Refined Isogeometric Analysis for a Preconditioned Conjugate Gradient Solver

Daniel Garcia[a,*], David Pardo[b,a,c], Lisandro Dalcin[d,e,f], Victor M. Calo[g,h,i]

[a] *Basque Center for Applied Mathematics, (BCAM), Bilbao, Spain.*
[b] *Department of Applied Mathematics, Statistics, and Operational Research, University of the Basque Country UPV/EHU, Leioa, Spain*
[c] *Ikerbasque (Basque Foundation for Sciences), Bilbao, Spain.*
[d] *Center for Numerical Porous Media, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia.*
[e] *Consejo Nacional de Investigaciones Científicas y Técnicas, Santa Fe, Argentina.*
[f] *Universidad Nacional del Litoral, Santa Fe, Argentina.*
[g] *Department of Applied Geology, Western Australian School of Mines, Curtin University, Perth, Australia.*
[h] *Mineral Resources, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Perth, Australia.*
[i] *Curtin Institute for Computation, Curtin University, Perth, Australia.*

Abstract

Starting from a highly continuous Isogeometric Analysis (IGA) discretization, refined Isogeometric Analysis (rIGA) introduces C^0 hyperplanes that act as separators for the direct LU factorization solver. As a result, the total computational cost required to solve the corresponding system of equations using a direct LU factorization solver dramatically reduces (up to a factor of 55) [1]. At the same time, rIGA enriches the IGA spaces, thus improving the best approximation error. In this work, we extend the complexity analysis of rIGA to the case of iterative solvers. We build an iterative solver as follows: we first construct the Schur complements using a direct solver over small subdomains (macro-elements). We then assemble those Schur complements into a global skeleton system. Subsequently, we solve this system iteratively using Conjugate Gradients (CG) with an incomplete LU (ILU) preconditioner. For a 2D Poisson model problem with a structured mesh and a uniform polynomial degree of approximation, rIGA achieves moderate savings with respect to IGA in terms of the number of Floating Point Operations (FLOPs) and computational time (in seconds) required to solve the resulting system of linear equations. For instance, for a mesh with four million elements and polynomial degree $p = 3$, the iterative solver is approximately 2.6 times faster (in time) when applied to the rIGA system than to the IGA one. These savings occur because the skeleton rIGA system contains fewer non-zero entries than the IGA one. The opposite situation occurs for 3D problems, and as a result, 3D rIGA discretizations provide no gains with respect to their IGA counterparts when considering iterative solvers.

Keywords: Isogeometric Analysis (IGA), Finite Element Analysis (FEA), refined Isogeometric Analysis (rIGA), solver-based discretization, iterative solvers, Conjugate gradient, Incomplete LU factorization, k-refinement.

1. Introduction

Galerkin-based discretizations such as Finite Element Analysis (FEA) and Isogeometric Analysis (IGA) are commonly employed nowadays to solve numerical problems governed by partial differential equations (PDEs) [2–16]. These methods employ a variational formulation with trial and test functions defined as a linear combination of basis functions to build a discretization of the governing PDEs.

Isogeometric Analysis (IGA) defines the geometry using conventional Computed-Aided Design (CAD) functions and, in particular, non-uniform rational basis spline (NURBS) [2]. These functions represent complex geometries commonly found in engineering design and are capable of preserving exactly the geometry description under refinement as required in the analysis. Moreover, the use of NURBS as basis functions is compatible with the isoparametric

[*]Corresponding author
Email address: `dgarcia@bcamath.org` (Daniel Garcia)
URL: `www.bcamath.org/dgarcia` (Daniel Garcia)

concept, that is, the same set of basis functions can be used for both geometry representation and analysis. Also, highly continuous NURBS often provide better approximation properties than traditional FEA on a per degree of freedom basis [17, 18]. This suggests that IGA is an accurate and robust scheme to approximate the solution of linear elasticity, structural vibration, wave propagation [19–22], and fluid mechanics [9–16, 23–28]. This method has also been implemented to solve problems in other engineering fields such as fluid-structure interaction (FSI) [6–8], phase transition phenomena [3–5, 29, 30], and medical applications [28, 31–33].

Once the problems governed by PDEs are in a discrete form, they can be solved with either direct or iterative solvers. In both cases, it is more expensive on a per degree of freedom basis to solve a C^{p-1} IGA discretization than a C^0 FEA one. Specifically, the cost of solving a C^{p-1} IGA system is $O(p^3)$ larger when using direct solvers and $O(p^2)$ larger when using iterative solvers, than the required cost to solve a C^0 FEA discretization with the same number of unknowns [34–37].

For the case of direct solvers, [1] introduces a refined isogeometric analysis (rIGA) discretization method to solve problems governed by PDEs. Starting from C^{p-1} IGA, the discretization technique reduces the continuity over certain hyperplanes on the mesh. By doing so, the cost of the LU factorization decreases. For fixed mesh sizes, a direct solver computes the solution of a rIGA discretization $O(p^2)$ times faster than for a C^{p-1} IGA one. For instance, authors of [1] solve a third order rIGA discretization corresponding to a 3D elliptic problem with two million elements in approximately one hour, while the corresponding IGA system requires 15 hours to be solved (despite being coarser), and the FEA discretization needs over 100 hours. Moreover, rIGA enriches the discrete space when reducing continuity along the macro-elements interfaces. This continuity reduction increases the number of basis functions into the macro-elements surrounding the interfaces, thus, augmenting the total number of degrees of freedom. This implies an improvement on the best approximation error with respect to C^{p-1} IGA discretization, although not necessarily on the stability constant.

We extend the analysis of rIGA to the case of iterative solvers and analyze the main features and limitations of the method. This extension involves an iterative solver that consists of four steps. First, the solver partitions the mesh problem into subdomains. We refer to these as blocks or macro-elements. The mesh partitioning uses hyperplanes that reduce the continuity over some inter-element boundaries. Then, the solver performs a static condensation of the macro-elements interior degrees of freedom. This consists of building the Schur complement of the interior macro-element degrees of freedom in terms of the boundary (interfacial) ones. This reduces the system size to just the degrees of freedom located along the macro-elements boundaries (mesh skeleton). Third, the iterative solver computes the solution of the reduced system. Finally, a backward substitution using the factors obtained when performing the static condensation allows us to recover the solution of the original system. Thus, our hybrid solver combines a direct solver to build the Schur complements of the macro-elements with an iterative solver to solve the skeleton system.

To simplify the analysis of the hybrid solver strategy, we restrict the study to a Poisson problem over a unitary domain discretized using regular and structured meshes containing the same number of elements in all spatial dimensions. Moreover, we perform the mesh partition in such a way that the macro-elements exhibit maximal continuity C^{p-1}, while the normal continuity on their interfaces is reduced to C^0. The theoretical analysis assumes that the mesh size and polynomial order of the approximation are fixed, and only the continuity at the element interfaces is modified.

We limit the study to only the Conjugate Gradient iterative method preconditioned with Incomplete LU factorization ILU(0) technique due to the large number of existing iterative solvers and the complexity required to analyze all of them simultaneously. Still, much of the analysis performed here can be easily generalized to the case of other iterative solvers. The Conjugate Gradient (CG) iterative solver is one of the best alternatives to study the hybrid solver strategy considering we use a boundary value problem based on Poisson's equation as the test problem. Besides, the Incomplete LU factorization ILU(0) is a method that performs adequately in terms of reducing the number of iterations and the computational times in IGA discretizations of Poisson equation [36]. For additional details on the iterative solver and preconditioner technique, we refer to [38, 39].

The paper is organized as follows: we briefly describe rIGA in Section 2. In Section 3, we discuss the implementation of rIGA on iterative solvers. Section 4 derives computational cost estimates. Sections 5 details the implementation. We describe the model problem and the numerical experiments in Section 6. The work concludes in Section 7 summarizing the main features and limitations of the proposed method.

2. Refined Isogeometric Analysis for direct solvers

Highly continuous discretizations as C^{p-1} IGA degrade the performance of the direct solvers on a per degree of freedom basis. The degradation comes from the growth in connectivities between the degrees of freedom in the mesh, as it occurs when we increase the continuity of the discretization. This results in an increment of the computational cost both in terms of FLOPs and memory requirements [34, 35].

The refined isogeometric analysis (rIGA) [1] is a discretization technique that optimizes the solver performance for a given mesh while preserving the optimal convergence order of the method with respect to the fixed element size. This strategy improves the performance of direct solvers when computing the solution using highly continuous discretizations. rIGA introduces an arbitrary number of C^0-hyperplanes that act as separators during the recursive mesh partitioning in direct solvers. Figure 1 illustrates the classical C^{p-1} IGA (left) and C^0 FEA (right) discretizations, in addition to a third intermediate discretization based on rIGA (center).

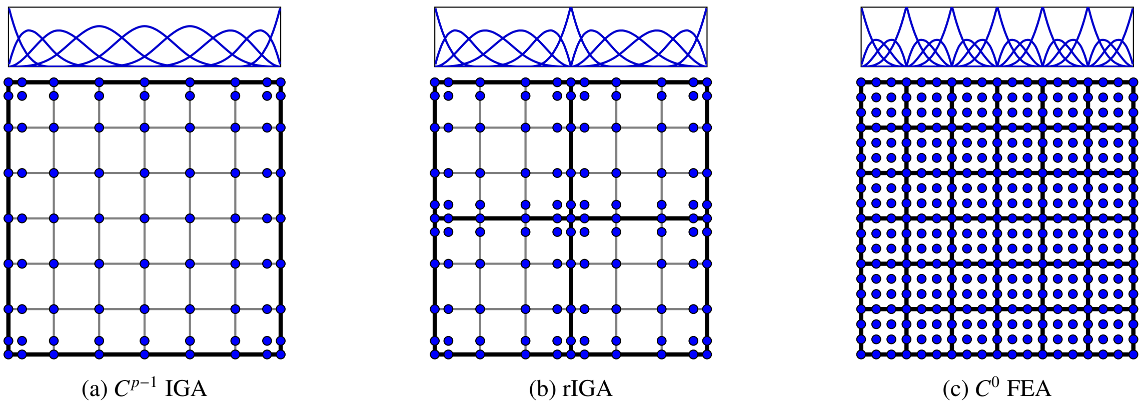
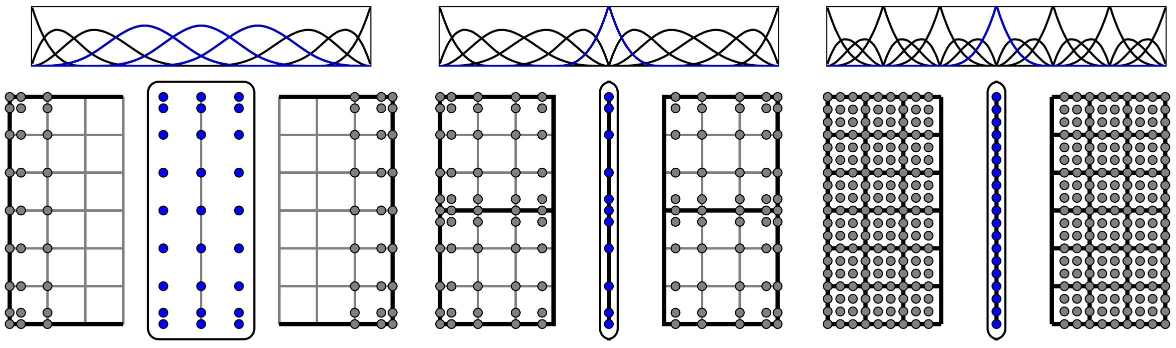


Figure 1: Illustration of Galerkin discretizations of a 2D system composed of 6×6 elements with polynomial basis functions of order $p = 3$. Blue circles represent the nodal degrees of freedom in the system, while black lines denote the mesh skeleton. Bold lines represent C^0 continuity.

