Alg. 4 is the aggregation of the CPU time of the solution of $(1 + m_{\text{unk}})$ linear ODEs with θ time steps for all levels. Thus, the parallel CPU time is $\text{CPU}_p \approx \ell\theta(m_{\text{unk}}^2 + m_{\text{unk}})(1 + m_{\text{unk}})$, whereas the serial CPU time is $\text{CPU}_0 \approx n_0(m_{\text{unk}}^2 + m_{\text{unk}})$. As a result, the CPU time linearly depends on the size of the global system in a very mild logarithmic way, i.e., it increases with $\log n_0$ due to the expression of ℓ, and the parallel algorithm will rapidly lead to a shorter time-to-solution than the serial solver.[1] As a result, the speed-up of the proposed direct solver is $S = \text{CPU}_0/\text{CPU}_p \approx \text{P}\ell^{-1}(1 + m_{\text{unk}})^{-1}$. The speed-up is quasi-linear (it only increases with $\log n_0$), and thus algorithmically strongly scalable.

[1] We note that the communications are not considered in the previous estimations. In any case, it has been experimentally observed that the communication time in (incomplete) multilevel Schur complement methods is small compared to the computation time up to almost half a million tasks in [23].

Analogously, the method is algorithmically weakly scalable, because the total CPU time does not depend on the number of processors or global system size (apart from the logarithmic term).

5.4 Iterative solvers for nonlinear ODEs

In this section, we assume that $\boldsymbol{\kappa}(\cdot,\cdot)$ is nonlinear. The nonlinear ODE (5.1) in $(0,T]$ can be stated in compact form as $\boldsymbol{\mathcal{A}}_0(\mathbf{u}_0)=\mathbf{0}$, which can also be split into interior and interface time steps as follows:

$$\begin{pmatrix} \boldsymbol{\mathcal{A}}_0^I(\mathbf{u}_0) \\ \boldsymbol{\mathcal{A}}_0^\Gamma(\mathbf{u}_0) \end{pmatrix} = \mathbf{0}. \tag{5.14}$$

We consider two different types of solvers for the nonlinear problem.

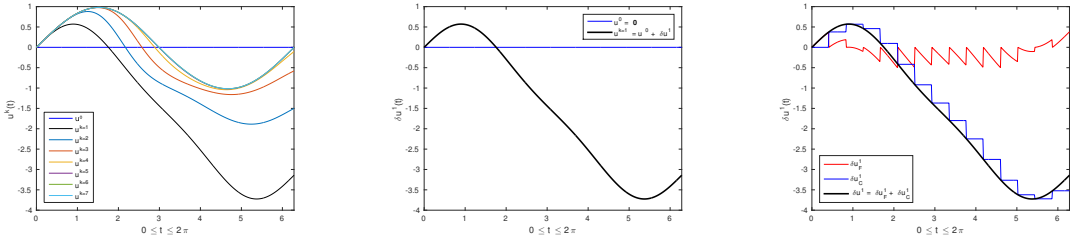
5.4.1 Newton-Schur complement methods

In order to solve the nonlinear ODE, we can use Newton's method over the global-in-time problem, and solve at every iteration a linear ODE using the multilevel Schur complement in Alg. 4. We can compute the Jacobian matrix related to (5.14) around a point $\bar{\mathbf{u}}_0$ as:

$$\boldsymbol{\mathcal{J}}_0(\bar{\mathbf{u}}_0) \doteq \frac{\partial \boldsymbol{\mathcal{A}}_0}{\partial \mathbf{u}_0}(\bar{\mathbf{u}}_0) = \frac{\mathrm{d}\bar{\mathbf{u}}_0}{\mathrm{d}t} + \frac{\partial \boldsymbol{\mathcal{K}}_0}{\partial \mathbf{u}_0}(\bar{\mathbf{u}}_0), \tag{5.15}$$

where we have used the fact that $\boldsymbol{\mathcal{A}}_0$ has two terms, one related to the time derivative and the other one related to $\boldsymbol{\kappa}(t,\cdot)$ (see 5.1). Thus, to solve (5.15) involves the solution of a linear ODE of the form (5.2).

The resulting multilevel Newton-Schur complement solver for nonlinear ODEs is stated in Alg. 5. Figure 5.4 shows the solution iterates using this algorithm for a selected nonlinear problem with known analytic solution $u = \sin(\mathrm{t})$. Each solution update obtained from a linearized problem is solved with the direct solver in Alg. 4 using two levels, i.e., $\ell = 1$. The nonlinear iterations of the Newton-Schur complement methods do not depend on the partition being used, since it is a linearization of the global problem and a (parallel) direct solver is used at every nonlinear iteration. In any case, the comparison against the sequential approach is more complicated here, since different nonlinear iterations (local vs. global) are being used in every case. We note that we can readily consider other iterative methods, e.g., Picard's method or Anderson acceleration techniques. In the case of Picard's method, the computation of the Jacobian is not required. Instead, the operator $\boldsymbol{\mathcal{A}}_0$ must be written as $\boldsymbol{\mathcal{A}}_0(\mathbf{u}_0) = \widetilde{\boldsymbol{\mathcal{A}}}_0(\mathbf{u}_0)\mathbf{u}_0$ and use at every nonlinear iteration the linear operator $\widetilde{\boldsymbol{\mathcal{A}}}_0(\bar{\mathbf{u}}_0)$ instead of $\boldsymbol{\mathcal{J}}_0(\bar{\mathbf{u}}_0)$. In Sect. 5.5, we consider a hybrid Picard-Newton nonlinear solver.



(A) Convergence history. (B) First iteration. (C) Additive solution.

FIGURE 5.4: Iterations for the solution of the nonlinear equation $\partial_t u - u^2 = \cos(t) - \sin^2(t)$ on $t = [0, 2\pi]$ using Newton's method and the parallel ODE direct solver. Fine and coarse solutions for the first nonlinear solution update δu^1. The time interval is discretized with 500 time steps divided into 15 subdomains. DG(0) (equivalent to Backward-Euler) is used.

Algorithm 5: Newton-Schur complement solver

Data: $\boldsymbol{u}^0$
Result: $\boldsymbol{u}_0$ such that $\boldsymbol{\mathcal{A}}_0(\boldsymbol{u}_0) = 0$

1:	$\mathbf{z} \leftarrow \mathbf{u}^0$ % Initial guess	k
2:	**while** *not convergence* **do**	
3:	Solve problem $\boldsymbol{\mathcal{J}}_0(\mathbf{z})\mathbf{y} = -\boldsymbol{\mathcal{A}}_0(\mathbf{z})$ using the multilevel Schur complement Alg. 4 with $(\boldsymbol{\mathcal{J}}_0(\mathbf{z}), -\boldsymbol{\mathcal{A}}_0(\mathbf{z}))$	$0, \ldots, \ell$
4:	Assign $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{y}$	k
5:	**end**	
6:	Return $\mathbf{z}$	$k, \ldots, \ell$

5.4.2 Nonlinear Schur complement-Newton methods

Following the ideas in nonlinear domain decomposition (see, e.g., [49, 87]), one can state the problem as a nonlinear Schur complement at a given level and next its linearization using, e.g., Newton's method. In order to present the problem, we consider the two-level case. Later, the algorithm will be extended to multiple levels. Let us define the level-1 nonlinear Schur complement problem

$$\boldsymbol{\mathcal{A}}_1(\mathbf{u}_1) = \boldsymbol{\mathcal{A}}_0^{\Gamma}(\boldsymbol{\mathcal{E}}_0(\mathbf{u}_1)), \quad \text{where} \quad \boldsymbol{\mathcal{E}}_0(\mathbf{u}_1) \doteq [\boldsymbol{\mathcal{E}}_0^I(\mathbf{u}_1), \mathbf{u}_1]^T,$$

is the nonlinear harmonic extension operator, solution of

$$\boldsymbol{\mathcal{A}}_0^I(\boldsymbol{\mathcal{E}}_0(\mathbf{u}_1)) \doteq \boldsymbol{\mathcal{A}}_0^I([\boldsymbol{\mathcal{E}}_0^I(\mathbf{u}_1), \mathbf{u}_1]^T) = \mathbf{0}. \tag{5.16}$$

We note that the computation of $\boldsymbol{\mathcal{E}}_0(\mathbf{u}_1)$ involves n_1 independent local nonlinear ODE solvers, using the same rationale as for the linear case. We denote by $[i]$ the time steps at level-0 in the time interval (t_1^i, t_1^{i+1}), i.e., $t_0^{m(i-1)+1}, \ldots, t_0^{m(i)-1}$. Thus, (5.16) can be computed as:

$$\boldsymbol{\mathcal{A}}_0^{[i]}(\boldsymbol{\mathcal{E}}_0^{[i]}(\mathbf{u}_1^i)) = \mathbf{0}, \quad \text{for } i = 0, \ldots, n_1 - 1. \tag{5.17}$$

Next, we apply linearization over the level-1 nonlinear Schur complement problem, e.g., Newton's method. In this case, using the implicit function theorem (see [87]), we know that

$$\begin{aligned}\boldsymbol{\mathcal{J}}_1(\bar{\mathbf{u}}_1) &\doteq \frac{\partial \boldsymbol{\mathcal{A}}_1}{\partial \mathbf{u}_1}(\bar{\mathbf{u}}_1) \\ &= \boldsymbol{\mathcal{J}}_0^{\Gamma\Gamma}(\boldsymbol{\mathcal{E}}_0(\bar{\mathbf{u}}_1)) - \boldsymbol{\mathcal{J}}_0^{I\Gamma}(\boldsymbol{\mathcal{E}}_0(\bar{\mathbf{u}}_1))\boldsymbol{\mathcal{J}}_0^{II}(\boldsymbol{\mathcal{E}}_0(\bar{\mathbf{u}}_1))^{-1}\boldsymbol{\mathcal{J}}_0^{I\Gamma}(\boldsymbol{\mathcal{E}}_0(\bar{\mathbf{u}}_1)),\end{aligned} \tag{5.18}$$

i.e., the level-1 Schur complement matrix of $\boldsymbol{\mathcal{J}}_0(\boldsymbol{\mathcal{E}}_0(\bar{\mathbf{u}}_1))$. The computation of the Schur complement operator can be done in parallel, as described in Sect. 5.3. As a result, the only difference between the Newton-Schur complement Alg. 5 and the nonlinear Schur complement-Newton Alg. 7 (for $k = 1$) is the nonlinear interior correction being computed in the second case (stated in Alg. 6). Using recursion, we can extend the nonlinear Schur complement-Newton algorithm to arbitrary levels. In Alg. 7 we state the algorithm to solve the nonlinear Schur complement at level-k using Newton's method for both the nonlinear extensions in Alg. 6 and the problem itself. Thus, the resulting method involves nested Newton iterations. Let us describe these algorithms in detail.

We first consider the nonlinear harmonic extension at level-k. Since the values at the previous level are known, i.e., $\boldsymbol{u}_k^\Gamma$ is fixed, the computation of $\boldsymbol{\mathcal{A}}_k(\boldsymbol{u}_k) = \mathbf{0}$ is to be computed solving local nonlinear ODE problems (5.17) (at level-k). Alg. 6 states the computation of such nonlinear ODE problem for a given level k and level-k time element i, i.e., (t_k^i, t_k^{i+1}). We can solve the local nonlinear ODE using Newton's method. In any case, unless $k = 0$, we do not have an explicit expression of the nonlinear ODE. Thus, in order to compute the Jacobian (Eq. (5.18) for level k), we require to compute the $k-1$ level extension in i. It leads to the deployment of $\mathrm{card}([i])$ nonlinear ODEs at the level $k-1$ in elements $j \in [i]$. Again, if the next level is not the level-0, we have to proceed recursively to solve these local problems. All these tasks are described in Alg. 6.

Once we have defined the nonlinear extension operator, we can state the level-k nonlinear Schur complement solved with Newton's method. Using the expression of the Jacobian in (5.18) (for level k), we require to compute the nonlinear extension at the next level, using Alg. 6 at all level-k elements. It leads to a level-k linear ODE to be solved, for which we can use the multilevel Schur complement approach in Alg. 4, exploiting levels $k, \ldots, \ell$ in its solution. As commented above, the level-1 nonlinear Schur complement consists in Newton iterations like in Alg. 5, but with the main difference that one performs a nonlinear interior correction of the values at level-0, solving the nonlinear ODE locally. The level-k nonlinear Schur complement requires nested nonlinear iterations to perform the nonlinear harmonic extension at that level, but the Jacobian problem to be solved only involves levels $k, \ldots, \ell$. As commented above, other linearization techniques can readily be used.

Algorithm 6: Nonlinear harmonic extension

Data: k, i, $\mathbf{v}$, $\mathbf{v}_0$
Result: $\boldsymbol{\mathcal{E}}_k^{[i]}(\mathbf{v})$, $\boldsymbol{\mathcal{J}}_k^{[i]}(\boldsymbol{\mathcal{E}}_k^{[i]}(\mathbf{v}))$, $-\boldsymbol{\mathcal{A}}_k^{[i]}(\boldsymbol{\mathcal{E}}_k^{[i]}(\mathbf{v}))$

1: **if** $k = 0$ **then**
2: Solve $\boldsymbol{\mathcal{A}}_0^{[i]}(\boldsymbol{\mathcal{E}}_0^{[i]}(\mathbf{v})) = \mathbf{0}$ using Newton's method (local nonlinear ODE), in order to get $\boldsymbol{\mathcal{E}}_0^{[i]}(\mathbf{v})$ — 0
3: Compute $\boldsymbol{\mathcal{J}}_0^{[i]}(\boldsymbol{\mathcal{E}}_0^{[i]}(\mathbf{v}))$, $-\boldsymbol{\mathcal{A}}_0^{[i]}(\boldsymbol{\mathcal{E}}_0^{[i]}(\mathbf{v}))$ — 0
4: **else**
5: % Solve $\boldsymbol{\mathcal{A}}_k^{[i]}(\boldsymbol{\mathcal{E}}_k^{[i]}(\mathbf{v}))) = \mathbf{0}$ using Newton's method as follows
6: $\mathbf{z}^{[i]} = \mathbf{v}_0^{[i]}$ % Initial guess — k
7: **while** *not convergence* **do**
8: **for** $j \in [i]$ **do**
9: Solve $\boldsymbol{\mathcal{A}}_{k-1}^{[j]}(\boldsymbol{\mathcal{E}}_{k-1}^{[j]}(\mathbf{z}^j)) = \mathbf{0}$ using Alg. 6 with $(k-1, j, \mathbf{z}^j, \mathbf{v}_0)$ — $k-1$
10: Compute $\boldsymbol{\mathcal{J}}_{k-1}^{[j]}(\boldsymbol{\mathcal{E}}_{k-1}^{[j]}(\mathbf{z}^j))$ and $-\boldsymbol{\mathcal{A}}_{k-1}^{[j]}(\boldsymbol{\mathcal{E}}_{k-1}^{[j]}(\mathbf{z}^j))$ — $k-1$
11: **end**
12: Assemble $\boldsymbol{\mathcal{J}}_k^{[i]}(\mathbf{z})$ and $-\boldsymbol{\mathcal{A}}_k^{[i]}(\mathbf{z})$ using the local contributions in l.10 and formula (5.18) — $k-1 \to k$
13: Solve $\boldsymbol{\mathcal{J}}_k^{[i]}(\mathbf{z})\mathbf{y} = -\boldsymbol{\mathcal{A}}_k^{[i]}(\mathbf{z})$ and assign $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{y}$ — k
14: **end**
15: Return $(\mathbf{z},\ \boldsymbol{\mathcal{J}}_k^{[i]}(\mathbf{z}), -\boldsymbol{\mathcal{A}}_k^{[i]}(\mathbf{z}))$ — k
16: **end**

Algorithm 7: level-k nonlinear Schur complement-Newton solver

Data: k, $\boldsymbol{u}^0$, $\mathbf{u}_0^0$
Result: $\mathbf{u}_0$ such that $\boldsymbol{\mathcal{A}}_0(\mathbf{u}_0) = 0$

1: $\mathbf{z}_0 = \mathbf{u}_0^0$ % Initial guess — k
2: **while** *not convergence* **do**
3: **for** $i = 0, \ldots, n_k$ **do**
4: Compute nonlinear harmonic extension $\boldsymbol{\mathcal{E}}_{k-1}^{[i]}(\mathbf{z}_k^i)$ and the local contributions $\boldsymbol{\mathcal{J}}_{k-1}^{[i]}(\boldsymbol{\mathcal{E}}_{k-1}^{[i]}(\mathbf{z}_k^i))$, $-\boldsymbol{\mathcal{A}}_{k-1}^{[i]}(\boldsymbol{\mathcal{E}}_{k-1}^{[i]}(\mathbf{z}_k^i))$ using Alg. 6 with $(k-1, i, \mathbf{z}_k^i, \mathbf{z})$ — $k-1, \ldots, 0$
5: **end**
6: Assemble the level-k Schur complement around $\boldsymbol{\mathcal{E}}_{k-1}(\mathbf{z}_k)$ using the local contributions in l.4 and formula (5.18) — $k-1 \to k$
7: Solve problem $\boldsymbol{\mathcal{J}}_k(\mathbf{z}_k)\mathbf{y} = -\boldsymbol{\mathcal{A}}_k(\mathbf{z}_k)$ using the multilevel Schur complement Alg. 4 with $(\boldsymbol{\mathcal{J}}_k(\mathbf{z}_k), -\boldsymbol{\mathcal{A}}_k(\mathbf{z}_k))$ — $k, \ldots, \ell$
8: Assign $\mathbf{z}_k \leftarrow \mathbf{z}_k + \mathbf{y}$ — k
9: **end**
10: Return $\mathbf{z}_0$ — $k, \ldots, \ell$

5.5 Numerical experiments

5.5.1 Experimental set-up

In this section we evaluate the weak scalability of the proposed methods. We consider the time interval $(0, T]$, which is divided into n_0 time elements. The parallel solver relies on a level-1 coarser time partition into n_1 time elements (time subdomains); thus, every level-1 time element is defined by aggregation of $\frac{n_0}{n_1}$ level-0 time elements. n_1 processors are used for the simulations in all cases. One higher level of coarsening gives a partition into n_2 elements; analogously, every level-2 time element is defined by aggregation of $\frac{n_1}{n_2}$ level-1 time elements. See Fig. 5.1 for a graphical illustration. Only a subset of n_2 processors have duties at level-2. Two-level methods involve level-0 and level-1 duties, whereas the three-level methods also involve level-2 duties. Higher levels have not been needed at the scales considered below but could be needed at largest scales to keep good weak scalability.

