Multilevel matrix-free method for high-performance isogeometric analysis of lattice structures

Clément Guillet, Thibaut Hirschler, Pierre Jolivet, Robin Bouclier

Multilevel matrix-free method for high-performance isogeometric analysis of lattice structures

C. Guillet[a,*], T. Hirschler[b], P. Jolivet[a] and R. Bouclier[c,d,e]

[a]Sorbonne Université, CNRS, LIP6, Paris, France
[b]ICB, Université de Technologie de Belfort-Montbéliard, Sévenans, France
[c]ICA, Université de Toulouse, INSA-ISAE-Mines Albi-UPS-CNRS, Toulouse, France
[d]IMT, Université de Toulouse, UPS-INSA-CNRS, Toulouse, France
[e]Institut Universitaire de France (IUF), France

ARTICLE INFO

Keywords:
isogeometric analysis
lattice structures
multigrid methods
domain decomposition methods
inexact FETI-DP

ABSTRACT

This paper presents a novel high-performance solver for the isogeometric analysis of lattice structures, designed to jointly exploit distributed-memory computing architectures and the specific nature of the problem. This work breaks with conventional approaches that primarily focus on multiscale homogenization or structural elements like beams and shells. Instead, it introduces a solver capable of meeting the overwhelming computational demands of full high-fidelity, fine-scale simulations of lattice structures. The solver features a two-level geometric preconditioner with a fine-level smoother based on overlapping domain decomposition, and a coarse-level correction utilizing an algebraic multigrid method. By leveraging the multiscale nature of the lattice structures, a matrix-free approach is employed at the fine level to perform matrix-vector products and apply transfer operators based on spline k-refinement. The structural similarities of the cells are also exploited through a reduced-order modeling procedure applied within each subdomain, which is used to efficiently compute the corresponding local solves within the fine-level smoother. A series of numerical experiments in both 2D and 3D, spanning various micro- and macro-geometries, are conducted to evaluate the efficiency of the solver in terms of memory usage, computational time, and robustness with respect to mesh refinement, spline degree, and problem size. Notably, an industrially representative spiral channel regenerative cooling thrust chamber lattice structure, consisting of over 66,000 cells, is simulated in minutes using thousands of processes.

1. Introduction

In the field of structural mechanics and materials, architected cellular materials are currently gaining significant momentum in both the scientific and industrial communities [8]. They rely upon a simple idea: mimicking nature, such as bones, to create a material or a structure mostly filled with voids but capable of bearing significant loads [71]. The advent of additive manufacturing has revolutionized their design and fabrication, making it possible to fabricate so-called *lattice structures*. These structures are created by tiling a well-designed unit-cell through geometrical deformation into a macro-shape structure, see Fig. 2 which will be detailed further in Section 2. These finely tuned multiscale heterogeneous structures offer unprecedented weight savings while maintaining stiffness and strength. Additionally, they can be engineered to be highly stretchable and auxetic, which is advantageous for applications such as energy absorption and vibration reduction [61]. Moreover, they can incorporate multi-functionality. All these characteristics make these structures valuable assets for many demanding applications, such as in aerospace [78], automotive manufacturing [43], and biomedical engineering [75].

In this work, we focus on the numerical simulation of the mechanical response of lattice structures, specifically at the design stage of a product, thus neglecting possible process-induced defects at this stage. As interest in designing representative lattice structures grows, along with the improvement of manufacturing technologies enabling their fabrication, the number of cells and their complexity have significantly increased. Consequently, the computational cost in terms of memory and time has become very demanding, or even intractable, if standard methods are applied as black-boxes.

*Corresponding author
clement.guillet@lip6.fr (C. Guillet)
ORCID(s): 0000-0003-3398-2533 (T. Hirschler); 0009-0000-3410-0884 (P. Jolivet); 0000-0002-6265-4726 (R. Bouclier)

Due to this complexity, most current computational methods used for simulating lattice structures rely on specific modeling techniques that avoid fine-scale computations at the architecture level, i.e., at the scale of the strut/beam of the cells. One primary approach involves leveraging multiscale methods based on homogenization that have been developed over the years for multiscale heterogeneous structures, and are now being enhanced with data-based techniques. These strategies aim to compute homogenized macro-scale behavior from heterogeneous microstructures through various scale-interchange methods. Such techniques include multilevel finite element methods (FEM), e.g., FE^2-type [7, 82], multiscale FEM (MsFEM [83]), global/local coupling [79], and direct numerical homogenization [30, 42, 54]. However, these methods often require a clear scale-separation, which is usually not the case with lattice structures since 3D printers constrain the achievable length-scale range of cells, and the macro-shape often follows a rather slender geometry, hence with few cells in one direction. Another class of approaches consists in using advanced beam or shell models (modeling the lattice architecture as a network of beams or shells) [51, 63, 80]. But connecting different beams or shells is challenging [13, 65] and, in any case, this may not suit all types of cell geometries (thick beams, etc.). As a result, it appears that focusing on efficient and scalable solvers capable of dealing with the high-fidelity, fine-scale problem, i.e., considering a volumetric model at the architecture scale, is desirable for lattice structures.

In view of performing high-performance computing (HPC), we begin with accurately modeling lattice geometries by adopting the computer-aided design (CAD) approach known as spline composition, as introduced in [24]. This methodology involves composing two scales, see again Fig. 2 which will be detailed further in Section 2: a microscopic model representing the lattice heterogeneities at the reference unit-cell level (see left-hand side of Fig. 2), and a macroscopic model representing the overall structure shape without these heterogeneities (see right-hand side of Fig. 2). Both models utilize smooth spline parametrization from CAD, employing linear combinations of B-spline or non-uniform rational B-spline (NURBS) shape functions, and thus offering great flexibility in describing locally and globally lattice structures. Additionally, this geometrical modeling is fully consistent with isogeometric analysis (IGA). In particular, it is possible to consider the (microscopic) spline space generating the cells to compute the deformation of the lattice structure within a Galerkin approach during numerical simulations [3, 33]. IGA, initially proposed in [18, 39], aims to bridge the gap between CAD and FEM, essentially serving as a natural extension of FEM with high-order and high-regular spline basis functions. Utilizing these splines allows for a more precise and lighter representation of curved geometries. Moreover, the inherent smoothness of these basis functions often yields greater accuracy per degree of freedom compared to traditional FEM approaches [25], proving advantageous across various applications, see [40, 44, 49, 58] to name a few.

Nevertheless, these advantages must be weighed against the computational cost associated with (i) the formation of operators and (ii) solving linear systems resulting from IGA discretizations. Spline functions, being of high-order, naturally lead to operators with a higher number of non-zero entries compared to standard low-order FEM. Moreover, it is recognized in IGA that the increased smoothness of spline functions induces poor conditioning numbers for the mass and stiffness matrices [28, 29]. Consequently, the standard element loop and Gauss quadrature are known to be far from optimal in IGA [15, 31, 41], and direct solvers and standard iterative methods may not offer efficient solutions [16, 17].

To address the substantial memory and computational cost requirements associated with solving representative problems in IGA and the poor conditioning of corresponding matrices, it appears essential to develop (i) dedicated fast assembly procedures and (ii) preconditioned iterative solvers. For the first point, a multitude of alternative formation procedures taking advantage of the tensor-product nature of spline patches have emerged, such as sum factorization [2, 11], weighted quadrature rules [15, 70], use of lookup tables [52, 62], and low-rank tensor techniques [53, 57]. Following this trend, a fast multiscale assembly procedure for lattice structures modeled by spline composition has recently been proposed in [33]. This method utilizes lookup tables with precomputed integrals associated with the cell pattern and macro-fields that encode the mechanical behavior related to the macro-shape. Given its efficiency, this strategy constitutes the starting point of our approach for the analysis of lattice structures. At this stage, it thus remains necessary to develop a solver that benefits from this specific data structure by allowing a matrix-free procedure, and that is suited to distributed-memory architectures. In this context, domain decomposition (DD) methods and multigrid methods as preconditioners for iterative solvers for IGA have garnered significant interest over the past decade.

The application of DD methods to IGA began with the adaptation of well-known DD methods from FEM to IGA, including finite element tearing and interconnecting (FETI) [48], balancing domain decomposition by constraints (BDDC) [6], and overlapping Schwarz methods [5]. Subsequently, several enhancements and variants of these methods

Table 1
Comparison of multigrid preconditioners for IGA in the literature. Although many interesting works have been performed, particularly from a mathematical perspective, the current state-of-the-art may appear limited from an application standpoint, specifically regarding the practical solution of very large-scale problems in parallel.

Reference	Preconditioner	Smoother	Robustness with h, p	Parallel solver	Multi-dimension solver	Main limitation
Gahalaut et al. [29]	h-multigrid	Gauss–Seidel	✓, ✗	✗	✗	Not robust with p
Donatelli et al. [22]	h-multigrid	Preconditioned Krylov smoother based on spectral information	✓, ✓	✗	✓	Sequential implementation
Hofreither et al. [38]	h-multigrid	Mass matrix with boundary correction	✓, ✓	✗	✗	Difficult extension to $d > 2$
Hofreither and Takacs [37]	h-multigrid	Additive subspace splitting correction (ASSC)	✓, ✓	✗	✓	Limited to single patch geometries
Hofer and Takacs [35], Takacs [74]	h-multigrid	Additive Schwarz with ASSC by patch	✓, ✓	✓	✓	Coarse problem expensive for large number of patches
de Prenter et al. [20]	h-multigrid	Multiplicative or additive Schwarz	✓, –	✗	✓	Sequential implementation
de la Riva et al. [19]	h-multigrid	Overlapping multiplicative Schwarz method	✓, ✓	✗	✗	Extension to parallel not straightforward
Tielen et al. [76]	p-multigrid	Gauss–Seidel	✓, ✗	✗	✗	Not robust with p
Tielen et al. [77]	p-multigrid	ILUT	✓, ✓	✗	✗	Difficult extension to parallel

have been proposed, such as a non-conforming version of the one-level FETI [32], a deluxe variant of BDDC [81], and several inexact FETI alternatives [9, 36, 56] that incorporate inexact local patch solves to possibly facilitate matrix-free procedures. For our specific case of spline composition-based lattice structures, an inexact FETI-DP strategy assisted with reduced order modeling (ROM) has recently been introduced in [34]. This solver leverages similarities between cells in a domain, enabling the approximation of many local (cell- and subdomain-wise) problems by solving only a few principal local problems. Combined with the fast assembly procedure developed in [33], this approach has demonstrated significant reductions in both memory consumption and computational time for the numerical simulation of lattice structures. However, due to the ROM strategy being applied across the entire domain, extending this method to a parallel solver compatible with distributed-memory architectures is not straightforward.

Multigrid methods, introduced several decades ago [27], quickly became one of the most efficient solvers for FEM applied to elliptic problems due to their optimal complexity. That is, the computational cost to solve the linear system grows only linearly with the number of degrees of freedom (DOFs). The first application of multigrid methods to IGA was studied in [29], where it was observed that standard smoothers, such as Gauss–Seidel iteration, result in a rapidly increasing condition number of the stiffness matrix with high-order splines. Further research [22] showed that the spectral radius of multigrid iteration matrices based on standard smoother approaches exponentially 1 as the spline degree p increases. Therefore, significant efforts have been made to achieve robustness with respect to the spline degree, primarily through h-multigrid methods. In h-multigrid, the grid hierarchy is generated by using coarser and coarser mesh sizes while keeping the same spline degree, combined with non-standard smoothers [19, 20, 22, 35, 37, 38, 74]. Additionally, p-multigrid methods have been proposed, where the grid hierarchy is generated by using lower and lower discretization orders while keeping the same mesh characteristic length, also employing both standard and non-standard smoothers [76, 77]. A comparison of all these multigrid solvers in terms of their main properties, such as numerical robustness, is provided in Table 1. From the latter, it appears that further work may be necessary to truly achieve high-performance from an application point of view, that is, to be able to solve efficiently large-scale problems in parallel.

Building on the aforementioned state-of-the-art techniques, our approach to perform isogeometric analysis of lattice structures involves developing a two-grid method based on a low-order correction, similar to p-multigrid methods, that aims to be robust with respect to mesh refinement h, spline degree p, problem size, and, last but not least, to be

compatible with distributed-memory architectures. In order to ensure robustness with respect to spline degree, we opt for a non-standard fine-level preconditioner (or smoother) based on an overlapping domain decomposition method. Practically, we decompose the computational domain into overlapping subdomains made of several cells which are assigned to different processes in a distributed-memory framework (using MPI), and consider a restricted additive Schwarz (RAS) method [14]. Then, the crucial point to solve efficiently the sub-problems (local to each subdomain, or equivalently, each MPI process) is to take advantage of the similarities between cells inside each subdomain thanks to the fast assembly procedure from [33] and the ROM-based solver introduced in [34]. In other words, we consider an inexact FETI-DP method inside each subdomain, in which only the local operators associated to a few cells are constructed and stored, the other being treated with a matrix-free procedure. In the same way, the grid-transfer operators of our two-grid method are never assembled explicitly: only their action on a distributed vector is provided to the algebraic backend. All these choices have also been made in relation to the capabilities offered by the PETSc library [4], on which our implementation is based. Ultimately, the entire approach can be interpreted as a two-grid overlapping Schwarz DD method with a matrix-free formulation on the fine grid and a low-order coarse problem, which may be solved efficiently with off-the-shelf solvers such as algebraic MG (AMG). At this stage, the method is developed to perform linear elastic simulations in the context of small displacements and rotations. With all the aforementioned components, the memory and computational cost are significantly reduced, enabling the simulation of lattice structures that were previously intractable.

The remainder of this paper is organized as follows. First, Section 2 provides the necessary background on lattice structure modeling and simulation using spline composition, and in particular, how this fits in a matrix-free framework. In Section 3, we present our proposed multilevel preconditioning approach involving a two-grid preconditioner with a DD smoother and an embedded AMG coarse solver, thoroughly detailing all its components such as the grid-transfer operators. Section 4 offers extensive numerical experiments to demonstrate the performance of our strategy. Finally, we draw conclusions in Section 5.

2. Lattice structure modeling and discretization

This section establishes the context of the study and introduces the corresponding notations. First, we provide necessary elements regarding B-spline and NURBS geometrical modeling techniques. Then, we outline the lattice structure modeling by spline composition. Finally, we present the corresponding linear elastic problem, along with its IGA discretization and the associated fast assembly procedure for the stiffness matrix needed to solve it.

2.1. B-spline and NURBS technologies

The technology behind IGA is now mature and relatively well-known in the scientific computing community, so only the fundamentals are provided here. For further details, the reader may consult, e.g., the books [10, 18, 64] and the references therein. From a practical point of view, IGA simply consists of using the spline-based parametrizations of CAD environments to build the approximation spaces when applying a Galerkin method. The B-spline and NURBS families are the spline technologies that have become the standard over the years for geometric modeling in CAD and computer graphics. NURBS functions allow for an exact representation of many shapes used in engineering, such as conical sections (circles, cylinders, spheres, ellipsoids, etc.). NURBS are a generalization of B-splines: they can be viewed as rational projections of B-splines. Therefore, they possess many of the properties of B-splines, with the most interesting being their potential for increased smoothness.

The n univariate B-spline basis functions are piecewise polynomials defined by their polynomial degree p and a set of non-decreasing parametric coordinates $\xi_i \in \mathbb{R}$, i being the knot index, collected into a knot-vector $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$. From this knot-vector Ξ, the B-spline basis functions are constructed recursively using the Cox–de Boor formula. The coordinates ξ_i, referred to as knots, divide the parametric space into (knot-span) elements, and the interval (ξ_1, ξ_{n+p+1}), which is usually equal to $(0, 1)$, constitutes the isogeometric patch in the parametric space. Multiple knots can coincide at the same coordinate in the parametric space, known as repeated knots. Across each knot, the basis functions have $p - m_i$ continuous derivatives, where m_i is the multiplicity of the knot ξ_i. Additionally, a knot-vector is termed open if its first and last knots appear $p + 1$ times. This results in the basis being interpolatory at the endpoints of the interval. In this work, we consider open knot-vectors as is standard practice, and are interested in splines with maximal continuity, i.e., C^{p-1}-regularity, so the knot-vectors will have multiplicity one, except for the first and last knots.