The C^0-hyperplanes reduce the continuity along certain inter-element boundaries. More precisely, the continuity becomes zero across the interface between the subdomains, which are the ones that the recursive partitioning algorithm for the mesh would select as it travels through the elimination process. This results in a weakening of the interconnection between the subdomains, which improves the performance of the factorization. Figure 2 illustrates different sizes of separators that interconnect two subdomains. The size of a separator consists of the number of basis functions with support over the corresponding inter-element boundary, and decreases as the continuity transversely to the separator is reduced (e.g., Figure 2b), weakening the interconnection between subdomains.


(a) Separator used to partition a C^{p-1} IGA system

(b) Separator used to partition an rIGA system

(c) Separator used to partition an C^0 FEA system

Figure 2: Illustration of a separator used to interconnect two subdomains that result from partitioning a 2D system with 6×6 elements and polynomial order $p = 3$. In this example, rIGA introduces two C^0-hyperplanes (separators).

The best discretization provided by rIGA simultaneously reduces solution time and memory requirements as well as the best approximation error with respect to those on C^{p-1} IGA. The total computational time of the direct solver becomes faster by a factor of approximately p^2 when compared to IGA, and the gains are even larger when compared to FEA, particularly in 3D (see [1]).

3. Refined Isogeometric Analysis for preconditioned conjugate gradients

In this section, we extend the computational complexity analysis of the refined Isogeometric Analysis (rIGA) to the case of a Conjugate Gradient (CG) iterative solver preconditioned with the Incomplete LU factorization ILU(0). The discretization method starts by partitioning the mesh into macro-elements using C^k-hyperplanes, being k the degree in continuity. The partitioning involves a reduction of continuity in such a way that it weakens the interconnection between the subdomains (macro-elements). In this work, we focus on C^{p-1} discretizations partitioned by an arbitrary number of C^0-hyperplanes in order to make the problem tractable. Figure 3 illustrates the partitioning of a mesh into four subdomains.

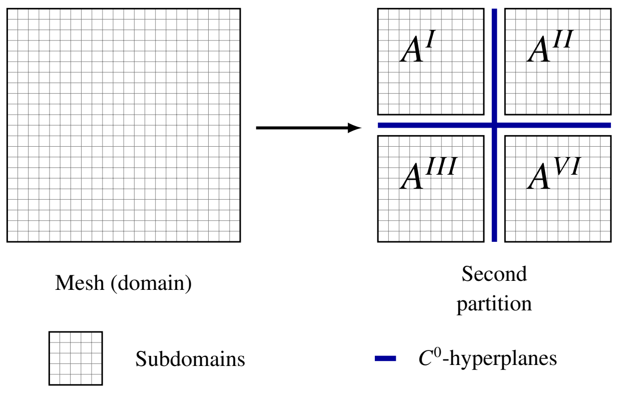


Figure 3: Partition of a 2D mesh into four subdomains (macro-elements).

Afterward, the method performs a static condensation in *all* macro-elements. This consists in a partial LU (Cholesky) factorization that eliminates the degrees of freedom inside every macro-element and results in a reduced system which involves only the degrees of freedom at the boundaries (skeleton). That is, we compute the Schur complement for *all* macro-elements and build the skeleton problem by assembling all these Schur complements. Considering the system $A^i\mathbf{x}^i = \mathbf{b}^i$ associated to the degrees of freedom of the i-th macro-element, we perform an LU factorization $A^i = L^i\, U^i$ which results in

$$\begin{pmatrix} A^i_{int,int} & A^i_{int,bnd} \\ A^i_{bnd,int} & A^i_{bnd,bnd} \end{pmatrix} \begin{pmatrix} \mathbf{x}^i_{int} \\ \mathbf{x}^i_{bnd} \end{pmatrix} = \begin{pmatrix} \mathbf{b}^i_{int} \\ \mathbf{b}^i_{bnd} \end{pmatrix} \Rightarrow \begin{pmatrix} L^i_{int,int} & 0 \\ L^i_{bnd,int} & I \end{pmatrix} \begin{pmatrix} U^i_{int,int} & U^i_{int,bnd} \\ 0 & S^i \end{pmatrix} \begin{pmatrix} \mathbf{x}^i_{int} \\ \mathbf{x}^i_{bnd} \end{pmatrix} = \begin{pmatrix} \mathbf{b}^i_{int} \\ \mathbf{b}^i_{bnd} \end{pmatrix}, \tag{1}$$

where the subscripts *int* and *bnd* refers to the degrees of freedom located in the interior and at the boundaries of the macro-element, respectively. S^i corresponds to the Schur complement of the A^i matrix, which is defined as

$$S^i = A^i_{bnd,bnd} - A^i_{bnd,int} \left(A^i_{int,int}\right)^{-1} A^i_{int,bnd}\,. \tag{2}$$

The reduced right-hand side $\mathbf{y}^i_{bnd}$ for the i-th macro-element is computed using Equation 3.

$$\begin{pmatrix} L^i_{int,int} & 0 \\ L^i_{bnd,int} & I \end{pmatrix} \begin{pmatrix} \mathbf{y}^i_{int} \\ \mathbf{y}^i_{bnd} \end{pmatrix} = \begin{pmatrix} \mathbf{b}^i_{int} \\ \mathbf{b}^i_{bnd} \end{pmatrix}, \tag{3}$$

and the reduced system for the i-th macro-element is

$$S^i \mathbf{x}^i_{bnd} = \mathbf{y}^i_{bnd}\,, \tag{4}$$

where $i = 1, ...N_{m-e}$, being N_{m-e} the number of macro-elements. The reduced systems are assembled all together obtaining the skeleton system $A_{skl}\,\mathbf{x}_{skl} = \mathbf{y}_{skl}$, which corresponds to the degrees of freedom located along the macro-element boundaries (C^0 skeleton mesh). Figure 4 illustrates the static condensation step.

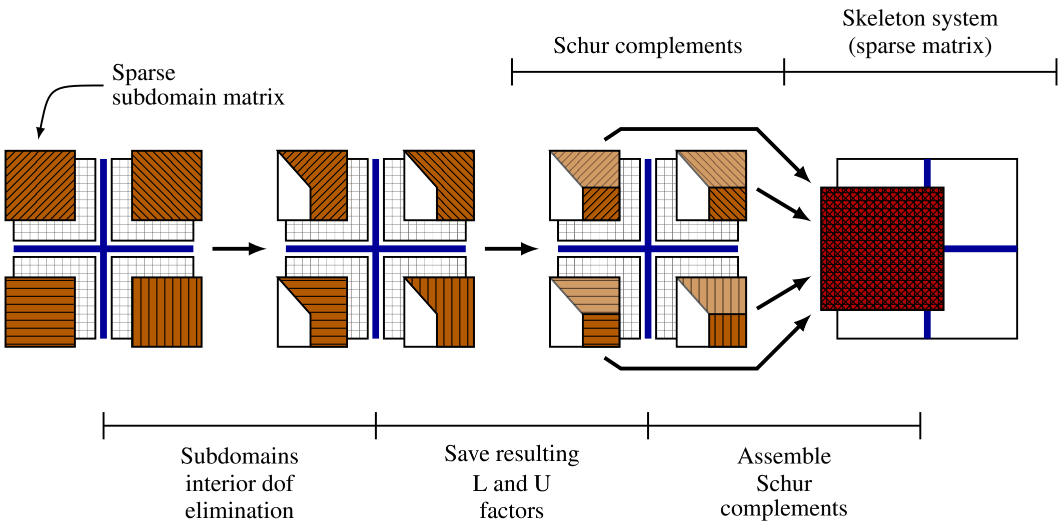


Figure 4: Static condensation procedure for a 2D system recursively partitioned into four subdomains.

We compute the solution of the skeleton system using the Conjugate Gradient iterative solver. First, we compute the preconditioner matrix P using the Incomplete LU factorization technique, ILU(0). Subsequently, we solve the preconditioned system. Lastly, we perform a backward substitution to obtain the solution of the original system. The backward substitution employs the factors obtained in the static condensation step.

4. Theoretical cost estimates for the preconditioned conjugate gradient solver

This section provides theoretical estimates of the number of FLOPs required to compute the solution of a problem when using the hybrid solver strategy described in the previous section. We restrict to fixed mesh sizes and polynomial orders $p > 1$. We separately analyze the computational cost corresponding to each step of the rIGA implementation for iterative solvers. For simplicity, we assume that the mesh has a size power of two in each dimension (i.e., $N_{elements} = (2^i)^d$ with $i \in \mathbb{Z}^+$) and the same number of elements in each spatial dimension (d). This configuration allows to split the mesh symmetrically, obtaining the same number of macro-elements in each dimension. Moreover, the resulting macro-elements have also a size power of two ($s = 2^j$ with $j = 1, 2, ..., \log_2(\sqrt[d]{N_{elements}})$). The use of those mesh sizes facilitates the construction of the cost estimates for static condesation since we can correlate the cost of one partition level with the previous partition levels, and also for the remaining operations since the skeleton mesh size is simple to estimate.

4.1. Static condensation of the macro-elements interior degrees of freedom

We partition the mesh into N_{m-e} macro-elements (Figure 3). Each macro-element corresponds to a C^{p-1} IGA system of size equal to $N_{dof} = (s + p)^d$, being d the dimension and s^d the number of elements into the macro-element (Figure 5). Depending on the size of the macro-element, we use a different approach to estimate the number of FLOPs

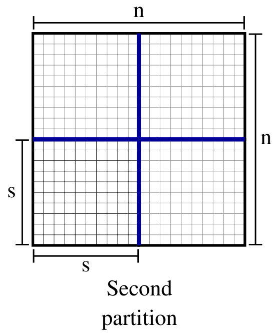


Figure 5: Illustration of a 2D mesh partitioned into four subdomains. The total number of elements is n^d, while the number of elements in each subdomains is s^d. The number of macro-elements is $N_{m-e} = (n/s)^d$. In this case $d = 2$ is the spatial dimension.

required to compute the reduced system. For small macro-elements sizes that involve nearly dense matrices, the cost is identical as performing the elimination of the interior degrees of freedom for a dense matrix problem. However, when the macro-elements are large, the corresponding cost is similar as computing the Schur complement for a sparse matrix.