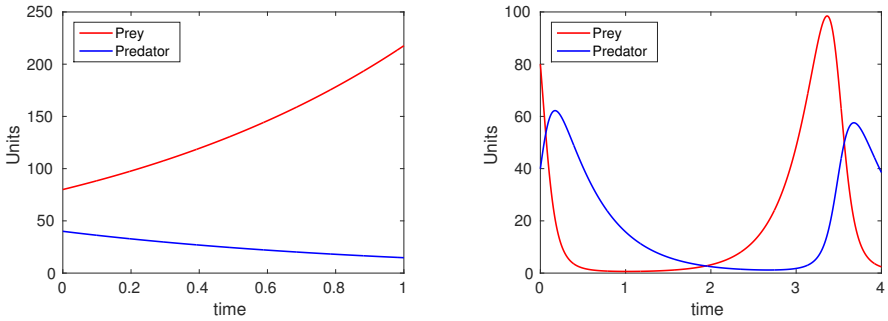
Three different approaches are considered for solving the Lotka-Volterra system of nonlinear ODEs: 1) the pure sequential approach (no parallelism is exploited, using linearization at every time step), 2) the Newton-Schur complement approach (that involves a global linearization) in Alg. 5, and 3) the 1-level nonlinear Schur complement-Newton approach in Alg. 7. Time integration is performed with the Backward Euler scheme with a constant time step. The linearization of the nonlinear problems is performed with a hybrid Picard-Newton technique, where Newton's method is activated when the discrete L_2-norm of the residual is below 10^2. (Even though all the algorithms have been stated using the full Newton's method for simplicity, its extension to Picard linearization (and hybrid approaches) is straightforward.) Sequentiality is exploited at the local level in each processor for all the different approaches (level-0 and higher level systems are lower block-triangular; see Eqs. 5.2 and 5.6, respectively). The stopping criteria for the sequential and Newton-Schur complement solvers is the reduction of the discrete L_2-norm of the nonlinear local/global residual below an absolute value of 10^{-8}. For the third approach, the global residual tolerance is fixed to 10^{-8}, while local nonlinear problems and Schur complement solutions converge below 10^{-10}.

The Lotka-Volterra equations are a system of nonlinear ODEs frequently used to describe the dynamics of biological systems in which two species interact, one as a predator and the other one as a prey. Our unknown functions, namely (u, v), represent the evolution in time of the number of units of each one of the species considered. The system of nonlinear, first order, differential equations reads

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = \alpha u - \beta uv \tag{5.19}$$

$$\frac{\mathrm{d}v(t)}{\mathrm{d}t} = \delta uv - \gamma v \tag{5.20}$$

where $\alpha, \beta, \gamma, \delta$ are positive parameters that model the interaction of the two species.



(A) No interaction. (B) $\alpha = 3$, $\beta = 0.2$, $\delta = 0.1$, $\gamma = 2$.

FIGURE 5.5: Solution to the Lotka-Volterra equations when different scenarios are considered.

The system is solved for $t \in (0, T]$. Appropriate initial conditions (initial number of preys and predators) must be provided, i.e., $u(0), v(0)$. Out of these equations, and graphically illustrated with an example in Fig. 5.5, if no interaction between the species is modelled (uncoupled linear equations), the preys grow in number while the predators decrease 5.5a. In the second case, the species interaction leads to the frequency plot 5.5b.

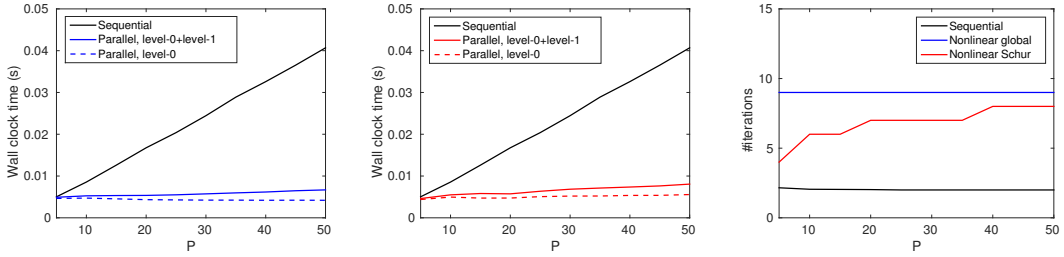
5.5.2 Two-level solvers

In this subsection, weak scalability results are presented for the solution of the Lotka-Volterra equations with a two-level method. The local problem size at level-0 is fixed to $\theta = \frac{n_0}{n_1}$. We consider a weak scalability analysis in which we keep fixed the local problem size $\frac{n_0}{n_1}$ and increase n_1, i.e., the number of subdomains (level-1 time elements). Thus, we are increasing the global time steps being used to solve the ODE. The Schur complement at level-1 has size n_1. We stop the analysis when $n_1 \approx \frac{n_0}{n_1}$, since we would require a three-level algorithm (see Sect. (5.3.3)).

Figs. 5.6 and 5.7 show a comparison between the weak scalability of the sequential solver (1 processor performs sequentially the full computation) and the parallel solvers for three different local problem sizes $\frac{n_0}{n_1}$. In these plots, the parallel solvers computing times are an aggregation of the local solvers (level-0) and the coarse solver (level-1) CPU times. The number of iterations for the sequential approach is an average value for all nonlinear time step problems. For parallel approaches, it shows the number of global nonlinear iterations required to meet the convergence requirements.

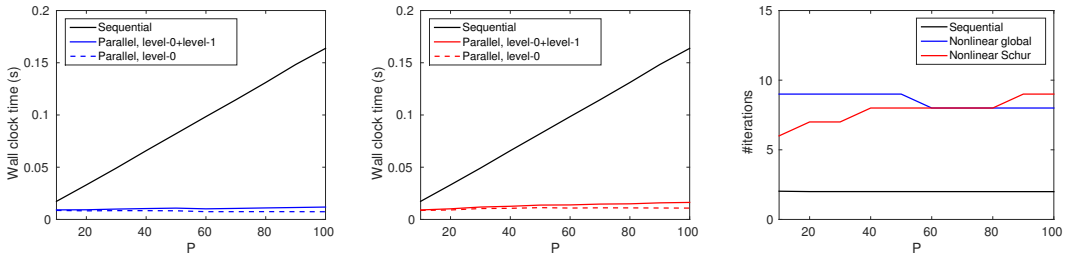
Out of these plots, we can draw some conclusions. First, parallel approaches reduce from the very beginning the time-to-solution of the simulations. Second, and most important, excellent weak scalability is observed for both parallel solvers; we can solve X times more time steps increasing X times the number of tasks, in approximately the same CPU time.

Regarding the Newton-Schur complement approach, one can observe that the CPU time at level-1 is always less than half the CPU time at level-0. It is due to the fact



(A) Newton-Schur complement solver. (B) Nonlinear Schur complement solver. (C) Number of iterations.

FIGURE 5.6: Weak scalability for local problem size $n_0/n_1 = 50$. The problem is solved with $\alpha = 3$, $\beta = 0.2$, $\delta = 0.1$, $\gamma = 2$ and $t \in (0, 3]$. Initial guess $u(0) = 10$, $v(0) = 40$.



(A) Newton-Schur complement solver. (B) Nonlinear Schur complement solver. (C) Number of iterations.

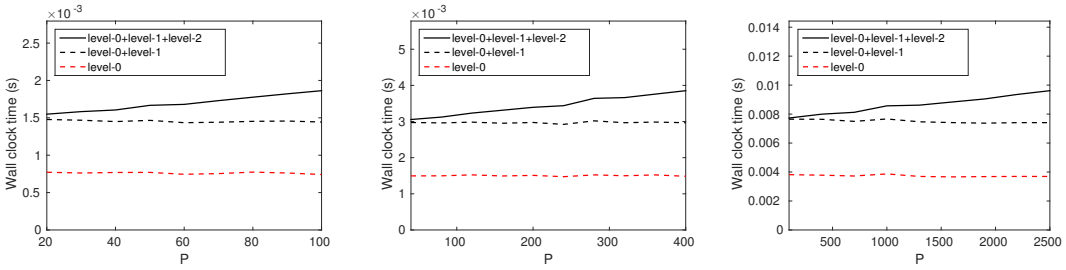
FIGURE 5.7: Weak scalability for $n_0/n_1 = 100$. The problem is solved with $\alpha = 3$, $\beta = 0.2$, $\delta = 0.1$, $\gamma = 2$ and $t \in (0, 3]$. Initial guess $u(0) = 10$, $v(0) = 40$

that we stop the scalability analysis before the coarse space is larger than the level-0 local problems and the fact that two local solvers are required at level-0 (see Alg. 3) per nonlinear iteration. For the nonlinear Schur complement-Newton approach, the number of global iterations to converge is lower in most cases, but it involves nested nonlinear iterations. The Newton-Schur complement approach is slightly faster than the nonlinear Schur complement-Newton one. In any case, it can strongly be affected by the tolerances being chosen in the nonlinear Schur complement-Newton method. Clearly, increasing the load per processor (local problem sizes), the benefit of the parallel solvers compared to the sequential approach becomes more notorious.

Results are presented up to the limit of $n_1 \approx \frac{n_1}{n_0}$, where the coarse problem size grows above the local problem sizes. Aiming to exploit further concurrency, the next section is devoted to show numerical results in a multilevel approach.

5.5.3 Multilevel solver

The multilevel (three-level) approach is be activated when the coarse problem size exceeds the local problem size. The first coarsening leads to a computation with n_1 level-0 local problems of size $\frac{n_0}{n_1}$ and a coarse problem of size n_1 (level-1 partition), such that $n_1 > \frac{n_0}{n_1}$.



(A) $n_0/n_1 = n_1/n_2 = 10$. (B) $n_0/n_1 = n_1/n_2 = 20$. (C) $n_0/n_1 = n_1/n_2 = 50$.

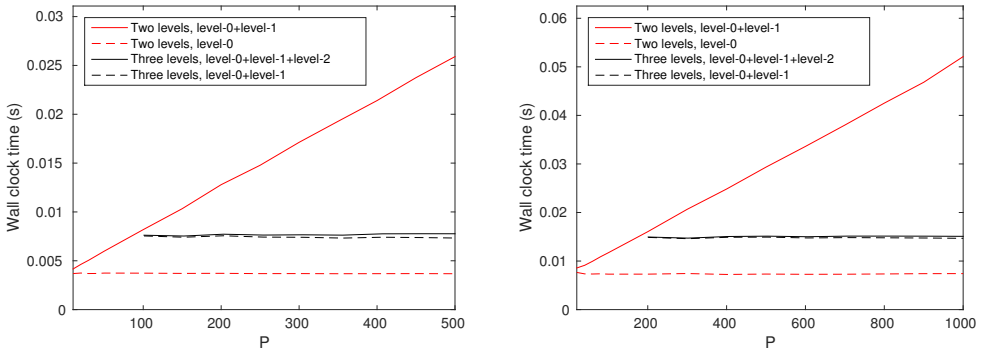
FIGURE 5.8: Three-level ($\ell = 2$) Schur complement solver with different local sizes.

At this point, another level of coarsening is introduced to exploit further concurrency. We consider a coarsening of the level-1 time partition into n_2 local problems of size $\frac{n_1}{n_2}$ and a coarse problem of size n_2 such that $n_2 < \frac{n_1}{n_2}$ (beyond this limit, a four-level method would be required). In the following plots, the parallel solvers computing times are an aggregation of the level-0 local solver CPU time (n_1 parallel tasks), the level-1 local solver CPU time (n_2 parallel tasks) and the level-2 global solver CPU times (Schur complement sequential solve in one processor).

In Fig. 5.8, CPU times for the three-level (i.e., $\ell = 2$) Schur complement solve are shown. Out of the plots, it can be observed that local solves computing times for the level-0 and level-1 tasks are of the same order, since local problem sizes is kept constant for both levels. The Schur complement computation (level-2) time grows with the number of processors, as expected. Again, the level-2 the Schur complement CPU time in Fig. 5.8 does not exceed half the CPU time of level-0 and level-1 local solves CPU time, due to the same reasons commented above, since $n_2 < \frac{n_1}{n_2}$. Beyond this limit, i.e., for $n_2 > \frac{n_1}{n_2}$, another level would be needed.

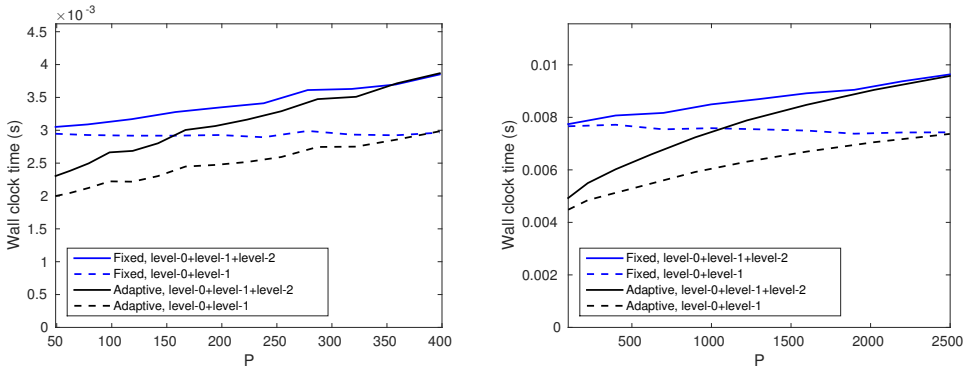
In Fig. 5.9, a comparison between the two-level ($\ell = 1$) and the three-level ($\ell = 2$) approaches is presented. In the first part of the plot, results are shown for the two-level approach only, since the size of the level-1 problem is still below the local sizes of the level-0 problems. The three-level technique is activated when the size of the coarse problem is two times the size of the local problems of the finest level, leading to a level-2 partition into two time elements. Out of the plots, the multilevel approach shows much better efficiency than the two-level approach. It is important to note that to include additional levels do not affect nonlinear convergence of Newton-Schur and 1-level nonlinear Schur complement-Newton methods, but it has benefits in the computation of the linear ODE systems. As a result, more levels show better computing times in these approaches.

In Fig. 5.10, an adaptive coarsening at level-2 is presented. Since $n_2 < \frac{n_1}{n_2}$ in the plot, we can reduce the coarsening from level-1 to level-2 to better balance the problems at these levels. It implies to enforce that $n_2 = \frac{n_1}{n_2}$. As expected, this choice is more efficient compared to the fixed size approach, when the size of the level-2 problem is below level-1 local problem sizes.



(A) $n_0/n_1 = 50$, $n_1/n_2 = 50$. (B) $n_0/n_1 = 100$, $n_1/n_2 = 100$.

FIGURE 5.9: Comparison between the two-level ($\ell = 1$) and the three-level ($\ell = 2$) Schur complement solver with different local sizes. Times for local solvers in the level-0, local solvers in the level-1 and the Schur complement solve in level-2.



(A) $n_0/n_1 = 20$. (B) $n_0/n_1 = 50$.

FIGURE 5.10: Comparison between the fixed coarsening and the adaptive coarsening between level-1 and level-2 for different level-0 local problem sizes. The adaptive coarsening uses $n_2 = n_1/n_2 = \sqrt{n_1}$.

5.6 Conclusions

In this chapter, we have considered a parallel-in-time multilevel direct method and two iterative parallel-in-time nonlinear solvers for the numerical approximation of ODEs using parallel computations. The time-parallel direct method computes explicitly multilevel Schur complements. It allows for arbitrary high levels of concurrency and has very good theoretical speed-up ratios compared to traditional parareal methods. The method can be considered a parareal method with an automatic definition of the coarse solver. However, such definition makes the method to converge in one iteration, i.e., it is a direct method, and thus does not suffer from the poor parallel efficiency of parareal schemes. The proposed scheme can be applied to θ-methods, DG methods, RK methods, and BDF methods. Next, we consider nonlinear ODEs, and propose two different strategies. First, we consider a Newton method over the global nonlinear ODE, using the multilevel Schur complement solver at every nonlinear iteration. Second, we state the global nonlinear

problem in terms of the nonlinear Schur complement (at an arbitrary level), and perform nonlinear iterations over it. Such approach leads to nested nonlinear multilevel iterations.

A detailed set of numerical experiments has been considered. Out of these results, the proposed methodology is observed to exhibit excellent scalability properties. The methods are weakly scalable in time, i.e., increasing X times the number of MPI tasks one can solve X times more time steps, in approximately the same amount of time, which is a key property to reduce time-to-solution in ODE simulations with heavy time stepping.

In Chapter 6, we will explore the combination of these ideas with space-parallel highly scalable BDDC implementations [18, 23, 22] to design space-time preconditioners for transient PDEs.

Chapter 6

Space-Time Balancing Domain Decomposition by Constraints preconditioners

In this chapter, we propose two-level space-time domain decomposition preconditioners for parabolic problems discretized using finite elements. They are motivated as an extension to space-time of balancing domain decomposition by constraints (BDDC) preconditioners. The key ingredients to be defined are the sub-assembled space and operator, the coarse degrees of freedom (DOFs) in which we want to enforce continuity among subdomains at the preconditioner level, and the transfer operator from the sub-assembled to the original finite element space. With regard to the sub-assembled operator, a perturbation of the time derivative is needed to end up with a well-posed preconditioner. The set of coarse DOFs includes the time average (at the space-time subdomain) of classical space constraints plus new constraints between consecutive subdomains in time. Numerical experiments show that the proposed schemes are weakly scalable in time, i.e., we can efficiently exploit increasing computational resources to solve more time steps in the same total elapsed time. Further, the scheme is also weakly space-time scalable, since it leads to asymptotically constant iterations when solving larger problems both in space and time. Excellent wall clock time weak scalability is achieved for space-time parallel solvers on some thousands of cores.

6.1 Introduction

At the beginning of the next decade supercomputers are expected to reach a peak performance of one exaflop/s, which implies a 100 times improvement with respect to current supercomputers. This improvement will not be based on faster processors, but on a much larger number of processors (in a broad sense). This situation will certainly have an impact in large scale computational science and engineering. Parallel algorithms will be required to exhibit much higher levels of concurrency, keeping good scalability properties.

In mesh-based implicit simulations, e.g., finite element (FE), finite volume, or finite difference methods, one ends up with a linear system to be solved. The linear system solve is a bottleneck of the simulation pipeline and weakly scalable algorithms require complex mathematical approaches, like algebraic multigrid (AMG) or multilevel domain decomposition (DD) techniques. When dealing with transient problems, since information always moves forward in time, one can exploit sequentiality. At every time step one has to solve a spatial problem before proceeding to the next step and parallelism can be exploited at the linear system solve. Although parallel-in-time methods are becoming popular, the sequential-in-time approach is the standard procedure in scientific simulations. However, the tremendous amounts of parallelism to be exploited in the near future certainly motivates to change this paradigm, since further concurrency opportunities must be sought.

As presented in Chapter 5, in transient simulations, a natural way to go is to exploit concurrency not only in space but also in time. The idea is to develop space-time solvers that do not exploit sequentiality (at least at the global level) and thus provide the solution in space at all time values in one shot. Space-time parallelism is a topic that is receiving rapidly increasing attention. Different iterative methods have been considered so far. One approach is to use the Parareal method [93], which is a time-only parallel algorithm, combined with a parallel space preconditioner (see, e.g., [73]). Another space-time algorithm is PFASST [63, 100], which combines a spectral deferred correction time integration with a nonlinear multigrid (MG) spatial solver; the viability of the PFASST method has been proved in [132] at JUQUEEN. Weakly scalable space-time AMG methods can be found in [74, 145, 65].