The extension to multi-dimensional problem, i.e., involving surfaces and volumes, is done using a tensor-product construction of univariate B-spline functions. Let $d \in \mathbb{N}^*$ be the dimension of the problem, $\boldsymbol{i} = (i_1, \dots, i_d)$ be the d-dimensional knot indexes, $\boldsymbol{p} = (p_1, \dots, p_d)$ be the B-spline degrees, $\boldsymbol{P_i} \in \mathbb{R}^d$ be a control net composed of $\boldsymbol{n}$ control points, and $\boldsymbol{\Xi} = \{\xi_1, \dots, \xi_{\boldsymbol{n}+\boldsymbol{p}+1}\}$ be a d-dimensional knot-vector, i.e., $\boldsymbol{\xi}_j \in \bar{\Omega} := [0,1]^d$, for $j = 1, \dots, d$. A tensor-product d-dimensional B-spline entity is defined by:

$$S_{\mathrm{BS}}(\boldsymbol{\xi}) = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \mathscr{M}_{\boldsymbol{i}}^{\boldsymbol{p}}(\boldsymbol{\xi}) \boldsymbol{P_i}, \quad \boldsymbol{\xi} \in \bar{\Omega}, \quad \text{with } \mathscr{M}_{\boldsymbol{i}}^{\boldsymbol{p}}(\boldsymbol{\xi}) = \prod_{j=1}^{d} \mathscr{M}_{i_j}^{p_j}(\boldsymbol{\xi}_j), \tag{1}$$

and where $\mathscr{M}_{i_j}^{p_j}$ denote the n_j univariate p_j-degree B-spline basis functions, for $j = 1, \dots, d$. Then, to be able to represent exactly conical sections, multivariate NURBS entities can be generated from multivariate B-spline entities as follows:

$$S(\boldsymbol{\xi}) = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \mathscr{N}_{\boldsymbol{i}}^{\boldsymbol{p}}(\boldsymbol{\xi}) \boldsymbol{P_i}, \quad \boldsymbol{\xi} \in \bar{\Omega}, \quad \text{with } \mathscr{N}_{\boldsymbol{i}}^{\boldsymbol{p}}(\boldsymbol{\xi}) = \frac{\omega_{\boldsymbol{i}} \mathscr{M}_{\boldsymbol{i}}^{\boldsymbol{p}}(\boldsymbol{\xi})}{\sum_{\boldsymbol{k}=1}^{\boldsymbol{n}} \omega_{\boldsymbol{k}} \mathscr{M}_{\boldsymbol{k}}^{\boldsymbol{p}}(\boldsymbol{\xi})}, \tag{2}$$

and where $\omega_{\boldsymbol{i}}$ denotes the NURBS weight associated to the $\boldsymbol{i}$th control point. Since NURBS are an extension of B-splines and the remainder of the paper applies to both B-splines and NURBS, we will only refer to Eq. (2) in the following to indicate a general spline mapping, i.e., either a B-spline or a NURBS mapping. At this stage, let us underline that only elementary geometries can be modeled with a single patch. Indeed, given the tensor-product structure of the parametric space, see Eq. (1), a one-patch spline geometry will not differ topologically from a square ($d = 2$) or a cube ($d = 3$). Therefore, we will use multi-patch models, i.e., spline models built from the combination of several spline patches, to construct lattice structures, see Section 2.2. With this in mind, the positions of the control points (and the values of the associated weights in case of NURBS) can be adjusted in order to create the complex shapes encountered in engineering.

One of the primary advantages of IGA lies in its refinement strategies for B-splines, which thus extend to NURBS. Not only does IGA alleviate the need for further communication with the CAD system, but its refinement process is also robust and efficient, and it offers an additional strategy compared to classical FEM. Alongside the direct use of knot-insertion and degree-elevation, which allows to recover h-refinement and p-refinement in FEM, respectively, a novel approach known as k-refinement is possible in IGA. This strategy boasts efficiency and robustness advantages over traditional p-refinement, enabling the increase of both the polynomial degree and regularity while maintaining the initial exact geometry.

The knot-insertion technique consists of adding new knots to knot-vectors. To perfectly replicate h-refinement, each new knot value needs to be inserted p times so that functions will be C^0 across the new element boundary. The degree-elevation technique involves increasing the polynomial degree of basis functions. It is important to note that during degree-elevation, all values in the knot-vector are repeated to preserve the initial discontinuities of function derivatives. Therefore, if one starts with a spline mesh involving C^0 basis functions at the knots, degree-elevation exactly coincides with p-refinement in FEM.

The method of k-refinement capitalizes on the non-commutative nature of knot-insertion and degree-elevation processes. By initially elevating the polynomial degree of splines and then inserting knots, k-refinement ensures the maximum available regularity of basis functions at inserted knots, namely C^{p-1}. k-refinement has emerged as a superior approach for achieving high-precision analysis compared to p-refinement. Indeed, for a given polynomial degree and a similar number of elements, a C^{p-1} spline mesh comes with fewer degrees of freedom than the corresponding C^0 FE mesh, which is totally understandable since the space of C^{p-1} functions is included into the space of C^0 functions. An example illustrating the differences between p-refinement and k-refinement is given in Fig. 1. It can be observed that providing C^0-regularity during the refinement process leads to a proliferation of nodes. In contrast, with k-refinement, only one basis function is added each time the degree is elevated, and only one more is added each time a knot is inserted. Furthermore, the smoother derivatives obtained through k-refinement have the potential to enhance the accuracy of mechanical quantities such as strains and stresses.

A final attractive feature is that matrix representations of the spline refinement procedures are possible. In other words, denoting by $\mathbf{N}^{\boldsymbol{p}_0,H}$ and $\mathbf{N}^{\boldsymbol{p},h}$ the matrices collecting, respectively, the $\boldsymbol{n}_H$ coarse (low-degree $\boldsymbol{p}_0$) and $\boldsymbol{n}_h$ fine

(potentially high-degree $\boldsymbol{p}$) spline functions, we can build the refinement operators P_ξ, P_η, and P_ζ associated to each of the parametric directions, respectively, and write:

$$\mathbf{N}^{\boldsymbol{p}_0,H} = P_\xi P_\eta P_\zeta \mathbf{N}^{\boldsymbol{p},h} \quad (\boldsymbol{n}_H < \boldsymbol{n}_h). \tag{3}$$

Obviously, the refinement operators P_ξ, P_η, and P_ζ result in the product of several univariate refinement matrices associated to the different knot-insertions and degree-elevations performed in the whole refinement. Eq. (3) offers a simple way to build the refined spline mesh from the coarse one.

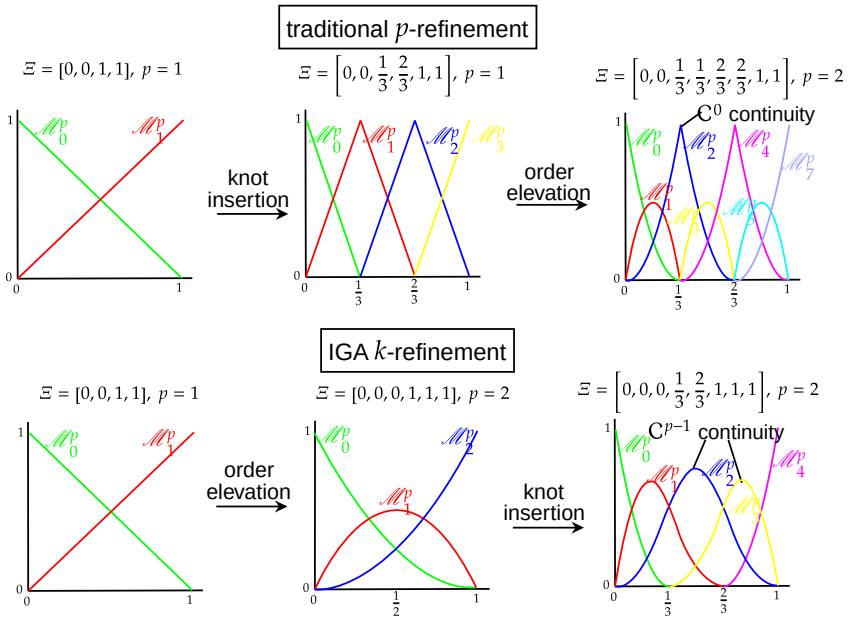


Figure 1: Difference between p-refinement and k-refinement strategies for a uniform open knot-vector Ξ. The more elements and higher polynomial degree, the more degrees of freedom are saved using k-refinement compared to traditional p-refinement.

2.2. Geometrical modeling by spline composition

In this work, lattice structures are accurately modeled at the architecture (micro) level using a functional composition approach with splines (either B-splines or NURBS), as first introduced in [24]. This approach involves two main components, see Fig. 2 for illustration: a macro-representation of the lattice structure where heterogeneities are not represented, and a reference microstructure that defines the pattern to be tiled into the macro-geometry (to generate the cells). The final heterogeneous structure is obtained by embedding the reference microstructure into the macro-model through composition.

First, we consider a reference tile, or reference microstructure denoted by Ω_{ref}.

Definition 2.1 (reference microstructure)**.** *Let $\Omega^{(k)}_{ref}$ be N_p reference patches, assumed to be fully matching, i.e., intersections $\Omega^{(k)}_{ref} \cap \Omega^{(l)}_{ref}$ for $k \neq l$ are either empty, common vertices, common edges, or common faces (in 3D), and the discretizations on common boundaries of the patches perfectly match, and let $\boldsymbol{P}^{(k)}_{m,\boldsymbol{i}}$ be associated control points. Then, the reference tile is constructed from these reference patches by a multi-patch spline model following general spline mappings Eq. (2):*

$$\Omega_{ref} := \bigcup_{k=1}^{N_p} \Omega^{(k)}_{ref}, \quad \Omega^{(k)}_{ref} := S^{(k)}_m(\bar{\Omega}_m), \quad S^{(k)}_m(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}_m} \mathcal{N}^{(k),\boldsymbol{p}_m,h}_{\boldsymbol{i}}(\boldsymbol{\theta}) \boldsymbol{P}^{(k)}_{m,\boldsymbol{i}}, \quad \boldsymbol{\theta} \in \bar{\Omega}_m, \tag{4}$$

where $\bar{\Omega}_m := [0,1]^d$ is the parametric space of the patches composing the micro-model, and $\boldsymbol{n}_m$ and $\boldsymbol{p}_m$ are the number of basis functions and polynomial degrees of the corresponding splines in each direction.

Note that the basis functions of this reference micro-model are indicated with a superscript $(\cdot)^h$, signifying that they pertain to the microstructure mapping.

We then consider a macroscopic model representing the global shape of the lattice structure, prescribed in our case by a second multi-patch spline model. This macro-mapping naturally introduces a partitioning of the domain into N_s macro-elements (each of which will include one cell or tile in the final geometry). Following the procedure in [33], we rewrite this macroscopic model using Bézier extraction so that the parameter spaces associated with each macro-element become identical, the difference between one macro-element and another being thus only the underlying geometric mapping. The Bézier macro-mappings associated with macro-elements are given by:

$$\mathcal{B}_M^{(s)}(\boldsymbol{\xi}) = \sum_{i=1}^{\boldsymbol{n}_M} \mathscr{B}_{\boldsymbol{i}}^{\boldsymbol{p}_M,\tilde{h}}(\boldsymbol{\xi}) \boldsymbol{P}_{M,\boldsymbol{i}}^{(s)}, \quad \boldsymbol{\xi} \in \bar{\Omega}_M, \quad \forall s = 1, \dots, N_s, \tag{5}$$

where $\bar{\Omega}_M := [0,1]^d$ is the parametric space of each macro-element, $\mathscr{B}_{\boldsymbol{i}}^{\boldsymbol{p}_M,\tilde{h}}$ denotes the Bézier basis functions, $\boldsymbol{P}_{M,\boldsymbol{i}}^{(s)}$ stands for the associated control points, and $\boldsymbol{n}_M$ and $\boldsymbol{p}_M$ are the corresponding number of basis functions and polynomial degrees of the splines in each direction. The superscript $(\cdot)^{\tilde{h}}$ refers to the macroscopic discretization parameter.

Definition 2.2 (full lattice structure)**.** *The full lattice structure domain Ω is defined by a union of N_s fully-matching tiles, denoted by $\Omega^{(s)}$, each of which being decomposed into N_p patches:*

$$\Omega = \bigcup_{s=1}^{N_s} \bigcup_{k=1}^{N_p} \Omega^{(s),(k)}, \quad \Omega^{(s),(k)} := \left\{ \boldsymbol{x} \in \mathbb{R}^d \mid \boldsymbol{x} = \mathscr{T}^{(s),(k)}(\boldsymbol{\theta}), \forall \boldsymbol{\theta} \in \bar{\Omega}_m \right\}.$$

For all $s = 1, \dots, N_s$, the cells $\Omega^{(s)}$ in the physical space are generated with the composition of the reference micro-model and the Bézier macro-mappings:

$$\begin{aligned} \mathscr{T}^{(s),(k)} : \bar{\Omega}_m &\to \Omega^{(s),(k)} \\ \boldsymbol{\theta} &\mapsto (\mathcal{B}_M^{(s)} \circ \mathcal{S}_m^{(k)})(\boldsymbol{\theta}). \end{aligned}$$

For the compositions to be admissible, the reference tile must lie in the parameter space of the macro-model, i.e., $\mathcal{S}_m^{(k)}(\bar{\Omega}_m) \subset \bar{\Omega}_M$. Once again, see Fig. 2 for an illustrative representation of this spline composition model.

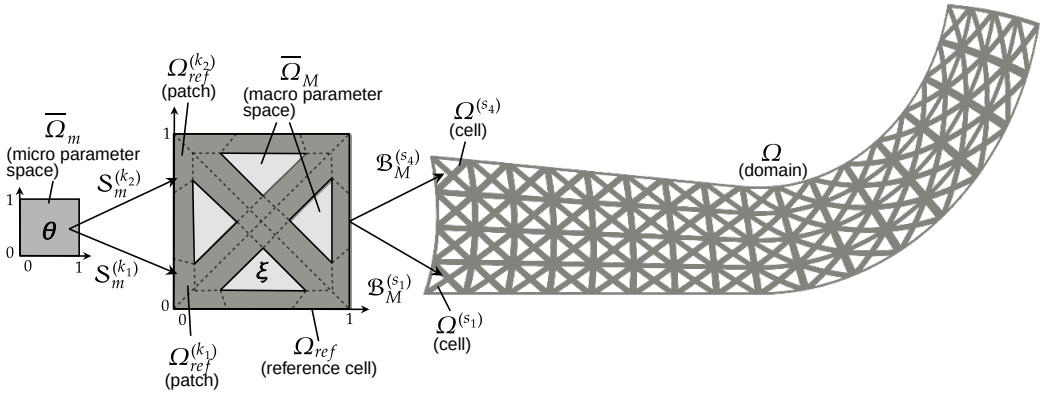


Figure 2: Brake pedal lattice structure, inspired from [60]: it is defined in this work by spline composition of a reference microstructure (unit-cell or tile) and a macro-geometry mapping.

2.3. Model problem and IGA discretization

2.3.1. Linear elasticity problem

We consider here the equations of linear elasticity modeling small displacements of a linear material under the action of internal and external forces. The structure is defined by a domain $\Omega \subset \mathbb{R}^d$. The boundary of the domain is denoted by $\partial\Omega$. For the presentation below, we assume that a part of the boundary, denoted $\partial\Omega_D$, is clamped with homogeneous Dirichlet conditions. The rest of the boundary, denoted by $\partial\Omega_N = \partial\Omega \backslash \partial\Omega_D$, is subject to a surface force $\boldsymbol{g}$, which may vanish on a part of Ω_N. We also introduce a body force $\boldsymbol{f}$, e.g., the gravity. The appropriate space for a variational formulation of this linear elasticity problem is the Sobolev space $\boldsymbol{H}^1_0(\Omega, \partial\Omega_D) := \{\boldsymbol{v} \in \boldsymbol{H}^1(\Omega) \mid \boldsymbol{v} = 0 \text{ on } \partial\Omega_D\}$, where $\boldsymbol{H}^1(\Omega) = \left[H^1(\Omega)\right]^d$ is the d-dimensional H^1 Sobolev space.

The linear elasticity problem then consists of finding the displacement $\boldsymbol{u} \in \boldsymbol{H}^1_0(\Omega, \partial\Omega_D)$ such that $\forall \boldsymbol{v} \in \boldsymbol{H}^1_0(\Omega, \partial\Omega_D)$:

$$a(\boldsymbol{u}, \boldsymbol{v}) = b(\boldsymbol{v}), \tag{6}$$

where:

$$a(\boldsymbol{u}, \boldsymbol{v}) := 2\mu \int_\Omega \varepsilon(\boldsymbol{u}) : \varepsilon(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} + \lambda \int_\Omega (\nabla \cdot \boldsymbol{u})(\nabla \cdot \boldsymbol{v}) \, \mathrm{d}\boldsymbol{x},$$
$$b(\boldsymbol{v}) := \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_N} \boldsymbol{g} \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{s}.$$

We use above the symmetric gradient notation $\varepsilon(\boldsymbol{v}) := (\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T)/2$ and the material Lamé coefficients λ and μ.

2.3.2. IGA discretization

IGA is based on the isoparametric element concept, in which the same basis functions are used for both the computational domain definition and the discretization spaces of the problem. The idea is to construct an approximation space for the variational formulation Eq. (6) using a Galerkin discretization with the same spline functions that we used for modeling lattice structures in Section 2.2. In accordance with [33, 55], we choose the splines from the micro-model as basis functions (see Eq. (4) from Definition 2.1) for both test and trial functions, rather than using the splines resulting from the composition. By making this selection, we deviate from strict adherence to the isoparametric concept. The rationale behind reusing the functions involved in the mapping of the reference microstructure is twofold [33]. First, it helps limit the degree of the approximation subspace, which may otherwise be excessively high due to the composition. Second, besides maintaining control over the degree of the solution, this approach ensures uniform discretization of the displacement field across all cells, which is a prerequisite for achieving efficient solution by taking advantage of the similarity of the cells [33, 34].

Here and in the following, we assume for the sake of clarity that the degree and number of basis functions are identical for all patches of the micro-model. We denote by p the degree and by n_m the number of basis functions per patch. Additionally, for the analysis, it is necessary to consider a functional space that is at least C^0 over the entire computational domain. Consequently, we derive the following continuous basis functions from those used in Eq. (4).