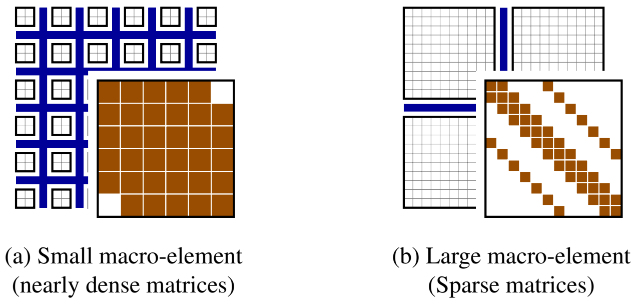


Figure 6: Illustration of the static condensation of the interior degrees of freedom for a single macro-element.

We realize that the macro-elements containing a number of elements $s^d \leq (p + 1)^d$, being $(p + 1)^d$ the support of the C^{p-1} basis functions, involve matrices nearly dense (Figure 6a). In this case, the cost of static condensation is the same as performing a partial factorization on a dense matrix with size N_{dof}. The cost to perform the partial factorization on the *i*-th macro-element is given by

$$\varphi_{sc}^{i} = \sum_{k=1}^{N^{\circ}_{dof}} \left(\sum_{i=k+1}^{N_{dof}} \left(1 + \sum_{j=k+1}^{N_{dof}} 2 \right) \right) = \frac{2}{3} \left(N^{\circ}_{dof} \right)^{3} + \left(\frac{1}{2} - 2N_{dof} \right) \left(N^{\circ}_{dof} \right)^{2} + \left(2 \left(N_{dof} \right)^{2} - N_{dof} - \frac{1}{6} \right) N^{\circ}_{dof}, \tag{5}$$

where $N^{\circ}_{dof} = (s + p - 2)^{d}$ is the number of interior degrees of freedom. The previous cost counts the number of operations required to perform Gaussian elimination of the interior degrees of freedom [40].

Furthermore, we consider that macro-elements containing a number of elements $s^{d} > (p + 1)^{d}$ involve sparse matrices (Figure 6b). In those cases, we estimate the cost of static condensation as if a multifrontal technique is used to compute the Schur complement. The cost in this case is given by

$$\psi_{sc}^{i} \approx \sum_{j=1}^{\ell} \left(\sum_{k=1}^{d} \left(2^{k-1} \cdot 2^{d(j-1)} \overbrace{\underbrace{\left(2^{-(j+k-2)} \left(\sqrt[d]{N^{\circ}_{dof}} - (p-1) \right) \right)}_{\text{Length}}{}^{3(d-1)} \underbrace{p^{3}}_{\text{Thickness}}}^{\text{Separator size}} \right) \right), \tag{6}$$

where ℓ is the number of partition levels. Term $(p - 1)$ refers to the number of degrees of freedom belonging to the separator that was removed by previous partition levels. For the case of a 2D problem, the equation simplifies to

$$\psi_{sc}^{i}|_{2D} \approx \sum_{j=1}^{\ell} \left(2^{2(j-1)} \left(2^{-(j-1)} \left(\sqrt{N^{\circ}_{dof}} - (p-1) \right) \right)^{3} p^{3} + 2^{2(j-1)+1} \left(2^{-(j)} \left(\sqrt{N^{\circ}_{dof}} - (p-1) \right) \right)^{3} p^{3} \right). \tag{7}$$

The first separator involves a cost of $\left(N^{\circ}_{dof} \right)^{3/2} p^{3}$ since we did not perform any prior partition. Therefore, the total cost becomes

$$\psi_{sc}^{i}|_{2D} = O\left(\left(N^{\circ}_{dof} \right)^{3/2} p^{3} + \frac{1}{2} \left(3 - 5 \cdot 2^{-\ell} \right) \left(\sqrt{N^{\circ}_{dof}} - (p-1) \right)^{3} p^{3} \right). \tag{8}$$

Finally, the estimate of the number of FLOPs required by static condensation of the *i*-th macro-element in 2D is given by

$$\theta_{sc}^{i}|_{2D} = \begin{cases} \varphi_{sc}^{i}|_{2D} & \forall\, s \leq p + 1 \\ \psi_{sc}^{i}|_{2D} & \forall\, s > p + 1, \end{cases}$$

that is

$$\theta_{sc}^{i}|_{2D} = \begin{cases} \frac{2}{3} \left(N^{\circ}_{dof} \right)^{3} + \left(\frac{1}{2} - 2N_{dof} \right) \left(N^{\circ}_{dof} \right)^{2} + \left(2 \left(N_{dof} \right)^{2} - N_{dof} - \frac{1}{6} \right) N^{\circ}_{dof} & \forall\, s \leq p + 1 \\ O\left(\left(N^{\circ}_{dof} \right)^{3/2} p^{3} + \frac{1}{2} \left(3 - 5 \cdot 2^{-\ell} \right) \left(\sqrt{N^{\circ}_{dof}} - (p-1) \right)^{3} p^{3} \right) & \forall\, s > p + 1. \end{cases} \tag{9}$$

Ultimately, the cost of the static condensation step consists of the number of FLOPs required to eliminate the interior degrees of freedom in *all* macro-elements. This cost is given by

$$\theta_{sc} = \sum_{i=1}^{N_{m-e}} \theta_{sc}^{i}, \tag{10}$$

where $N_{m-e} = (n/s)^{d}$, being n^{d} the total number of elements of the original mesh. For a fixed macro-element size, the above step scales linearly with respect to the total problem size.

The number of operations required to compute the reduced right-hand-side and assemble the skeleton system are omitted in the estimates, since their cost is negligible in comparison to the total solution cost.

4.2. ILU(0) preconditioner

To estimate the number of FLOPs needed to set up the preconditioner, we need to know the number of non-zero entries of the matrix A_{skl} in the skeleton system. The number of non-zero entries in the skeleton problem corresponds to that of a statically condensed C^0 FEA discretization with a matrix that has the same number of degrees of freedom on each macro-element interface. Figure 7 illustrates both the full and statically condensed C^0 FEA (right) and the original and statically condensed rIGA (left) discretizations for a 2D problem.

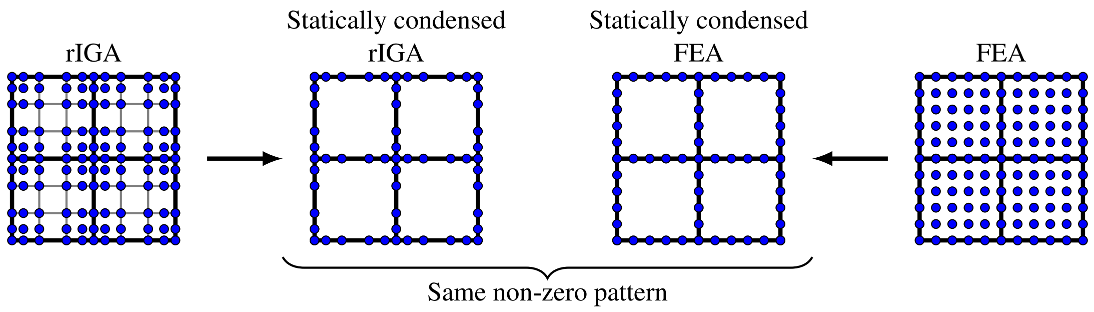


Figure 7: Illustration of both FEA and rIGA discretizations of a 2D system. The rIGA discretization composes of 2×2 subdomains, 6×6 elements and polynomial basis functions of order $p = 3$, while the FEA discretization consists of 2×2 elements and polynomial degree $\hat{p} = 5$. Blue circles represent the nodal degrees of freedom in the system, while black lines denote the mesh skeleton. Bold lines represent C^0 continuity.

We define as $\hat{p}$ the polynomial degree of the C^0 FEA system that after being statically condensed has the same non-zero pattern than that of the rIGA skeleton system. While the number of elements is equal to the number of macro-elements in which the original C^{p-1} system was partitioned, the size of the FEA elements corresponds to the number of degrees of freedom in a single macro-element. Therefore, we compute $\hat{p}$ by comparing the number of degrees of freedom between a FEA element and a rIGA macro-element.

$$\begin{aligned}
\text{FEA element} &= \text{rIGA subdomain} \\
(\hat{p}+1)^d &= (s+p)^d \\
\hat{p}+1 &= s+p \\
\hat{p} &= s+p-1.
\end{aligned}$$

The number of non-zero entries in the j-th matrix row corresponds to the number of nodes with which the j-th node interacts. Depending on the location of this node, the number of interactions nodes varies [36, 41]. Figure 8 illustrates the possible location of the nodes over the elements of the statically condensed C^0 FEA system. Moreover, Table 1 presents the number of nodes at each location as well as the number of interactions per node.

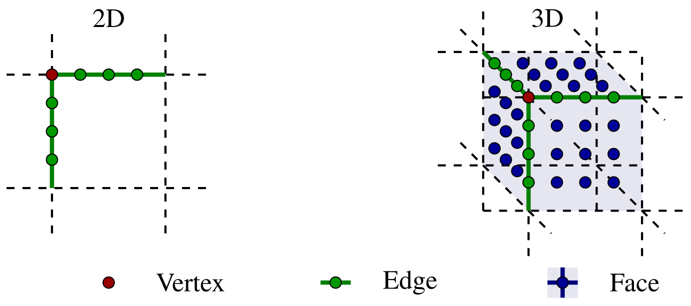


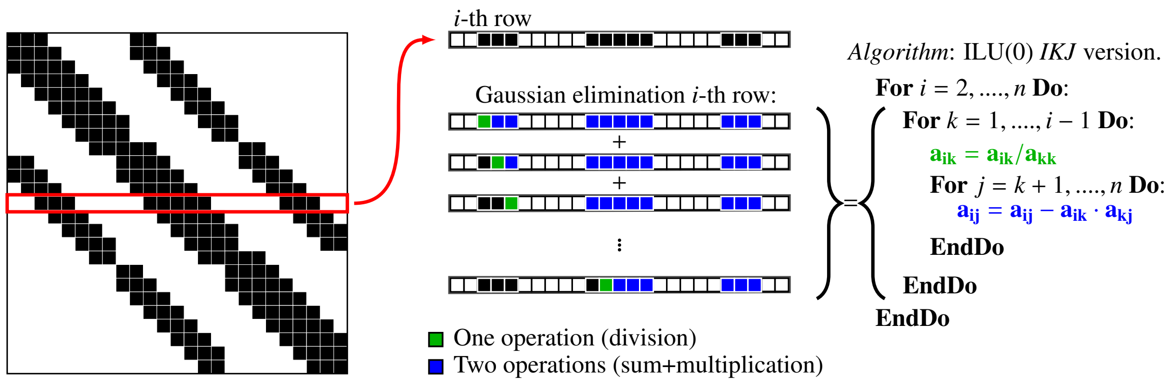
Figure 8: Illustration of the location of nodes over a single element in both 2D and 3D after static condensation.