The multilevel (ML) BDDC preconditioner [98, 140] has recently been proved to be an excellent candidate for extreme scale simulations in [23], where a recursive implementation that permits overlapping among communication and computation at all levels has scaled up to almost half a million cores and two million subdomains (MPI tasks), for both structured and unstructured meshes with tens of billions of elements. The key ingredient of these methods relies on the definition of a FE space with relaxed inter-element continuity. These spaces are defined by choosing the quantities to be continuous among processors, i.e., the DOFs [57]. As far as we know, scalable DD methods in space-time have not been considered so far.

In this chapter, we develop *weakly scalable space-time preconditioners based on BDDC methods*. In order to do that, we extend the key ingredients in the space-parallel BDDC framework, namely the sub-assembled space and operator, coarse DOFs, and transfer operators, to space-time. We prove that the resulting method only involves a set of well-posed problems, and time causality can still be exploited at the local level. We have solved a set of linear and nonlinear problems that show the excellent weak scalability of the proposed preconditioners.

The outline of the chapter is as follows. In Sect. 6.2 we set the problem and introduce notation. In Sect. 6.3 we introduce the classical space-parallel BDDC preconditioners.

In Sect. 6.4 we develop space-time (ST) BDDC (STBDDC) preconditioners. In Sect. 6.5 we present a detailed set of numerical experiments showing the scalability properties of the proposed methods. Finally, in Sect. 6.6 we draw some conclusions.

6.2 Problem setting

In this section, we introduce the problem to be solved, the partition of the domain in space and time, and the space and time discretization. In the sequel, calligraphic letters are used for operators. $\mathcal{M}$ denotes mass matrix operators related to the time derivative discretization, $\mathcal{K}$ is used for the rest of terms in the PDE operator, and $\mathcal{A}$ is used for the sum of these two operators. Given an operator $\mathcal{A}$, we will use the notation $\mathcal{A}(u,v) \doteq \langle \mathcal{A}u, v\rangle$. Uppercase letters ($V,\ldots$) are used for (FE-type) functional spaces whereas functions are represented by lowercase letters ($v,\ldots$). We use classical functional analysis notation for Sobolev spaces.

6.2.1 Domain partitions

We consider a bounded space-time domain $\Omega \times (0,T]$, where Ω is an open polyhedral domain $\Omega \subset \mathbb{R}^d$, d being the space dimension. Let us construct two partitions of Ω, a *fine* partition into *elements* and a *coarse* partition into *subdomains*. The partition of Ω into elements is represented by θ. In space, elements $e \in \theta$ are tetrahedra/hexahedra for $d = 3$ or triangles/quadrilaterals for $d = 2$. The *coarse* partition Θ of Ω into subdomains is obtained by *aggregation of elements* in θ, i.e., there is an element partition $\theta_\omega \doteq \{e \in \theta : e \subset \omega\} \subset \theta$ for any $\omega \in \Theta$. The interface of the subdomain partition is $\Gamma \doteq \cup_{\omega\in\Theta}\partial\omega \setminus \partial\Omega$.

For the time interval $(0,T]$, we define a time partition $\{0 = t^0, t^1, \ldots, t^K = T\}$ into K time elements. We denote the k-th element by $\delta_k \doteq (t_{k-1}, t_k]$, for $k = 1,\ldots,K$.

6.2.2 Space-time discretization

Let us consider as a model problem the following transient convection-diffusion-reaction (CDR) equation: find $u \in H^1(\Omega)$ such that

$$\begin{aligned} &\partial_t u - \nabla\cdot\nu\nabla u + \beta\cdot\nabla u + \sigma u = f \quad \text{on } \Omega, \quad \text{almost everywhere in } (0,T], \\ &u = g \quad \text{in } \partial\Omega, \qquad u(0,x) = u^0, \end{aligned} \tag{6.1}$$

where ν and σ are positive constants, $\beta \in \mathbb{R}^d$, and $f \in H^{-1}(\Omega)$. We supplement this system with the initial condition $u(0) = u^0$. Homogeneous Dirichlet data is assumed for the sake of clarity in the exposition, but its extension to the general case is obvious. Besides, let us consider $\beta = 0$ and $\sigma = 0$ for simplicity in the exposition of the algorithm. In Sect. 6.5 we will take a nonlinear viscosity $\nu(u) = \nu_0|\nabla u|^p$ (with $p \geq 0$ and $\nu_0 > 0$), i.e., the transient p-Laplacian problem.

For the space discretization, we use H^1-conforming FE spaces on conforming meshes with strong imposition of Dirichlet conditions. The discontinuous Galerkin (DG) case will not be considered in this work, but we refer to [60] for the definition of BDDC methods for DG discretizations. We define $\bar{V} \subset H^1_0(\Omega)$ as the global FE space related to the FE mesh θ. Further, we define the FE-wise operators:

$$\mathcal{M}_e(u,v) \doteq \int_e uv, \quad \mathcal{K}_e(u,v) \doteq \int_e \nu \nabla u \cdot \nabla v,$$
$$\mathcal{A}_e(u,v) \doteq \mathcal{M}_e(u,v) + |\delta_k| \mathcal{K}_e(u,v).$$

The global FE problem $\bar{\mathcal{A}} : \bar{V} \to \bar{V}$ can be written as the sum of element contributions, i.e.,

$$\bar{\mathcal{A}}(u,v) \doteq \sum_{e \in \theta} \mathcal{A}_e(u,v), \qquad \text{for } u, v \in \bar{V}. \qquad \text{(Analogously for } \bar{\mathcal{M}} \text{ and } \bar{\mathcal{K}}.)$$

In time, we make use of a collocation-type method of lines. For the sake of clarity, we will use the Backward-Euler time integration scheme. In any case, the resulting preconditioner can readily be applied to any θ-method or Runge-Kutta method. We are interested in solving the following fully discrete system: given $u(t^0) = 0$, find at every time step $k = 1, \ldots, K$ the solution $u(t^k) \in \bar{V}$ of

$$\bar{\mathcal{A}} u(t^k) = \bar{\mathcal{M}} u(t^k) + |\delta_k| \bar{\mathcal{K}} u(t^k) = \bar{g}(t^k), \qquad \text{for any } v \in \bar{V}, \tag{6.2}$$

with $\bar{g}(t^k) \doteq |\delta_k| \bar{f}(t^k) + \bar{\mathcal{M}} u(t^{k-1}) \in \bar{V}'$, where $\bar{V}'$ denotes the dual space of $\bar{V}$. Non-homogeneous boundary conditions can be enforced by simply modifying the right-hand side at t^1, i.e., $\bar{g}(t^1) \doteq |\delta_k| \bar{f}(t^1) + \bar{\mathcal{M}} u^0 \in \bar{V}'$. Such imposition of boundary conditions, i.e., by enforcing homogeneous conditions in the FE space plus the modification of the right-hand side, is better suited for the space-time framework. (We note that this is the way strong Dirichlet boundary conditions are imposed in FE codes.)

6.3 Space BDDC preconditioning

In this section, we present first a parallel solver for the transient problem Eq. (6.2), which combines a sequential-in-time approach with a space-parallel BDDC preconditioned Krylov solver at every time step [57]. It will serve to introduce space-parallel BDDC methods and related concepts that will be required in the space-time section. BDDC preconditioners involve the definition of three key ingredients: (1) a sub-assembled problem that involves independent subdomain corrections, (2) a set of coarse DOFs and the corresponding subspace of functions with continuous coarse DOFs among subdomains, and (3) the interior correction and transfer operators. Let us elaborate these ingredients.

6.3.1 Sub-assembled problem

Non-overlapping DD preconditioners rely on the definition of a sub-assembled FE problem, in which contributions between subdomains have not been assembled. In order to do so, at every subdomain $\omega \in \Theta$, we consider the FE space V_ω associated to the element partition θ_ω with homogeneous boundary conditions on $\partial\omega \cap \partial\Omega$. One can define the subdomain operator $\mathcal{A}_\omega(u,v) = \sum_{e\in\theta_\omega} \mathcal{A}_e(u,v)$, for $u, v \in V_\omega$. (Analogously for $\mathcal{M}_\omega$ and $\mathcal{K}_\omega$.)

Subdomain spaces lead to the sub-assembled space of functions $V \doteq \Pi_{\omega\in\Theta} V_\omega$. For any $u \in V$, we define its restriction to a subdomain $\omega \in \Theta$ as u_ω. Any function $u \in V$ can be represented by its unique decomposition into subdomain functions as $\{u_\omega \in V_\omega\}_{\omega\in\Theta}$. We also define the sub-assembled operator $\mathcal{A}(u,v) \doteq \Pi_{\omega\in\Theta}\mathcal{A}_\omega(u_\omega, v_\omega)$. (Analogously for $\mathcal{M}$ and $\mathcal{K}$.)

With these definitions, $\bar{V}$ can be understood as the subspace of functions in V that are continuous on the interface Γ, and $\bar{\mathcal{A}}$ can be interpreted as the Galerkin projection of $\mathcal{A}$ onto $\bar{V}$. We note that θ and the FE type defines $\bar{V}$, whereas Θ is also required to define the local spaces $\{V_\omega\}_{\omega\in\Theta}$ and the sub-assembled space V, respectively.

At this point, we can state the following sub-assembled problem: given $g \in V'$, find $u \in V$ such that

$$\mathcal{A}_\omega u_\omega = \mathcal{M}_\omega u_\omega + |\delta_k|\mathcal{K}_\omega u_\omega = g_\omega, \quad \omega \in \Theta. \tag{6.3}$$

With the previous notation, we can write the sub-assembled problem in a compact manner as $\mathcal{A}u = g$.

6.3.2 Coarse DOFs

A key ingredient in DD preconditioners is to classify the set of nodes of the FE space $\bar{V}$. The interface ∂e of every FE in the mesh θ can be decomposed into vertices, edges, and faces. By a simple classification of these entities, based on the set of subdomains that contain them, one can also split the interface Γ into faces, edges, and vertices (at the subdomain level), that will be called geometrical objects. We represent the set of geometrical objects by Λ. In all cases, edges and faces are open sets in their corresponding dimension. By construction, faces belong to two subdomains and edges belong to more than two subdomains in three-dimensional problems. This classification of Ω into objects automatically leads to a partition of interface DOFs into DOF objects, due to the fact that every DOF in a FE does belong to only one geometrical entity. These definitions are heavily used in DD preconditioners (see, e.g., [138, p. 88]).

Next, we associate to some (or all) of these geometrical objects a *coarse* DOF. In BDDC methods, we usually take as coarse DOFs mean values on a subset of objects Λ_O. Typical choices of Λ_O are $\Lambda_O \doteq \Lambda_C$, when only corners are considered, $\Lambda_O \doteq \Lambda_C \cup \Lambda_E$, when corners and edges are considered, or $\Lambda_O \doteq \Lambda$, when corners, edges, and faces are considered. These choices lead to three common variants of the BDDC method referred

as BDDC(c), BDDC(ce) and BDDC(cef), respectively. This classification of DOFs into objects can be restricted to any subdomain $\omega \in \Theta$, leading to the set of subdomain objects $\Lambda_O(\omega)$.

With the classification of the interface nodes and the choice of the objects in Λ_O, we can define the coarse DOFs and the corresponding BDDC space. Given an object $\lambda \in \Lambda_O(\omega)$, let us define its restriction operator τ_λ^ω on a function $u \in V_\omega$ as follows: $\tau_\lambda^\omega(u)(\xi) = u(\xi)$ for a node ξ that belongs to the geometrical object λ, and zero otherwise. We define the BDDC space $\widetilde{V} \subset V$ as the subspace of functions $v \in V$ such that the constraint

$$\int_\lambda \tau_\lambda^\omega(v_\omega) \qquad \text{is identical for all } \omega \in \text{neigh}(\lambda), \tag{6.4}$$

where $\text{neigh}(\lambda)$ stands for the set of subdomains that contain the object λ. (The integral on λ is just the value at the vertex, when λ is a vertex.) Thus, every $\lambda \in \Lambda_O$ defines a coarse DOF value Eq. (6.4) that is continuous among subdomains. Further, we can define the BDDC operator $\widetilde{\mathcal{A}}$ as the Galerkin projection of $\mathcal{A}$ onto $\widetilde{V}$.

6.3.3 Transfer operator

The next step is to define a transfer operator from the sub-assembled space V to the continuous space $\bar{V}$. The transfer operator is the composition of a weighting operator and a harmonic extension operator.

1. The weighting operator $\mathcal{W}$ takes a function $u \in V$ and computes mean values on interface nodes, i.e.,

$$\mathcal{W}u(\xi) \doteq \frac{\sum_{\omega \in \text{neigh}(\xi)} u_\omega(\xi)}{|\text{neigh}(\xi)|}, \tag{6.5}$$

   at every node ξ of the FE mesh θ, where $\text{neigh}(\xi)$ stands for the set of subdomains that contain the node ξ. It leads to a continuous function $\mathcal{W}u \in \bar{V}$. It is clear that this operator only modifies the DOFs on the interface. Other choices can be defined for non-constant physical coefficients.

2. Next, let us define the bubble space $V_0 \doteq \{v \in V : v = 0 \text{ on } \Gamma\}$ and the Galerkin projection $\mathcal{A}_0$ of $\mathcal{A}$ onto V_0. We also define the trivial injection $\mathcal{I}_0$ from V_0 to $\bar{V}$. The *harmonic* extension reads as $\mathcal{E}v \doteq (I - \mathcal{I}_0\mathcal{A}_0^{-1}\mathcal{I}_0^T\bar{\mathcal{A}})v$. The computation of $\mathcal{A}_0^{-1}$ involves to solve problem Eq. (6.3) with homogeneous boundary conditions on Γ. This operator corrects interior DOFs only.

The transfer operator $\mathcal{Q} : V \to \bar{V}$ is defined as $\mathcal{Q} \doteq \mathcal{E}\mathcal{W}$.

6.3.4 Space-parallel preconditioner

With all these ingredients, we are now in position to define the BDDC preconditioner. This preconditioner is an additive Schwarz preconditioner (see, e.g. [138, chap. 2]), with

corrections in V_0 and the BDDC correction in $\widetilde{V}$ with the transfer $\mathcal{Q}$. As a result, the BDDC preconditioner reads as:

$$\mathcal{B} \doteq \mathcal{I}_0 \mathcal{A}_0^{-1} \mathcal{I}_0^T + \mathcal{Q} \widetilde{\mathcal{A}}^{-1} \mathcal{Q}^T. \tag{6.6}$$

6.4 Space-time BDDC preconditioning

As commented in Sect. 6.1, the huge amounts of parallelism of future supercomputers will require to seek for additional concurrency. In the simulation of Eq. (6.2) using the space-parallel preconditioner Eq. (6.6), we are using a sequential-in-time approach by exploiting time causality. The objective of this section is to solve Eq. (6.2) at all time steps in one shot, by using a space-time-parallel preconditioner and a Krylov subspace method for non-symmetric problems. In order to do so, we want to extend the BDDC framework to ST.

In the following, we will use bold symbols, e.g., $\boldsymbol{u}$, $\boldsymbol{V}$, or $\boldsymbol{\mathcal{A}}$, for space-time functions, functional spaces, and operators, respectively. $\boldsymbol{I}$ is the identity matrix, which can have different dimension in different appearances.

First, we must start with a space-time partition of $\Omega \times (0, T]$. We consider a time subdomain partition by aggregation of time elements, $\{0 = T_0, T_1, \ldots, T_N = T\}$ into N time subdomains. We denote the n-th subdomain as $\Delta_n \doteq (T_{n-1}, T_n]$, for $n = 1, \ldots, N$. By definition, Δ_n admits a partition into K_n time elements $\{T_{n-1} = t_n^0, \ldots, t_n^{K_n} = T_n\}$. Next, we define the space-time subdomain partition as the Cartesian product of the space subdomain partition Θ and the time subdomain partition defined above; for every space subdomain ω and time subdomain Δ_n, we have the space-time subdomain $\boldsymbol{\omega}_n \doteq \omega \times \Delta_n$.

The global space of continuous space-time functions in which we want to solve Eq. (6.2) is the FE space $\bar{V}$ of spatial functions times $K+1$ time steps, i.e., $\bar{\boldsymbol{V}} \doteq [\bar{V}]^{K+1}$, constrained to zero initial condition. Thus, by definition, $\boldsymbol{u} \in \bar{\boldsymbol{V}}$ can be expressed as $\boldsymbol{u} = (u(t^0) = 0, u(t^1), \ldots, u(t^K))$, and the original problem Eq. (6.2) (for all time step values) can be stated in a compact manner as

$$\bar{\boldsymbol{\mathcal{A}}} \boldsymbol{u} = \bar{\boldsymbol{f}}, \qquad \text{in } \bar{\boldsymbol{V}}. \tag{6.7}$$

In order to define the STBDDC preconditioner for Eq. (6.7), we will use the same structure as above, extending the three ingredients in Sect. 6.3 to the space-time case.

6.4.1 Sub-assembled problem

Using the space-time partition above, the trial (and test) space for the local space-time subdomain $\boldsymbol{\omega}_n$ is $\boldsymbol{V}_{\boldsymbol{\omega}_n} \doteq [V_\omega]^{K_n+1}$. Thus, by definition, $\boldsymbol{u}_{\boldsymbol{\omega}_n} \in \boldsymbol{V}_{\boldsymbol{\omega}_n}$ can be expressed as $\boldsymbol{u}_{\boldsymbol{\omega}_n} = (u_\omega(t_n^0), \ldots, u_\omega(t_n^{K_n}))$. Analogously, the sub-assembled space is

$\boldsymbol{V} \doteq \Pi_{n=1}^{N} \Pi_{\omega \in \Theta} \boldsymbol{V}_{\boldsymbol{\omega}_n}$. Let us note that, using these definitions, functions in $\boldsymbol{V}$ have duplicated values at T_n, for $n = 1, \ldots, N-1$. The global space of continuous space-time functions $\bar{\boldsymbol{V}}$ can be understood as the subspace of functions in $\boldsymbol{V}$ that are continuous on the space-time interface $\Gamma \times T_n$, for $n = 1, \ldots, N-1$.