Definition 2.3 (C^0 basis functions over the reference cell)**.** *To begin, we introduce the n^{ref} C^0-continuous basis functions associated with the reference cell as follows:*

$$\mathcal{N}^{ref}_i : \Omega_{ref} \to \mathbb{R}, \quad i = 1, \dots, n^{ref},$$

defined such that:

$$\operatorname{span}\left\{\mathcal{N}^{ref}_i \mid i = 1, \dots, n^{ref}\right\} = \left\{v \in C^0(\Omega_{ref}) \mid \forall k, v_{|\Omega^{(k)}_{ref}} \in \operatorname{span}\{\mathcal{N}^{(k),p,h}_j \circ \mathcal{S}^{(k)-1}_m \mid j = 1, \dots, n_m\}\right\},$$

Clearly, this implies $n^{ref} < N_p n_m$.

Definition 2.4 (C^0 basis functions over the computational domain)**.** *Next, we introduce the n C^0-continuous basis functions associated with the entire computational domain as follows:*

$$\mathcal{N}_i : \Omega \to \mathbb{R}, \quad i = 1, \dots, n,$$

defined such that:

$$\begin{aligned}\text{span}\left\{\mathcal{N}_i \mid i=1,\dots,n\right\} &= \left\{v \in C^0(\Omega) \mid \forall s, v_{|\Omega^{(s)}} \in \text{span}\{\mathcal{N}_j^{ref} \circ \mathcal{B}_M^{(s)-1} \mid j=1,\dots,n^{ref}\}\right\},\\ &= \left\{v \in C^0(\Omega) \mid \forall s, \forall k, v_{|\Omega^{(s),(k)}} \in \text{span}\{\mathcal{N}_j^{(k),p,h} \circ \mathscr{T}^{(s),(k)-1} \mid j=1,\dots,n_m\}\right\}.\end{aligned}$$

This implies that $n < N_s n^{ref} < N_s N_p n_m$.

A standard choice for these C^0 basis functions consists in strongly gluing matching inter-patch or inter-cell basis functions. Making use of these new notations, the test and trial displacement fields can be expressed as:

$$\boldsymbol{u}_h = \sum_{i=1}^{n} \mathcal{N}_i \mathbf{u}_{h,i} \text{ in } \Omega,$$

where $\mathbf{u}_{h,i}$ is a vector of size d that contains the coefficients (one per spatial direction) corresponding to the ith element of the spline basis. We finally introduce the d-dimensional vector of basis functions, where each component corresponds to one degree of freedom (DOF) of the spline model:

$$\boldsymbol{\mathcal{N}}_i = \begin{pmatrix} \mathcal{N}_i \\ \vdots \\ \mathcal{N}_i \end{pmatrix}, \quad i=1,\dots,n. \tag{7}$$

The approximation space for our Galerkin discretization is then:

$$\boldsymbol{V}_{p,h} := \left\{\boldsymbol{v} \in \boldsymbol{H}_0^1(\Omega, \partial\Omega_D) \mid \boldsymbol{v} \in \text{span}\left\{\boldsymbol{\mathcal{N}}_i \mid i=1,\dots,n\right\}\right\}, \tag{8}$$

and the discrete problem, resulting from the Galerkin formulation, is given by: find $\boldsymbol{u}_h \in \boldsymbol{V}_{p,h}$ such that $\forall \boldsymbol{v}_h \in \boldsymbol{V}_{p,h}$:

$$a(\boldsymbol{u}_h, \boldsymbol{v}_h) = b(\boldsymbol{v}_h),$$

which can be written as a linear system by decomposing the test and trial functions in the basis of $\boldsymbol{V}_{p,h}$:

$$\mathbf{K}_h \mathbf{u}_h = \mathbf{f}_h. \tag{9}$$

The size of the stiffness matrix is equal to dn, which is also the sum of DOFs of our mechanical model.

Two main challenges arise from Eq. (9) from a computational cost and memory storage point of view: (i) the assembly of the stiffness matrix $\mathbf{K}_h$ and (ii) solving the associated linear system. In this paper, we start by making use of the fast assembly procedure proposed in [33], and then develop an application-specific preconditioner to efficiently solve iteratively Eq. (9) on distributed-memory architectures.

2.3.3. Fast assembly procedure

For intricate high-order reference microstructures and large numbers of macro-elements, *i.e.*, large numbers of cells, the assembly of the stiffness matrix $\mathbf{K}_h$ can be very demanding. This issue is well-known in the context of IGA, and successful procedures have been developed to reduce the assembly time compared to the standard FEM procedure based on element loops and Gauss quadrature. As stated above, we consider here the multiscale assembly strategy introduced in [33] specifically for IGA of lattice structures defined by spline composition. Instead of performing a naive assembly strategy, the key is to rely on the similarities between cells to avoid computing the same quantities multiple times. We briefly recall the main elements of this approach here, but the reader is encouraged to refer to [33] for further details.

The starting point is to pull back the macro-geometry mapping from the local stiffness matrices so that the integral is defined on the reference tile:

$$\mathbf{K}_{h,i,j}^{(s)} = \sum_{l=1}^{d}\sum_{r=1}^{d} \int_{\Omega_{\text{ref}}} \frac{d\mathcal{N}_i^{ref}}{d\xi_l} \frac{d\mathcal{N}_j^{ref}}{d\xi_r} \mathbf{T}_{l,r}^{\tilde{h}(s)} d\boldsymbol{\xi}, \tag{10}$$

where $\mathbf{T}_{l,r}^{\tilde{h}(s)} : \bar{\Omega}_M \to \mathbb{R}^{d\times d}$ are the macro-fields and involve geometrical and material quantities associated only with the macro-scale model, such as, e.g., the Jacobian of the macro-geometry mapping. The full expression of these fields in the context of linear elasticity can be found in [33, Eq. (37)]. Usually, these macro-fields are smooth (rational) polynomial functions defined over the macro-geometry. The idea is to build and use a unique lookup table with precomputed integrals gathering all the reference microstructure information common to every cells. Therefore, the macro-fields are projected onto a spline space identical for each tile and defined by multivariate Bernstein polynomials of degree p_π:

$$\Pi^{\tilde{h}}\mathbf{T}_{l,r}^{\tilde{h}(s)}(\boldsymbol{\xi}) = \sum_{t=1}^{n_\pi} \mathscr{B}_t^{p_\pi,\tilde{h}}(\boldsymbol{\xi})\boldsymbol{\tau}_t^{\tilde{h}(s)}, \quad \boldsymbol{\xi} \in \bar{\Omega}_M,$$

where n_p is the number of basis functions for the projection spline space, and $\boldsymbol{\tau}_t^{\tilde{h}(s)}$ are the coefficients of the macro-fields in this basis. These coefficients are determined during an initialization phase by solving multiple linear systems. Because they depend solely on the macro-model basis functions and mapping, these linear systems are small (macro-element-wise) and are independent, making them computationally inexpensive to solve. By approximating the macro-fields with their projections onto the spline space and substituting them into Eq. (10), we get:

$$\mathbf{K}_{h,i,j}^{(s)} = \sum_{l=1}^{d}\sum_{r=1}^{d}\sum_{t=1}^{n_\pi} \boldsymbol{\tau}_t^{\tilde{h}(s)} \underbrace{\int_{\Omega_{\text{ref}}} \frac{\mathrm{d}\mathcal{N}_i^{ref}}{\mathrm{d}\xi_l}\frac{\mathrm{d}\mathcal{N}_j^{ref}}{\mathrm{d}\xi_r}\mathscr{B}_t^{p_\pi,\tilde{h}}(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}}_{\text{independent of } \mathcal{B}_M^{(s)}}.$$

The remaining integrals depend only on the geometric model of the reference microstructure and the macro-projection space, so they can be precomputed via Gauss quadrature during the initialization phase and stored in lookup tables. Obviously, the same treatment can be performed with the right-hand side $\mathbf{f}_h$ in Eq. (9).

3. Proposed multilevel preconditioning approach

Building on the context introduced in previous sections, we now thoroughly describe our proposed multilevel preconditioner, specifically designed for solving large-scale lattice structure problems. To facilitate understanding, we first provide a brief overview of the developed approach. We then delve into each component of the method in more detail. Specific care is given to presenting the approach using a top-down approach: from the fine level, where the reference problem is posed, to the coarser levels that correct approximations made throughout the grid hierarchy. Finally, a discussion is conducted at the end to justify the choices made in our approach compared to other possible options.

3.1. Global overview of the method

Multilevel methods serve as efficient preconditioners for linear systems by employing a hierarchy of discretizations to decrease the contraction factor of iterative methods. These methods comprise two main components: relaxation operations (also called smoothers) that allow for partial solving (focusing on the high-frequency part) at the fine levels of the hierarchy, and coarse corrections that globally correct the error.

In this context, we focus on a geometric two-grid method which hierarchy is generated through spline k-refinement. The proposed strategy can be summarized by its main components, as follows:

(i) two geometrical levels are considered: a fine- and a coarse- level, each with corresponding nested discretization spaces;

(ii) a matrix-free approach is employed at the fine level to compute matrix-vector products, as needed by iterative methods;

(iii) a dedicated non-standard DD-based fine-level preconditioner (smoother) is used;

(iv) transfer (restriction and prolongation) operators, based on the k-refinement procedure, are used between the two levels in a matrix-free fashion as well;

(v) on the coarse (second) level, an AMG correction is applied.

These components are briefly introduced here but shall be presented in more detail in the subsequent sections.

A fine mesh is constructed by applying the k-refinement strategy to a "reference" coarse mesh until the desired order p and sufficient mesh refinement are achieved for accurate numerical simulation. In practice, the reference coarse mesh is the coarsest spline mesh (large elements and low-degree) that exactly represents the desired lattice structure geometry in an analysis-suitable manner. Note that this implies that the Bézier macro-mappings from Eq. (5) are the same for both the coarse and fine meshes, so the refinement of the latter differs only at the microstructure level (see Eq. (4) from Definition 2.1). Based on the fine mesh, a fine discretization space is introduced at the fine level, similarly as in Eq. (8).

Definition 3.1 (fine discretization)**.** *Let $\{\boldsymbol{\mathcal{N}}_i^{p,h} \mid i = 1, \dots, n_h\}$ be the C^0 basis functions over the entire computational domain associated with the fine mesh (refer to Definition 2.4 with parameters h and p for the general form), then the fine approximation space is defined by an IGA discretization such that:*

$$\boldsymbol{V}_{p,h} := \left\{ \boldsymbol{v} \in \boldsymbol{H}^1_0(\Omega, \partial\Omega_D) \mid \boldsymbol{v} \in \operatorname{span}\left\{ \boldsymbol{\mathcal{N}}_i^{p,h} \mid i = 1, \dots, n_h \right\} \right\}.$$

This fine discretization space is constructed from n_h vectors of basis functions, see Eq. (7), of high-order (p) and high-regularity (generally C^{p-1}) within each patch. The number of DOFs of the associated fine model is dn_h.

The fine stiffness matrix of dimension dn_h associated to this fine discretization is denoted by $\mathbf{K}_h$. Our approach does not require the explicit assembly of this matrix. Specifically, matrix-vector products are performed on-the-fly in a matrix-free fashion using the fast assembly procedure described in Section 2.3.3.

At this level, a DD preconditioner, which involves an overlapping decomposition of the computational domain into subdomains, is defined to act as a (post-)smoother in our two-grid method.

Definition 3.2 (subdomains)**.** *The computational domain Ω is decomposed into N_Σ subdomains. A subdomain consists of a union of several adjacent cells. There are actually two notions of subdomains, distinguished here through the terminologies non-overlapping subdomains and overlapping subdomains (see Fig. 3 for illustration). As their names suggest, the non-overlapping subdomains result from a non-overlapping decomposition of the computational domain, while the overlapping subdomains include an overlapping layer with a minimum symmetric width of one cell. In the following, the overlapping subdomains are denoted by $\mathcal{S}^{[\sigma]}$, where $\sigma = 1, \dots, N_\Sigma$. The overlap used in our method typically corresponds to one cell, which is of order $\tilde{h}$, where $\tilde{h}$ denotes the macroscopic discretization parameter.*

An illustration of a four-way decomposition is provided in Fig. 3, reusing the lattice structure example depicted in Fig. 2. The smoother consists of a single application of the restricted additive Schwarz (RAS) method. The action of the local-to-each-overlapping-subdomain inverse of the stiffness matrix is iteratively computed using the ROM-based inexact FETI-DP solver introduced in [34]. The ROM strategy leverages the repetitive nature of cells to build reduced bases, efficiently approximating the numerous local systems (pertaining to each cell) from a few principal cells (see again Fig. 3 for illustration). The inexact FETI-DP framework does not require the representation of the assembled matrix on the fine discretization, making it well-suited to our matrix-free approach.

A second level, which is typically needed to improve the numerical efficiency of single-level preconditioners/smoothers, is introduced in the grid hierarchy by utilizing the previously mentioned reference coarse mesh.

Definition 3.3 (coarse discretization)**.** *Let $\{\boldsymbol{\mathcal{N}}_i^{p_0,H} \mid i = 1, \dots, n_H\}$ be the C^0 basis functions over the computational domain associated with the coarse mesh of the reference microstructure. The coarse approximation space is then defined using these low-degree functions such that:*

$$\boldsymbol{V}_{p_0,H} := \left\{ \boldsymbol{v} \in \boldsymbol{H}^1_0(\Omega, \partial\Omega_D) \mid \boldsymbol{v} \in \operatorname{span}\left\{ \boldsymbol{\mathcal{N}}_i^{p_0,H} \mid i = 1, \dots, n_H \right\} \right\}.$$

This coarse discretization space is constructed from n_H vectors basis functions of low-order (p_0) and low-regularity (generally C^0) within each patch. The number of DOFs of the associated coarse model is dn_H.

Remark 3.1. *Note that since the fine mesh is constructed by applying k-refinement to the coarse mesh, the approximation spaces are nested, i.e., $\boldsymbol{V}_{p_0,H} \subset \boldsymbol{V}_{p,h}$.*

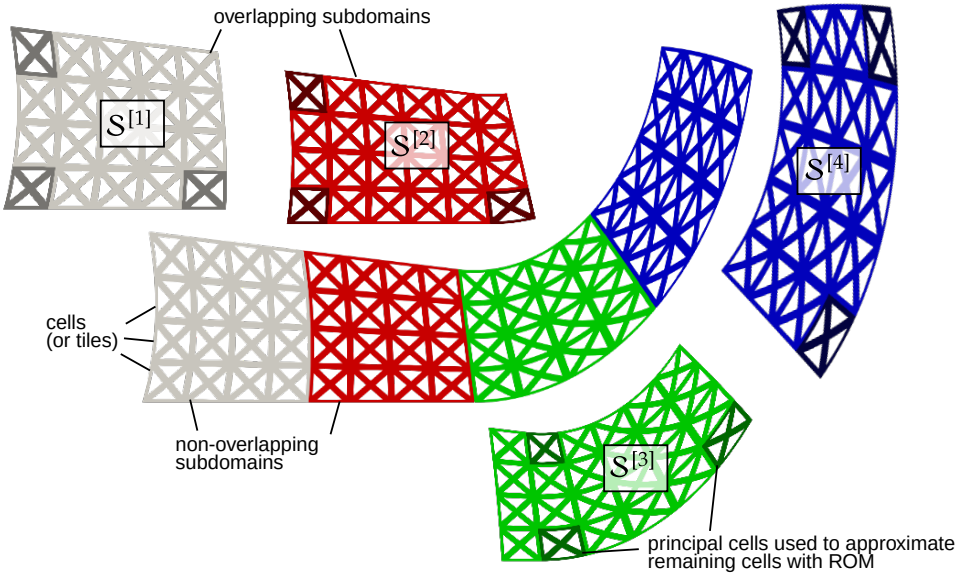


Figure 3: Illustration of the domain decomposition into subdomains and cells.

The coarse stiffness matrix of dimension dn_H associated to this coarse discretization is denoted by $\mathbf{K}_H$. Since this matrix is constructed from (a limited number of) low-order basis functions, it is assumed to be inexpensive to compute and to store and is therefore assembled explicitly.

The grid-transfer operators, used whenever the two-grid method is applied on a vector for restricting residuals from $\boldsymbol{V}_{p,h}$ to $\boldsymbol{V}_{p_0,H}$ and prolongating corrections from $\boldsymbol{V}_{p_0,H}$ to $\boldsymbol{V}_{p,h}$, are constructed using the matrix representation of the spline k-refinement procedure , see Eq. (3). For the coarse (second) level, corrections involving the action of the inverse of $\mathbf{K}_H$ are computed using AMG, thus making the resulting complete preconditioner multilevel (two geometric and several algebraic levels) with a priori no computational bottleneck, since AMG is known to be efficient for low-order elliptic discretizations.

A schematic representation of the complete multilevel preconditioner is provided in Fig. 4.

3.2. Detailed components of the method

Now that the overall approach is outlined, let us discuss each aspect of the method in finer detail.

3.2.1. Matrix-free formulation

At the fine level, a matrix-free approach is employed to avoid explicitly assembling the fine stiffness matrix $\mathbf{K}_h$. The motivation behind this is to significantly reduce memory costs associated with storing a matrix resulting from high-order discretization. Matrix-free methods necessitate defining an efficient procedure for performing matrix-vector products without explicit matrix assembly.