Dimension	Type	# nodes	# interactions $(nnz\,(A_{skl}\vert_{row}))$
2D	Vertex	1	$(2\hat{p}+1)^2-4(\hat{p}-1)^2$
	Edge	$2(\hat{p}-1)$	$(2\hat{p}+1)(\hat{p}+1)-2(\hat{p}-1)^2$
3D	Vertex	1	$(2\hat{p}+1)^3-8(\hat{p}-1)^3$
	Edge	$3(\hat{p}-1)$	$(2\hat{p}+1)^2(\hat{p}+1)-4(\hat{p}-1)^3$
	Face	$3(\hat{p}-1)^2$	$(2\hat{p}+1)(\hat{p}+1)^2-2(\hat{p}-1)^3$

* Data for a single element

Table 1: Summary table of the nodes interactions for both 2D and 3D statically condensed C^0 FEA systems [36].

We consider an ILU(0) preconditioner based on the *IKJ* version of Gaussian elimination shown in [38]. The implementation performs as illustrated in Figure 9.



i-th row
Gaussian elimination i-th row:
+
+
⋮
🟩 One operation (division)
🟦 Two operations (sum+multiplication)

```
Algorithm: ILU(0) IKJ version.
For $i = 2, ...., n$ Do:
    For $k = 1, ...., i-1$ Do:
        $\mathbf{a_{ik} = a_{ik}/a_{kk}}$
        For $j = k+1, ...., n$ Do:
            $\mathbf{a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}}$
        EndDo
    EndDo
EndDo
```

Figure 9: Implementation of the ILU(0) preconditioning method. For simplicity, we illustrate the procedure of Gaussian elimination for a single row of the original matrix.

The number of FLOPs executed when performing a truncated Gaussian elimination in the j-th row is given by

$$\theta_{A_{skl}|_j} = \frac{1}{4}\left(3\,nnz(A_{skl}|_j)^2 - 2\,nnz(A_{skl}|_j) - 1\right), \tag{11}$$

where $nnz(A_{skl}|_j)$ refers to the number of non-zero entries in the j-th row of matrix A_{skl}. Finally, the number of FLOPs required to build the ILU(0) preconditioner is given by

$$\theta_{pre} = \begin{cases} N_{m-e}\left(2(\hat{p}-1)\,\theta_{A_{skl}|_{Edge}} + (1)\,\theta_{A_{skl}|_{Vertex}}\right) & 2D \\ N_{m-e}\left(3(\hat{p}-1)^2\theta_{A_{skl}|_{Face}} + 3(\hat{p}-1)\,\theta_{A_{skl}|_{Edge}} + (1)\,\theta_{A_{skl}|_{Vertex}}\right) & 3D. \end{cases} \tag{12}$$

For a fixed macro-element size, the cost of building the ILU(0) preconditioner scales linearly with respect to the total problem size.

4.3. Iterative solver

The major cost when solving a system with an iterative solver comes from the matrix-vector products. Each of those matrix-vector product consists of two operations, one sum and one multiplication, per non-zero entry of the sparse matrix [36]. The total number of operations is proportional to the number of non-zero entries in both the system and the preconditioner matrix. Therefore, the cost associated to solve the skeleton system once is given by

$$\theta_{it} = O\left(2\left(nnz\,(P) + nnz\,(A_{skl})\right)\right), \tag{13}$$

where P is the preconditioner, A_{skl} is the skeleton matrix. Both the skeleton and preconditioner matrix have the same number of non-zero entries, which allows to express the cost as

$$\theta_{it} = O\left(4\, nnz\left(A_{skl}\right)\right), \tag{14}$$

The estimate of the number of non-zero entries is computed based on Table 1, and given by

$$nnz\left(A_{skl}\right) = \begin{cases} N_{m-e}\Big(2(\hat{p}-1)\Big((2\hat{p}+1)(\hat{p}+1) - 2(\hat{p}-1)^2\Big) \\ \qquad + (1)\Big((2\hat{p}+1)^2 - 4(\hat{p}-1)^2\Big)\Big) & 2D \\ \\ N_{m-e}\Big(3(\hat{p}-1)^2\Big((2\hat{p}+1)(\hat{p}+1)^2 - 2(\hat{p}-1)^3\Big) \\ \quad +3(\hat{p}-1)\Big((2\hat{p}+1)^2(\hat{p}+1) - 4(\hat{p}-1)^3\Big) & 3D. \\ \qquad + (1)\Big((2\hat{p}+1)^3 - 8(\hat{p}-1)^3\Big)\Big). \end{cases} \tag{15}$$

As it occurred in the previous steps, the cost of matrix-vector multiplication scales linearly with respect to the total problem size for a fixed macro-element size.

5. Implementation details

We set the stopping criteria of the CG iterative solver as:

$$||e||_{H1} = ||e_I + e_D||_1 \leq \xi, \tag{16}$$

where e is the total error consisting of the sum of the error of the iterative solver (e_I) and the discretization error (e_D). We select $\xi = 2||e_D|_{IGA}||_1$, where $||e_D|_{IGA}||_1$ is the H^1-norm of the discretization error (e_D) resulting from solving the model problem with C^{p-1} IGA.

We consider a range of cases for every mesh size. Each case splits the mesh into a particular number of macro-elements using C^0-hyperplanes. The first one corresponds to traditional C^{p-1} IGA with no reduction of continuity, while the last case involves a total reduction of continuity which results in a traditional C^0 FEA.

The implementation of our method is based on the library PetIGA, which is a high-performance software platform for IGA [42] based on PETSc [43, 44]. PetIGA has been used to solve multiple problems in engineering [4, 5, 34–36, 42, 45–52]. With PetIGA, we can directly use the sequential version of the multifrontal solver MUMPS [53, 54] to perform the static condensation of the macro-elements interior degrees of freedom. Also, we can use the PETSc version of Conjugate Gradient (CG) iterative solver preconditioned with the Incomplete LU technique ILU(0) to solve the reduced system resulting from static condensation.

All computational tests are solved sequentially on TACC Stampede system. Each node is outfitted with turbo boost 2.7 GHz cores (up to peak 3.5 GHz in turbo mode) and 1TB of memory.

6. Numerical Results

We focus on both, static condensation of the macro-elements interior degrees of freedom and solution of the skeleton system using the iterative solver. The remaining operations only contribute with lower order terms (*L.O.T.*). We report the FLOPs and computational times (in seconds) with respect to the macro-element size in each direction (s), for all the macro-element sizes power of two ($s = 2^i$ with $i = 1, 2..., \log_2(n)$). We shall plot the cost of both static condensation and the preconditioner set-up. In the following numerical results, lines connect data of the same type, which allows us to compare the behavior of these sets of data.

The theoretical estimates that will be displayed with solid lines in the plots accurately predict the number of FLOPs of static condensation and the preconditioner set-up for cases with macro-element sizes power of two. For other macro-element sizes, we expect that the theoretical estimates will provide a close approximation to the real cost.

To simplify the discussion, we only focus on the cases which deliver the maximum reduction in solution time.

6.1. Model Problem

We use the 2D Poisson model problem presented in Equation (17) to analyze the performance of the hybrid solver strategy. In particular, we study the impact of using refined Isogeometric Analysis (rIGA) with a preconditioned CG iterative solver in the solution cost.

$$\begin{cases} \text{Find } u \text{ such that:} & \\ \nabla \cdot (\nabla u) = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{17}$$

where $\Omega = (0,1)^2$ and $f = 2\alpha^2\pi^2 \sin(\alpha\pi x)\sin(\alpha\pi y)$. We use a different value of α for each polynomial degree to have approximately the same $||e_D|_{IGA}||_{H^1}$ in all the examples.

We build the model problem using unmapped B-splines, assuming a unitary domain. This domain is defined by the tensor product of the unmapped B-splines resulting in a regular and structured mesh. The same number of elements in all spatial dimensions results in uniform macro-elements after performing the domain partitioning. We discretize the model problem with a mesh of 2048^2 and 4096^2 elements, and polynomial degrees p = 2, 3, 4, 5 and 6. The polynomial degree is kept constant in the entire mesh.

6.2. Fit of estimates

We fit the theoretical FLOPs estimates with the numerical results by using three constants, namely A, B and C. Thus, the theoretical estimate of the number of FLOPs required to solve the problem is given by:

$$\theta = \underbrace{A\,\theta_{SC}}_{\text{Static Condensation}} + \underbrace{B\,\theta_{pre}}_{\text{Preconditioning}} + \underbrace{N_{ite}\,(C\,\theta_{it})\,,}_{\text{Iterative solver}} \tag{18}$$

where N_{ite} is the number of iterations that the solver requires to converge. Table 2 lists the values of the constants A, B and C.

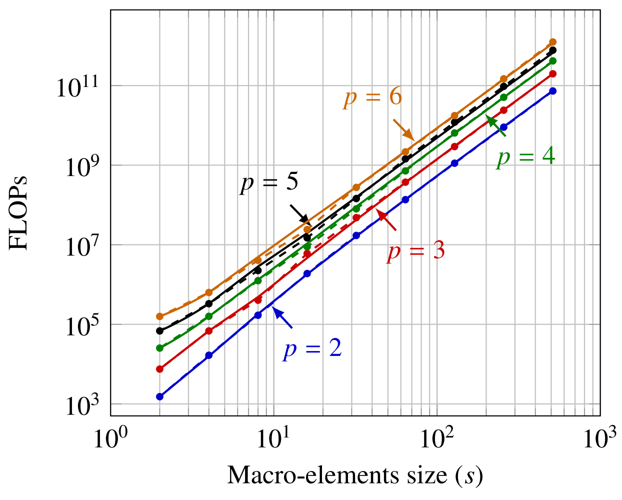
	p				
Constants	2	3	4	5	6
A	28	22	18	16	16
B			0.7		
C			1		

Table 2: Fitting constants computed for every polynomial degree in 2D.