Now, we propose the following sub-assembled problem in $\boldsymbol{V}$: find the solution $\boldsymbol{u} \in \boldsymbol{V}$ such that, at every $\boldsymbol{\omega}_n$ in the space-time partition, it satisfies the space-time problem

$$\begin{aligned} &\sum_{k=1}^{K_n} \left\{ \mathcal{M}_\omega(u_\omega(t_n^k) - u_\omega(t_n^{k-1}), v_\omega(t_n^k)) + |\delta_n^k| \mathcal{K}_\omega(u_\omega(t_n^k), v_\omega(t_n^k)) \right\} \\ &+ \frac{(1 - \mathrm{Kr}_{1,n})}{2} \mathcal{M}_\omega(u_\omega(t_n^0), v_\omega(t_n^0)) - \frac{(1 - \mathrm{Kr}_{N,n})}{2} \mathcal{M}_\omega(u_\omega(t_n^{K_n}), v_\omega(t_n^{K_n})) \\ &= \sum_{k=1}^{K_n} |\delta_n^k| f_\omega(t_n^k)(v_\omega(t_n^{K_n})), \end{aligned} \tag{6.8}$$

for any $\boldsymbol{v} \in \boldsymbol{V}$, where $\mathrm{Kr}_{i,j}$ is the Kronecker delta. Note that the perturbation terms in the second line of Eq. (6.8) are introduced only on time interfaces, i.e., in the first and last time steps of the time subinterval, as long as the corresponding time step is not a time domain boundary. For subdomains with $n = 1$, and thus $t_n^0 = 0$, the first stabilization term vanishes. Analogously, the second stabilization term vanishes for $n = N$ and $t_n^{K_n} = T$. We can write Eq. (6.8) in compact manner as

$$\boldsymbol{\mathcal{A}} \boldsymbol{u} = \boldsymbol{f} \qquad \text{in } \boldsymbol{V}. \tag{6.9}$$

The motivation of the perturbation terms is to have a positive semi-definite sub-assembled operator. In any case, the perturbation terms are such that, after inter-subdomain assembly, we recover the original space, i.e., $\boldsymbol{\mathcal{A}}$ is in fact a sub-assembled operator with respect to $\bar{\boldsymbol{\mathcal{A}}}$, since interface perturbations between subdomains cancel out. We prove that these properties hold.

Proposition 6.4.1. *The Galerkin projection of the sub-assembled space-time problem Eq. (6.9) onto $\bar{\boldsymbol{V}}$ reduces to the original problem Eq. (6.7). Further, the sub-assembled operator $\boldsymbol{\mathcal{A}}$ is positive definite.*

Proof. In order to prove the equivalence, we need to show that $\bar{\boldsymbol{\mathcal{A}}}$ is the Galerkin projection of $\boldsymbol{\mathcal{A}}$ onto $\bar{\boldsymbol{V}}$, which amounts to say that the perturbation terms vanish for $\boldsymbol{u} \in \bar{\boldsymbol{V}}$. First, we note that the following equality

$$\sum_{n=1}^{N} \left(\frac{(1 - \mathrm{Kr}_{1,n})}{2} u_\omega(t_n^0) - \frac{(1 - \mathrm{Kr}_{N,n})}{2} u_\omega(t_n^{K_n}) \right) = 0$$

holds for functions that are continuous in time, since $u_\omega(t_{n-1}^{K_{n-1}}) = u_\omega(t_n^0)$ for $n = 2, \ldots, N$. On the other hand, multiplying the right-hand of Eq. (6.8) against $\boldsymbol{v} = \boldsymbol{u}$,

using the fact that $(a-b)a = \frac{1}{2}(a^2-b^2) + \frac{1}{2}(a-b)^2$, we get

$$\boldsymbol{\mathcal{A}}(\boldsymbol{u},\boldsymbol{u}) = \frac{1}{2}\|u(T)\|^2 + \sum_{n=1}^{N}\sum_{k=1}^{K}\left(\frac{1}{2}\|u(t_n^k) - u(t_n^{k-1})\|^2 + |\delta_n^k|\mathcal{K}(u(t_n^k), u(t_n^k))\right). \quad (6.10)$$

Since $\mathcal{K}$ is positive semi-definite, $\boldsymbol{\mathcal{A}}$ is positive semi-definite. On the other hand, $\boldsymbol{\mathcal{A}}$ is a lower block triangular matrix. Restricted to one subdomain block, it has diagonal blocks $\frac{1}{2}\mathcal{M}_\omega$ at the first time step, $\mathcal{M}_\omega + |\delta_k|\mathcal{K}_\omega$ at intermediate time steps, and $\frac{1}{2}\mathcal{M}_\omega + |\delta_k|\mathcal{K}_\omega$ at the last time step. Since all these matrices are invertible, $\boldsymbol{\mathcal{A}}$ is non-singular. Further, $\boldsymbol{\mathcal{A}}^T$ is an upper triangular non-singular matrix. As a result, $\boldsymbol{\mathcal{A}}$ is positive-definite. □

6.4.2 Coarse DOFs

Let us define the continuity to be enforced among space-time subdomains. Let us consider a set of space objects Λ_O (see Sect. 6.3). We define $\widetilde{\boldsymbol{V}} \subset \boldsymbol{V}$ as the subspace of functions $\boldsymbol{v} \in \boldsymbol{V}$ such that the constraint

$$\sum_{k=1}^{K_n-1} |\delta_n^k| \int_\lambda \tau_\lambda^\omega(v_\omega(t_n^k)) \qquad \text{is identical for all } \omega \in \text{neigh}(\lambda), \quad (6.11)$$

holds for every $\lambda \in \Lambda_O$, and

$$\int_\omega v_\omega(t_n^0) = \int_\omega v_\omega(t_{n-1}^{K_{n-1}}), \qquad \text{for all } \omega \in \Theta, \quad n = 2,\dots,N. \quad (6.12)$$

The first set of constraints are the mean value of the space constraints in Eq. (6.4) over time sub-intervals Δ_n. The second constraint enforces continuity between two consecutive-in-time subdomains $\boldsymbol{\omega}_{n-1}$ and $\boldsymbol{\omega}_n$ of the mean value of the function on their corresponding space subdomain ω. The Galerkin projection of $\boldsymbol{\mathcal{A}}$ onto $\widetilde{\boldsymbol{V}}$ is denoted by $\widetilde{\boldsymbol{\mathcal{A}}}$.

Additionally, motivated by a space-time definition of objects, i.e., applying the object generation above to space-time meshes, we could also enforce the continuity of the coarse DOFs

$$\int_\lambda \tau_\lambda^\omega(v_\omega(t_n^0)),\ n = 2,\dots,N \quad \text{and} \quad \int_\lambda \tau_\lambda^\omega(v_\omega(t_n^{K_n})),\ n = 1,\dots,N-1, \quad (6.13)$$

for every $\lambda \in \Lambda_O$. Thus, we are enforcing pointwise in time (in comparison to the mean values in Eq. (6.11)) space constraints on time interfaces. Figure 6.1 illustrates the resulting space-time set of objects where continuity is to be enforced in a sub-assembled space $\boldsymbol{V}$.

6.4.3 Transfer operator

Next, we have to define a transfer operator from $\boldsymbol{V}$ to $\bar{\boldsymbol{V}}$, and the concept of harmonic extension in the space-time setting.

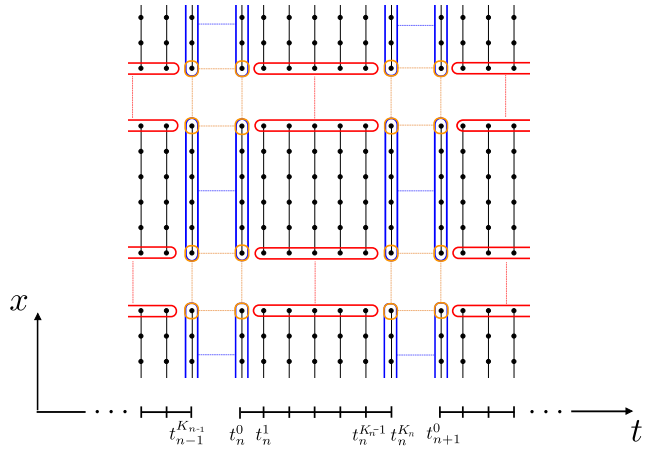


FIGURE 6.1: Continuity to be enforced among space-time subdomains, for the 1-dimensional spatial domain case. The sets of nodes in red are related to the space constraints time averages over time sub-intervals in Eq. (6.11), the ones in blue are the space mean value constraints on time interfaces in Eq. (6.12), and the ones in orange are spatial constraints on time interfaces in Eq. (6.13).

1. In order to define the space-time weighting operator, we make use of the spatial-only definition in Eq. (6.5). Let us define the subdomain restriction of the weighting operator as $\mathcal{W}_\omega u \doteq (\mathcal{W}u)_\omega$. We define the space-time weighting operator restricted to $\boldsymbol{\omega}_n$ as

$$\boldsymbol{\mathcal{W}}_{\boldsymbol{\omega}_n}\boldsymbol{u} \doteq (\mathcal{W}_\omega u(t_{n-1}^{K_{n-1}}), \mathcal{W}_\omega u(t_n^1), \ldots, \mathcal{W}_\omega u(t_n^{K_n})), \tag{6.14}$$

and $\boldsymbol{\mathcal{W}} \doteq \Pi_{n=1}^{N}\Pi_{\omega\in\Theta}\boldsymbol{\mathcal{W}}_{\boldsymbol{\omega}_n}\boldsymbol{u}_{\boldsymbol{\omega}_n}$. We can observe that this weighting operator uses the space-only weighting operator in Eq. (6.5), in order to make the functions continuous in space. On the other hand, on the time interfaces T_n between subdomains $\boldsymbol{\omega}_n$ and $\boldsymbol{\omega}_{n+1}$ (for $n = 1, \ldots, N-1$, where functions in $\boldsymbol{V}$ can be discontinuous in time) we take the value at the preceding subdomain, i.e., $\boldsymbol{\omega}_n$. This choice is motivated by the causality of the problem in time.

2. Next, we define the space-time interior correction. In order to do so, we first define the space-time "bubble" space as $\boldsymbol{V}_0$, where its local component at $\boldsymbol{\omega}_n$ is

$$\boldsymbol{v}_{\boldsymbol{\omega}_n} = (0, v_\omega(t_n^1), \ldots, v_\omega(t_n^{K_n})), \qquad v(\cdot) \in V_0.$$

This definition of $\boldsymbol{V}_0$ naturally arises from the definition of the weighting operator Eq. (6.14). The nodes that are enforced to be zero in $\boldsymbol{V}_0$ are the ones that are modified by Eq. (6.14). $\boldsymbol{\mathcal{I}}_0$ is the trivial injection from $\boldsymbol{V}_0$ to $\bar{\boldsymbol{V}}$ and we denote the Galerkin projection of $\boldsymbol{\mathcal{A}}$ as $\boldsymbol{\mathcal{A}}_0$. Its inverse involves local subdomain problems like Eq. (6.8) with zero initial condition and homogeneous boundary conditions on Γ. Finally, we define the space-time "harmonic" extension operator as $\boldsymbol{\mathcal{E}}\boldsymbol{v} \doteq (\boldsymbol{I} - \boldsymbol{\mathcal{I}}_0\boldsymbol{\mathcal{A}}_0^{-1}\boldsymbol{\mathcal{I}}_0^T\bar{\boldsymbol{\mathcal{A}}})\boldsymbol{v}$. Functions $\boldsymbol{v} \in \bar{\boldsymbol{V}}$ such that $\boldsymbol{\mathcal{E}}\boldsymbol{v} = \boldsymbol{v}$ are denoted as "harmonic"

functions.

We finally define the transfer operator $\boldsymbol{\mathcal{Q}} : \boldsymbol{V} \to \bar{\boldsymbol{V}}$ as $\boldsymbol{\mathcal{Q}} \doteq \boldsymbol{\mathcal{E}}\boldsymbol{\mathcal{W}}$.

6.4.4 Space-time-parallel preconditioner

After extending the previous ingredients to space-time, we can now define the STBDDC preconditioner as

$$\boldsymbol{\mathcal{B}} \doteq \boldsymbol{\mathcal{I}}_0 \boldsymbol{\mathcal{A}}_0^{-1} \boldsymbol{\mathcal{I}}_0^T + \boldsymbol{\mathcal{Q}} \widetilde{\boldsymbol{\mathcal{A}}}^{-1} \boldsymbol{\mathcal{Q}}^T. \tag{6.15}$$

In the following section, we will analyse the quality of $\boldsymbol{\mathcal{B}}$ as a preconditioner for $\bar{\boldsymbol{\mathcal{A}}}$. We are particularly interested in the weak scalability properties of the preconditioner. Again, this preconditioner can be cast in the additive Schwarz theory.

6.4.5 Implementation aspects

Let us make some comments about the efficient implementation of the STBDDC preconditioner. We want to solve system Eq. (6.2) (or equivalently Eq. (6.7)) for all time steps in one shot by using a Krylov iterative solver preconditioned with the STBDDC preconditioner Eq. (6.15). As usual in DD preconditioning, it is common to take as initial guess for the Krylov solver the interior correction $\boldsymbol{u}^0 = \boldsymbol{\mathcal{I}}_0 \boldsymbol{\mathcal{A}}_0^{-1} \boldsymbol{\mathcal{I}}_0^T \boldsymbol{f}$. In this case, it can be proved by induction that applying a Krylov method with $\boldsymbol{\mathcal{B}}$ as a preconditioner will give at each iterate $\boldsymbol{V}_0$-orthogonal residuals of the original problem Eq. (6.7) and "harmonic" directions (see, e.g., [96]). Thus, the application of the BDDC preconditioner applied to $\boldsymbol{r} \in \boldsymbol{V}'$ such that $\boldsymbol{r} \perp \boldsymbol{V}_0$ can be simplified as:

$$\boldsymbol{\mathcal{B}}\boldsymbol{r} = \boldsymbol{\mathcal{E}}\boldsymbol{\mathcal{W}} \widetilde{\boldsymbol{\mathcal{A}}}^{-1} \boldsymbol{\mathcal{W}}^T \boldsymbol{r}.$$

It involves the following steps.

1. Compute $\boldsymbol{s} \doteq \boldsymbol{\mathcal{W}}^T \boldsymbol{r}$.

   By the definition in Eq. (6.14), the restriction of $\boldsymbol{s} = \boldsymbol{\mathcal{W}}^T \boldsymbol{r}$ to $\boldsymbol{\omega}_n$ is $\boldsymbol{s}_{\boldsymbol{\omega}_n} = (0, \mathcal{W}_\omega^T r(t_n^1), \ldots, \mathcal{W}_\omega^T r(t_n^{K_n}))$, where $\mathcal{W}_\omega = \mathrm{diag}(1/|\mathrm{neigh}(\xi)|)$. This operation implies nearest neighbour communications only.

2. Compute $\widetilde{\boldsymbol{\mathcal{A}}}^{-1} \boldsymbol{s}$. In order to compute this problem, we first use the following decomposition of $\widetilde{\boldsymbol{V}}$ into the subspaces $\widetilde{\boldsymbol{V}}_F$ and $\widetilde{\boldsymbol{V}}_C$. $\widetilde{\boldsymbol{V}}_F$ is the set of functions that vanish on the coarse DOFs Eq. (6.11)-Eq. (6.13). $\widetilde{\boldsymbol{V}}_C$ is the complement of $\widetilde{\boldsymbol{V}}_F$, which provides the values on the coarse DOFs. We define $\widetilde{\boldsymbol{V}}_C$ as the span of the columns of $\boldsymbol{\Phi}$, where $\boldsymbol{\Phi}$ is the solution of

   $$\begin{bmatrix} \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}_n} & \boldsymbol{\mathcal{C}}_{\boldsymbol{\omega}_n}{}^T \\ \boldsymbol{\mathcal{C}}_{\boldsymbol{\omega}_n} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}_{\boldsymbol{\omega}_n} \\ \boldsymbol{l}_{\boldsymbol{\omega}_n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix}, \tag{6.16}$$

   where we have introduced the notation $\boldsymbol{\mathcal{C}}_{\boldsymbol{\omega}_n}$ for the matrix associated to the coarse DOFs constraints. We can check that (see [42, p. 206] for the symmetric case)

(1) $\widetilde{\boldsymbol{V}}_F \perp_{\boldsymbol{\mathcal{A}}} \widetilde{\boldsymbol{V}}_C$, (2) $\widetilde{\boldsymbol{V}} = \widetilde{\boldsymbol{V}}_F \oplus \widetilde{\boldsymbol{V}}_C$. The local problems in Eq. (6.16) are indefinite (and couple all time steps in one subdomain). In order to be able to use sequential-in-time local solvers and sparse direct methods for positive-definite matrices, we propose the following approach (see [57] for the space-parallel BDDC preconditioner). Using the fact that $\boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}_n}$ is non-singular (see Proposition 6.4.1), we can solve Eq. (6.16) using the Schur complement:

$$-\boldsymbol{\mathcal{C}}_{\boldsymbol{\omega}_n}\boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}_n}{}^{-1}\boldsymbol{\mathcal{C}}_{\boldsymbol{\omega}_n}{}^{T}\boldsymbol{l}_{\boldsymbol{\omega}_n} = \boldsymbol{I}, \qquad \boldsymbol{\Phi}_{\boldsymbol{\omega}_n} = -\boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}_n}{}^{-1}\boldsymbol{\mathcal{C}}_{\boldsymbol{\omega}_n}{}^{T}\boldsymbol{l}_{\boldsymbol{\omega}_n}. \tag{6.17}$$

Further, for non-symmetric problems (as the space-time problem considered herein), we also require to define $\widetilde{\boldsymbol{V}}_C^*$ as the span of the columns of $\boldsymbol{\Psi}$, where $\boldsymbol{\Psi}$ is the solution of

$$\begin{bmatrix} \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}_n}{}^{T} & \boldsymbol{\mathcal{C}}_{\boldsymbol{\omega}_n}{}^{T} \\ \boldsymbol{\mathcal{C}}_{\boldsymbol{\omega}_n} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Psi}_{\boldsymbol{\omega}_n} \\ \boldsymbol{l}_{\boldsymbol{\omega}_n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix}.$$

This problem is similar to Eq. (6.16), but replacing $\boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}_n}$ by $\boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}_n}{}^{T}$ ($\boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}_n}{}^{T}$ is an upper triangular non-singular matrix from Proposition 6.4.1). Thus, we can use the Schur complement approach (like in Eq. (6.17)) to exploit sequentiality (backward in time) for the local problems.

With these spaces, the original problem to be solved, $\widetilde{\boldsymbol{\mathcal{A}}}\boldsymbol{u} = \boldsymbol{s}$, can be written as: find $\boldsymbol{u} = \boldsymbol{u}_F + \boldsymbol{u}_C \in \widetilde{\boldsymbol{V}}$, where $\boldsymbol{u}_F \in \widetilde{\boldsymbol{V}}_F$ and $\boldsymbol{u}_C \in \widetilde{\boldsymbol{V}}_C$ satisfy

$$\boldsymbol{\mathcal{A}}(\boldsymbol{u}_F, \boldsymbol{v}_F) + \boldsymbol{\mathcal{A}}(\boldsymbol{u}_C, \boldsymbol{v}_C^*) = (\boldsymbol{s}, \boldsymbol{v}_F) + (\boldsymbol{s}, \boldsymbol{v}_C^*), \text{ for any } \boldsymbol{v}_F \in \widetilde{\boldsymbol{V}}_F,\ \boldsymbol{v}_C^* \in \widetilde{\boldsymbol{V}}_C^*,$$

where we have used the orthogonality property

$$\boldsymbol{\mathcal{A}}(\boldsymbol{u}_F + \boldsymbol{u}_C, \boldsymbol{v}_F + \boldsymbol{v}_C^*) = \boldsymbol{\mathcal{A}}(\boldsymbol{u}_F, \boldsymbol{v}_F) + \boldsymbol{\mathcal{A}}(\boldsymbol{u}_C, \boldsymbol{v}_C^*).$$

Thus, it involves a fine problem and a coarse problem that are independent. The computation of the fine problem has the same structure as Eq. (6.16) (with a different forcing term), and can be computed using the Schur complement approach in Eq. (6.17). The Petrov-Galerkin type coarse problem couples all subdomains and is a basis for having a weakly scalable preconditioner. Its assembly, factorization, and solution is centralised in one processor or a subset of processors.