Here, the matrix-vector product of the fine stiffness matrix is efficiently computed using the fast multiscale assembly technique outlined in Section 2.3. During the initialization phase, the macro-field projection coefficients and the lookup tables are precomputed. Consequently, performing the matrix-vector product involves only the multiplication of these macro-field projection coefficients, the lookup tables, and an assembly operator (from tile to full structure). Once again, we advise the interested reader to consult [33] for further details on the fast assembly procedure. In this work, the strategy is efficiently implemented to compute the matrix-vector product in parallel within each non-overlapping subdomain. Technically speaking, the implementation uses a PETSc MatIS, with local (to each MPI process corresponding to each non-overlapping subdomain) matrices represented as MatShell.

3.2.2. Fine-level preconditioner/smoother

Given the state-of-the-art introduced in Section 1, the choice of the smoother at all levels of the hierarchy is crucial for multigrid methods applied to IGA discretizations. In standard FEM discretizations of elliptic problems, smoothers

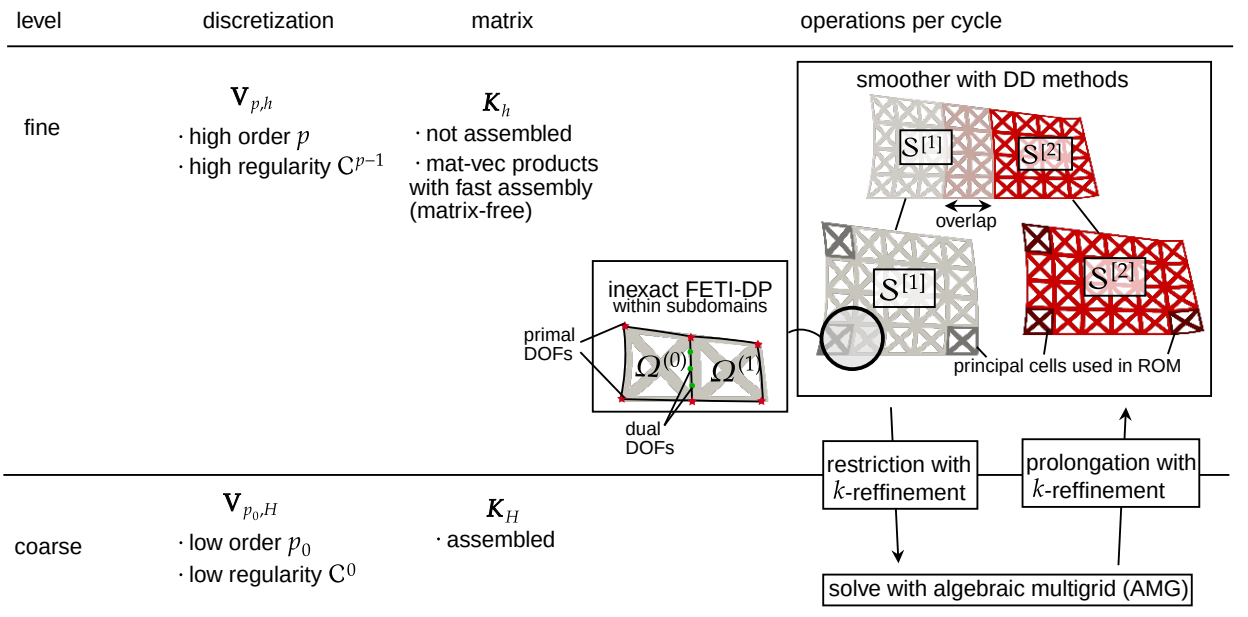


Figure 4: Schematic representation of the proposed multilevel preconditioner.

like Jacobi or Gauss–Seidel iterations are typically used. However, for IGA, these smoothers cause significant convergence deterioration as the spline degree p increases.

3.2.2.a. Non-standard DD-based preconditioner/smoother

To address this issue, we employ a non-standard smoother based on the RAS method [14]. Schwarz methods with overlap are known for efficiently damping high-frequency error components due to the overlap [21], making them suitable candidates for effective smoothing. RAS relies on the decomposition of the domain into N_Σ overlapping subdomains, as defined in Definition 3.2. These subdomains include an extra layer of one tile of overlap, and are illustrated in Fig. 3. Let $\mathbf{M}^{-1}_{h,\mathrm{RAS}}$ be the fine-level smoother defined by:

$$\mathbf{M}^{-1}_{h,\mathrm{RAS}} = \sum_{\sigma=1}^{N_\Sigma} \left(\tilde{\mathbf{R}}_h^{[\sigma]}\right)^T \left(\mathbf{K}_h^{[\sigma]}\right)^{-1} \mathbf{R}_h^{[\sigma]},$$

where $\mathbf{R}_h^{[\sigma]}$ are restriction matrices from the global domain to overlapping subdomains. The matrices $\tilde{\mathbf{R}}_h^{[\sigma]}$ are the same as the matrices $\mathbf{R}_h^{[\sigma]}$ except that they are weighted to take into account duplicated DOFs on the overlap. The matrix $\mathbf{K}_h^{[\sigma]}$ represents the restriction of the globally assembled stiffness matrix to the subdomain labelled as $[\sigma]$. Technically speaking, this is implemented using the PETSc PCASM machinery, with a custom geometric overlap defined via PCASMSetLocalSubdomains. Next, we will explain how the action of the subdomain solvers, i.e., $(\mathbf{K}_h^{[\sigma]})^{-1}$, on a local vector are computed.

3.2.2.b. Inexact FETI-DP with ROM

To approximate the action of subdomain-local inverses $(\mathbf{K}_h^{[\sigma]})^{-1}$ on a given vector, we utilize, within each overlapping subdomain, a non-overlapping substructuring method at the level of the cells: the ROM-based inexact FETI-DP method introduced in [34] (see again Fig. 4 to locate where this method fits within the overall proposed approach). This inexact FETI-DP has been specifically developed to take advantage of the similarities of the cells in a lattice structure from a computational cost and memory storage point of view. In brief, it involves three main components.

(i) In FETI-DP, the continuity constraints on the displacement at the cell corners are maintained throughout the iterative process, naturally leading to a coarse problem at each iteration, while other constraints are enforced using Lagrange multipliers.

(ii) We utilize the framework of inexact FETI-DP algorithms [45], developing an inexact version that avoids solving numerous local systems. This is achieved by iterating on the initial complete saddle-point problem and designing a block preconditioner consistent with the mathematical foundation established in [72].

(iii) We apply a ROM approach [66], particularly a greedy technique, to exploit the repetitiveness of the cells. This technique extracts the "principal" cells in terms of stiffness. The corresponding few principal local stiffness operators are then used in the preconditioner of the inexact FETI-DP to build reduced bases for efficiently approximating the numerous other local systems.

It has been demonstrated that the resulting algorithm significantly reduces computational costs compared to the standard FETI-DP solver, particularly when the macro-mapping deformation is minimal, as the number of reduced basis elements is very small in this situation [34]. Here, since we apply the ROM technique only inside each overlapping subdomain, and not over the entire computational domain at once as done in [34], we find ourselves in an ideal situation for the method. Indeed, the cells within a subdomain are closer to each other than they are across the entire structure. Furthermore, performing ROM subdomain-by-subdomain is necessary to be compatible with distributed-memory architectures, allowing different local-to-each-subdomain ROM to be carried out concurrently on each MPI process. Technically speaking, this is implemented using a PETSc PCShell for which we write a callback which implements the local solution of a linear system with a given local input vector using the inexact FETI-DP preconditioner. This PCShell is then passed to the outer PCASM, cf. previous paragraph, using PCASMGetSubKSP.

In the rest of this section, we summarize the key concepts of the ROM-based inexact FETI-DP method. For illustration purposes, the main components of the strategy are also depicted in Fig. 5. As the method has already been thoroughly presented in [34], we attempt to be brief in the following. Obviously, readers are encouraged to refer to [34] for a comprehensive description. We place ourselves inside an overlapping subdomain and omit all subscripts and superscripts related to subdomain decompositions, discretizations, etc. Let $\mathbf{K}^{(s)}$ denote the local stiffness matrix of the cell labeled (s), $s = 1, \dots, N_s$, where N_s is here the number of cells in the considered subdomain. The core idea of the ROM technique is to exploit the similarities between cells within the subdomain. To this end, we introduce an interpolatory reduced basis space for the stiffness matrices:

$$\forall \varepsilon > 0,\ \exists \mathscr{K} = \left\{ \mathbf{K}^{(s_1)}, \dots, \mathbf{K}^{(s_{N_B})} \right\} \ \text{ s.t. } \ \forall s = 1, \dots, N_s,\ \exists \mathring{\mathbf{K}}^{(s)} \in \mathscr{K} \text{ s.t. } \left\| \mathbf{K}^{(s)} - \mathring{\mathbf{K}}^{(s)} \right\| < \varepsilon, \tag{11}$$

where $\| \cdot \|$ denotes an appropriate norm. N_B represents the number of local matrices selected for the reduced basis, corresponding to the number of principal cells used to approximate all remaining cells (see again Fig. 3). The ROM-approximated stiffness matrix $\mathring{\mathbf{K}}^{(s)}$ is defined by:

$$\mathring{\mathbf{K}}^{(s)} = \sum_{\nu=1}^{N_B} \alpha_\nu^{(s)} \mathbf{K}^{(s_\nu)},$$

where the coefficients $\alpha_\nu^{(s)} \in \mathbb{R}$ depend on the cell (s). The construction of this reduced basis using a greedy approach is detailed in [34].

Then, the starting point of the inexact FETI-DP method is the following saddle-point problem:

$$\begin{pmatrix} \mathbf{K} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{d} \end{pmatrix}, \quad \text{with} \quad \mathbf{K} = \begin{pmatrix} \mathbf{K}_{RR} & \mathbf{K}_{RP} \\ \mathbf{K}_{RP}^T & \mathbf{K}_{PP} \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} \mathbf{u}_R \\ \mathbf{u}_P \end{pmatrix} \quad \text{and} \quad \mathbf{f} = \begin{pmatrix} \mathbf{f}_R \\ \mathbf{f}_P \end{pmatrix}.$$

The displacement DOFs have been partitioned into primal DOFs (at cell corners, denoted by subscript P), which are assembled globally, and remaining DOFs (denoted by subscript R, which include (i) interior DOFs, and (ii) dual DOFs at cell interfaces, the latter being enforced through Lagrange multipliers). $\lambda \in \mathbb{R}^L$ are these Lagrange multipliers, where L is the total number of discrete continuity equations imposed weakly (encapsulated in $\mathbf{d}$). $\mathbf{B}$ is the coupling matrix, associated to the cell-to-cell interface continuity conditions ($\mathbf{B}$ vanishes for the primal DOFs). A multiplicity scaling is

applied in the algorithm to account for the number of cells sharing each interface DOF. The idea behind inexact FETI-DP methods is to iterate on this saddle-point problem using a suitable block preconditioner. These iterations will be referred to as "global" iterations for the ROM-based inexact FETI-DP algorithm, as they pertain to the external loop of this method (see again Fig. 5). This offers the opportunity to approximate the solutions to the local (cell-wise) stiffness matrices, which is done here using the ROM strategy, unlike in the standard FETI-DP method where local problems must be solved exactly. Additionally, by adopting a matrix-free approach, we avoid assembling all the preconditioner sub-blocks, and instead compute their application to a vector on-the-fly. In practice, only the principal local stiffness matrices are assembled explicitly. To apply this block preconditioner to a vector, it is also necessary to solve an interface problem, which is done iteratively (see "local" iterations in Fig. 5), using a finely-tuned preconditioner that involves, once again, the ROM strategy (see [34] for details). Based on the substructuring technique of the FETI-DP method, this interface problem is defined by condensing displacement DOFs and constructing the global dual Schur complement. This dual Schur complement is determined by solving a coarse problem posed over primal DOFs (involving the global primal Schur complement), which is constructed by eliminating the remaining DOFs. All operators involved in solving this interface problem are constructed from the operators of the principal cells using the ROM strategy. Fig. 5 provides additional details on the overall view of the method.

For three-dimensional elasticity problems, constructing the coarse problem by selecting only DOFs at cell corners may not be sufficient to ensure a small condition number and achieve convergence within a reasonable number of iterations for the interface problem. To address this issue, methods based on adding primal constraints, such as averages over selected edges [26] and first-order moments [46], have been proposed. An adaptive coarse space approach, enriched by a small number of additional local edge eigenvalue problems, has also been suggested for problems with large coefficient jumps within and across cell boundaries [47].

In this work, we adopt the method introduced in [26], which augments the coarse problem by adding only edge averages (and not first-order moments), as we are focusing on homogeneous problems without coefficient jumps across cell boundaries. All cells share the same properties, being constructed from a reference cell. Specifically, we introduce a set of additional Lagrange multipliers for all the edges at the interfaces between cells, denoted as $\boldsymbol{\mu} \in \mathbb{R}^E$, where E is the number of such edges. The primal variables are unchanged by this modification and remain associated with the cell corners. We also introduce a matrix $\mathbf{Q} \in \mathbb{R}^{L\times E}$ corresponding to the three translation rigid body modes of each edge in the x-, y-, and z-directions, see [26] for details. The resulting saddle-point problem for the augmented coarse problem is then formulated as follows:

$$\begin{pmatrix} \mathbf{K} & \mathbf{B}^T & \mathbf{B}^T\mathbf{Q} \\ \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{Q}^T\mathbf{B} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \lambda \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{d} \\ \mathbf{Q}^T\mathbf{d} \end{pmatrix}.$$

By eliminating the remaining primal DOFs, $\mathbf{u}_R$ and $\mathbf{u}_P$, along with the additional Lagrange multipliers $\boldsymbol{\mu}$, an interface problem including the augmented coarse problem is constructed.

3.2.3. Grid-transfer operators

Now that the fine level of the hierarchy is treated in its integrity (matrix-vector products and smoother applications), to achieve proper numerical robustness, special treatments such as coarse space correction are required to handle low-frequency error components. In our two-grid preconditioner, it is handled by the grid-transfer operators and the coarse grid correction introduced here and in next paragraph, respectively.

The prolongation and restriction operators, which transfer quantities between the fine-level and the coarse-level, are based on the spline k-refinement strategy. Given that the approximation spaces are nested, see Remark 3.1, we opt for canonical operators.

A prolongation matrix is defined on the N_p reference patches involved in the reference microstructure, see Eq. (4), using the transformation matrices obtained through degree-elevation and knot-refinement algorithms, see Eq. (3). For each reference patch $k = 1, \dots, N_p$, let $n_H^{(k),\mathrm{ref}}$ and $n_h^{(k),\mathrm{ref}}$ be the number of DOFs of the patch $\Omega_{\mathrm{ref}}^{(k)}$, for the coarse and fine discretizations, respectively. Similarly, let n_H^{ref} and n_h^{ref} be the number of DOFs of the reference coarse and fine tile, respectively. Let

$$(\bar{\mathbf{A}}_{H,i,j}^{(k)})_{\substack{1\le i\le n_H^{\mathrm{ref}} \\ 1\le j\le n_H^{(k),\mathrm{ref}}}}, \quad (\bar{\mathbf{A}}_{h,i,j}^{(k)})_{\substack{1\le i\le n_h^{\mathrm{ref}} \\ 1\le j\le n_h^{(k),\mathrm{ref}}}}, \quad (\tilde{\mathbf{A}}_{H,i,j}^{(s)})_{\substack{1\le i\le n_H \\ 1\le j\le n_H^{\mathrm{ref}}}}, \quad (\tilde{\mathbf{A}}_{h,i,j}^{(s)})_{\substack{1\le i\le n_h \\ 1\le j\le n_h^{\mathrm{ref}}}}$$

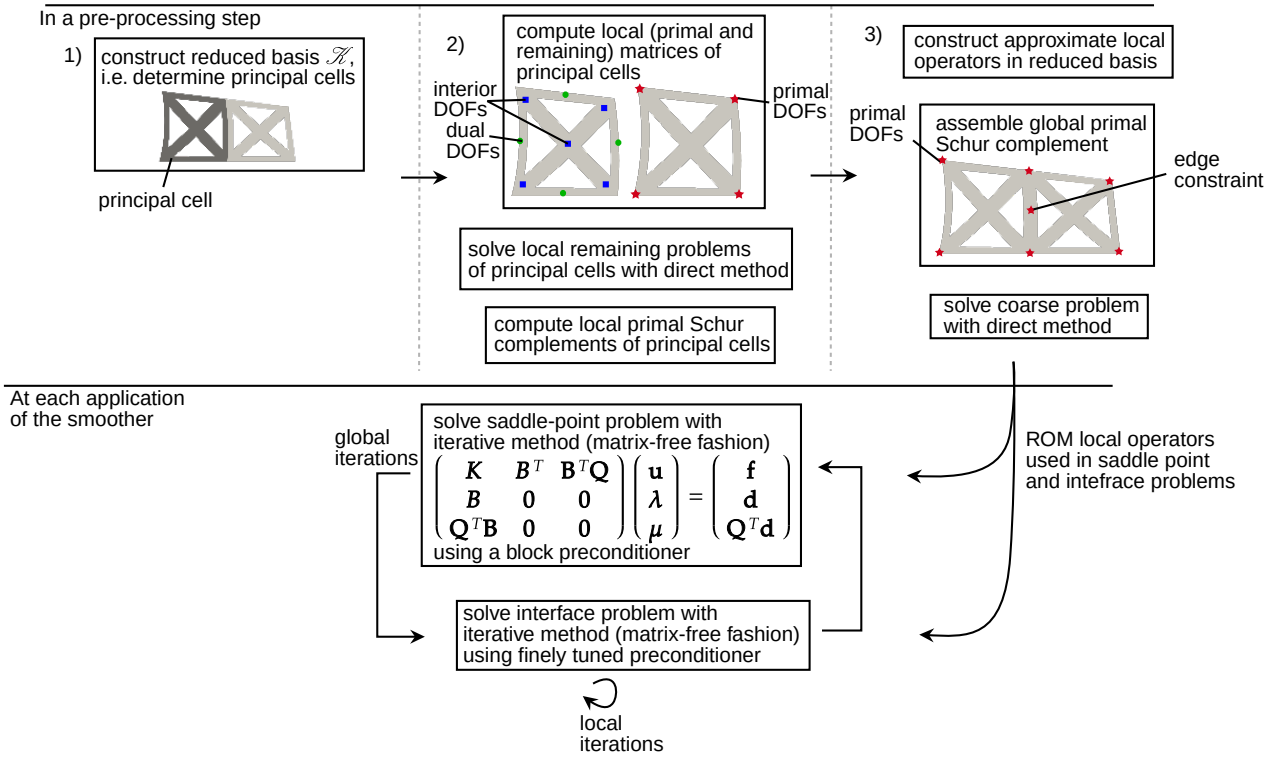


Figure 5: Summary of the inexact FETI-DP method with ROM for an overlapping subdomain.