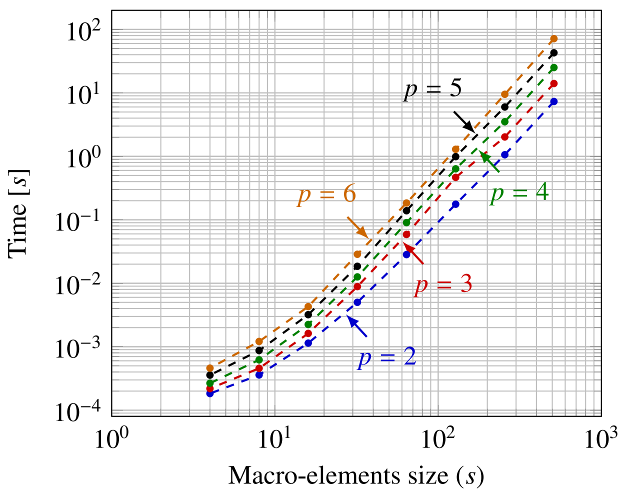
Constant A captures the total number of FLOPs employed by MUMPS to reduce the system matrices and to form the Schur complements of each macro-element [55]. In particular, A accounts for the constant accompanying the leading term of the LU factorization cost performed by LAPACK, and the contribution of the matrix-matrix and matrix-vector multiplication and subtraction required to form the Schur matrices. For small p, the contribution of forming those matrices is significant compared with the cost of LU factorization, influencing the value of constant A. However, the contribution of those operations becomes smaller as the polynomial degree grows ($p >> 1$). Constant B corrects the overestimation in the number of operations of the truncated Gaussian elimination that occurs at some non-zero entries of the skeleton mesh. This overestimate is related to the fact that θ_{pre} estimates the cost of preconditioning by assuming a periodic mesh. Thus, θ_{pre} counts additional operations at some non-zero entries. Finally, θ_{it} exactly predicts the number of FLOPs required per iteration by the iterative solver to converge. Therefore, constant C is one. We take N_{ite} equal to the number of iterations that results from the numerical experiments.

6.3. Cost of static condensation

Figure 10 illustrates the number of FLOPs (left) as well as the average computational time (right) required to eliminate the internal degrees of freedom for various macro-element sizes.



(a) Number of FLOPs.

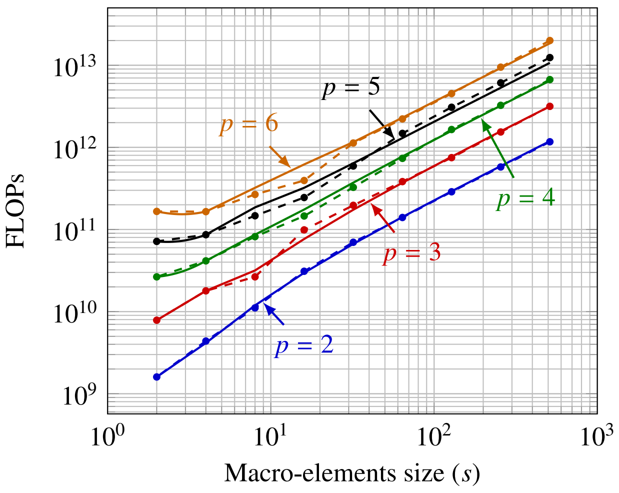


(b) Computational time.

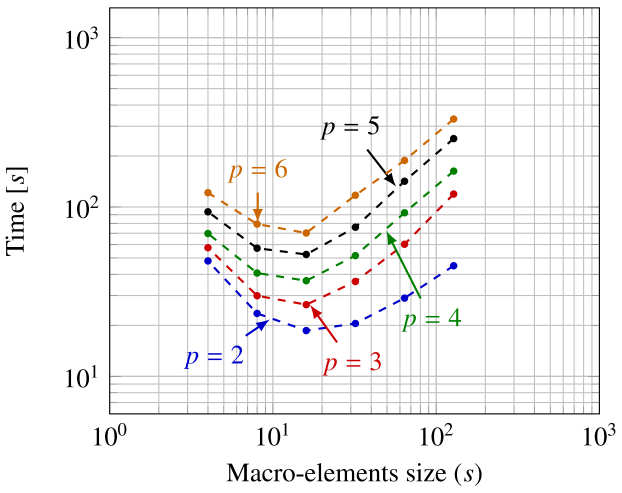
Figure 10: Number of FLOPs (left) and computational time (right) required to eliminate the interior degrees of freedom in a single macro-element. The solid lines ($-$) correspond to the theoretical estimates, and the dashed lines with rounded markers ($-o-$) represent the numerical results.

When the macro-element size is large ($s >> 1$), the computational time required to eliminate the internal degrees of freedom mainly consists of the time spent in performing floating point operations (FLOPs). For instance, when we increase the macro-element size from 64^2 to 128^2, the computational time (Figure 10b) grows linearly with the number of FLOPs (Figure 10a). However, when the macro-element size becomes closer to one ($s \to 1$), many other factors affect the computational cost, e.g., the bandwidth limit of global memory access [56]. Indeed, the procedure of computing LU factors of multiple small sparse systems becomes memory bound. Thus, in such limit, the cost of memory access becomes dominant, increasing the factorization time as shown in Figure 10b.

The cost to build the skeleton system (perform the static condensation) is equal to the sum of the cost of performing the partial factorization over *all* the macro-elements. Figure 11 illustrates the number of FLOPs (left) and computational time (right) required to perform the static condensation in *all* the macro-elements.



(a) Number of FLOPs.



(b) Computational time.

Figure 11: Number of FLOPs (left) and computational time (right) required to eliminate the interior degrees of freedom in *all* the macro-elements for the 2D Poisson problem with a mesh size of $N_e = 2048^2$ elements and polynomial degrees from 2 to 6. The solid lines (−) correspond to the theoretical estimates, and the dashed lines with rounded markers ($-o-$) represent the numerical results.

The rate of change of the number of FLOPs required to eliminate the interior degrees of freedom in *all* macro-element with respect to those on the element interfaces is

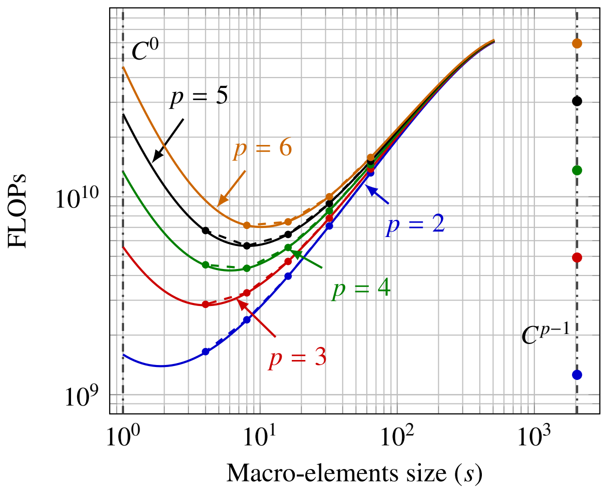
$$O\left(\frac{(N_{m-e}\,\theta_{SC})_{\ell}}{(N_{m-e}\,\theta_{SC})_{\ell+1}}\right) \sim O\left(\left(\frac{(n/s)^2_{\ell}}{(n/s)^2_{\ell+1}}\right)\left(\frac{(N^{\circ}_{dof})^{\beta}_{\ell}}{(N^{\circ}_{dof})^{\beta}_{\ell+1}}\right)\right) = O\left(\left(\frac{s^2_{\ell+1}}{s^2_{\ell}}\right)\left(\frac{(s+p-2)^{2\beta}_{\ell}}{(s+p-2)^{2\beta}_{\ell+1}}\right)\right), \tag{19}$$

where ℓ refers to the ℓ-th level of partition. For macro-element sizes larger than the polynomial degree ($s >> p$), $\beta = 3/2$ (sparse matrix) and the rate of change becomes approximately $O(s_{\ell}/s_{\ell+1})$. This explains the linear reduction of the number of FLOPs as the macro-elements become smaller. Moreover, as soon as the macro-element size becomes of the same order of the polynomial degree ($s \sim p$), β becomes three (dense matrix) and the rate of change starts to behaves approximately as $O\left(s^2_{\ell+1}/s^2_{\ell}\right)$. This explains the growth in the number of FLOPs observed when the macro-element size is close to one (Figure 11). Also, we can notice that the rates of change of the static condensation cost do not depend on any particular type of macro-element size. Therefore, we expect that the same behavior remains for cases with macro-element sizes different than a power of two.

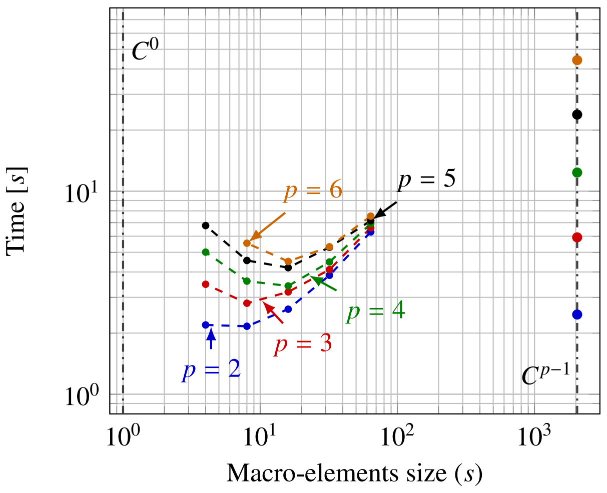
In addition to the numerical factorization, the direct methods perform an analysis phase. This phase involves the reordering of the matrix as well as additional operations to allocate memory. Assuming that these operations can be computed once and then reused by *all* macro-elements, we considered this cost as L.O.T. and we do not include it in the total computational cost.

6.4. *Cost of preconditioner set-up*

Before solving the skeleton system, we use the Incomplete LU factorization strategy ILU(0) to build the preconditioner matrix. Figure 12 illustrates the number of FLOPs as well as the computational time required for this step.



(a) Number of FLOPs.



(b) Computational time.

Figure 12: Number of FLOPs (left) and computational time (right) required to build the preconditioner matrix P for the reduced system when solving the 2D Poisson model problem with a mesh size of $N_e = 2048^2$ elements and polynomial degrees from 2 to 6. The solid lines ($-$) correspond to the theoretical estimates, and the dashed lines with rounded markers ($-o-$) represent the numerical results.

The cost to compute the preconditioner matrix becomes almost independent of p for cases with macro-element sizes larger than the polynomial degree ($s >> p$). For those cases, $\hat{p} \approx s$ and therefore θ_{pre} mostly depends on s, given by Equation 12.

We observe a reduction in the cost to build the preconditioner only when using discretizations with polynomial degrees higher than two ($p > 2$). For instance, the case with 2048^2 elements and $p = 2$ involves no reduction in the cost, but the case with $p = 6$ involves a reduction factor of approximately 13 in both the number of FLOPs and computational time.

6.5. *Cost of solving the reduced system*

The cost to solve the skeleton system depends on the number of non-zero entries of the matrix A_{skl}, as well as the number of iterations required to converge within the desired tolerance $\xi = 2\|e_D|_{IGA}\|_1$. Table 3 illustrates the value of $\|e_D|_{IGA}\|_1$ for all polynomial degrees and a mesh size of 2048^2 elements.

Polynomial degree p				
2	3	4	5	6
1.95E-07	2.88E-08	1.59E-08	1.30E-08	1.08E-08

Table 3: H^1-norm of the discretization error (e_D) resulting from approximating the model problem solution with C^{p-1} IGA for a mesh size of $N_e = 2048^2$ elements and polynomial degrees from 2 to 6.