Summarising, the STBDDC preconditioner can be implemented in such a way that standard sequential-in-time solvers can still be applied for the local problems. Due to the fact that coarse and fine problems are independent, we can exploit an overlapping implementation, in which computations at fine/coarse levels are performed in parallel. This implementation has been proved to be very effective at extreme scales for space-parallel BDDC solvers in [18, 23, 22]. The implementation of the STBDDC preconditioner used in Sect. 6.5 also exploits this overlapping strategy.

6.5 Numerical experiments

In this section we evaluate the weak scalability for the CDR problem Eq. (6.1) of the proposed STBDDC preconditioner, when combined with the right-preconditioned version of the iterative Krylov-subspace method generalized minimal residual method (GMRES). The STBDDC-GMRES solver is tested with the 2D convection-diffusion-reaction (CDR) partial differential equation (PDE) on regular domains. Domains are discretized with structured Q1 FE meshes and Backward-Euler (BE) time integration is performed with a constant step size $|\delta_k|$. As performance metrics, we focus on the number of STBDDC preconditioned GMRES iterations required for convergence, and the total computation time. This time will include both preconditioner set-up and the preconditioned iterative solution of the linear system in all the experiments reported. The nonlinear case is linearized with a Picard algorithm and a relaxation factor of $\alpha = 0.75$. The stopping criteria for the iterative linear solver is the reduction of the initial residual algebraic L_2-norm by a factor 10^{-6}. The nonlinear Picard algorithm stopping criteria is the reduction of the algebraic L_2-norm of the nonlinear residual below 10^{-3}.

The problem to be solved is the CDR problem Eq. (6.1). We may consider the Poisson problem for $\beta = (0, 0)$ and $\sigma = 0$. Further, we will also tackle the p-Laplacian problem, by taking the nonlinear viscosity $\nu(u) = \nu_0|\nabla u|^p$ (with $p \geq 0$ and $\nu_0 > 0$), $\beta = (0, 0)$ and $\sigma = 0$.

6.5.1 Experimental framework

The novel techniques proposed in this paper for the STBDDC-GMRES solver have been implemented in `FEMPAR` (see Chapter 1 for an introduction). The parallel codes in `FEMPAR` heavily use standard computational kernels provided by (highly-efficient vendor implementations of) the BLAS and LAPACK. Besides, through proper interfaces to several third party libraries, the local constrained Neumann problems and the global coarse-grid problem can be solved via sparse direct solvers. Here, we use the overlapped BDDC implementation proposed in [18], with excellent scalability properties. It is based on the overlapped computation of coarse and fine duties. As long as coarse duties can be fully overlapped with fine duties, perfect weak scalability can be attained. We refer to [18] and [23] for more details. Results reported in this section were obtained on two different distributed-memory platforms: the Gottfried complex of the HLRN-III Cray system, located in Hannover (Germany) and the MareNostrum III, in the Barcelona Supercomputing Centre (BSC). In all cases, we consider a one-to-one mapping among subdomains, cores and MPI tasks, plus one additional core for the coarse problem (see [21, 18] for details).

6.5.2 Weak scalability setting

In computer science parlance weak scalability is related to how the solution time varies with the number of processors for a fixed problem size per processor. When the time

remain asymptotically constant, we say that we have a scalable algorithm. When we consider problems that are obtained after discretization of differential operators, the concept of weak scalability is suitable *as soon as* the relation between the different terms in the (discretization of the) PDE remains constant in the weak scalability analysis. This is the case in most scalability analyses of PDE solvers, which usually deal with steady Poisson or linear elasticity problems. However, the situation becomes more involved as one faces more complicated problems, that combine multiple differential terms of different nature. The simplest example is the CDR equation Eq. (6.1). One can consider a fixed domain Ω and fixed physical properties, and produce a weak scalability analysis by increasing the number of elements (i.e., reducing h) in FEs, and the number of subdomains (i.e., reducing H) in the same proportion. However, as we go to larger scales, the problem to be solved tends to a simple Poisson problem (convective terms are $\mathcal{O}(1/h)$ whereas diffusive terms are $\mathcal{O}(1/h^2)$). The same situation happens for space-only parallelization of transient problems because the CFL changes in the scalability analysis. This situation has already been identified in [65, 54], leading to what the authors call CFL-constant scalability. In these simulations, the CFL is constant, but still, spatial differential terms can change their relative weight in the scalability analysis, e.g., one keeps the convective CFL, i.e., $\text{CFL}_\beta = |\beta|\frac{|\delta|}{h}$, but not the diffusive CFL, i.e., $\text{CFL}_\nu = \nu\frac{|\delta|}{h^2}$ (see [54]).

Weak scalability analysis of PDE solvers should be such that the relative weight of all the discrete differential operators is kept. To do that, we keep fixed the physical problem to be solved (boundary conditions, physical properties, etc), the FE mesh size h, and the subdomain size H, but increase (by scaling) the physical domain $\Omega \to \alpha\Omega$ and subsequently the number of subdomains and FEs. Let us consider that $\Omega = [0,1]^d$, a FE mesh of size $h = (1/n_h)^d$, and a subdomain size $H = (1/n_H)^d$. Now, we consider $\Omega' = \alpha\Omega = [0,\alpha]^d$, $\alpha \in \mathbb{N}^+$. The FE partition now must involve αn_h FE partitions per dimension ($\alpha^d n_h^d$ FEs) and αn_H subdomain partitions per dimension ($\alpha^d n_H^d$ subdomains). It is also possible to apply this approach to unstructured meshes and space-time domains. Weak scalability in the sense proposed herein is not only about the capability to solve larger problems but also more complicated ones. E.g., we keep fixed the local Reynolds or Péclet number or CFLs, but we increase the global Reynolds or Péclet number, facing not only a larger problem but also a harder one, in general. We have used this definition of weak scalability for PDE solvers in the numerical experiments below for time and space-time parallel solvers.

6.5.3 Weak scalability in Time

In this case, the spatial domain is not partitioned and only the time integration is distributed through P_t processors. This fact leads to enforced continuity of mean values of the function on the spatial domain Ω on time interfaces, i.e., constraint Eq. (6.12) with $\omega = \Omega$. In order to maintain a constant CFL number, the original time interval $(0,T]$ is scaled with the number of processors, i.e., $T' = P_t T$. As a result, using P_t processors

we solve a P_t times larger time domain (and time steps). Note that with this approach neither $|\delta_k|$ nor $|\Delta_n|$ are modified through the analysis.

Time-parallel Poisson solver

Consider the transient Poisson equation (Eq. (6.1) with $\beta = (0,0)$ and $\sigma = 0$) with $\nu = 1$ on the unit square spatial domain $\Omega = [0,1]^2$ and $T = 1$. The source term f is chosen such that $u(x,t) = \sin(\pi x)\sin(\pi y)\sin(\pi t)$ is the solution of the problem. Homogeneous Dirichlet boundary conditions and zero initial condition are imposed. We perform the weak scalability analysis of the TBDDC-GMRES solver with element size $h = \frac{1}{30}$ and several values of $K_n = \frac{|\Delta_n|}{|\delta_k|} = \{10, 15, 30, 60\}$.

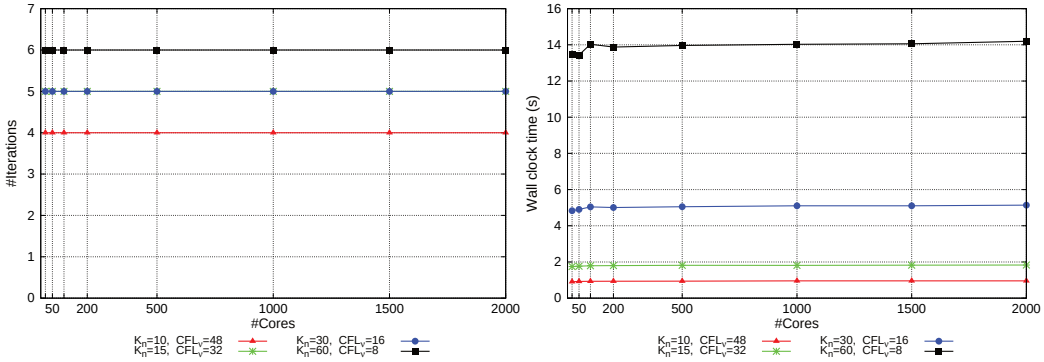
Fig. 6.2 reports the weak scalability analysis of the TBDDC-GMRES solver for this experiment. While h is kept fixed, we evaluate different values of K_n and $|\Delta_n|$, which lead to a wide range of diffusive CFLs, shown in 6.2(a), 6.2(b), and 6.2(c) .

Out of these plots, we can draw some conclusions. First, for a fixed local problem size and physical coefficients, reducing the diffusive CFL by reducing the time step size results in more iterations. Second, and most salient, the algorithm is in fact weakly scalable. In fact, for a fixed local problem size and diffusive CFL, as one increases the number of processors, i.e., computes more time steps, the number of iterations is asymptotically constant. In this range, the overlapped fine/coarse strategy leads to perfect weak scalability for time-parallel solvers too. As a result, this analysis shows the capability of the method to compute X times more time steps with X times more cores for the same total elapsed time, which is the main motivation of time-parallelism.

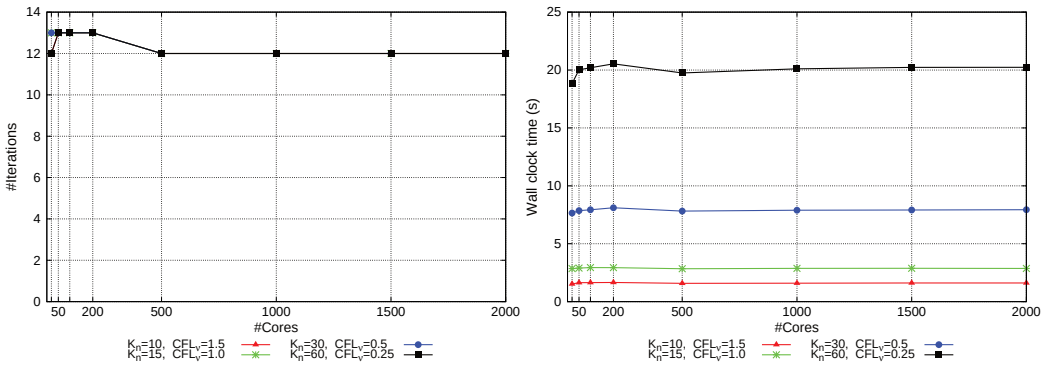
Time-parallel CDR solver

Consider now the transient CDR Eq. (6.1) with $\nu = 10^{-3}$ and $\sigma = 10^{-4}$ on $\Omega = [0,1]^2$ with homogeneous Dirichlet boundary conditions and null initial solution. We take $T = \frac{1}{10}$. In order to show results for different convective CFL ranges, two convection velocity fields are analysed, namely $\beta = (1,0)$ and $\beta = (10,0)$. The CFL values shown are those corresponding to the convective term since it is more restrictive than the diffusive CFL in all the cases being considered. The SUPG stabilization technique is employed (see [45]).

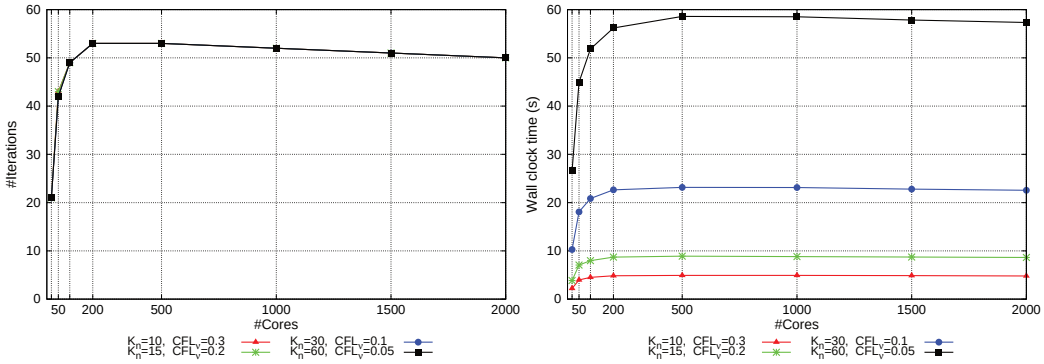
We perform the study with several values of $K_n = \frac{|\Delta_n|}{|\delta_k|} = \{10, 30, 60\}$, which lead to different convective CFLs. For the first case (Fig. 6.3), the source term is chosen such that the function $u(x,t) = \sin(\pi x)\sin(\pi y)\sin(\pi t)$ is the solution of the problem. For the second test (Fig. 6.4), we take $f = 1$, with a boundary layer formation.
Out of these plots we can extract the same conclusions from Figs. 6.3 and 6.4 as for the Poisson problem. The method is weakly scalable in time for transient CDR problems.



(a) Iteration counter and computing time for $|\Delta_n| = \frac{8}{15}$.



(b) Iteration counter and computing time for $|\Delta_n| = \frac{1}{60}$.

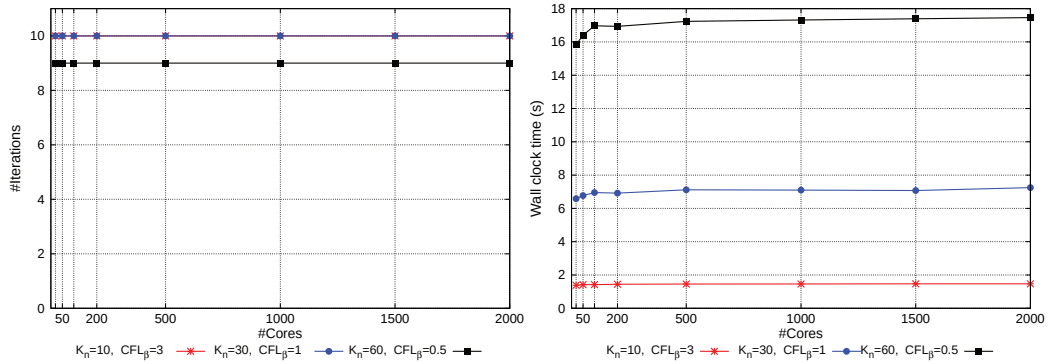


(c) Iteration counter and computing time for $|\Delta_n| = \frac{1}{300}$.

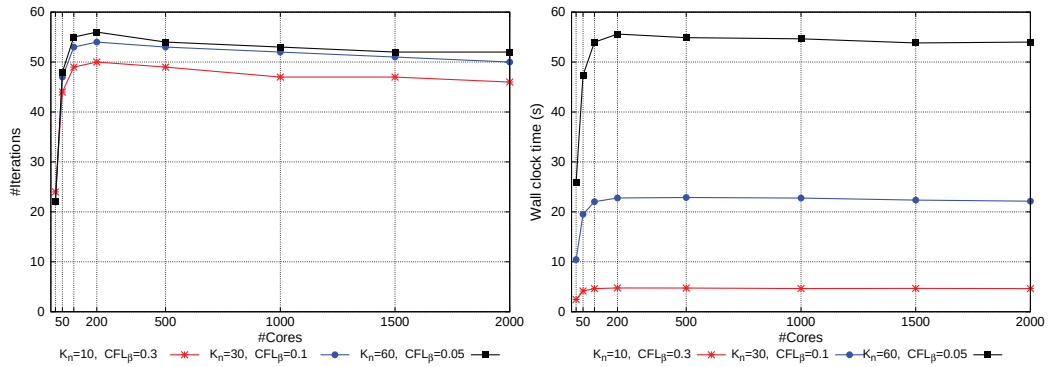
FIGURE 6.2: Weak scalability for the iterations count (left) and total elapsed time (right) of the TBDDC-GMRES solver in the solution of the unsteady 2D Poisson problem on HLRN. Fixed element size $h = 1/30$ while time partition on P_t subdomains.

6.5.4 Weak scalability in space-time

The STBDDC preconditioner is considered here, for the set of constraints Eq. (6.11)-Eq. (6.13). In this case, the spatial domain Ω is scaled by P_x and P_y in the corresponding directions, where $P_x = P_y$ in all cases. On the other hand, the time interval $(0, T]$ is scaled by P_t, leading to a $P = P_x \times P_y \times P_t$ partition of the scaled ST domain $P_x\Omega \times P_t(0, T]$. Therefore, the relative weight of the operators is kept constant through a weak scalability analysis. Local problem loads will be given by $\frac{H}{h}$ in space and $K_n = \frac{|\Delta_n|}{|\delta_k|}$ in time, i.e.,



(a) Iteration counter and computing time with $\beta = (10, 0)$.



(b) Iteration counter and computing time with $\beta = (1, 0)$.

FIGURE 6.3: Weak scalability for the total elapsed time (right) and number of GMRES iterations (left) of the TBDDC solver in the solution of the 2D CDR equation on HLRN (sinusoidal solution). Fixed element size to $h = 1/30$.