be the assembly operators from the patch $\Omega_{\text{ref}}^{(k)}$ to the reference tile and from the reference tile to the full structure. These are Boolean matrices constructed with a single one and zeros for each row. Incorporating renormalization, a tentative prolongation operator is then defined by:

$$(\tilde{\mathbf{P}}^{p,h}_{\mathbf{1},H,i,j})_{\substack{1\le i\le n_h\\ 1\le j\le n_H}} = \kappa_i \left(\sum_{s=1}^{N_s} \sum_{k=1}^{N_p} \tilde{\mathbf{A}}_h^{(s)} \bar{\mathbf{A}}_h^{(k)} P_\xi P_\eta P_\zeta \left(\tilde{\mathbf{A}}_H^{(s)} \bar{\mathbf{A}}_H^{(k)} \right)^T \right)_{i,j}, \quad \text{where } \kappa_i \text{ s.t. } \sum_{i=1}^{n_h} \tilde{\mathbf{P}}^{p,h}_{\mathbf{1},H,i,j} = 1.$$

Additionally, boundary conditions are imposed to define the final prolongation operator:

$$(\mathbf{P}^{p,h}_{\mathbf{1},H,i,j})_{\substack{1\le i\le n_h\\ 1\le j\le n_H}} = \begin{cases} (\tilde{\mathbf{P}}^{p,h}_{\mathbf{1},H,i,j})_{i,j}, & \text{if } j \text{ is not a DOF associated to } \partial\Omega_D, \\ \delta_{\tilde{i},j}, & \text{if } j \text{ is a DOF associated to } \partial\Omega_D, \end{cases} \tag{12}$$

where $\tilde{i}$ is the closest DOF to j in the fine discretization and $\delta_{\tilde{i},j}$ represents the Kronecker delta symbol, which equals one if $\tilde{i} = j$ and zero otherwise. The restriction operator is defined as the transpose of the prolongation, i.e.:

$$\mathbf{R}^{\mathbf{1},H}_{p,h} = (\mathbf{P}^{p,h}_{\mathbf{1},H})^T.$$

Note that the boundary conditions prescribed by Eq. (12) ensure that the imposed displacements on the boundary $\partial\Omega_D$ are preserved by the restriction operator, a property naturally verified by the prolongation operator. These grid-transfer operators could be represented by sparse matrices, containing only a few non-zero entries.

However, similarly to the fine stiffness matrix, a matrix-free approach is employed. A reference prolongation operator from patches to the reference tile, defined by:

$$(\tilde{\mathbf{P}}^{p,h}_{\mathbf{1},H,i,j})_{\substack{1\le i\le n_h\\ 1\le j\le n_H}} = \left(\sum_{k=1}^{N_p} \bar{\mathbf{A}}_h^{(k)} P_\xi P_\eta P_\zeta \left(\bar{\mathbf{A}}_H^{(k)} \right)^T \right)_{i,j},$$

is computed during the initialization phase. This operator can be assembled and stored inexpensively in memory because it has the size of the reference tile which is supposed to be small in comparison to the full heterogeneous structure. Eventually, the matrix-vector product is performed by applying assembly operators from a tile to the full structure. Renormalization and boundary conditions are also applied at this stage. Technically speaking, this is implemented using a PETSc MatShell for the prolongation operator, and then calling PCMGSetInterpolation. Since we do not provide an additional restriction operator, PETSc will simply use the transpose of the prolongation. Therefore, we must provide two callbacks for the MatShell, one for its action on a vector, and one for its transposed action on a vector.

3.2.4. Coarse grid correction

Traditionally, multigrid methods involve recursively applying lower-dimensional corrections until reaching a sufficiently coarse level that allows for an inexpensive solution using a direct solver. In our approach, we combine our two-grid method with an algebraic multigrid (AMG) method at the second (coarse) level. Combining geometric multigrid with algebraic multigrid is not a new paradigm in itself, see similar techniques in [12, 73], but to the best of our knowledge, it is the first time we show the applicability of such a methodology applied to IGA. The aim is to leverage the efficiency of AMG methods on our coarse level, as these methods have proven to be particularly effective for solving low-order finite element problems, which aligns with our coarse discretization. The algebraic approach offers several advantages, including adaptability and memory requirements, while maintaining satisfactory convergence performance. By combining this method with our two-grid preconditioner with k-refinement, we aim for a hybrid solution strategy that capitalizes on the strengths of both approaches.

Accurately describing the near-null space of the coarse level operator is crucial for AMG. In linear elasticity, the null-space contains the rigid body motions of the structure. The structure does not undergo strain, but is rather subjected to simple translations and/or rotations. More specifically, in linear elasticity (and with standard notations), an arbitrary kernel element reads as:

$$\begin{array}{rccc} \boldsymbol{u}^{\mathrm{rb}} : & \Omega & \to & \mathbb{R}^d \\ & \mathbf{x} & \mapsto & \boldsymbol{R}(\mathbf{x})\mathbf{a} \end{array},$$

with $\mathbf{a} \in \mathbb{R}^{3(d-1)}$ being the amplitudes of the translations and rotations, and:

$$\boldsymbol{R}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & -x_2 \\ 0 & 1 & x_1 \end{bmatrix}, \text{ if } d = 2, \quad \boldsymbol{R}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & 0 & x_3 & -x_2 \\ 0 & 1 & 0 & -x_3 & 0 & x_1 \\ 0 & 0 & 1 & x_2 & -x_1 & 0 \end{bmatrix}, \text{ else.}$$

Therefore, building a discrete basis that defines the kernel of the stiffness matrix can be formulated as finding $\mathbf{U}_i^{\mathrm{rb}} \in \mathbb{R}^{d \times 3(d-1)}$, $i = 1 \dots n$, such that:

$$\forall \mathbf{x}_h \in \Omega^h, \sum N_i(\mathscr{G}^{-1}(\mathbf{x}_h))\mathbf{U}_i^{\mathrm{rb}} = \boldsymbol{R}(\mathbf{x}_h), \tag{13}$$

where the N_i are some basis functions and $\mathscr{G}$ is a geometric mapping (for instance, a multipatch spline model) that discretizes the domain. In a standard isoparametric formulation, the solution and the geometric mapping share the discretization, which makes the identification of the $\mathbf{U}_i^{\mathrm{rb}}$ rather straightforward:

$$\forall i,\ \mathbf{U}_i^{\mathrm{rb}} = \boldsymbol{R}(\mathbf{x}_i),$$

where $\mathbf{x}_i \in \mathbb{R}^d$ are the nodes (or control points) of the mesh. However, in the case of a non-isoparametric approach (as it is the case in this study), the solution of Eq. (13) is not straightforward. There might be even no solution if the solution space is not able to reproduce the geometric map as it is likely our case. Indeed, here the geometric maps are made of composed splines (thus involved high-order polynomials) whereas the solution spaces are built upon standard spline spaces (see Section 2.3). However, as we are looking for a near-null space of the operator (and not the exact null space), we can allow ourselves to compute an approximate solution of Eq. (13). The approach followed in this work consists in selecting as many physical points as basis functions, denoted $\mathbf{x}_i^*$, which then enable to set:

$$\forall i,\ \mathbf{U}_i^{\mathrm{rb}} = \boldsymbol{R}(\mathbf{x}_i^*). \tag{14}$$

An appropriate choice for spline-based discretizations relies on Greville abscissae [64], i.e. we define the $\mathbf{x}_i^*$ as:

$$\forall i,\ \mathbf{x}_i^* = (\mathcal{B}\circ\mathcal{S})(\boldsymbol{\theta}_i^*),$$

where the $\boldsymbol{\theta}_i^*$ are the Greville abscissae associated to the reference cell (see Definition 2.1). Therefore, the procedure for building the basis of the (near) null space operator is done by firstly mapping the Greville abscissae through the geometric maps (as they are initially defined however the micro parameter spaces $\bar{\Omega}_m$), and then using Eq. (14) for each physical point.

3.3. Discussions

In this work, we propose a preconditioner for linear systems arising from IGA discretizations based on multilevel matrix-free methods. From a multigrid perspective, on the fine level, our method is designed with a one-level DD preconditioner/smoother. On the second (coarse) level, we make full use of an efficient AMG preconditioner, which is particularly well-suited for explicitly assembled low-order discretizations such as our coarse discretization.

We now elucidate the rationale behind our solver design choices over alternative options. First, we note that for multigrid methods applied to IGA discretizations, standard smoothers such as Jacobi and Gauss–Seidel iterations are computationally inexpensive but lack robustness with respect to spline degree, necessitating the use of non-standard smoothers. Furthermore, to address large-scale problems with moderate-to-high spline degrees, we employ a matrix-free formulation for the fine discretization stiffness matrix, requiring our smoother to be compatible with this formulation.

Focusing on DD approaches and the current state-of-the-art, the first most natural choice might be to employ the ROM-based FETI-DP method across the entire domain, as in [34]. However, for cases involving macro-mappings with significant deformation, applying the ROM strategy to the entire domain proves inefficient, leading to memory limitations. Therefore, we chose to restrict this strategy to each overlapping subdomain, where the macro-mapping deformation is expected to be limited. In other words, the cells are anticipated to be closer in terms of stiffness within each subdomain compared to across the entire domain. Moreover, this effect becomes more pronounced as the number of tiles and subdomains increases. The second, and perhaps most important reason for this choice is that this workload is also embarrassingly parallel and can thus be computed concurrently by each MPI process, which was not the case with the previous contribution [34].

Another alternative could have been to design a two-level FETI-DP preconditioner, making use of a global (standard) FETI-DP instead of RAS, and applying the proposed ROM-based inexact FETI-DP to solve the Neumann problems within each subdomain. A coarse problem would then be constructed to ensure continuity between subdomains. However, with this approach, selecting the primal DOFs for the coarse problem is not straightforward, making it difficult to achieve both good scalability and efficiency. Another reason for favoring overlapping Schwarz methods over FETI-DP on the fine level is the flexibility of PCASM over PCBDDC when using PCShell as subdomain solvers.

4. Numerical results

In this section we explore the performance of our solver on four numerical examples. All of them share the same material parameters, specifically an isotropic elastic material with a Young's modulus of $E = 5000\,\text{MPa}$ and Poisson's ratio $\nu = 0.40$. For each application, the computational domain is built from a reference microscopic (reference unit-cell) and a macroscopic geometry, according to the spline composition technique presented in Section 2.2. The macroscopic geometry is refined to obtain the desired number of cells, and the reference tile is also refined for accurate mechanical analysis. The various test cases considered in this paper, along with their corresponding reference unit-cell and macroscopic geometries, are as follows.

(i) **2D auxetic rectangular domain.** This test involves a two-dimensional planar domain under bending. The macro-geometry is initially described with a single linear B-spline patch. The micro-geometry is an auxetic lattice pattern, modeled with a total of 20 matching patches, consisting of 16 quadratic NURBS patches and 4 linear patches (see Fig. 6). For more information on the cell geometry, refer to [34].

(ii) **2D brake pedal.** This case simulates a two-dimensional brake pedal under operating conditions. The macro-geometry is described with a curved quadratic NURBS patch. The micro-geometry is a hollow square linked by a cross, constructed with 24 linear Bézier patches to create an analysis-suitable model, i.e., matching interfaces

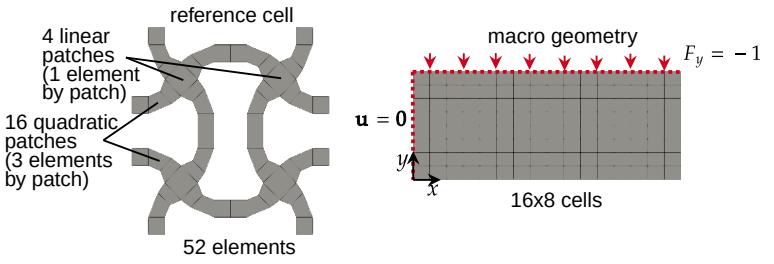


Figure 6: 2D auxetic rectangular domain test case: reference unit-cell and example of refined macro-geometry to multiply the number of cells. The lattice structure is clamped on the left side and subjected to a constant vertical load distributed on the top side.

between the patches and non-trimmed patches (see Fig. 7). For more information on the macro-geometry, the interested reader is advised to consult [34].

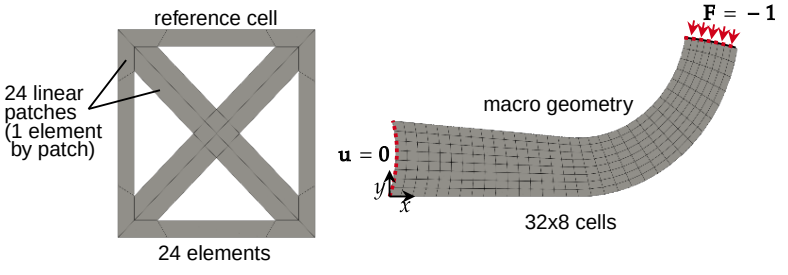


Figure 7: 2D brake pedal test case: reference unit-cell and example of refined macro-geometry to multiply the number of cells. The lattice structure is clamped on the left side and subjected to a constant normal load distributed on the right side.

(iii) **3D straight beam.** This test involves a three-dimensional straight beam under bending. The macro-geometry is initially modeled with a linear B-spline patch containing one single element. A reference lattice unit-cell, modeled as a body-centered cubic (BCC) cell using 32 linear B-spline patches, is associated with this macro-geometry (see Fig. 8).

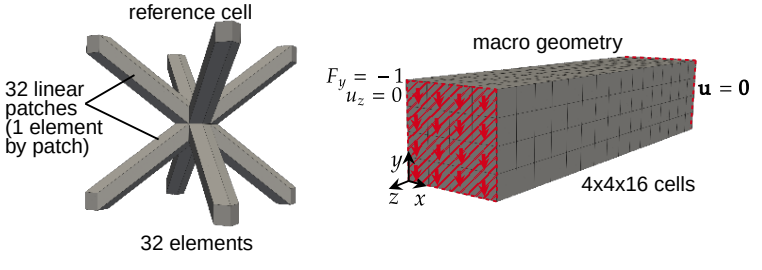


Figure 8: 3D straight beam test case: reference cell and example of refined macro-geometry to multiply the number of cells. The lattice structure is clamped on the right side and subjected to a constant vertical load distributed on the left side. The left surface is also prescribed to remain in the same plane during deformation.

(iv) **3D thrust chamber.** Inspired by [50, 59, 67], this test simulates a three-dimensional industrially representative spiral channel regenerative cooling thrust chamber subjected to a uniform pressure. The macro-geometry is modeled with two matching NURBS patches, each comprising 1230 elements. The associated reference unit-cell is modeled as a straight channel cell using 40 linear B-spline patches (see Fig. 9).