Table 4 provides the number of non-zero entries (nnz) of matrices A and A_{skl}. The number of non-zero entries of matrix A_{skl} (from the second to the last row in the table) is lower than that of matrix A (first row in the table). Matrix A_{skl} has approximately $7(s + p)$ non-zero entries per column instead of the $(2p + 1)^2$ in matrix A. This implies that, in most cases, matrix A_{skl} is denser than matrix A. However, matrix A_{skl} is significantly smaller than A. This results in a lower total number of non-zero entries.

s	Polynomial degree p				
	2	3	4	5	6
2048 (IGA)	1.05E+08	2.06E+08	3.40E+08	5.09E+08	7.11E+08
64	6.06E+07	6.25E+07	6.44E+07	6.63E+07	6.38E+07
32	6.21E+07	6.59E+07	6.98E+07	7.39E+07	7.81E+07
16	6.52E+07	7.32E+07	8.16E+07	9.05E+07	9.99E+07
8	7.19E+07	8.91E+07	1.08E+08	1.29E+08	1.52E+08

Table 4: Number of non-zero entries of the 2D skeleton system used to solve the 2D Poisson model problem with a mesh size of $N_e = 2048^2$ elements and polynomial degrees from 2 to 6.

Table 5 shows the number of iterations that the CG iterative solver preconditioned with ILU(0) requires to converge within the desired tolerance. The hybrid strategy reduces the number of iterations in most of the cases. This is a result of reducing the system by eliminating the degrees of freedom interior to the macro-elements. In the skeleton problem, the macro-elements communicate only to the neighboring macro-elements. Therefore, the number of iterations required to solve the system depends on the size of each macro-elements. This explains the growth in the number of iterations that we observe when the macro-element size reduces (the number of macro-elements increases).

s	Polynomial degree p					size of A (first row) and A_{skl} (remaining rows)				
	2	3	4	5	6	2	3	4	5	6
2048 (IGA)	194	222	167	118	78	4202500^2	4206601^2	4210704^2	4214809^2	4218916^2
64	37	38	47	46	52	136257^2	138369^2	140481^2	142593^2	144705^2
32	57	56	53	57	54	270465^2	278785^2	287105^2	295425^2	303745^2
16	84	89	90	80	65	545025^2	578049^2	611073^2	644097^2	677121^2
8	106	150	150	150	106	1118721^2	1250305^2	1381889^2	1513473^2	1645057^2

Table 5: Number of iterations required by the preconditioned iterative solver CG+ILU(0) to converge, when solving the 2D Poisson model problem with a mesh size of $N_e = 2048^2$ elements and polynomial degrees from 2 to 6. The numbers of iterations provided here are obtained from PETSc measurements.

Tables 6 and 7 show, respectively, the number of FLOPs and the computational time required to solve the reduced system using the preconditioned iterative solver CG+ILU(0). The reduction in the cost observed when using the hybrid strategy is explained by both, the lower number of non-zero entries that the skeleton matrix A_{skl} has, and the reduction in the number of iterations due to performing static condensation at the level of the macro-elements.

s	Polynomial degree p				
	2	3	4	5	6
2048 (IGA)	8.98E+10	1.92E+11	2.35E+11	2.46E+11	2.26E+11
64	9.14E+09	9.67E+09	1.23E+10	1.24E+10	1.44E+10
32	1.44E+10	1.50E+10	1.51E+10	1.72E+10	1.72E+10
16	2.25E+10	2.67E+10	3.01E+10	2.97E+10	2.66E+10
8	3.18E+10	5.55E+10	6.72E+10	7.99E+10	6.64E+10

Table 6: Number of FLOPs required to solve the 2D skeleton system using the CG+ILU(0) solver, when solving the 2D Poisson model problem with a mesh size of $N_e = 2048^2$ elements and polynomial degrees from 2 to 6. The numbers of FLOPs provided here are obtained from PETSc measurements.

s	Polynomial degree p				
	2	3	4	5	6
2048 (IGA)	63.14	115.10	137.84	149.72	132.72
64	5.94	6.24	7.96	8.029	9.31
32	9.24	9.58	9.57	10.91	11.04
16	14.30	16.89	18.92	18.68	16.71
8	19.59	34.25	41.30	50.74	42.13

Table 7: Computational time (in seconds) required to solve the 2D skeleton system using the CG+ILU(0) solver, when solving the 2D Poisson model problem with a mesh size of $N_e = 2048^2$ elements and polynomial degrees from 2 to 6.

6.6. *Total cost*

The total cost required to solve the 2D model problem comprehends the costs of static condensation, the preconditioner construction and the computation of the skeleton system solution. Tables 8 and 9 show, respectively, the number of FLOPs (θ) and the computational time (τ) required by each of the three steps, and the total cost required to solve the 2D model problem.

p	s	θ_{SC}	θ_{pre}	θ_{it}	θ
2	2048 (IGA)	***	1.26E+09	8.98E+010	9.11E+10
	64	1.40E+11	1.32E+10	9.14E+009	1.62E+11
	32	6.99E+10	7.12E+09	1.44E+010	9.14E+10
	16	3.10E+10	3.97E+09	2.25E+010	5.75E+10
	8	1.11E+10	2.39E+09	3.18E+010	4.53E+10
3	2048 (IGA)	***	4.93E+09	1.92E+11	1.97E+11
	64	3.83E+11	1.38E+10	9.67E+09	4.07E+11
	32	1.97E+11	7.79E+09	1.50E+10	2.19E+11
	16	9.89E+10	4.71E+09	2.67E+10	1.30E+11
	8	2.65E+10	3.27E+09	5.55E+10	8.52E+10
4	2048 (IGA)	***	1.36E+10	2.35E+11	2.49E+11
	64	7.35E+11	1.45E+10	1.23E+10	7.62E+11
	32	3.26E+11	8.49E+09	1.51E+10	3.50E+11
	16	1.46E+11	5.54E+09	3.01E+10	1.81E+11
	8	8.18E+10	4.35E+09	6.72E+10	1.53E+11
5	2048 (IGA)	***	3.04E+10	2.46E+11	2.76E+11
	64	1.48E+12	1.51E+10	1.24E+10	1.50E+12
	32	5.93E+11	9.24E+09	1.72E+10	6.19E+11
	16	2.45E+11	6.45E+09	2.97E+10	2.81E+11
	8	1.47E+11	5.65E+09	7.99E+10	2.33E+11
6	2048 (IGA)	***	5.95E+10	2.26E+11	2.86E+11
	64	2.22E+12	1.58E+10	1.44E+10	2.25E+12
	32	1.13E+12	1.00E+10	1.72E+10	1.16E+12
	16	3.95E+11	7.47E+09	2.66E+10	4.29E+11
	8	2.67E+11	7.17E+09	6.64E+10	3.40E+11

Table 8: Number of FLOPs required to perform static condensation (θ_{SC}), construct the preconditioner (θ_{pre}), compute the skeleton system solution (θ_{it}) and solve the 2D model problem (θ) with a mesh size of $N_e = 2048^2$ elements and polynomial degrees from 2 to 6.

p	s	τ_{SC}	τ_{pre}	τ_{it}	τ
2	2048 (IGA)	***	2.45	63.13	65.58
	64	27.18	6.45	5.94	39.57
	32	19.86	4.06	9.24	33.18
	16	18.40	2.64	14.30	35.34
	8	23.33	2.23	19.59	45.15
3	2048 (IGA)	***	5.92	115.99	121.92
	64	49.26	6.60	6.24	62.10
	32	34.78	4.18	9.58	48.54
	16	26.26	3.33	16.89	46.49
	8	32.18	3.00	34.24	69.43
4	2048 (IGA)	***	12.36	137.84	150.20
	64	80.49	6.92	7.96	95.37
	32	49.79	4.47	9.57	63.84
	16	35.88	3.64	18.92	58.45
	8	41.46	3.63	41.30	86.39
5	2048 (IGA)	***	23.97	149.72	173.69
	64	133.13	7.16	8.03	148.32
	32	75.20	5.29	10.90	91.41
	16	50.37	4.19	18.68	73.24
	8	58.69	4.76	50.74	114.20
6	2048 (IGA)	***	44.21	132.72	176.94
	64	188.05	7.54	9.31	204.89
	32	117.35	5.32	11.04	133.71
	16	70.50	4.51	16.71	91.73
	8	79.05	5.54	42.13	126.72

Table 9: Computational time (in seconds) required to perform static condensation (τ_{SC}), construct the preconditioner (τ_{pre}), compute the skeleton system solution (τ_{it}) and solve the 2D model problem (τ) with a mesh size of $N_e = 2048^2$ elements and polynomial degrees from 2 to 6.

The total number of FLOPs (θ) shows a similar behavior than θ_{SC} when the size of the macro-elements decreases. Moreover, no increment in the value of θ is observed even with the growing contribution of θ_{it}. This implies that the predominant factor contributing to the total cost is the static condensation. This is best observed in cases with a large polynomial degree (Table 8). The use of the hybrid method slightly reduces the total number of FLOPs (θ). In particular, when using a polynomial degree $p = 3$.

In terms of computational time, the static condensation is the factor that contributes most to the total cost. In most of the cases, the discretization that delivers the lowest computational time is the one that involves the minimum cost in the static condensation. Moreover, τ_{it} has a significant contribution to the total time cost. This contribution is growing as the macro-elements size decreases. The reduction of the computational time obtained by comparing the best computational time of rIGA with the IGA computational time when using the hybrid solver is approximately a factor of two, independently of the polynomial degree.

When the problem size increases to 4096^2, both, the number of FLOPs and the total computational time show a similar behavior to the problem with mesh size of 2048^2 (Table 10). Furthermore, as the problem size increases to 4096^2, the reduction factor of the total computational time increases and becomes slightly above three. This is a result of the decrease of the iterative solver computational time. For sufficiently large problems, the reduction factor associated to the iterative solver will determine the total reduction factor of rIGA with respect to IGA, since that is the only term scaling non-linearly with respect to the problem size.

p	s	θ_{SC}	θ_{pre}	θ_{it}	θ	τ_{SC}	τ_{pre}	τ_{it}	τ
2	4096 (IGA)	***	5.05E+09	7.04E+011	7.09E+11	***	9.77	492.65	502.42
	128	1.15E+12	1.03E+11	4.08E+010	1.29E+12	154.79	45.58	26.20	226.55
	64	5.58E+11	5.43E+10	6.76E+010	6.80E+11	108.09	25.74	43.42	177.25
	32	2.80E+11	2.89E+10	1.11E+011	4.19E+11	78.55	15.45	70.84	164.84
	16	1.24E+11	1.60E+10	1.62E+011	3.02E+11	73.67	11.13	103.46	188.26
3	4096 (IGA)	***	1.97E+10	1.30E+012	1.32E+12	***	23.93	780.50	804.43
	128	3.00E+12	1.06E+11	3.85E+010	3.15E+12	297.31	46.59	24.69	368.60
	64	1.53E+12	5.68E+10	5.76E+010	1.65E+12	196.75	26.82	37.06	260.63
	32	7.86E+11	3.16E+10	1.04E+011	9.22E+11	137.44	17.97	65.92	221.32
	16	3.95E+11	1.90E+10	1.95E+011	6.09E+11	104.70	12.86	123.77	241.32

Table 10: Number of FLOPs and Computational time (in seconds) required to perform static condensation (θ/τ_{SC}), construct the preconditioner (θ/τ_{pre}), compute the skeleton system solution (θ/τ_{it}) and solve the 2D model problem (θ/τ) with a mesh size of $N_e = 4096^2$ elements and polynomial degrees $p = 2$ and $p = 3$.