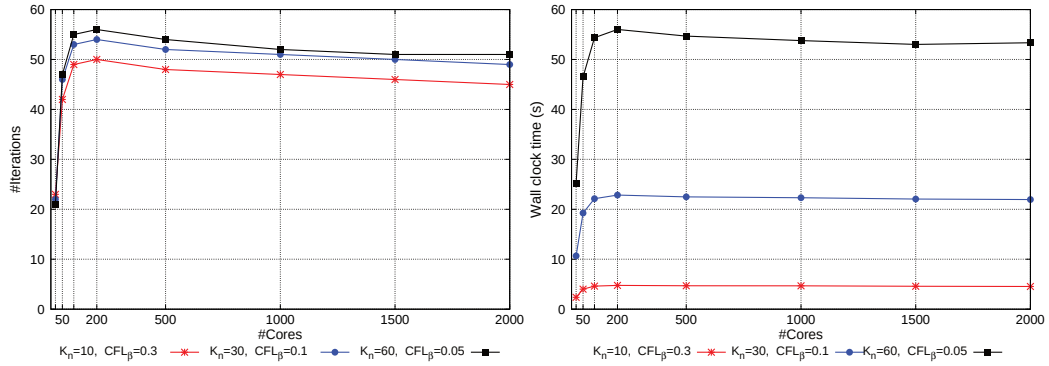
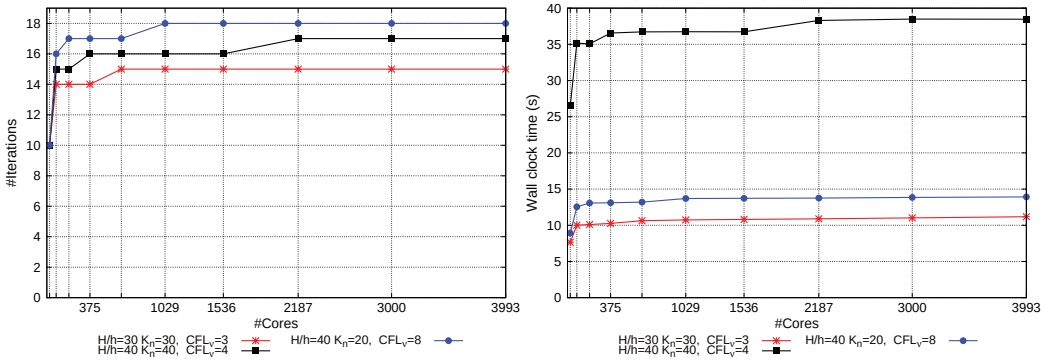


FIGURE 6.4: Weak scalability for the total elapsed time (right) and number of GMRES iterations (left) of the TBDDC solver in the solution of the 2D CDR equation (boundary layer) on HLRN. Fixed element size to $h = 1/30$.

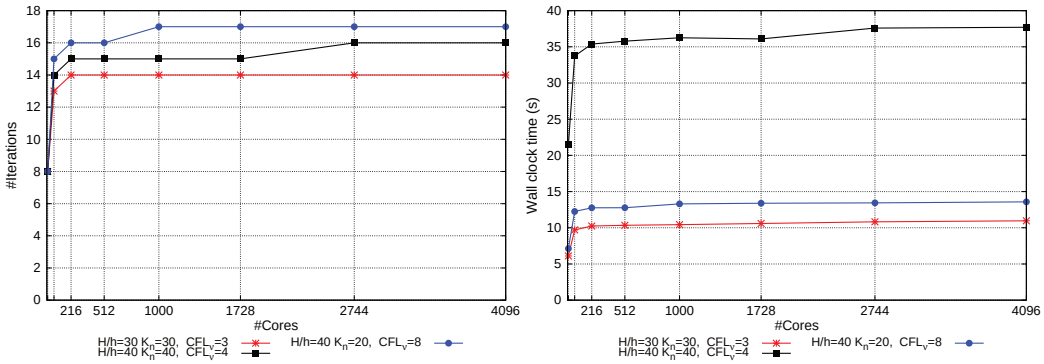
$(\frac{H}{h})^d \times \frac{|\Delta_n|}{|\delta_k|}$.

Space-time Poisson solver

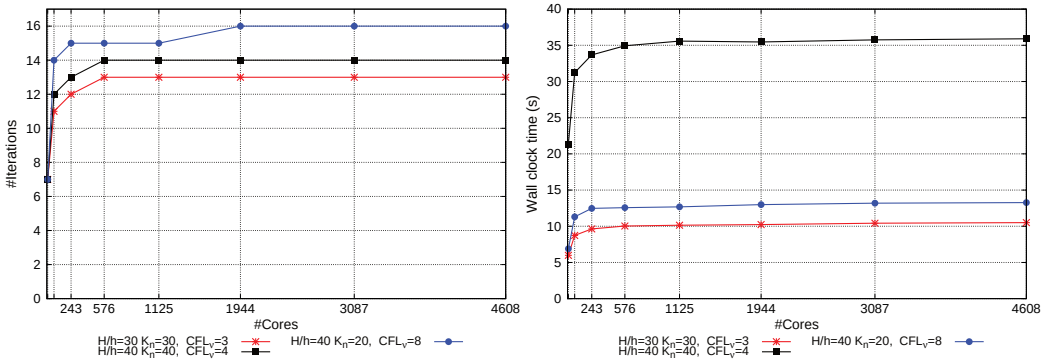
Consider the Poisson problem (Eq. (6.1) with $\beta = (0,0)$ and $\sigma = 0$) with $\nu = 1$, $f = 1$, and homogeneous Dirichlet boundary conditions and zero initial condition. The original spatial domain is $\Omega = [0,1]^2$ while the original time domain is $(0, 0.1]$.



(a) Iteration counter and wall clock times for $P_t = 3P_x$



(b) Iteration counter and wall clock times for $P_t = P_x$



(c) Iteration counter and wall clock times for $P_x = 3P_t$

FIGURE 6.5: Weak scalability for the total elapsed time (right) and number of GMRES iterations (left) of the STBDDC solver in the solution of the 2D Poisson problem on HLRN.

Fig. 6.5 shows weak scalability results for the STBDDC-GMRES solver. Number of iterations (left) and total elapsed times (right) have been reported for three different ratios between spatial and time partitions $\frac{P_x}{P_t}$ in 6.5(a), 6.5(b), and 6.5(c). Also, at every figure, three different local problem sizes, and thus diffusive CFLs, have been considered.

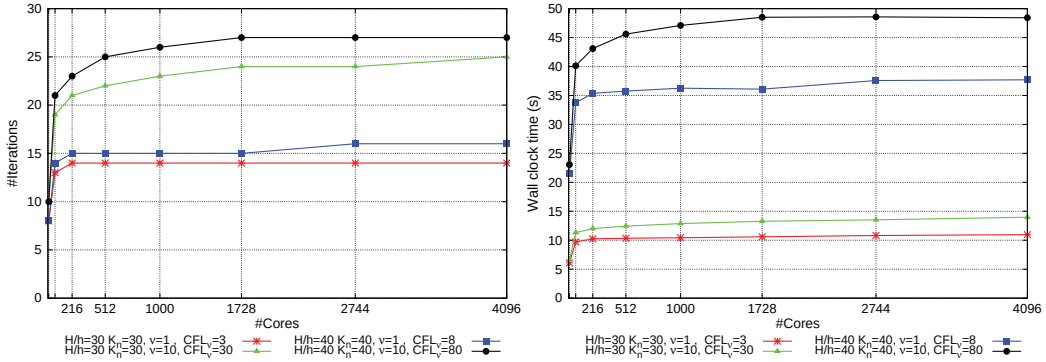


FIGURE 6.6: Weak scalability for the total elapsed time (right) and number of GMRES iterations (left) of the STBDDC solver in the solution of the 2D Poisson problem on HLRN. Partition is done equally in time and space, i.e., $P_x = P_t$.

The most salient information from these figures is the fact that the scheme is also scalable in space-time simulations. Here, one is not only solving a larger problem in time (as above) but also in space. This result is not surprising since the spatial BDDC preconditioner is known to be weakly scalable and the time-parallel version has also been proved to be weakly scalable in Sect. 6.5.3. The influence of $\frac{P_x}{P_t}$ on the number of iterations is very mild; also the effect of the diffusive CFL is mild in this case. The overlapping strategy is fully effective in the range under consideration, because perfect weak scalability can be observed. The effect of the diffusive CFL for a fixed local problem size, obtained by multiplying by 10 the viscosity, is reported in Fig. 6.6. In this case, a larger diffusive CFL leads to more iterations but weak scalability is also achieved.

Next, we want to compare the space-time solver against a sequential-in-time approach. We fix the time step size to $|\delta_k| = 10^{-3}$ and $\frac{H}{h} = 30$. Thus, the time interval is $T = K|\delta_k|$, when considering K time steps. The sequential approach makes use of $P_s = P_x \times P_y = 4^2$ processors (space-parallelism only) to solve recursively the spatial problem for increasing values of K. The space-time approach is using $P_s \times P_t = 4^2 P_t$ processors to solve the same problem, with a local number of time steps $K_n = 10$. The motivation for such analysis is to assess the benefit of time-parallelism in linear problems when spatial parallelism cannot be further exploited efficiently due to very low load per processor.

Figure 6.7 shows the comparison between the sequential approach and the STBDDC preconditioned ST solver up to $K = 1000$ time steps. The theoretical cost of the sequential approach is proportional to K steps times the elapsed time of the A_ω^{-1} local solve (a preconditioned GMRES iteration). On the other hand, the theoretical cost of the space-time solver is proportional to the cost of the $\boldsymbol{A}_{\boldsymbol{\omega}_n}^{-1}$ local space-time solves, and in turn, K_n times spatial A_ω^{-1} local solves (exploiting locally sequentiality in time). The number of local solves is plotted for both sequential and space-time approaches in Fig. 6.7(a). The sequential approach shows a linear growth of the computing time, as expected, since

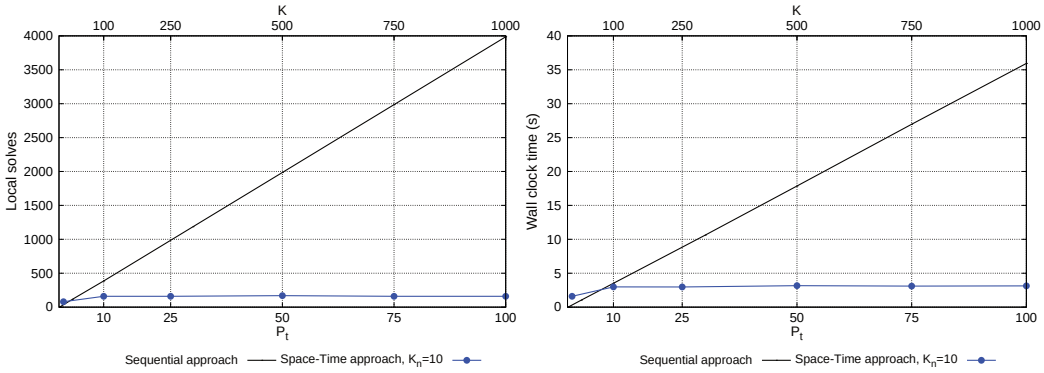


FIGURE 6.7: Comparison between the sequential and space-time solvers for the transient Poisson problem on MareNostrum supercomputer. Spatial partition is fixed to $P_x = P_y = 4$. The space-time approach is using $P = 4^2 P_t$ MPI fine-level tasks.

it is solving K times problems in a sequential fashion. Since the current implementation of space-time preconditioners in `FEMPAR` does not exploit local sequentiality in time, we observe a discrepancy between the intersection of curves in terms of local solves and elapsed time, due to the quadratic complexity of sparse direct methods. In any case, the space-time approach starts to be competitive with less than 10 time partitions. Out of these plots, we are able to reduce the time-to-solution of simulations with the space-time approach by adding more processors to exploit time-parallelism, both for the linear and nonlinear problems considered herein. Besides, the method shows excellent weak scalability properties.

Space-time CDR solver

Consider the CDR equation Eq. (6.1) with $\sigma = 10^{-4}$, $\beta = (1, 0)$, and $f = 1$, on an original domain $\Omega \times (0, T] = [0, 0.3]^2 \times (0, 0.3]$, and scaled through the weak scalability analysis by P_x, P_t respectively. Homogeneous Dirichlet and initial conditions are enforced. Local problem size is fixed with $K_n = 30$ and $\frac{H}{h} = 30$. The ratio between spatial and time partition is $\frac{P_x}{P_t} = 3$. Several diffusion parameters are considered in order to present different scenarios: from a diffusive-dominated case ($\nu = 1.0$) to a convection-dominated one ($\nu = 10^{-6}$). We have the convective CFL equal to 1.0 in all cases. SUPG stabilization is again used.

Table 6.1 presents the iteration count for different diffusion values that lead to different scenarios. In the diffusive-dominated case the STBDDC preconditioned GMRES tends to an asymptotically constant number of iterations, thus independent of the number of subdomains. Moving to the convective case, the number of iterations slightly grows with the decrease of the diffusive CFL number.

	$\nu =$	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-6}
	$\text{CFL}_\nu =$	10^2	10	1	10^{-1}	10^{-2}	10^{-4}
	Péclet=	$5 \cdot 10^{-3}$	$5 \cdot 10^{-2}$	0.5	5	50	$5 \cdot 10^3$
$(P_x \times P_y) \times P_t$	#Sbd						
$(3 \times 3) \times 1$	9	18	11	7	5	5	5
$(6 \times 6) \times 2$	72	28	16	11	11	11	11
$(9 \times 9) \times 3$	243	35	17	11	12	12	13
$(12 \times 12) \times 4$	576	37	17	12	13	14	15
$(15 \times 15) \times 5$	1125	38	17	13	14	15	17
$(18 \times 18) \times 6$	1944	39	17	13	15	15	18
$(21 \times 21) \times 7$	3087	40	17	14	16	16	19
$(24 \times 24) \times 8$	4608	41	18	14	17	17	21
$(27 \times 27) \times 9$	6561	41	18	15	18	18	22
$(30 \times 30) \times 10$	9000	42	18	16	19	19	24

TABLE 6.1: Iteration count for CDR equation. Convective CFL equal to 1.0 in all cases. CFL_ν represents the diffusive CFL.

6.5.5 Nonlinear space-time p-Laplacian solver

In this experiment we compare the sequential-in-time method (solving the spatial problem with a BDDC approach for every time step) against the proposed STBDDC solver. We consider the p-Laplacian problem (with $p = 1$), i.e., Eq. (6.1) with $\nu = |\nabla u|$, $\beta(0, 0)$, and $\sigma = 0$, on $\Omega = [0, 1]^2 \times (0, T]$ with the initial solution $u^0 = x + y$, Dirichlet data $g = x + y$, and the forcing term $f = 1$. We fix the time step size to $|\delta_k| = 10^{-3}$ and $\frac{H}{h} = 30$.

We consider the same setting as the experiment reported in Fig. 6.7 to compare the sequential-in-time and space-time parallel solvers for a nonlinear problem. Thus, we consider an increasing number of time steps, and $P_s = P_x \times P_y = 4^2$ processors for the space-time solver. The sequential solver exploits P_s processors to extract space-parallelism only. Figure 6.8 shows the comparison between the sequential approach up

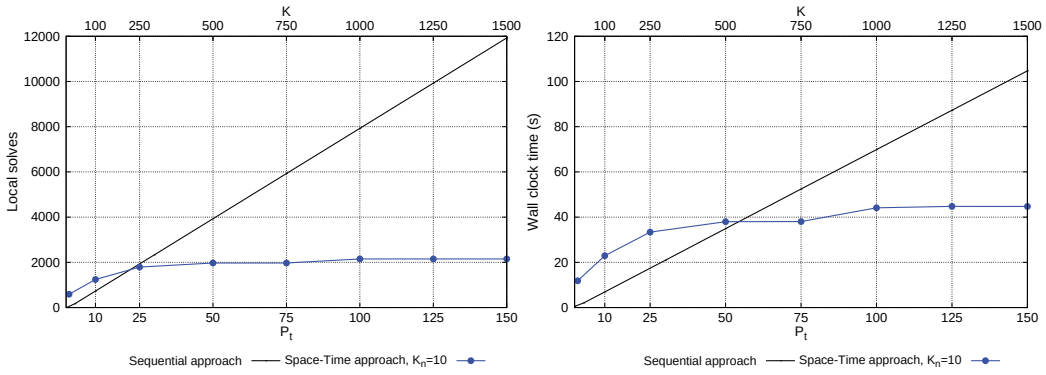


FIGURE 6.8: Comparison between the sequential and space-time solvers for the p-Laplacian transient problem on MareNostrum supercomputer. Spatial partition is fixed to $P_x = P_y = 4$. The space-time approach is using $P = 4^2 P_t$ MPI fine-level tasks.

to $T = 1.5$ and $K = 1500$ time steps. In Fig. 6.8(a) we plot the number of space solves, which is now proportional to the number of accumulated linear iterations through the nonlinear iterations. Remarkable algorithmic scalability is also obtained in the nonlinear setting. Elapsed time plots in 6.8(b) show a similar behaviour in the nonlinear case as in the linear one. The nonlinear case also exhibits excellent weak scalability properties. The nonlinear space-time solver is competitive in terms of number of local problems to be computed at about 20 processors, whereas it requires slightly more than 50 processors in order to be superior in terms of elapsed time. Discrepancies should be substantially reduced exploiting causality for the local problems, as commented above. Out of these plots, we are able to reduce the time-to-solution of simulations with the space-time approach by adding more processors to exploit time-parallelism for the nonlinear problem considered herein. In any case, there is still room for improvement when considering nonlinear problems. In this sense, nonlinear space-time preconditioning [87, 49] and more elaborated linearization strategies have the potential to lead to better performance.

6.6 Conclusions

In this chapter, we have considered a space-time iterative solver based on DD techniques. In particular, we have considered the GMRES iterative solver with space-time preconditioning obtained by extending the BDDC framework to space-time for parabolic problems discretized with FEs. Since the time direction has a very different nature than the spatial one, i.e., it is a transport-type operator, a particular definition of the coarse DOFs and transfer operators is considered, taking into account time causality. Further, perturbation terms must be included to lead to a well-posed system. The exposition has been carried out for a Backward-Euler time integrator, but the extension to θ-methods and Runge-Kutta methods is straightforward. Further, the well-posedness of the proposed space-time preconditioner has been checked.

On the other hand, we have carried out a detailed set of numerical experiments on parallel platforms. Out of these results, the proposed methodology is observed to exhibit excellent scalability properties. The methods are weakly scalable in time, i.e., increasing X times the number of MPI tasks one can solve X times more time steps, in approximately the same amount of time, which is a key property to reduce time-to-solution in transient simulations with heavy time stepping. We have also shown weak scalability in space-time, where one is not only facing larger problems in time but also in space. Further, we have applied the STBDDC preconditioner to nonlinear problems, by considering a linearization of the full space-time system, and applying the proposed space-time solver at every nonlinear iteration. The use of the space-time solvers proposed becomes faster than a sequential-in-time approach for a modest number of processors.

Future work will include the development of nonlinear space-time BDDC preconditioners, extending the concept of nonlinear preconditioning (see, e.g., the recent work in [87] for nonlinear finite element tearing and interconnect (FETI) preconditioners in

space) to space-time. As it has already been observed in space [49], the use of nonlinear preconditioning should make space-time preconditioning more effective for nonlinear problems. Further extensions of this work will involve the extension to multilevel space-time algorithms (to keep perfect weak scalability at larger scales), and their application to solid mechanics and incompressible fluid dynamics simulations.

Chapter 7

Conclusions and future work

7.1 Conclusions

The development of balancing domain decomposition by constraints (BDDC) methods for the linear systems arising from the finite element (FE) discretization of complex electromagnetic and transient problems has been explored in this dissertation. Due to the structure of the thesis, each chapter contains its own detailed conclusions restricted to the scope of the same, therefore the author aims to provide here a more general overview. In order to present a summary of the major contributions of the present thesis, the complete list of established objectives in Chapter 1 is recovered.