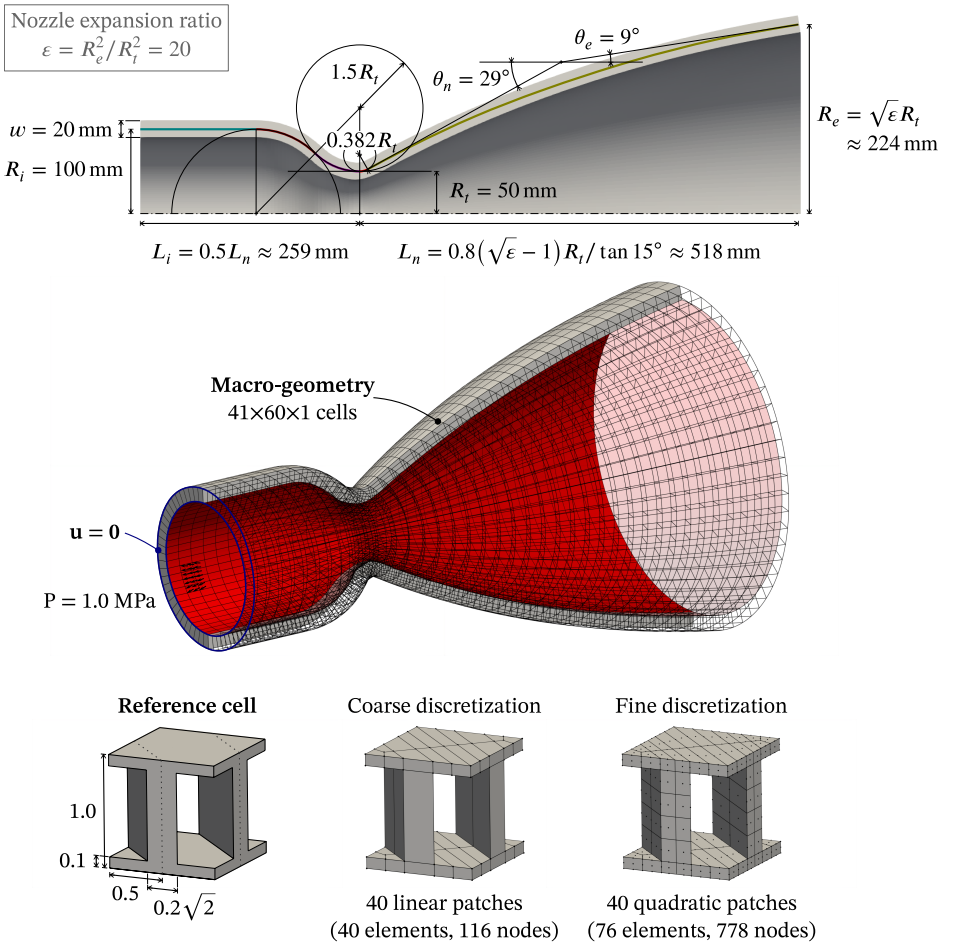


Figure 9: 3D thrust chamber test case: reference unit-cell and example of macro-geometry to multiply the number of cells.

All numerical examples are computed on the Skylake partition of the Joliot-Curie supercomputer[1], consisting of 1656 nodes with two Intel Skylake 8168 processors clocked at 2.7 GHz, each with 24 cores, and 192 GB of DDR4 memory per node. The code is implemented in Python using petsc4py as the linear algebra backend [4], and the libraries YETI [23] and IGAlattice[2] for geometry and IGA discretization tools.

Our approach involves four different interleaved iterative solvers, which are summarized in Table 2. The default parameters for the different solvers, unless otherwise specified, are chosen as follows.

(i) **Global linear system [outer].** The flexible generalized minimal residual method (FGMRES [68]) is used with a restart parameter of 50 and a tolerance of 10^{-5}. Right preconditioning is applied using our multilevel method, embedded within the PCMG machinery of PETSc.

(ii) **Fine-level preconditioner/smoother.** The Jacobi-preconditioned Chebyshev method is used as the pre-smoother. As the post-smoother, a single application of RAS (PCASM preconditioner from PETSc) is considered. For both cases, only one smoothing step is considered. Subdomain-local solves needed by RAS are computed using the ROM-based inexact FETI-DP method, for which two linear systems are solved recursively with iterative methods. The overlapping subdomains are constructed by adding a single layer of cells to the non-overlapping subdomains. Numerical investigations have shown that using no additional cell layer (i.e., including

[1] https://www-hpc.cea.fr/en/Joliot-Curie.html

[2] IGAlattice is a Python code dedicated to the isogeometric analysis of lattice structures. This code has been developed by the Chair of Numerical Modelling and Simulation of EPFL through the ADAM2 project co-funded by the H2020 Horizon programme of the European Union.

Table 2
Name of the different iterative solvers used in the proposed method.

name	description	iterative method	tolerance
outer	global problem	FGMRES	10^{-5}
saddle	saddle-point problem in inexact FETI-DP	GMRES	10^{-4}
interface	interface problem in inexact FETI-DP	PCG	10^{-9}
coarse	AMG coarse correction	GMRES	10^{-2}

only interface DOFs in the overlap) is insufficient to achieve fast convergence, while adding more than a single layer of cells results in a higher computational cost without significant improvement in convergence rate.

(a) **Saddle-point problem [saddle].** The saddle-point problem is solved using a (user-defined) GMRES method preconditioned with our ROM-based block preconditioner. The tolerance for the saddle-point problem is 10^{-4} and the ROM construction tolerance, from Eq. (11), is 10^{-5}, as prescribed in [34] for optimal efficiency.

(b) **Interface problem [interface].** For constructing this block preconditioner, the interface problem is solved with a preconditioned conjugate gradient (PCG) method (from scipy.sparse.linalg). The tolerance for the interface problem is 10^{-9}.

(iii) **Coarse grid correction [coarse].** The coarse grid correction is referred to as the coarse iterative problem. The GMRES method [69] preconditioned with an algebraic multigrid method (PCGAMG [1] preconditioner from PETSc) is used. The tolerance is set at 10^{-2}. The smoother at all levels for GAMG is SOR, and for building the grid hierarchy, we use a coarsening threshold of 0.1, threshold scale set at 0.75, and aggressive coarsening set at 2.

4.1. Numerical investigation of the developed preconditioner

In this first section, we investigate the numerical properties of our solver, such as its robustness with respect to discretization parameters and the efficiency of the ROM-based strategy, using the two-dimensional test cases: the 2D brake pedal (see Fig. 7) and the 2D auxetic rectangular domain (see Fig. 6).

4.1.1. Robustness with respect to spline degree, mesh refinement, and number of tiles

The robustness of the solver concerning different parameters: the spline degree p, the mesh refinement h, and the number of tiles N_s is investigated. The number of outer iterations is shown in Table 3, with respect to the spline degree p and mesh refinement h, for the brake pedal test case in a configuration with 40 tiles in the x-direction and 10 tiles in the y-direction, totaling 400 tiles, using 32 MPI processes. The configurations of p and h_i correspond to k-refinement through degree elevation until degree p is reached, combined with mesh refinement via i successive knot insertions (in other words, the element size is divided by 2^i in each direction). For comparison purpose, the Jacobi method has also been used as a post-smoother in contrast to a single application of RAS. A † in Table 3 means that the outer FGMRES solver is not able to reach the prescribed tolerance in less than 3000 iterations. This notation will be used consistently in the tables presented in this paper. Results from the structural analyses (magnitude of the displacement field and von Mises stress field) are provided in Fig. 10 for this test case.

As expected, the standard Jacobi smoother at the fine level lacks robustness with increasing spline degree p, as the number of iterations rises significantly with higher degrees. It is also not robust with mesh refinement h (knot-insertion for splines). Conversely, the RAS smoother demonstrates robustness with both spline degree p and mesh refinement h as the number of outer iterations remains approximately constant across all scenarios.

Furthermore, the number of iterations for the outer, saddle, interface, and coarse iterative problems is provided in Table 4 and Table 5 for the brake pedal and auxetic rectangular test cases, respectively, as the number of tiles increases. The number of MPI processes is set to 32 and the discretization parameters are $(p, h) = (2, h_1)$ for the brake pedal

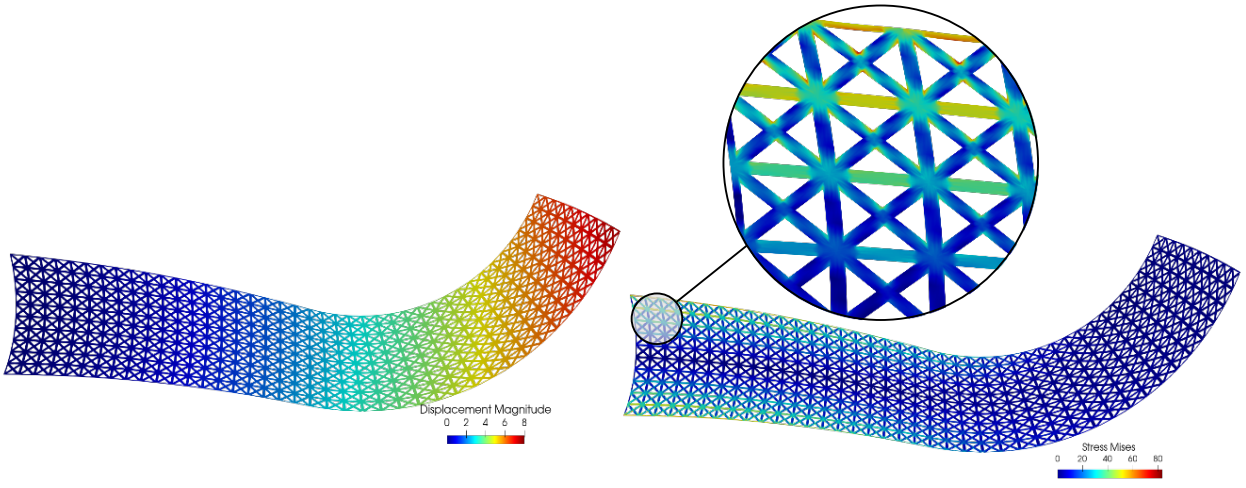


Figure 10: Structural analysis results for the 2D brake pedal with 400 tiles, $p = 3$, $h = h_1$.

Table 3
Number of outer iterations for different configurations of k-refinement and fine-level preconditioner. 2D brake pedal with 400 tiles and 32 MPI processes. The term "Jacobi" refers to using the Jacobi method as both a pre- and a post-smoother, while "RAS" denotes our approach, which involves a single application of RAS as a post-smoother.

	# iterations							
	$p = 1$		$p = 2$		$p = 4$		$p = 6$	
	Jacobi	RAS	Jacobi	RAS	Jacobi	RAS	Jacobi	RAS
h_1	84	9	184	10	828	11	1224	12
h_2	260	10	591	11	1303	12	1744	12
h_3	969	10	1561	11	2009	12	†	13

Table 4
Number of (outer, saddle, interface, coarse) iterations for the developed multilevel preconditioner with $p = 2$, $h = h_1$, and different number of tiles. 2D brake pedal test case with 32 MPI processes.

# tiles	# DOFs		# iterations			
	fine	coarse	outer	saddle	interface	coarse
32x8 (256)	0.1M	0.01M	10	2	23	8
128x32 (4,096)	1.8M	0.2M	10	2	24	10
512x128 (65,536)	30M	3.4M	11	2	24	11

case and $(p, h) = (3, h_2)$ for the auxetic case. For both test cases, the number of iterations of the four iterative solvers remains constant as the problem size increases, which demonstrates that the multilevel preconditioner with the RAS smoother is robust to the size of the problem.

4.1.2. Impact of the coarse grid correction on the global solver

The multilevel preconditioner introduced in this paper consists of a fine-level smoother/preconditioner and a coarse correction, specifically an AMG solver. In this section, we examine the impact of the coarse correction on the convergence of the global solver. For this purpose, we compare our multilevel preconditioner with a strategy that consists of only performing the fine-level preconditioner, i.e., only performing the smoother (using similarly RAS and ROM-based inexact FETI-DP local solves, so that only the coarse correction is removed). For this study, we consider the 2D brake pedal test case with 400 tiles and $(p, h) = (2, h_1)$. Table 6 presents the number of outer iterations for both solvers.

Table 5
Number of (outer, saddle, interface, coarse) iterations for the developed multilevel preconditioner with $p = 3$, $h = h_2$, and different number of tiles. 2D auxetic rectangular domain test case with 32 MPI processes.

# tiles	# DOFs		# iterations			
	fine	coarse	outer	saddle	interface	coarse
16×8 (128)	0.55M	0.049M	9	1	22	9
64×32 (2,048)	8.8M	0.79M	9	1	22	9
128×64 (65,536)	35.3M	3.1M	10	1	23	10

Table 6
Number of outer iterations for different solvers: one-level RAS, i.e., using only the proposed smoother, and our multilevel preconditioner with an AMG coarse correction, with respect to the number of MPI processes. 2D brake pedal with 400 tiles, $(p, h) = (2, h_1)$.

#MPI processes	# iterations	
	one-level RAS	multilevel
2	19	8
4	400	9
8	†	9
32	†	10
64	†	11

The corresponding results reveal that the iteration count increases significantly for the one-level DD preconditioner, leading to solver failure beyond four subdomains (one subdomain per MPI process). In contrast, the iteration count remains constant for the proposed multilevel preconditioner, highlighting the crucial role of the coarse grid correction in transmitting low-frequency solution components between subdomains.

4.1.3. Investigation of the ROM-based strategy

In this section, we examine the robustness of the ROM-based strategy with an increasing number of tiles and MPI processes. We consider the auxetic rectangular domain test case with $(p, h) = (3, h_2)$ and the brake pedal test case with $(p, h) = (2, h_1)$. The ROM-based strategy is expected to be more efficient when tiles are more similar to each other, making it particularly effective for the rectangular test case where macro-mapping deformation is minimal, compared to the brake pedal test case. Tables 7 and 8 show, for each test case, the ratio of the number of principal tiles (used to build the local operators) to the number of tiles in the subdomain, across various configurations of different number of tiles and MPI processes, with a ROM tolerance of 10^{-5}.

As expected, for the rectangular test case, a single cell per subdomain is sufficient to accurately represent the local operators because all tiles are the same. For the brake pedal test case, we observe that multiplying the number of tiles in the subdomains requires storing almost no additional local operators. Thus, the ROM strategy becomes all the more efficient as the problem size grows. We also see that keeping the ratio of number of tiles to MPI processes constant while increasing the problem size slightly enhances ROM efficiency. For example, the maximum ratio decreases from 80% to 66% between 256 tiles for 32 MPI processes and 4096 tiles for 128 MPI processes. This feature can be explained by the fact that increasing the problem size while maintaining this ratio reduces the subdomain size in the macro-model, and thus their macro-mapping deformation. Therefore, the proposed algorithm constitutes an efficient HPC extension of the ROM-based strategy initially introduced in [34] without parallel implementation.

In conclusion, to maximize the benefits of the ROM strategy, it is advantageous to consider a large number of subdomains with a large number of tiles, in order to make the tiles similar.

4.2. Performance of the solver on more industrial realistic examples

In a next step, we investigate the performance of the solver on the two three-dimensional test cases: the 3D straight beam (see Fig. 8) and 3D thrust chamber (see Fig. 9) test cases.

4.2.1. Straight beam

First, a weak scaling study of the developed multilevel preconditioner is conducted using the straight beam test case. To this end, a series of configurations with an increasing number of cells (thus a similar increasing number of DOFs)

Table 7
Minimum and maximum ratio of the number of principal tiles per subdomain obtained with a ROM tolerance of 10^{-5}. 2D auxetic rectangular domain test case, $p = 3$, $h = h_2$. As expected, a single tile per subdomain is sufficient since all the tiles are the same.

# processes \ # tiles	256		4,096	
	min	max	min	max
32	1/36 (2.7%)	1/25 (4%)	1/100 (1%)	1/81 (1.2%)

Table 8
Minimum and maximum ratio of the number of principal tiles per subdomain obtained with a ROM tolerance of 10^{-5}. 2D brake pedal test case, $p = 2$, $h = h_1$.

# processes \ # tiles	256		4,096		16,384		65,536	
	min	max	min	max	min	max	min	max
32	6/24 (37.5%)	12/15 (80%)	8/180 (4.4%)	12/162 (7.4%)	8/612 (1.3%)	13/578 (2.2%)	-	-
128	-	-	8/24 (33%)	12/18 (66%)	7/180 (3.8%)	11/180 (6.1%)	7/612 (1.1%)	10/612 (1.6%)

and processes (i.e., subdomains), while maintaining approximately a constant ratio between them, is considered. The results of structural analysis for this test case are provided on figure Fig. 11. Table 9 presents the number of iterations and the computational time for both the initialization phase and solver across configurations with increasing numbers of tiles and MPI processes, and where $(p, h) = (2, h_1)$. The left panel of Fig. 12 shows the computational time of the initialization and solving steps as a function of the number of processes. It also shows the number of DOFs per process as a function of the total number of processes. This ratio of number of tiles per MPI process ranges from between 52 to 64 across all configurations. For all the scenarios considered, we observe that the number of iterations (for the outer, saddle, interface, and coarse iterative solvers) does not depend on the size of the problem. In addition, the computational time for solving the linear system is of the same order of magnitude for all configurations, ensuring near-ideal weak-scaling. Furthermore, from a broader HPC perspective, it seems that these runs go far beyond the state-of-the-art. To the best of our knowledge, (i) such large lattice structures have never been simulated using a fine-scale high-fidelity approach, and (ii) the simulations represent some of the largest IGA runs ever conducted in a distributed-memory context.

A strong scaling study is also performed on a configuration with $10 \times 10 \times 40$ tiles, totaling 4000 tiles, with $p = 2$ and $h = h_1$. The number of MPI processes ranges from 8 to 2048, corresponding to an average number of tiles per MPI process ranging from 500 to 2. We observe that the results deviate from the ideal scaling rate as the number of processes increases. This behavior is due to the reduced efficiency of the ROM strategy (at the level of the overlapping subdomains) when the number of tiles within each MPI process decreases.