Having a reduction factor greater in computational time than in the number of FLOPs can be an effect of the domain partitioning. The partitioning improves the performance of the transfer of data. This benefits the cache and results in a speed-up of the computational time. Large sparse systems degrade the access to memory and transfer data performance. In those cases, the machine has more probabilities to fail in optimally filling the cache. This results in delay operations until the required data is transferred. The division of the problem in sub-problems (macro-elements) permits to improve the performance of the cache-based machines. In particular, the solution of all the small problems results in a smaller global system that better exploit the data locality, and benefits the filling of the cache [56].

6.7. *Hybrid solver strategy in 3D*

The extension of the aforementioned rIGA hybrid solver to 3D is straightforward, since no significant modifications in the implementation are required to build the solver. Unfortunately, in 3D, the number of non-zero entries of matrix A_{skl} increases with respect to those of A (see Table 12), as opposed to what we encounter in the 2D case (see Table 11). This occurs because the number of non-zero entries per column of matrix A_{skl} is approximately $11(s + p)^2$. This value is larger than the number of non-zero entries per column of matrix A ($(2p + 1)^3$). Moreover, for $s > p$ (the most important scenario), nnz becomes gigantic. For instance, in a problem discretized with 2048^3 elements and a polynomial degree $p = 2$, using macro-elements of size 32^3, we obtain in average 12085 non-zero entries per column (unknown). This is approximately two orders of magnitude larger than the 125 non-zero entries per column observed when using traditional IGA.

The large value of nnz implies that the cost of a single CG iteration to approximate the solution of the skeleton system is more expensive than a single CG iteration to approximate the solution of the full problem discretized with C^{p-1} IGA and no static condensation. Therefore, the hybrid solver strategy combined with rIGA is unsuitable to solve 3D problems. An alternative approach is needed.

7. Conclusions

We extend the refined Isogeometric Analysis (rIGA) to solve problems governed by partial differential equations using a Conjugate Gradient iterative method preconditioned with Incomplete LU factorization ILU(0). The strategy we analyze splits the mesh into sub-domains and eliminates in each sub-domain the internal degrees of freedom. This reduces the size of the system, which now is only composed of the degrees of freedom located along the boundaries of all the sub-domains. We then turn to an iterative method to solve the reduced system. Lastly, a backward substitution is performed to recover the eliminated degrees of freedom and obtain the solution of the original system. To illustrate the impact of mesh partitioning on the computation of the solution of a linear problem, we report the floating point operations (FLOPs) and computational times required to solve a linear system with structured meshes and uniform polynomial degrees.

p	s	size (number of unknowns)			nnz		
		A	A_{skl}	A/A_{skl}	A	A_{skl}	A/A_{skl}
2	2048 (IGA)	4.20E+06	–	–	1.05E+08	–	–
	512	–	2.05E+04	204.95	–	6.10E+07	1.72
	128	–	6.99E+04	60.10	–	6.00E+07	1.75
	32	–	2.70E+05	15.54	–	6.21E+07	1.69
	8	–	1.12E+06	3.76	–	7.19E+07	1.46
	2	–	5.25E+06	0.80	–	1.19E+08	0.88
6	2048 (IGA)	4.22E+06	–	–	7.13E+08	–	–
	512	–	2.07E+04	204.16	–	6.20E+07	11.50
	128	–	7.21E+04	58.52	–	6.38E+07	11.17
	32	–	3.04E+05	13.89	–	7.81E+07	9.13
	8	–	1.65E+06	2.56	–	1.52E+08	4.70
	2	–	1.36E+07	0.31	–	6.89E+08	1.03

Table 11: Results comparison between the number of non-zero entries of the original matrix A and that of the skeleton matrix A_{skl} of the 2D Poisson problem with a mesh size of $N_e = 2048^2$ elements and polynomial degrees $p = 2$ and $p = 6$.

p	s	size (number of unknowns)			nnz		
		A	A_{skl}	A/A_{skl}	A	A_{skl}	A/A_{skl}
2	2048 (IGA)	8.62E+09	–	–	1.08E+12	–	–
	512	–	6.31E+07	136.600	–	1.50E+14	0.00720
	128	–	2.16E+08	39.942	–	3.75E+13	0.02868
	32	–	8.44E+08	10.206	–	1.02E+13	0.10594
	8	–	3.66E+09	2.356	–	3.50E+12	0.30777
	2	–	2.04E+10	0.422	–	2.59E+12	0.41518
6	2048 (IGA)	8.67E+09	–	–	1.90E+13	–	–
	512	–	6.41E+07	135.28	–	1.54E+14	0.12
	128	–	2.29E+08	37.79	–	4.24E+13	0.45
	32	–	1.06E+09	8.14	–	1.61E+13	1.18
	8	–	7.90E+09	1.10	–	1.54E+13	1.24
	2	–	1.37E+11	0.06	–	8.11E+13	0.23

Table 12: Results comparison between the number of non-zero entries of the original matrix A and that of the skeleton matrix A_{skl} of the 3D Poisson problem with a mesh size of $N_e = 2048^3$ elements and polynomial degrees $p = 2$ and $p = 6$.

The iterative (hybrid) solver-based discretization delivers moderate savings in terms of computational time when solving the 2D Poisson model problem. These savings are smaller than those obtained using rIGA with direct solvers. For instance, in a mesh with 2048^2 elements and polynomial degree $p = 5$, the hybrid solver strategy solves the model problem approximately 2.37 times faster than with C^{p-1} IGA, while with direct solvers, rIGA reduces the computational time by a factor of approximately 22 with respect to C^{p-1} IGA [1].

The cost of all operations is linear with respect to the number of elements, except for the case of the number of iterations, which grows as $n^{\alpha \cdot d}$ with $\alpha > 1$. For the 2D case, this increases the computational savings of rIGA as compared to IGA up to an asymptotic limit. For instance, the model problem discretized with a mesh of size 4096^2 and polynomial degree $p = 3$ deliver a reduction factor of approximately 3, which is larger than when using a mesh size of 2048^2 elements (reduction factor of approximately 2).

Comparing the performance of the iterative solver when applied to rIGA discretizations vs. when applied to C^{p-1} IGA, we observe a larger reduction in terms of time than in terms of the number of FLOPs. This may be a result of the domain partitioning, which simplifies the data transfers with respect to the bandwidth limitations of the machine. Large sparse systems (as in IGA) degrade the performance of the memory access and the data transmission. In those cases, the machine often fails to optimally fill the cache memory resulting in a delay of some operations until the required data is transferred. The division of the problem in subproblems (macro-elements) improves the performance of the cache-based machines. The solution of the subproblems delivers a smaller global system that not only better exploits the data locality but also benefits in the filling of the cache [56]. However, when the macro-elements are small ($s \sim p$), the cost to access the memory becomes significant with respect to the cost to operate on the data (eliminate degrees of freedom) [56]. This memory access impacts the scaling of the computational time of the static condensation (Figure 11). Once the size of the macro-elements is large enough ($s >> p$), data operations dominate the timings. At this point, the reduction factor of computational time corresponds to that of the number of FLOPs.

In 3D, the number of non-zero entries of the rIGA skeleton matrix is larger than that of the IGA matrix. This results in a significant increment in the relative cost of matrix-vector multiplication. Thus, rIGA combined with the hybrid solver strategy delivers no savings in the computational cost.

We developed a hybrid solver based on a CG iterative solver preconditioned with ILU(0). Nevertheless, rIGA could be combined with other iterative solvers and/or preconditioners. As future work, we will focus on using the hybrid solver strategy with CG and preconditioned with the Balancing Domain Decomposition by Constrains (BDDC) strategy [57–59]. By using this preconditioner technique, we will be able to represent the skeleton system in a reduced form using fewer non-zero entries. We shall also analyze rIGA performance when solving non-elliptic partial differential equations.

Acknowledgement

David Pardo has received funding from the Project of the Spanish Ministry of Economy and Competitiveness with reference MTM2016-76329-R (AEI/FEDER, EU), and MTM2016-81697-ERC/AEI, the BCAM "Severo Ocho" accreditation of excellence SEV-2013-0323, and the Basque Government through the BERC 2014-2017 program and the Consolidated Research Group Grant IT649-13 on "Mathematical Modeling, Simulation, and Industrial Applications (M2SI)". This publication was also made possible in part by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 644602, the CSIRO Professorial Chair in Computational Geoscience at Curtin University, the Deep Earth Imaging Enterprise Future Science Platforms of the Commonwealth Scientific Industrial Research Organisation, CSIRO, of Australia, the Mega-grant of the Russian Federation Government (N14.Y26.31.0013) and the Curtin Institute for Computation. The J. Tinsley Oden Faculty Fellowship Research Program at the Institute for Computational Engineering and Sciences (ICES) of the University of Texas at Austin has partially supported the visits of VMC to ICES. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper.

References

[1] D. Garcia, D. Pardo, L. Dalcin, M. Paszyski, N. Collier, VM. Calo, The value of continuity: Refined isogeometric analysis and fast direct solvers, Computer Methods in Applied Mechanics and Engineering 316 (2017) 586–605, special Issue on Isogeometric Analysis: Progress and Challenges.

[2] TJR. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (3941) (2005) 4135 – 4195.

[3] H. Gómez, VM. Calo, Y. Bazilevs, TJR. Hughes, Isogeometric analysis of the Cahn-Hilliard phase-field model, Computer Methods in Applied Mechanics and Engineering 197 (4950) (2008) 4333 – 4352.

[4] P. A. Vignal, N. Collier, VM. Calo, Phase Field Modeling Using PetIGA, Procedia Computer Science 18 (2013) 1614 – 1623.

[5] P. Vignal, L. Dalcin, D. L. Brown, N. Collier, VM. Calo, An energy-stable convex splitting for the phase-field crystal equation, Computers & Structures 158 (2015) 355 – 368.

[6] Y. Bazilevs, VM. Calo, Y. Zhang, TJR. Hughes, Isogeometric Fluid-structure Interaction Analysis with Applications to Arterial Blood Flow, Computational Mechanics 38 (4-5) (2006) 310–322.