- **Arbitrary order edge FEs.** In Chapter 2, the edge FE of arbitrary order is theoretically defined both for tetrahedra and hexahedra. The implementation is based on the definition of heterogeneous polynomial pre-bases that span local FE spaces, which combined with a change of basis provide the bases of shape functions for arbitrary order edge FEs. The author believes that the comprehensive description of the nontrivial implementation issues behind the edge FE may be of high value for other researchers and developers aiming to work in similar FE frameworks. The most salient conclusion out of Chapter 2 is the agreement between theoretical and experimental convergence rates in the solution of manufactured solutions, hence the availability of the implementation in `FEMPAR` to solve problems posed in curl-conforming spaces, discretized with tetrahedral/hexahedral edge FEs of arbitrary order.

- **h-adaptive edge FE framework.** An original implementation strategy of $p-$ and h-adaptive edge (Nédélec) FEs on hierarchically refined octree-based non-conforming meshes is carefully presented in Chapter 2. The implementation, which is grounded on the current interface of `FEMPAR` with the `p4est` library, is satisfactorily tested against analytical solutions. The method, which is implemented and tested for arbitrary order hexahedral FE in both 2D and 3D problems, allows to reduce significantly the number of unknowns for a goal precision.

- **Highly scalable BDDC solvers for problems in H(curl).** One of the major topics of the present thesis is the development and implementation of highly

scalable solvers for problems in curl-conforming spaces. Firstly, the problem is faced with existing approaches for arbitrary order discretizations of homogeneous problems. Indeed, in Chapter 3 an implementation of the BDDC solver is described, putting emphasis on delicate implementation details. Then, the implementation of the BDDC preconditioned solver is shown to have excellent weak scalability properties in terms of both iterations and computing times for core counts above 16K when applied to homogeneous problems discretized with hexahedral FE of order up to 4.

- **Robust BDDC solvers for heterogeneous problems in $H(\mathbf{curl})$.** In Chapter 3 we define the physics-based (PB)-BDDC preconditioner for problems in curl-conforming spaces. It leverages the implementation of the standard BDDC preconditioner for problems posed in curl-conforming spaces, which includes a crucial change of basis, and the PB partition definition proposed in [16]. The PB-BDDC solver is also shown to have excellent weak scalability properties in terms of both iterations and computing times for core counts above 16K for multi-material problems discretized using hexahedral FE of order up to 4. On the other hand, the proposed preconditioner is shown to be robust with regard to coefficient jumps for arbitrary domain shapes and tetrahedral/hexahedral edge FEs. Chapter 3 also includes satisfactory weak scalability results in the application of the PBBDDC preconditioner to heterogeneous problems with prescribed heterogeneous coefficients and the High Temperature Superconductors (HTS) problem.

- **Build a complete FE parallel framework for the simulation of realistic 3D HTS problems.** In Chapter 4 a fully-parallel FE framework suitable for the solution of (nonlinear) 3D problems modelling the electromagnetic behaviour of HTS is presented. The H-formulation, which is widely used in the HTS modelling community, is selected as a demonstrator of the potential of the scheme. Basic building blocks of the parallel scheme leverage the presented developments in Chapters 2 and 3: We successfully combine a mesh generation with an advanced adaptive mesh refinement (AMR) technique, which allows to efficiently reduce the size of the global problem whilst providing an accurate solution on the superconducting device, with robust PB-BDDC solvers for arbitrary order edge FE discretizations of the problem. Besides, a tailored nonlinear parallel solver based on Newton-Raphson (NR) methods is used. Two outstanding results arise in the numerical experiments section Sect. 4.5. First, the implementation is validated against experimental data for HTS tapes, which shows the availability of the numerical tool to model the phenomena. Secondly, the reduction of the time-to-solution of a selected 3D benchmark by a factor of 50 by using our computational tool in High Performance Computing (HPC) platforms with respect to serial (i.e., not parallel) computations. All the developments are available in the open source

software project `FEMPAR`, which can become a powerful tool for the HTS modelling community.

- **Design parallel-in-time nonlinear ordinary differential equation (ODE) solvers.** In Chapter 5 different direct solvers have been proposed for both linear and nonlinear ODEs. The methods allowed to identify key ideas for the latter combination with space-parallel scalable BDDC implementations to design space-time preconditioners. All the presented implementations showed excellent weak scalability results. In short, increasing X times the number of MPI tasks (e.g. time subintervals in a one-to-one mapping between tasks and subintervals) one can solve X times more time steps in approximately the same amount of time.

- **Develop a time-extension of the BDDC preconditioner.** In Chapter 6 a highly scalable space-time (ST) preconditioner based on BDDC methods is provided. In particular, and due to the *causality* of the time direction, a particular definition for the continuity across interfaces and averaging operators is defined. The implementation in `FEMPAR` does not exploit the classical sequentiality of the discretizations (at least at the global level) and exhibits excellent weak scalability properties with regard to the time direction, i.e., for a fixed problem in space, increasing X times the number of MPI tasks one can solve X times more steps. Further, the methods also exhibit outstanding weak scalability properties in space-time, i.e., one can solve not only larger problems in time but also in space.

7.2 Open lines of research

The present thesis addressed the development of highly scalable BDDC methods in a broad sense. Research in any field never truly ends, it is just delimited by a period of time. In this sense, some ideas that arise as a natural continuation path of the present work are still to be explored. In the other hand, other tangent ideas that emerged during the development of the present thesis are also shortly described.

- **Full *hp*-adaptive code.** In Chapter 2, an implementation of *h*-adaptive methods and arbitrary order edge FEs has been presented and tested. In the simulations, one of the two parameters (i.e., the mesh or the elements order) is fixed. The presented *h*-adaptive methods are based on the construction of a restriction operator to enforce the conformity of the FE. Its construction in order to enforce continuity between adjacent, conforming FEs of different order (see, e.g., [55]) would lead to the case where also the element order may vary among elements, i.e., *p*-adaptivity, but it is not presented in this thesis.

- **Multilevel BDDC methods for problems in $H(\mathbf{curl})$.** The coarse problem is the bottleneck of all BDDC methods where the original domain is partitioned into a large number of subdomains. Besides, the PB approach has been shown

to be robust, but the price to pay is a larger coarse problem size. FEMPAR provides a multilevel implementation of the BDDC preconditioner [23] that can be straightforwardly defined for a PB approach. However, as pointed out in [149], the situation is more involved for problems in $H(\text{curl})$ since the Nédélec definition of degrees of freedom (DOFs) does not lead to weakly scalable BDDC algorithms. Instead, for the multilevel case one must employ an analogous change of variables (see Chapter 3) for the Nédélec-like DOFs in coarser levels to obtain multilevel scalable algorithms.

- **Explore BDDC methods on non-conforming meshes.** In Chapter 4 a fully-parallel FE framework for the solution of HTS problems is built. In the solver part, the physics-based BDDC (PB-BDDC) solver is employed. However, the non-conformity of the mesh is not exploited at the solver level, which contains an implementation of the most conservative coarse space proposed in [137], called Alg. C. There is room for improvement in this direction, since significantly smaller coarse problems could be obtained with tailored definitions of the non-conforming geometrical entities that conform the coarse triangulation.

- **Time integration with alternative methods.** In the current work numerical results are shown for Backward-Euler (BE) time integrator, perhaps one of the most simple (and useful) schemes for the time discretization of a transient partial differential equation (PDE). Although exposed, no numerical results are provided for other time integrators such as Runge-Kutta (RK) or Backward differentiation formula (BDF), which would allow to increase the order of the approximation in time.

- **Methods with space-time FEs.** In space-time formulations, the usual approach is to first discretize the spatial problem with a FE method, leading to a problem discretized in space but not yet in time. Secondly, by means of time integrators one is able to express the approximation of the transient solution. The idea of space-time FE discretizations is to break with this two-step process and integrate the problem in one shot with space-time FEs. In short, $D+1$ space-time FEs can be composed as a tensor product of a regular FE in the spatial (2D or 3D) direction and a time FE in the (1D) time direction. Note that the definition allows for *heterogeneous* FEs, where the user can select different sort of FE and/or approximation orders for both space and time dimensions.

- **Implementation of a transient 3D time-parallel scheme.** Transient PDEs pose the problem in a space of $D+1$ dimensions, where $D = \{2, 3\}$. That implies a 4D space in the case of transient 3D simulations. Although theoretically defined in Chapter 6, no numerical results are shown for such simulations, restricting our implementation to 2D plus time dimensional problems.

- **Nonlinear (Space-)Time BDDC methods.** In Chapter 4 the BDDC preconditioner is applied to the HTS problem, which is governed by a nonlinear PDE. In this case, the preconditioner is applied for the solution of linearized problems, which are obtained by means of a NR method over the global problem. Therefore, the preconditioner is not aware of the nonlinear nature of the problem. On the other hand, the concept of nonlinear preconditioning is presented in this thesis in Chapter 5 for ODEs, but not extended to PDEs. The combination of the introduced ideas with BDDC preconditioners would lead to nonlinear preconditioning techniques, see, e.g., [87].

- **Space-Time BDDC preconditioners for the HTS problem.** On one hand, robust and scalable parallel BDDC solvers are presented for a wide range of heterogeneous problems in Chapters 3 and 4, including the HTS problem. On the other hand, STBDDC methods are defined, implemented and satisfactorily tested in Chapter 6 for elliptic problems discretized with standard Lagrangian FEs. Therefore, in the present thesis all the basic algorithms are set towards the complete STBDDC approach for curl-conforming problems, but an implementation in `FEMPAR` is still to be explored.

Bibliography

[1] Intel MKL PARDISO - Parallel Direct Sparse Solver Interface. `https://software.intel.com/en-us/articles/intel-mkl-pardiso`.

[2] MFEM – a free, lightweight, scalable C++ library for finite element methods. `http://mfem.org/`.

[3] Netgen/NGSolve. `http://ngsolve.org/`.

[4] Marenostrum IV website. `https://www.bsc.es/marenostrum/marenostrum`, 2018.

[5] R. Agelek, M. Anderson, W. Bangerth, and W. Barth. On orienting edges of unstructured two- and three-dimensional meshes. *ACM Transactions on Mathematical Software*, 2017. To appear.

[6] M. D. Ainslie, D. Hu, V. M. R. Zermeno, and F. Grilli. Numerical simulation of the performance of high-temperature superconducting coils. *J. Supercond. Novel Magn.*, 30(7):1987–1992, 2017.

[7] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. The FEniCS Project Version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[8] A. Alonso and A. Valli. An optimal domain decomposition preconditioner for low-frequency time-harmonic maxwell equations. *Math. Comput.*, 68(226):607–631, 1999.

[9] N. Amemiya, K. Miyamoto, N. Banno, and O. Tsukamoto. Numerical analysis of AC losses in high T_c superconductors based on E-J characteristics represented with n-value. *IEEE Trans. Appl. Supercond.*, 7(2):2110–2113, 1997.

[10] N. Amemiya, S. Sato, and T. Ito. Magnetic flux penetration into twisted multifilamentary coated superconductors subjected to AC transverse magnetic fields. *J. Appl. Phys.*, 100(12):3907, 2006.

[11] A. Amor-Martin, L. E. Garcia-Castillo, and D. D. Garcia-Doñoro. Second-order nédélec curl-conforming prismatic element for computational electromagnetics. *IEEE Transactions on Antennas and Propagation*, 64(10):4384–4395, Oct 2016.

[12] I. Anjam and J. Valdman. Fast MATLAB assembly of FEM matrices in 2D and 3D: Edge elements. *Applied Mathematics and Computation*, 267:252 – 263, 2015.

[13] D. Arnold, D. Boffi, and R. Falk. Approximation by quadrilateral finite elements. *Mathematics of computation*, 71(239):909–922, 2002.

[14] S. Badia and R. Codina. A Nodal-based Finite Element Approximation of the Maxwell Problem Suitable for Singular Solutions. *SIAM Journal on Numerical Analysis*, 50(2):398–417, 2012.

[15] S. Badia, A. F. Martín, E. Neiva, and F. Verdugo. A generic finite element framework on parallel tree-based adaptive meshes. *in preparation*, 2018.

[16] S. Badia, A. F. Martín, and H. Nguyen. Physics-based balancing domain decomposition by constraints for multi-material problems. *Journal of Scientific Computing, in press*, 2017.

[17] S. Badia, A. F. Martín, and M. Olm. Scalable solvers for complex electromagnetic problems. *in preparation*, 2018.

[18] S. Badia, A. F. Martín, and J. Principe. A highly scalable parallel implementation of balancing domain decomposition by constraints. *SIAM Journal on Scientific Computing*, 36(2):C190–C218, Jan. 2014.

[19] S. Badia, A. F. Martín, and J. Principe. `FEMPAR`: An object-oriented parallel finite element framework. *Arch. Comput. Methods Eng.*, 25(2):195–271, 2018.

[20] S. Badia, A. F. Martín, and J. Principe. `FEMPAR` Web page. http://www.fempar.org, 2018.

[21] 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.

[22] S. Badia, A. F. Martín, and J. Principe. On the scalability of inexact balancing domain decomposition by constraints with overlapped coarse/fine corrections. *Parallel Computing*, 50:1–24, 2015.

[23] S. Badia, A. F. Martín, and J. Principe. Multilevel Balancing Domain Decomposition at Extreme Scales. *SIAM Journal on Scientific Computing*, pages C22–C52, 2016.

[24] S. Badia and M. Olm. Space-time balancing domain decomposition. *SIAM Journal on Scientific Computing*, 39(2):C194–C213, 2017.

[25] S. Badia and M. Olm. Nonlinear parallel-in-time schur complement solvers for ordinary differential equations. *Journal of Computational and Applied Mathematics*, 344:794 – 806, 2018.

[26] S. Badia, F. Verdugo, and A. F. Martín. The aggregated unfitted finite element method for elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 336:533 – 553, 2018.

[27] A. Badía-Majós and C. López. Electromagnetics close beyond the critical state: thermodynamic prospect. *Supercond. Sci. Technol.*, 25(10):104004, 2012.

[28] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Web page. `http://www.mcs.anl.gov/petsc`, 2018.

[29] W. Bangerth, D. Davydov, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and D. Wells. The deal.II Library, Version 8.4. *Journal of Numerical Mathematics*, 24, 2016.

[30] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II-A general-purpose object-oriented finite element library. *ACM Trans. Math. Softw.*, 33(4), 2007.

[31] W. Bangerth and O. Kayser-Herold. Data structures and requirements for hp finite element software. *ACM Trans. Math. Softw.*, 36(1):4:1–4:31, 2009.

[32] C. Bean. Magnetization of High-Field superconductors. *Rev. Mod. Phys.*, 36(1):31–39, 1964.

[33] A. Bellen and M. Zennaro. Parallel algorithms for initial-value problems for difference and differential equations. *Journal of Computational and Applied Mathematics*, 25(3):341–350, May 1989.

[34] M. Bergot and M. Duruflé. High-order optimal edge elements for pyramids, prisms and hexahedra. *Journal of Computational Physics*, 232(1):189–213, 2013.

[35] M. Bergot and P. Lacoste. Generation of higher-order polynomial basis of Nédélec H(curl) finite elements for Maxwell's equations. *Journal of Computational and Applied Mathematics*, 234(6):1937–1944, 2010.

[36] M. Bonazzoli, V. Dolean, F. Hecht, and F. Rapetti. An example of explicit implementation strategy and preconditioning for the high order edge finite elements applied to the time-harmonic Maxwell's equations. *Computers & Mathematics with Applications*, 75(5):1498 – 1514, 2018.

[37] M. Bonazzoli and F. Rapetti. High-order finite elements in numerical electromagnetism: degrees of freedom and generators in duality. *Numerical Algorithms*, 74(1):111–136, 2017.

[38] A. Bossavit. Numerical modelling of superconductors in three dimensions: a model and a finite element method. *IEEE Trans. Magn.*, 30(5):3363–3366, 1994.

[39] M. C. Bouzo, F. Grilli, and Y. Yang. Modelling of coupling between superconductors of finite length using an integral formulation. *Supercond. Sci. Technol.*, 17(10):1103, 2004.

[40] E. H. Brandt. Square and rectangular thin superconductors in a transverse magnetic field. *Phys. Rev. Lett.*, 74:3025–3028, 1995.

[41] E. H. Brandt. Superconductors of finite thickness in a perpendicular magnetic field: Strips and slabs. *Phys. Rev. B*, 54:4246–4264, 1996.

[42] S. C. Brenner and R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer, 3rd edition edition, 2010.

[43] S. C. Brenner and L.-Y. Sung. BDDC and FETI-DP without matrices or vectors. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1429–1435, Jan. 2007.

[44] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer-Verlag, 1991.

[45] A. N. Brooks and T. J. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1–3):199–259, Sept. 1982.

[46] P. Brune, M. Knepley, B. Smith, and X. Tu. Composing scalable nonlinear algebraic solvers. *Argonne National Laboratory, Preprint ANL/MCS-P2010-0112*, 2013.

[47] C. Burstedde, L. C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM J. Sci. Comput.*, 33(3):1103–1133, 2011.

[48] O. Bíró. Edge element formulations of eddy current problems. *Comput. Methods Appl. Mech. Eng.*, 169(3):391–405, 1999.

[49] X.-C. Cai and D. E. Keyes. Nonlinearly preconditioned inexact Newton algorithms. *SIAM Journal on Scientific Computing*, 24(1):183–200, 2002.

[50] J. Calvo and O. Widlund. An adaptive choice of primal constraints for bddc domain decomposition algorithms. *Electronic Transactions on Numerical Analysis*, 45:524–544, 2016.

[51] A. Campbell. A direct method for obtaining the critical state in two and three dimensions. *Supercond. Sci. Technol.*, 22(3):034005, 2009.

[52] A. J. Christlieb, C. B. Macdonald, and B. W. Ong. Parallel high-order integrators. *SIAM J. Sci. Comput.*, 32(2):818–835, 2010.

[53] G. Cohen and S. Pernet. *Definition of Different Types of Finite Elements*, pages 39–93. Springer Netherlands, Dordrecht, 2017.

[54] 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.

[55] L. Demkowicz. *Computing with hp-adaptive finite elements: Volume I: One and Two dimensional elliptic and Maxwell problems.* Chapman & Hall/CRC Press, 2006.

[56] L. Demkowicz, J. Kurtz, D. Pardo, M. Paszenski, W. Rachowicz, and A. Zdunek. *Computing with hp-adaptive finite elements: Volume II: Three Dimensional Elliptic and Maxwell Problems with applications.* Chapman & Hall/CRC Press, 2007.

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

[58] C. R. Dohrmann and O. B. Widlund. Some recent tools and a bddc algorithm for 3d problems in h(curl). In R. Bank, M. Holst, O. Widlund, and J. Xu, editors, *Domain Decomposition Methods in Science and Engineering XX*, pages 15–25, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[59] C. R. Dohrmann and O. B. Widlund. A BDDC algorithm with deluxe scaling for three-dimensional H(curl) problems. *Communications on Pure and Applied Mathematics*, 69(4):745–770, 2016.