4.2.2. Thrust chamber

As an additional step towards the transfer of the proposed method to an industrial environment, the complex spiral channel regenerative cooling thrust chamber of Fig. 9 is computed here. More precisely, we consider three configurations: (i) a configuration with 41x60x1 cells, resulting in 2,460 total cells, (ii) a configuration with 82x120x2 cells, resulting in 19,680 total cells, and (iii) a configuration with 123x180x3 cells, resulting in 66,420 total cells. A representation of the decomposition of the computational domain into subdomains is provided in Fig. 13 (a) for the second configuration, as well as the structural analysis results in Fig. 13 (b). The number of iterations and the computational time for the setup and solution phases are provided in Table 10 for the three configurations. The tolerance for the interface problem is set here to 10^{-4}, as it has been found to be sufficient to make the saddle-point problem converge on this test case. We observe that across all configurations, the number of iterations remains within the same order of magnitude. The computational time is primarily dominated by the initialization and the solution of the subdomain-local FETI-DP problems. Although the first and second configurations have the same number of tiles per subdomain, an increase in computational time is observed, which can be attributed to the higher number of iterations in the outer and interface problems. Overall, these last results suggest the effectiveness of the developed approach in addressing complex industrial problems, such as those in the aerospace domain.

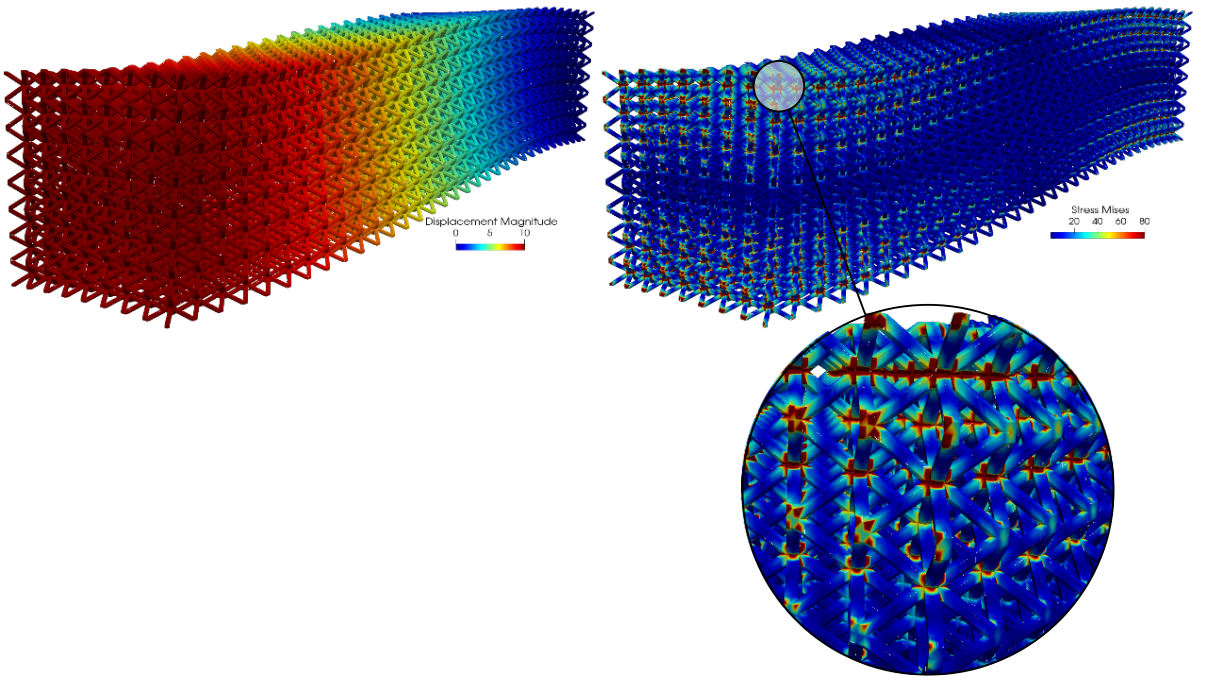


Figure 11: Results of the structural analyses for the 3D straight beam with 2,048 tiles, $p = 2$, $h = h_1$.

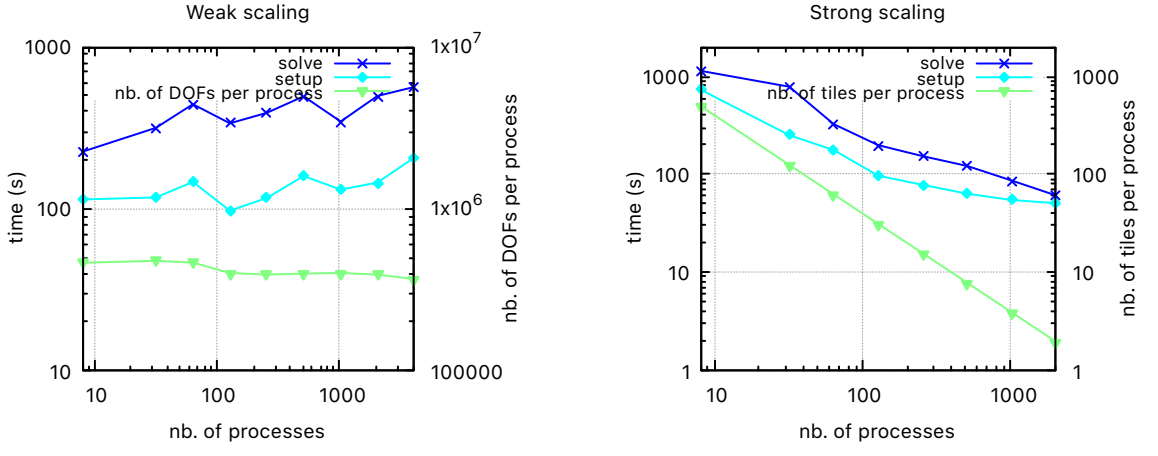


Figure 12: Weak and strong scaling (on a configuration with $10 \times 10 \times 40$ tiles) on the 3D straight beam test case for the proposed multilevel preconditioner with $p = 2$, $h = h_1$.

5. Conclusion

In this work, we introduced a high-performance solver dedicated to the isogeometric analysis of lattice structures. The solver is designed to fully leverage the computational power of distributed-memory architectures, enabling full fine-scale simulations of lattice structures for problems that were previously computationally intractable. It utilizes a two-level preconditioner, consisting of a fine-level preconditioner (or smoother) and a coarse-level correction. The fine-level preconditioner is based on an overlapping domain decomposition method, where the restricted additive Schwarz method is used. The computational domain is divided into overlapping subdomains, each containing several cells. This fine-level correction ensures robustness with respect to mesh refinement (h), spline degree (p), and problem size,

Table 9
Weak scaling study on the 3D straight beam test case for the proposed multilevel preconditioner with $p = 2$, $h = h_1$.

# tiles	# processes	# DOFs		# iterations				time (s)			
								solve			setup
		fine	coarse	outer	saddle	interface	coarse	apply smoother	fine matrix multiplication	total	total
5x5x20 (500)	8	3.7M	0.7M	15	1	45	23	92	84	224	114
8x8x32 (2,048)	32	15.2M	2.7M	16	1	45	19	181	66	316	117
10x10x40 (4,000)	64	29.7M	5.3M	16	1	43	17	246	125	441	146
12x12x48 (6,912)	128	51.2M	9.1M	16	1	44	19	188	95	337	97
15x15x60 (13,500)	256	100M	17.8M	16	1	44	19	219	104	390	116
19x19x76 (27,436)	512	203M	36M	16	1	43	23	260	137	494	159
24x24x96 (55,296)	1,024	409.4M	72.8M	16	1	42	21	214	64	345	131
30x30x120 (108,000)	2,048	800M	142.1M	17	1	43	25	261	114	496	145
37x37x148 (202,612)	4,096	1.5B	266.5M	16	1	42	26	285	141	565	207

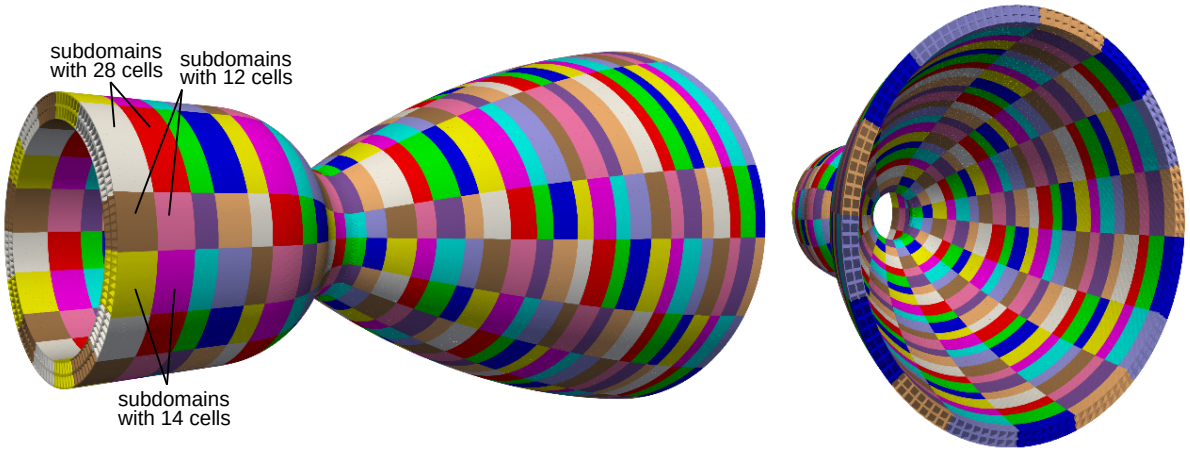
Table 10
Results for three configurations of the 3D thrust chamber with the proposed multilevel preconditioner, $p = 2$, $h = h_1$. The tolerance of the interface solver is 10^{-4}.

# tiles	# processes	# DOFs		# iterations				time (s)			
								solve			setup
		fine	coarse	outer	saddle	interface	coarse	apply smoother	fine matrix multiplication	total	total
41x60x1 (2,460)	64	4.5M	0.5M	11	2	17	20	376	12	394	464
82x120x2 (19,680)	512	33.6M	3.7M	14	2	31	23	925	15	961	498
123x180x3 (66,420)	4,096	111M	11.9M	16	2	33	25	664	9	684	512

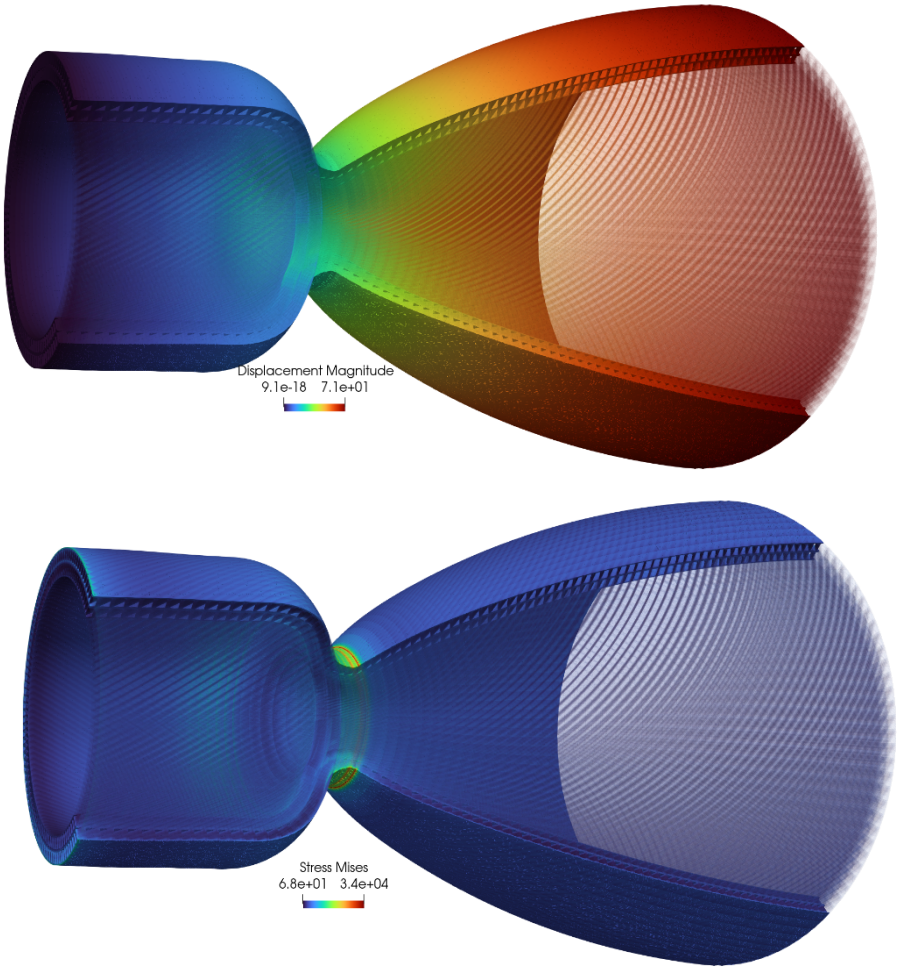
while remaining compatible with an implementation using a distributed-memory paradigm. The coarse-level correction utilizes an algebraic multigrid method.

A key feature of the solver is its use of a ROM-based approach, as introduced in [34], to solve subdomain-local problems efficiently. This is made possible by the similarity between the cells within each subdomain. Additionally, by employing a matrix-free formulation, the fine-level operators are never assembled explicitly. Instead, only their action on local portions of distributed vectors is computed. This is efficiently handled using the fast assembly procedure from [33], now embedded in the Portable, Extensible Toolkit for Scientific Computation (PETSc).

The performance and properties of the solver were evaluated through a series of numerical experiments across various two- and three-dimensional micro- and macro-geometries. When compared to standard smoothers (e.g., Jacobi, SOR), our dedicated fine-level smoother significantly reduced the number of iterations, especially for high spline-degrees (p). The solver demonstrated robustness with respect to mesh refinement, spline degree, and problem size. Furthermore, the matrix-free formulation and ROM-based strategy within the fine-level preconditioner drastically reduced the memory footprint, allowing us to perform a 3D simulation with over one billion DOFs in minutes using thousands of processes.



(a) Decomposition of the computational domain of the 3D thrust chamber into 512 subdomains.



(b) Structural analysis results for the 3D thrust chamber.

Figure 13: Result for the 3D thrust chamber: test case of 82x120x2 = 19,680 cells, 512 subdomains, and $p = 2$, $h = h_1$.

As future works, we intend to extend the solver to non-linear regimes and introduce manufacturing-induced defects into the simulation.

Acknowledgments

This research was supported by the French "Agence Nationale de la Recherche" under grant ANR-22-CE46-0007 (AVATAR). ANR is gratefully acknowledged. For the purpose of Open Access, a [CC-BY public copyright license] has been applied by the authors to the present document and will be applied to all subsequent versions up to the Author Accepted Manuscript arising from this submission. This work was granted access to the HPC resources of TGCC@CEA under the allocation AD010607519R2 made by GENCI.

References

[1] M. Adams, H. H. Bayraktar, T. M. Keaveny, and P. Papadopoulos. Ultrascalable implicit finite element analyses in solid mechanics with over a half a billion degrees of freedom. In *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, SC04, pages 34:1–34:15. IEEE Computer Society, 2004.

[2] P. Antolin, A. Buffa, F. Calabro, M. Martinelli, and G. Sangalli. Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization. *Computer Methods in Applied Mechanics and Engineering*, 285:817–828, 2015.

[3] P. Antolin, A. Buffa, E. Cohen, J. F. Dannenhoffer, G. Elber, S. Elgeti, R. Haimes, and R. Riesenfeld. Optimizing micro-tiles in micro-structures as a design paradigm. *Computer-Aided Design*, 115:23–33, 2019.

[4] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, and J. Zhang. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.23, Argonne National Laboratory, 2025.

[5] L. Beirao da Veiga, D. Cho, L. F. Pavarino, and S. Scacchi. Overlapping Schwarz methods for isogeometric analysis. *SIAM Journal on Numerical Analysis*, 50(3):1394–1416, Jan. 2012. ISSN 1095-7170. doi: 10.1137/110833476. URL http://dx.doi.org/10.1137/110833476.

[6] L. Beirao da Veiga, D. Cho, L. F. Pavarino, and S. Scacchi. BDDC preconditioners for isogeometric analysis. *Mathematical Models and Methods in Applied Sciences*, 23(06):1099–1142, Mar. 2013. ISSN 1793-6314. doi: 10.1142/s0218202513500048. URL http://dx.doi.org/10.1142/S0218202513500048.

[7] M. A. Benaimeche, J. Yvonnet, B. Bary, and Q.-C. He. A k-means clustering machine learning-based multiscale method for anelastic heterogeneous structures with internal variables. *International Journal for Numerical Methods in Engineering*, 123(9):2012–2041, 2022.

[8] M. Benedetti, A. Du Plessis, R. Ritchie, M. Dallago, N. Razavi, and F. Berto. Architected cellular materials: A review on their mechanical properties towards fatigue-tolerant design and fabrication. *Materials Science and Engineering: R: Reports*, 144:100606, 2021.

[9] M. Bosy, M. Montardini, G. Sangalli, and M. Tani. A domain decomposition method for isogeometric multi-patch problems with inexact local solvers. 80(11):2604–2621, 2020. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2020.08.024. URL https://www.sciencedirect.com/science/article/pii/S0898122120303321. High-Order Finite Element and Isogeometric Methods 2019.

[10] R. Bouclier and T. Hirschler. *IGA: Non-conforming Coupling and Shape Optimization of Complex Multipatch Structures, Volume 1*. John Wiley & Sons, 2022.

[11] A. Bressan and S. Takacs. Sum factorization techniques in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 352:437–460, 2019.

[12] J. Brown, V. Barra, N. Beams, L. Ghaffari, M. Knepley, W. Moses, R. Shakeri, K. Stengel, J. L. Thompson, and J. Zhang. Performance portable solid mechanics via matrix-free *p*-multigrid. *arXiv preprint arXiv:2204.01722*, 2022.