[7] Y. Bazilevs, VM. Calo, TJR. Hughes, Y. Zhang, Isogeometric fluid-structure interaction: theory, algorithms, and computations, Computational Mechanics 43 (1) (2008) 3–37.

[8] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, TJR. Hughes, An immersogeometric variational framework for fluid-structure interaction: Application to bioprosthetic heart valves, Computer Methods in Applied Mechanics and Engineering 284 (2015) 1005 – 1053.

[9] Y. Bazilevs, VM. Calo, J. A. Cottrell, TJR. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Computer Methods in Applied Mechanics and Engineering 197 (1-4) (2007) 173 – 201.

[10] Y. Bazilevs, C. Michler, VM. Calo, TJR. Hughes, Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes, Computer Methods in Applied Mechanics and Engineering 199 (13-16) (2010) 780 – 790.

[11] K. Chang, TJR. Hughes, VM. Calo, Isogeometric variational multiscale large-eddy simulation of fully-developed turbulent flow over a wavy wall, Computers & Fluids 68 (2012) 94–104.

[12] I. Akkerman, Y. Bazilevs, VM. Calo, TJR. Hughes, S. Hulshoff, The role of continuity in residual-based variational multiscale modeling of turbulence, Computational Mechanics 41 (3) (2008) 371–378.

[13] Y. Bazilevs, I. Akkerman, Large eddy simulation of turbulent Taylor-Couette flow using isogeometric analysis and the residual-based variational multiscale method, Journal of Computational Physics 229 (9) (2010) 3402 – 3414.

[14] Y. G. Motlagh, H. T. Ahn, TJR. Hughes, VM. Calo, Simulation of laminar and turbulent concentric pipe flows with the isogeometric variational multiscale method, Computers & Fluids 71 (2013) 146 – 155.

[15] P. N. Nielsen, A. R. Gersborg, J. Gravesen, N. L. Pedersen, Discretizations in isogeometric analysis of Navier-Stokes flow, Computer Methods in Applied Mechanics and Engineering 200 (45-46) (2011) 3242 – 3253.

[16] A. Tagliabue, L. Dedé, A. Quarteroni, Isogeometric Analysis and error estimates for high order partial differential equations in fluid dynamics, Computers & Fluids 102 (2014) 277 – 303.

[17] V. P. Nguyen, C. Anitescu, S. P. Bordas, T. Rabczuk, Isogeometric analysis: An overview and computer implementation aspects, Mathematics and Computers in Simulation 117 (2015) 89 – 116.

[18] J. A. Evans, Y. Bazilevs, I. Babuka, TJR. Hughes, n-widths, supinfs, and optimality ratios for the k-version of the isogeometric finite element method, Computer Methods in Applied Mechanics and Engineering 198 (2126) (2009) 1726 – 1741, advances in Simulation-Based Engineering Sciences Honoring J. Tinsley Oden.

[19] J. A. Cottrell, TJR. Hughes, A. Reali, Studies of refinement and continuity in isogeometric structural analysis, Computer Methods in Applied Mechanics and Engineering 196 (4144) (2007) 4160 – 4183.

[20] J. A. Cottrell, A. Reali, Y. Bazilevs, TJR. Hughes, Isogeometric analysis of structural vibrations, Computer Methods in Applied Mechanics and Engineering 195 (41-43) (2006) 5257 – 5296.

[21] F. Auricchio, L. B. da Veiga, A. Buffa, C. Lovadina, A. Reali, G. Sangalli, A fully locking-free isogeometric approach for plane linear elasticity problems: A stream function formulation, Computer Methods in Applied Mechanics and Engineering 197 (1-4) (2007) 160 – 172.

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

[23] H. Gómez, TJR. Hughes, X. Nogueira, VM. Calo, Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations, Computer Methods in Applied Mechanics and Engineering 199 (25-28) (2010) 1828 – 1840.

[24] A. Buffa, C. de Falco, G. Sangalli, Isogeometric analysis: Stable elements for the 2D Stokes equation, International Journal for Numerical Methods in Fluids 65 (11-12) (2011) 1407–1422.

[25] J. A. Evans, TJR. Hughes, Isogeometric divergence-conforming B-splines for the unsteady Navier-Stokes equations, Journal of Computational Physics 241 (2013) 141 – 167.

[26] M.-C. Hsu, Y. Bazilevs, VM. Calo, T. Tezduyar, TJR. Hughes, Improving stability of stabilized and multiscale formulations in flow simulations at small time steps, Computer Methods in Applied Mechanics and Engineering 199 (13-16) (2010) 828–840.

[27] Y. Bazilevs, C. Michler, VM. Calo, TJR. Hughes, Weak Dirichlet boundary conditions for wall-bounded turbulent flows, Computer Methods in Applied Mechanics and Engineering 196 (49-52) (2007) 4853–4862.

[28] Y. Zhang, Y. Bazilevs, S. Goswami, C. L. Bajaj, TJR. Hughes, Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow, Computer Methods in Applied Mechanics and Engineering 196 (29-30) (2007) 2943 – 2959.

[29] P. Vignal, A. Sarmiento, A. M. Côrtes, L. Dalcin, VM. Calo, Coupling Navier-Stokes and Cahn-Hilliard equations in a two-dimensional annular flow configuration, Procedia Computer Science 51 (2015) 934 – 943.

[30] P. Vignal, L. Dalcin, N. Collier, VM. Calo, Modeling Phase-transitions Using a High-performance, Isogeometric Analysis Framework, Procedia Computer Science 29 (2014) 980 – 990.

[31] Y. Bazilevs, J. Gohean, TJR. Hughes, R. Moser, Y. Zhang, Patient-specific isogeometric fluidstructure interaction analysis of thoracic aortic blood flow due to implantation of the Jarvik 2000 left ventricular assist device, Computer Methods in Applied Mechanics and Engineering 198 (45-46) (2009) 3534 – 3550.

[32] S. S. Hossain, S. F. A. Hossainy, Y. Bazilevs, VM. Calo, TJR. Hughes, Mathematical modeling of coupled drug and drug-encapsulated nanoparticle transport in patient-specific coronary artery walls, Computational Mechanics 49 (2) (2011) 213–242.

[33] VM. Calo, N. F. Brasher, Y. Bazilevs, TJR. Hughes, Multiphysics model for blood flow and drug transport with application to patient-specific coronary artery flow, Computational Mechanics 43 (1) (2008) 161–177.

[34] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, VM. 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 (0) (2012) 353 – 361.

[35] N. Collier, L. Dalcin, VM. Calo, On the computational efficiency of isogeometric methods for smooth elliptic problems using direct solvers, International Journal for Numerical Methods in Engineering 100 (8) (2014) 620–632.

[36] N. Collier, L. Dalcin, D. Pardo, VM. Calo, The Cost of Continuity: Performance of Iterative Solvers on Isogeometric Finite Elements, SIAM Journal on Scientific Computing 35 (2) (2013) A767–A784.

[37] D. Pardo, M. Paszynski, N. Collier, J. Alvarez, L. Dalcin, VM. Calo, A survey on direct solvers for Galerkin methods, SeMA Journal 57 (1) (2012) 107–134.

[38] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[39] M. R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, Journal of research of the National Bureau of Standards 49 (1952) 409–436.

[40] J. E. Gentle, Iterative Methods for Sparse Linear Systems, Springer-Verlag, Berlin, Ge, 1998.

[41] D. Pardo, J. Álvarez-Aramberri, M. Paszynski, L. Dalcin, VM. Calo, Impact of element-level static condensation on iterative solver performance, Computers & Mathematics with Applications 70 (10) (2015) 2331 – 2341.

[42] L. Dalcin, N. Collier, P. Vignal, A. Côrtes, VM. Calo, PetIGA: A framework for high-performance isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 308 (2016) 151–181.

[43] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Web page (2016).
URL `http://www.mcs.anl.gov/petsc`

[44] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.7, Argonne National Laboratory (2016).
URL `http://www.mcs.anl.gov/petsc`

[45] VM. Calo, N. O. Collier, D. Pardo, M. R. Paszynski, Computational complexity and memory usage for multi-frontal direct solvers used in p finite element analysis, Procedia Computer Science 4 (0) (2011) 1854 – 1861.

[46] L. M. Bernal, VM. Calo, N. Collier, G. A. Espinosa, F. Fuentes, J. C. Mahecha, Isogeometric Analysis of Hyperelastic Materials Using PetIGA, Procedia Computer Science 18 (2013) 1604 – 1613.

[47] A. Côrtes, A. Coutinho, L. Dalcin, VM. Calo, Performance evaluation of block-diagonal preconditioners for the divergence-conforming B-spline discretization of the Stokes system, Journal of Computational Science 11 (2015) 123 – 136.

[48] L. F. R. Espath, A. F. Sarmiento, P. Vignal, B. O. N. Varga, A. M. A. Côrtes, L. Dalcin, VM. Calo, Energy exchange analysis in droplet dynamics via the Navier-Stokes-Cahn-Hilliard model 797 (2016) 389–430.

[49] L. Bernal, VM. Calo, N. Collier, G. Espinosa, F. Fuentes, J. Mahecha, Isogeometric analysis of hyperelastic materials using PetIGA, Procedia Computer Science 18 (2013) 1604 – 1613.

[50] A. Sarmiento, D. Garcia, L. Dalcin, N. Collier, V. Calo, Micropolar fluids using B-spline divergence conforming spaces, Procedia Computer Science 29 (2014) 991 – 1001.

[51] A. Sarmiento, A. Côrtes, D. Garcia, L. Dalcin, N. Collier, VM. Calo, PetIGA-MF: A multi-field high-performance toolbox for structure-preserving B-splines spaces, Journal of Computational Science 18 (2017) 117–131.

[52] L. F. R. Espath, A. F. Sarmiento, L. Dalcin, VM. Calo, On the thermodynamics of the Swift-Hohenberg theory, Continuum Mechanics and Thermodynamics (2017) 1–11.

[53] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, J. Koster, A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling, SIAM Journal on Matrix Analysis and Applications 23 (1) (2001) 15–41.

[54] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, S. Pralet, Hybrid scheduling for the parallel solution of linear systems, Parallel Computing 32 (2) (2006) 136 – 156.

[55] S. Blackford, J. Dongarra, LAPACK Working Note 41 "Installation Guide for LAPACK", Department of Computer Science, University of Tennessee.

[56] J. Dongarra, V. Eijkhout, H. Van der Vorst, An iterative solver benchmark, Scientific Programming 9 (4) (2001) 223–231. doi:10.1155/2001/527931.

[57] J. Mandel, Balancing domain decomposition, Communications in Numerical Methods in Engineering 9 (3) (1993) 233–241.

[58] J. Mandel, C. R. Dohrmann, Convergence of a balancing domain decomposition by constraints and energy minimization, Numerical Linear Algebra with Applications 10 (7) (2003) 639–659.

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