[60] M. Dryja, J. Galvis, and M. Sarkis. BDDC methods for discontinuous Galerkin discretization of elliptic problems. *Journal of Complexity*, 23(4–6):715–739, 2007.

[61] B. Dutoit, J. Duron, S. Stavrev, and F. Grilli. Dynamic field mapping for obtaining the current distribution in high-temperature superconducting tapes. *IEEE Trans. Appl. Supercond.*, 15(2):3644–3647, 2005.

[62] C. M. Elliott and Y. Kashima. A finite-element analysis of critical-state models for type-II superconductivity in 3D. *IMA J. Numer. Anal.*, 27(2):293–331, 2007.

[63] M. Emmett and M. J. Minion. Toward an efficient parallel in time method for partial differential equations. *Comm. App. Math. and Comp. Sci.*, 7(1):105–132, 2012.

[64] G. Escamez, F. Sirois, V. Lahtinen, A. Stenvall, A. Badel, P. Tixador, B. Ramdane, G. Meunier, R. Perrin-Bit, and C. É. Bruzek. 3-D Numerical modeling of AC losses in multifilamentary MgB2 wires. *IEEE Trans. Appl. Supercond.*, 26(3):1–7, 2016.

[65] R. Falgout, S. Friedhoff, T. Kolev, S. MacLachlan, and J. Schroder. Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635–C661, 2014.

[66] R. S. Falk, P. Gatto, and P. Monk. Hexahedral H(div) and H(curl) finite elements. *ESAIM: M2AN*, 45(1):115–143, 2011.

[67] C. Farhat, M. Lesoinne, and K. Pierson. A scalable dual-primal domain decomposition method. *Numerical Linear Algebra with Applications*, 7(7-8):687–714.

[68] S. Farinon, G. Iannone, P. Fabbricatore, and U. Gambardella. 2D and 3D numerical modeling of experimental magnetization cycles in disks and spheres. *Supercond. Sci. Technol.*, 27(10):104005, 2014.

[69] F. Fuentes, B. Keith, L. Demkowicz, and S. Nagaraj. Orientation embedded high order shape functions for the exact sequence elements of all shapes. *Computers & Mathematics with Applications*, 70(4):353 – 458, 2015.

[70] S. Fukui, H. Tonsho, H. Nakayama, M. Yamaguchi, S. Torii, K. Ueda, and T. Takao. Numerical evaluation of measured ac loss in hts tape in ac magnetic field carrying ac transport current. *Physica C*, 392-396(Part 1):224 – 228, 2003. Proceedings of the 15th International Symposium on Superconductivity (ISS 2002): Advances in Superconductivity XV. Part I.

[71] M. J. Gander. 50 years of time parallel time integration. In *Multiple shooting and time domain decomposition*. Springer, 2015.

[72] M. J. Gander and S. Güttel. ParaExp: A parallel integrator for linear initial–value problems. *SIAM J. Sci. Comput.*, 35(2):C123–C142, 2013.

[73] M. J. Gander, R.-J. Li, and Y.-L. Jiang. Parareal Schwarz waveform relaxation methods. In *Domain Decomposition Methods in Science and Engineering XX*, volume Part II of *Lecture Notes in Computational Science and Engineering*, pages 451–458. Springer Berlin Heidelberg, 2013.

[74] M. J. Gander and M. Neumüller. Analysis of a New Space-Time Parallel Multigrid Algorithm for Parabolic Problems. *arXiv:1411.0519 [math]*, 2014. arXiv: 1411.0519.

[75] L. E. Garcia-Castillo, A. J. Ruiz-Genoves, I. Gomez-Revuelto, M. Salazar-Palma, and T. K. Sarkar. Third-order nedelec curl-conforming finite element. *IEEE Transactions on Magnetics*, 38(5):2370–2372, Sep. 2002.

[76] L. E. García Castillo and M. Salazar Palma. Second order Nédélec tetrahedral element for computational electromagnetics. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 13(3):261–287.

[77] J. Gopalakrishnan, L. García-Castillo, and L. Demkowicz. Nédélec spaces in affine coordinates. *Computers & Mathematics with Applications*, 49(7):1285–1294, 2005.

[78] X. Granados, G. Gonçalves Sotelo, M. Carrera, and J. Lopez-Lopez. H-Formulation FEM Modeling of the current distribution in 2G HTS tapes and its experimental validation using Hall Probe mapping. *IEEE Trans. Appl. Supercond.*, 26(8):1–10, 2016.

[79] F. Grilli. *Numerical modelling of high temperature superconducting tapes and cables.* PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, 2004.

[80] F. Grilli, R. Brambilla, F. Sirois, and S. Memiaghe. Development of a three-dimensional finite-element model for high-temperature superconductors based on the H-formulation. *Cryogenics*, 53:142–147, 2013.

[81] F. Grilli, E. Pardo, A. Stenvall, D. N. Nguyen, W. Yuana, and F. Gömöry. Computation of losses in HTS under the action of varying magnetic fields and currents. *IEEE Trans. Appl. Supercond.*, 24(1):78–110, 2014.

[82] F. Grilli, S. Stavrev, Y. L. Floch, M. Costa-Bouzo, E. Vinot, I. Klutsch, G. Meunier, P. Tixador, and B. Dutoit. Finite-element method modeling of superconductors: from 2-D to 3-D. *IEEE Trans. Appl. Supercond.*, 15(1):17–25, 2005.

[83] F. Hecht. New development in FreeFem++. *Journal of Numerical Mathematics*, 20(3-4):251–265, 2012.

[84] Z. Hong, A. M. Campbell, and T. Coombs. Numerical solution of critical state in superconductivity by finite element software. *Supercond. Sci. Technol.*, 19(12):1246, 2006.

[85] M. Kapolka, V. M. R. Zermeno, S. Zou, A. Morandi, P. L. Ribani, E. Pardo, and F. Grilli. Three-dimensional modeling of the magnetization of superconducting rectangular-based bulks and tape stacks. *IEEE Trans. Appl. Supercond.*, 28(4):1–6, 2018.

[86] Y. Kim, C. Hempstead, and A. Strnad. Magnetization and critical supercurrents. *Phys. Rev. Lett.*, 129:528–535, 1963.

[87] A. Klawonn, M. Lanser, and O. Rheinbach. Nonlinear FETI-DP and BDDC Methods. *SIAM Journal on Scientific Computing*, 36(2):A737–A765, Jan. 2014.

[88] A. Klawonn, P. Radtke, and O. Rheinbach. Adaptive coarse spaces for bddc with a transformation of basis. In T. Dickopf, M. J. Gander, L. Halpern, R. Krause, and L. F. Pavarino, editors, *Domain Decomposition Methods in Science and Engineering XXII*, pages 301–309, Cham, 2016. Springer International Publishing.

[89] A. Klawonn, O. Widlund, and M. Dryja. Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients. *SIAM Journal on Numerical Analysis*, 40(1):159–179, 2002.

[90] V. Lahtinen, A. Stenvall, F. Sirois, and M. Pellikka. A finite element simulation tool for predicting hysteresis losses in superconductors using an H-oriented formulation with cohomology basis functions. *J. Supercond. Novel Magn.*, 28(8):2345–2354, 2015.

[91] J. Li and O. B. Widlund. FETI-DP, BDDC, and block Cholesky methods. *International Journal for Numerical Methods in Engineering*, 66(2):250–271, 2006.

[92] Y. L. Li, S. Sun, Q. I. Dai, and W. C. Chew. Vectorial Solution to Double Curl Equation With Generalized Coulomb Gauge for Magnetostatic Problems. *IEEE Transactions on Magnetics*, 51(8):1–6, 2015.

[93] J. Lions, Y. Maday, and A. Turinici. A parareal in time discretization of PDEs. *Acad. Sci. Paris*, 332(Seria I):661–668, 2001.

[94] G. P. Lousberg, M. Ausloos, C. Geuzaine, P. Dular, P. Vanderbemden, and B. Vanderheyden. Numerical simulation of the magnetization of high-temperature superconductors: a 3D finite element method using a single time-step iteration. *Supercond. Sci. Technol.*, 22(5):055005, 2009.

[95] J. Mandel. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9(3):233–241, 1993.

[96] J. Mandel and C. R. Dohrmann. Convergence of a balancing domain decomposition by constraints and energy minimization. *Numerical Linear Algebra with Applications*, 10(7):639–659, 2003.

[97] J. Mandel, C. R. Dohrmann, and R. Tezaur. An algebraic theory for primal and dual substructuring methods by constraints. *Applied Numerical Mathematics*, 54(2):167–193, July 2005.

[98] J. Mandel, B. Sousedík, and C. Dohrmann. Multispace and multilevel BDDC. *Computing*, 83(2):55–85, 2008.

[99] J. Mandel and B. Sousedík. Adaptive selection of face coarse degrees of freedom in the BDDC and the FETI-DP iterative substructuring methods. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1389–1399, Jan. 2007.

[100] M. L. Minion, R. Speck, M. Bolten, M. Emmett, and D. Ruprecht. Interweaving PFASST and Parallel Multigrid. *SIAM Journal on Scientific Computing*, 37(5):S244–S263, Jan. 2015.

[101] P. Monk. *Finite Element Methods for Maxwell's Equations*. Oxford Science Publications, 2003.

[102] G. Mur. Edge elements, their advantages and their disadvantages. *IEEE Transactions on Magnetics*, 30(5):3552–3557, 1994.

[103] C. Navau, A. Sanchez, N. Del-Valle, and D. Chen. Alternating current susceptibility calculations for thin-film superconductors with regions of different critical-current densities. *J. Appl. Phys.*, 103:113907, 2008.

[104] J. Nédélec. Mixed finite elements in R^3. *Numer. Math.*, 35:315–341, 1980.

[105] J. Nédélec. A new family of mixed finite elements in R^3. *Numer. Math.*, 50:57–81, 1986.

[106] M. Neilan and D. Sap. Stokes elements on cubic meshes yielding divergence-free approximations. *Calcolo*, 53(3):1–21, 2015.

[107] N. Nibbio and S. Stavrev. Effect of the geometry of HTS on AC loss by using finite element method simulation with B-dependent E-J power law. *IEEE Trans. Appl. Supercond.*, 11(1):2627–2630, 2001.

[108] S. Nicaise. Edge Elements on anisotropic meshes and approximation of the Maxwell Equations. *SIAM Journal of Numerical Analysis*, 39(3):784–816, 2001.

[109] W. T. Norris. Calculation of hysteresis losses in hard superconductors: polygonal-section conductors. *J. Phys. D*, 4(9):1358, 1971.

[110] D. Oh, O. Widlund, Z. Zampini, and C. Dohrmann. BDDC algorithms with deluxe scaling and adaptive selection of primal constraints for Raviart-Thomas vector fields. *Math. Comp.*

[111] M. Olm, S. Badia, and A. F. Martín. On a general implementation of h and p-adaptive curl-conforming finite elements. *submitted*, 2018.

[112] M. Olm, S. Badia, and A. F. Martín. Simulation of high temperature superconductors and experimental validation. *Computer Physics Communications*, in press, 2018.

[113] R. Ortigosa and A. J. Gil. A new framework for large strain electromechanics based on convex multi-variable strain energies: Finite Element discretisation and computational implementation. *Computer Methods in Applied Mechanics and Engineering*, 302:329–360, 2016.

[114] E. Pardo, D.-X. Chen, A. Sanchez, and C. Navau. Alternating current loss in rectangular superconducting bars with a constant critical-current density. *Supercond. Sci. Technol.*, 17(1):83, 2004.

[115] E. Pardo, D.-X. Chen, A. Sanchez, and C. Navau. The transverse critical-state susceptibility of rectangular bars. *Supercond. Sci. Technol.*, 17(3):537, 2004.

[116] E. Pardo and M. Kapolka. 3D computation of non-linear eddy currents: Variational method and superconducting cubic bulk. *J. Comput. Phys.*, 344:339–363, 2017.

[117] E. Pardo and M. Kapolka. 3D magnetization currents, magnetization loop, and saturation field in superconducting rectangular prisms. *Supercond. Sci. Technol.*, 30(6):064007, 2017.

[118] E. Pardo, J. Šouc, and L. Frolek. Electromagnetic modelling of superconductors with a smooth current–voltage relation: variational principle and coils from a few turns to large magnets. *Supercond. Sci. Technol.*, 28(4):044003, 2015.

[119] R. Pecher, M. McCulloch, S. Chapman, C. Prigozhin, and L. Elliott. 3D-modelling of bulk type-II superconductors using unconstrained H-formulation. In *Proceedings of the 6th European Conference on Applied Superconductivity, EUCAS*, volume 181, 2003.

[120] C. Pechstein and C. R. Dohrmann. A unified framework for adaptive BDDC. *Electronic Transactions on Numerical Analysis*, 46:273–336, 2017.

[121] C. Pechstein and R. Scheichl. Analysis of FETI methods for multiscale PDEs. part ii: interface variation. *Numerische Mathematik*, 118(3):485–529, Jul 2011.

[122] L. Prigozhin. The bean model in superconductivity: Variational formulation and numerical solution. *J. Comput. Phys.*, 129(1):190–200, 1996.

[123] L. Prigozhin. Solution of Thin Film Magnetization Problems in Type-II Superconductivity. *J. Comput. Phys.*, 144:180–193, 1998.

[124] M. Rognes, R. Kirby, and A. Logg. Efficient Assembly of H(div) and H(curl) Conforming Finite Elements. *SIAM J. Sci. Comput.*, 31(6):4130–4151, 2009.

[125] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.

[126] A. Schneebeli. An H(curl;Ω)-conforming FEM: Nédélec elements of first type. Technical report, 2003.

[127] J. Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and Visualization in Science*, 1(1):41–52, Jul 1997.

[128] J. Shadid, R. Pawlowski, E. Cyr, R. Tuminaro, L. Chacón, and P. Weber. Scalable implicit incompressible resistive MHD with stabilized FE and fully-coupled Newton–Krylov-AMG. *Computer Methods in Applied Mechanics and Engineering*, 304:1–25, 2016.

[129] K. Sim, S. Kim, S. Lee, J. Cho, and T. K. Ko. The estimation of the current distribution on the hts cable by measuring the circumferential magnetic field. *IEEE Trans. Appl. Supercond.*, 20(3):1981–1984, 2010.

[130] P. Solin, K. Segeth, and I. Dolezel. *Higher-Order Finite Element Methods*. Chapman & Hall/CRC Press, 2003.

[131] B. Sousedík, J. Šístek, and J. Mandel. Adaptive-Multilevel BDDC and its parallel implementation. *Computing*, 95(12):1087–1119, Dec. 2013.

[132] R. Speck, D. Ruprecht, M. Emmett, M. Bolten, and R. Krause. A space-time parallel solver for the three-dimensional heat equation. *arXiv:1307.7867 [cs, math]*, 2013.

[133] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. A robust two-level domain decomposition preconditioner for systems of pdes. *Comptes Rendus Mathematique*, 349(23):1255 – 1259, 2011.

[134] N. Spillane, V. Dolean, P. Hauret, F. Nataf, and D. J. Rixen. Solving generalized eigenvalue problems on the interfaces to build a robust two-level FETI method. *Comptes Rendus Mathematique*, 351(5):197 – 201, 2013.

[135] A. Stenvall, V. Lahtinen, and M. Lyly. An H-formulation-based three-dimensional hysteresis loss modelling tool in a simulation including time varying applied field and transport current: the fundamental problem and its solution. *Supercond. Sci. Technol.*, 27(10):104004, 2014.

[136] A. Stenvall and T. Tarhasaari. Programming finite element method based hysteresis loss computation software using non-linear superconductor resistivity and $T - \varphi$ formulation. *Supercond. Sci. Technol.*, 23(7):075010, 2010.

[137] A. Toselli. Dual-primal FETI algorithms for edge finite-element approximations in 3D. *IMA Journal of Numerical Analysis*, 26(1):96–130, 2006.

[138] A. Toselli and O. Widlund. *Domain Decomposition Methods: Algorithms and Theory*. Springer-Verlag, Berlin, 2005.

[139] T. Tu, D. R. O'Hallaron, and O. Ghattas. Scalable parallel octree meshing for terascale applications. In *Proceedings of the ACM/IEEE SC 2005 Conference, Supercomputing, 2005*, pages 1–15, 2005.

[140] X. Tu. Three-level BDDC in three dimensions. *SIAM Journal on Scientific Computing*, 29(4):1759–1780, 2007.

[141] A. S. Ulrich Trottenberg, Cornelius Oosterlee. *Multigrid*. Academic Press, 2001.

[142] J. van Nugteren. *High temperature superconductor accelerator magnets*. PhD thesis, University of Twente, Enschede, 2016.

[143] J. van Nugteren, B. van Nugteren, P. Gao, L. Bottura, M. Dhallé, W. Goldacker, A. Kario, H. ten Kate, G. Kirby, E. Krooshoop, G. de Rijk, L. Rossi, C. Senatore, S. Wessel, K. Yagotintsev, and Y. Yang. Measurement and numerical evaluation of AC losses in a ReBCO roebel cable at 4.5 K. *IEEE Trans. Appl. Supercond.*, 26(3):1–7, 2016.

[144] G. Wang, S. Wang, N. Duan, Y. Huangfu, H. Zhang, W. Xu, and J. Qiu. Extended Finite-Element Method for Electric Field Analysis of Insulating Plate With Crack. *IEEE Transactions on Magnetics*, 51(3):1–4, 2015.

[145] T. Weinzierl and T. Köppl. A geometric space-time multigrid algorithm for the heat equation. *Numerical Mathematics: Theory, Methods and Applications*, 5(01):110–130, 2012.

[146] S. Zampini. PCBDDC: A class of robust dual-primal methods in PETSc. *SIAM J. Sci. Comput.*, 38:S282–S306, 2016.

[147] S. Zampini. Adaptive BDDC deluxe methods for H(curl). In C.-O. Lee, X.-C. Cai, D. E. Keyes, H. H. Kim, A. Klawonn, E.-J. Park, and O. B. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XXIII*, pages 285–292, Cham, 2017. Springer International Publishing.

[148] S. Zampini and X. Tu. Multilevel balancing domain decomposition by constraints deluxe algorithms with adaptive coarse spaces for flow in porous media. *SIAM Journal on Scientific Computing*, 39(4):A1389–A1415, 2017.

[149] S. Zampini, P. S. Vassilevski, V. Dobrev, T. Kolev, S. Zampini, P. Vassilevski, V. Dobrev, and T. Kolev. Balancing Domain Decomposition by Constraints Algorithms for Curl-conforming Spaces of Arbitrary Order. pages 1–13, 2017.

[150] M. Zhang and T. A. Coombs. 3D modeling of high-T_c superconductors by finite element software. *Supercond. Sci. Technol.*, 25(1):015009, 2012.

[151] S. Zou. *Magnetization of High Temperature Superconducting Trapped-Field Magnets*. PhD thesis, Karlsruhe Institute of Technology, 2017.