[13] T. Cadart, T. Hirschler, S. Bahi, S. Roth, F. Demoly, and N. Lebaal. An optimal penalty method for the joint stiffening in beam models of additively manufactured lattice structures. *International Journal of Solids and Structures*, page 113107, Oct. 2024. ISSN 00207683. doi: 10.1016/j.ijsolstr.2024.113107.

[14] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 1999.

[15] F. Calabro, G. Sangalli, and M. Tani. Fast formation of isogeometric Galerkin matrices by weighted quadrature. *Computer Methods in Applied Mechanics and Engineering*, 316:606–622, 2017.

[16] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, and V. Calo. The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers. *Computer Methods in Applied Mechanics and Engineering*, 213-216:353–361, 2012. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2011.11.002. URL https://www.sciencedirect.com/science/article/pii/S0045782511003392.

[17] N. Collier, L. Dalcin, D. Pardo, and V. M. Calo. The cost of continuity: Performance of iterative solvers on isogeometric finite elements. *SIAM Journal on Scientific Computing*, 35(2):A767–A784, Jan. 2013. ISSN 1095-7197. doi: 10.1137/120881038. URL http://dx.doi.org/10.1137/120881038.

[18] J. A. Cottrell, T. J. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[19] Á. P. de la Riva, C. Rodrigo, and F. J. Gaspar. An efficient multigrid solver for isogeometric analysis. 06 2018. URL https://arxiv.org/pdf/1806.05848.pdf.

[20] F. de Prenter, C. V. Verhoosel, E. H. van Brummelen, J. A. Evans, C. Messe, J. Benzaken, and K. Maute. Multigrid solvers for immersed finite element methods and immersed isogeometric analysis. *Computational Mechanics*, 65(3):807–838, Mar 2020. ISSN 1432-0924. doi: 10.1007/s00466-019-01796-y. URL https://link.springer.com/content/pdf/10.1007/s00466-019-01796-y.pdf.

[21] V. Dolean, P. Jolivet, and F. Nataf. *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*. Society for Industrial and Applied Mathematics, Nov. 2015. ISBN 9781611974065. doi: 10.1137/1.9781611974065. URL http://dx.doi.org/10.1137/1.9781611974065.

[22] M. Donatelli, C. Garoni, C. Manni, S. Serra-Capizzano, and H. Speleers. Robust and optimal multi-iterative techniques for IGA Galerkin linear systems. *Computer Methods in Applied Mechanics and Engineering*, 284:230–264, 2015. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2014.06.001. URL https://www.sciencedirect.com/science/article/pii/S0045782514001844. Isogeometric Analysis Special Issue.

[23] A. Duval, T. Hirschler, J. E. C. Fuentes, M. Guerder, and T. Elguedj. YETI: YET another IGA code. 2023.

[24] G. Elber. *Precise Construction of Micro-structures and Porous Geometry via Functional Composition*, pages 108–125. Springer International Publishing, 2017. ISBN 9783319678856. doi: 10.1007/978-3-319-67885-6_6. URL http://dx.doi.org/10.1007/978-3-319-67885-6_6.

[25] J. A. Evans, Y. Bazilevs, I. Babuška, and T. J. Hughes. n-widths, sup–infs, and optimality ratios for the k-version of the isogeometric finite element method. *Computer Methods in Applied Mechanics and Engineering*, 198(21-26):1726–1741, 2009.

[26] C. Farhat, M. Lesoinne, and K. Pierson. A scalable dual-primal domain decomposition method. *Numerical linear algebra with applications*, 7:687–714, 10 2000. doi: 10.1002/1099-1506(200010/12)7:7/83.0.CO;2-S.

[27] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *Scientific communications*, 1961.

[28] K. Gahalaut and S. Tomar. Condition number estimates for matrices arising in the isogeometric discretizations. Technical report, RICAM-Report, 2012-2013.

[29] K. Gahalaut, J. Kraus, and S. Tomar. Multigrid methods for isogeometric discretization. *Computer Methods in Applied Mechanics and Engineering*, 253:413–425, 2013. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2012.08.015. URL https://www.sciencedirect.com/science/article/pii/S0045782512002678.

[30] R. N. Glaesener, S. Kumar, C. Lestringant, T. Butruille, C. M. Portela, and D. M. Kochmann. Predicting the influence of geometric imperfections on the mechanical response of 2D and 3D periodic trusses. *Acta materialia*, 254:118918, 2023.

[31] R. R. Hiemstra, G. Sangalli, M. Tani, F. Calabrò, and T. J. Hughes. Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity. *Computer Methods in Applied Mechanics and Engineering*, 355:234–260, 2019. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2019.06.020. URL https://www.sciencedirect.com/science/article/pii/S0045782519303639.

[32] T. Hirschler, R. Bouclier, D. Dureisseix, A. Duval, T. Elguedj, and J. Morlier. A dual domain decomposition algorithm for the analysis of non-conforming isogeometric Kirchhoff–Love shells. *Computer Methods in Applied Mechanics and Engineering*, 357:112578, 2019. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2019.112578. URL https://www.sciencedirect.com/science/article/pii/S0045782519304438.

[33] T. Hirschler, P. Antolin, and A. Buffa. Fast and multiscale formation of isogeometric matrices of microstructured geometric models. *Computational Mechanics*, 69(2):439–466, Feb 2022. ISSN 1432-0924. doi: 10.1007/s00466-021-02098-y. URL https://link.springer.com/content/pdf/10.1007/s00466-021-02098-y.pdf.

[34] T. Hirschler, R. Bouclier, P. Antolin, and A. Buffa. Reduced order modeling based inexact FETI-DP solver for lattice structures. *International Journal for Numerical Methods in Engineering*, 125(8):e7419, 2024. doi: https://doi.org/10.1002/nme.7419. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.7419.

[35] C. Hofer and S. Takacs. *A Parallel Multigrid Solver for Multi-Patch Isogeometric Analysis*, pages 205–219. Springer International Publishing, 2019. ISBN 9783030142445. doi: 10.1007/978-3-030-14244-5_11. URL http://dx.doi.org/10.1007/978-3-030-14244-5_11.

[36] C. Hofer, U. Langer, and S. Takacs. *Inexact Dual-Primal Isogeometric Tearing and Interconnecting Methods*, pages 393–403. Springer International Publishing, 2018. ISBN 9783319938738. doi: 10.1007/978-3-319-93873-8_37. URL http://dx.doi.org/10.1007/978-3-319-93873-8_37.

[37] C. Hofreither and S. Takacs. Robust multigrid for isogeometric analysis based on stable splittings of spline spaces. *SIAM Journal on Numerical Analysis*, 55(4):2004–2024, Jan. 2017. ISSN 1095-7170. doi: 10.1137/16m1085425. URL http://dx.doi.org/10.1137/16M1085425.

[38] C. Hofreither, S. Takacs, and W. Zulehner. A robust multigrid method for isogeometric analysis in two dimensions using boundary correction. *Computer Methods in Applied Mechanics and Engineering*, 316:22–42, 2017. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2016.04.003. URL https://www.sciencedirect.com/science/article/pii/S0045782516301414. Special Issue on Isogeometric Analysis: Progress and Challenges.

[39] T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39):4135–4195, 2005. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2004.10.008. URL https://www.sciencedirect.com/science/article/pii/S0045782504005171.

[40] T. Hughes, A. Reali, and G. Sangalli. Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: Comparison of p-method finite elements with k-method NURBS. *Computer Methods in Applied Mechanics and Engineering*, 197(49):4104–4124, 2008. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2008.04.006. URL https://www.sciencedirect.com/science/article/pii/S0045782508001618.

[41] T. J. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Computer methods in applied mechanics and engineering*, 199(5-8):301–313, 2010.

[42] S. Khakalo, V. Balobanov, and J. Niiranen. Modelling size-dependent bending, buckling and vibrations of 2D triangular lattices by strain gradient elasticity models: Applications to sandwich beams and auxetics. *International Journal of Engineering Science*, 127:33–52, 2018. ISSN 0020-7225. doi: https://doi.org/10.1016/j.ijengsci.2018.02.004. URL https://www.sciencedirect.com/science/article/pii/S0020722517319535.

[43] S. Ki Moon, Y. En Tan, J. Hwang, and Y.-J. Yoon. Application of 3D printing technology for designing light-weight unmanned aerial vehicle wing structures. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 2015.

[44] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff–Love elements. *Computer methods in applied mechanics and engineering*, 198(49-52):3902–3914, 2009.

[45] A. Klawonn and O. Rheinbach. Inexact FETI-DP methods. *International Journal for Numerical Methods in Engineering*, 2006.

[46] A. Klawonn and O. B. Widlund. Dual-primal FETI methods for linear elasticity. *Communications on Pure and Applied Mathematics*, 59(11): 1523–1572, 2006. ISSN 1097-0312. doi: 10.1002/cpa.20156. URL http://dx.doi.org/10.1002/cpa.20156.

[47] A. Klawonn, M. Kühn, and O. Rheinbach. Adaptive coarse spaces for FETI-DP in three dimensions. *SIAM Journal on Scientific Computing*, 38(5):A2880–A2911, Jan. 2016. ISSN 1095-7197. doi: 10.1137/15m1049610. URL http://dx.doi.org/10.1137/15M1049610.

[48] S. K. Kleiss, C. Pechstein, B. Jüttler, and S. Tomar. IETI – Isogeometric Tearing and Interconnecting. *Computer Methods in Applied Mechanics and Engineering*, 247-248:201–215, 2012. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2012.08.007. URL https://www.sciencedirect.com/science/article/pii/S0045782512002599.

[49] S. Lipton, J. A. Evans, Y. Bazilevs, T. Elguedj, and T. J. Hughes. Robustness of isogeometric structural discretizations under severe mesh distortion. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):357–373, 2010.

[50] J. Lv, G. Du, P. Jin, and R. Li. Heat transfer analysis and structural optimization for spiral channel regenerative cooling thrust chamber. *International Journal of Aerospace Engineering*, 2023(1):8628107, 2023.

[51] K. Ma and Y. Bazilevs. Isogeometric analysis of architected materials and structures. *Engineering with Computers*, pages 1–15, 2024.

[52] A. Mantzaflaris and B. Jüttler. Integration by interpolation and look-up for Galerkin-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:373–400, 2015.

[53] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, and U. Langer. Low rank tensor methods in Galerkin-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:1062–1085, 2017.

[54] F. Masi and I. Stefanou. Multiscale modeling of inelastic materials with Thermodynamics-based Artificial Neural Networks (TANN). *Computer Methods in Applied Mechanics and Engineering*, 398:115190, 2022.

[55] F. Massarwi, J. Machchhar, P. Antolin, and G. Elber. Hierarchical, random and bifurcation tiling with heterogeneity in micro-structures construction via functional composition. *Computer-Aided Design*, 102:148–159, 2018. ISSN 0010-4485. doi: https://doi.org/10.1016/j.cad.2018.04.017. URL https://www.sciencedirect.com/science/article/pii/S0010448518302264. Proceeding of SPM 2018 Symposium.

[56] M. Montardini, G. Sangalli, R. Schneckenleitner, S. Takacs, and M. Tani. A IETI-DP method for discontinuous Galerkin discretizations in isogeometric analysis with inexact local solvers. *Mathematical Models and Methods in Applied Sciences*, 33(10):2085–2111, Aug. 2023. ISSN 1793-6314. doi: 10.1142/s0218202523500495. URL http://dx.doi.org/10.1142/S0218202523500495.

[57] M. Montardini, G. Sangalli, and M. Tani. A low-rank isogeometric solver based on Tucker tensors. *Computer Methods in Applied Mechanics and Engineering*, 417:116472, 2023.

[58] S. Morganti, F. Auricchio, D. Benson, F. Gambarin, S. Hartmann, T. Hughes, and A. Reali. Patient-specific isogeometric structural analysis of aortic valve closure. *Computer methods in applied mechanics and engineering*, 284:508–520, 2015.

[59] R. Newlands. The thrust optimised parabolic nozzle. Technical paper, Aspirespace, 2017. URL http://aspirespace.org.uk/downloads/Thrust%20optimised%20parabolic%20nozzle.pdf.

[60] nTopology. Element pro case study: F1 brake pedal. *https://ntopology.com/wp- content/uploads/2019/01/CaseStudy_Brake_Pedal.pdf*, 2019.

[61] C. Pan, Y. Han, and J. Lu. Design and optimization of lattice structures: A review. *Applied Sciences*, 10(18):6374, Sept. 2020. ISSN 2076-3417. doi: 10.3390/app10186374. URL http://dx.doi.org/10.3390/app10186374.

[62] M. Pan, B. Jüttler, and A. Giust. Fast formation of isogeometric Galerkin matrices via integration by interpolation and look-up. *Computer Methods in Applied Mechanics and Engineering*, 366:113005, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113005. URL https://www.sciencedirect.com/science/article/pii/S0045782520301894.

[63] J.-C. Passieux, R. Bouclier, and O. Weeger. Image-based isogeometric twins of lattices with virtual image correlation for varying cross-section beams. *International Journal for Numerical Methods in Engineering*, 124(10):2237–2260, 2023.

[64] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[65] C. M. Portela, J. R. Greer, and D. M. Kochmann. Impact of node geometry on the effective stiffness of non-slender three-dimensional truss lattice architectures. *Extreme mechanics letters*, 22:138–148, 2018.

[66] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations*. Springer International Publishing, 2016. ISBN 9783319154312. doi: 10.1007/978-3-319-15431-2. URL http://dx.doi.org/10.1007/978-3-319-15431-2.

[67] G. Rao. Exhaust nozzle contour for optimum thrust. *Journal of Jet Propulsion*, 28(6):377–382, 1958.

[68] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[69] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[70] G. Sangalli and M. Tani. Matrix-free weighted quadrature for a computationally efficient isogeometric k-method. *Computer Methods in Applied Mechanics and Engineering*, 338:117–133, 2018.

[71] T. A. Schaedler and W. B. Carter. Architected cellular materials. *Annual Review of Materials Research*, 46(1):187–210, 2016.

[72] V. Simoncini. Block triangular preconditioners for symmetric saddle-point problems. *Applied Numerical Mathematics*, 49(1):63–80, 2004. ISSN 0168-9274. doi: https://doi.org/10.1016/j.apnum.2003.11.012. URL https://www.sciencedirect.com/science/article/pii/S0168927403001958. Numerical Algorithms, Parallelism and Applications.

[73] H. Sundar, G. Biros, C. Burstedde, J. Rudi, O. Ghattas, and G. Stadler. Parallel geometric-algebraic multigrid on unstructured forests of octrees. In *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2012. doi: 10.1109/SC.2012.91.

[74] S. Takacs. Fast multigrid solvers for conforming and non-conforming multi-patch isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 371:113301, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113301. URL https://www.sciencedirect.com/science/article/pii/S0045782520304862.

[75] W. Tao and M. C. Leu. Design of lattice structure for additive manufacturing. *Conference: 2016 International Symposium on Flexible Automation (ISFA)*, 2016.

[76] R. Tielen, M. Moller, and C. Vuik. Efficient multigrid based solvers for isogeometric analysis. In *6th European Conference on Computational Mechanics (ECCM 6) 7th European Conference on Computational Fluid Dynamics (ECFD 7)*, 2018.

[77] R. Tielen, M. Möller, D. Göddeke, and C. Vuik. p-multigrid methods and their comparison to h-multigrid methods within isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 372:113347, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113347. URL https://www.sciencedirect.com/science/article/pii/S0045782520305326.

[78] V. Vasiliev, V. Barynin, and A. Razin. Anisogrid composite lattice structures – development and aerospace applications. *Composite Structures*, 94(3):1117–1127, 2012. ISSN 0263-8223. doi: https://doi.org/10.1016/j.compstruct.2011.10.023. URL https://www.sciencedirect.com/science/article/pii/S0263822311004004.

[79] M. Wangermez, O. Allix, P.-A. Guidault, O. Ciobanu, and C. Rey. Interface coupling method for the global–local analysis of heterogeneous models: A second-order homogenization-based strategy. *Computer Methods in Applied Mechanics and Engineering*, 365:113032, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113032. URL https://www.sciencedirect.com/science/article/pii/S0045782520302164.

[80] O. Weeger. Isogeometric sizing and shape optimization of 3D beams and lattice structures at large deformations. *Structural and Multidisciplinary Optimization*, 65(2):43, 2022.

[81] O. B. Widlund, S. Scacchi, and L. F. Pavarino. BDDC deluxe algorithms for two-dimensional H(*curl*) isogeometric analysis. *SIAM Journal on Scientific Computing*, 44(4):A2349–A2369, Aug. 2022. ISSN 1095-7197. doi: 10.1137/21m1438839. URL http://dx.doi.org/10.1137/21M1438839.

[82] R. Xu, J. Yang, W. Yan, Q. Huang, G. Giunta, S. Belouettar, H. Zahrouni, T. B. Zineb, and H. Hu. Data-driven multiscale finite element method: From concurrence to separation. *Computer Methods in Applied Mechanics and Engineering*, 363:112893, 2020.

[83] J. Yan, S. Huo, T. Yu, C. Zhang, X. Chai, D. Hu, and K. Yan. Multiscale analysis for 3D lattice structures based on parallel computing. *International Journal for Numerical Methods in Engineering*, 122(22):6756–6776, 2021.