Parallel refined Isogeometric Analysis in 3D

Leszek Siwik[a], Maciej Woźniak[a], Victor Trujillo[b],
David Pardo[c,d,e], Victor Manuel Calo[f,g,h], Maciej Paszyński[a]

[a]*AGH University of Science and Technology*
Faculty of Computer Science, Electronics and Telecommunication
Department of Computer Science
al. A Mickiewicza 30, 30-059 Kraków, Poland
email: maciej.paszynski@agh.edu.pl, phone: +48-12-328-3314, fax: +48-12-617-5172
[b]*Pontifical Catholic University of Valparaiso*
Department of Mathematics
Valparaiso, Chile
[c]*The University of The Basque Country, Bilbao, Spain*
[d]*Basque Center for Applied Mathematics, Bilbao, Spain*
[e]*IKERBASQUE*
[f]*Chair in Computational Geoscience, Applied Geology Department, Western Australian School of Mines, Faculty of Science and Engineering, Curtin University, Perth, WA, Australia*
[g]*Mineral Resources, Commonwealth Scientific and Industrial Research Organization (CSIRO), Kensington, WA, Australia 6152*
[h]*Curtin Institute for Computation, Curtin University, Perth, WA, Australia 6845*

Abstract

We study three-dimensional isogeometric analysis (IGA) and the solution of the resulting system of linear equations via a direct solver. IGA uses highly continuous C^{p-1} basis functions, which provide multiple benefits in terms of stability and convergence properties. However, smooth basis significantly deteriorate the direct solver performance and its parallel scalability. As a partial remedy for this, refined Isogeometric Analysis (rIGA) method improves the sequential execution of direct solvers. The refinement strategy enriches traditional highly-continuous C^{p-1} IGA spaces by introducing low-continuity C^0-hyperplanes along the boundaries of certain pre-defined macro-elements. In this work, propose a solution strategy for rIGA for parallel distributed memory machines and compare the computational costs of solving rIGA vs IGA discretizations. We verify our estimates with parallel numerical experiments. Results show that the weak parallel scalability of the direct solver improves approximately by a factor of p^2 when considering rIGA discretizations rather

Preprint submitted to International Journal of Parallel and Distributed ComputingDecember 29, 2017

than highly-continuous IGA spaces.

Keywords: isogeometric analysis, direct solvers, parallel computing

---

1. Introduction

We focus on parallel three-dimensional isogeometric analysis (IGA) [15] Isogeometric analysis uses the same basis functions, for example B-splines or NURBS, for representing the geometry of the modeled objects, as well as for the numerical simulations performed using a Galerkin method.

For tensor product IGA discretizations, each geometrical component of the model object is mapped into a regular patch of elements, with basis functions defined as tensor products of one-dimensional B-splines or by Non-Uniform Ratonal B-splines (NURBS) [32], which in general result in non-tensor product basis. Adaptive grids often combine B-splines and T-spline basis functions [11]. Nevertheless, we only focus on regular patches of elements.

IGA has multiple applications such as shear deformable shell theory [12], phase field models for topology optimization [20, 21], phase separation simulations with possible application to cancer growth simulations, using either Cahn-Hilliard [26] or Navier-Stokes-Korteweg higher order models [27], wind turbine aerodynamics [31], incompressible hyper-elasticity [24], turbulent flow simulations [16], tumor growth [33], transport of drugs in cardiovascular applications [30] or the blood flow simulations itself [10, 9, 14]. Nevertheless, IGA presents some limitations. Namely, (a) non-trivial integration techniques need to be carefully designed in order to gain efficiency (see e.g. [7]) and (b) the cost of solving the resulting system of linear equations per unknown is significantly more expensive when using highly-continuous C^{p-1} IGA discretizations than when employing their C^0 traditional finite element counterparts [19, 18].

We refine the IGA space by adding C^0 separators along certain hyperplanes of the original IGA mesh. Thus, we reduce the solution cost by a factor of order $\mathcal{O}(p^2)$ for a sequential direct solver (see [19]). That is, we reduce the cost while increasing the problem size. The cost reduction is explained by the critical role that separators play on the direct solve performance since they "separate" (break) the system into smaller subsystems.

In this work, we extend the analysis of rIGA to parallel distributed memory machines. We show that the same separators that reduce the computational cost of the direct solver also benefit the domain-decomposition scheme since communication among different processors across a given separator diminishes with respect to that used in traditional IGA discretizations.

Another major advantage of rIGA is its simple implementation which only requires repeated konts at the selected separators in a tensor product grid. We first introduce notation as well as a description of how to easily implement rIGA spaces in Section 2. Section 3 analyzes the parallel scalability of rIGA and IGA and estimates the improved performance of using rIGA discretizations rather than IGA ones when computing in distributed memory parallel machines. Section 4 describes numerical experiments performed to verify the theoretical estimates. We conclude the paper in Section 5.

2. Parallel refined Isogeometric Analysis (rIGA) discretizations

To simplify the explanation, we only consider a model diffusion equation problem governed by

$$-\nabla \cdot K \nabla u = f \tag{1}$$

where $K = \begin{pmatrix} K_{11} & 0 \\ 0 & K_{22} \end{pmatrix}$ is the matrix of the material diffusion coefficients matrix, u is the concentration and f is the source. The above partial differential equation (PDE) is solved in a computational domain $\Omega = (0,1)^3$. We impose homogeneous Dirichlet boundary conditions

$$u = 0 \text{ on } \Gamma_D, \tag{2}$$

where $\Gamma_D = \partial\Omega$.

The weak variational formulation is obtained by taking the L^2-scalar product with functions $v \in H^1_{\Gamma_D}(\Omega) = \{v \in H^1(\Omega) : v|_{\Gamma_D} = 0\}$, integrating by parts, and imposing the Dirichlet boundary conditions:

$$\text{Find } u \in V = H^1_{\Gamma_D}(\Omega) \text{ such that} \tag{3}$$

$$b(v,u) = l(v), \forall v \in V, \tag{4}$$

where

$$b(v,u) = \int_\Omega K\nabla v \cdot \nabla u dx, \text{ and} \tag{5}$$

$$l(v) = \int_\Omega fv dx \tag{6}$$

The computational cost estimates associated with the direct solver performance obtained for problem (6) remain valid for other H^1 problems, since the non-zero pattern of the matrix is invariant.

In 3D, we utilize the basis functions defined as tensor products of one-dimensional B-splines defined by using knot vectors for each direction. We use a uniform grid spacing in each direction. We use open knot vectors, that is, we repeat a knot-point $p+1$ times at the beginning and at the end of the knot-vector. If no interior knot is repeated, the B-splines are C^{p-1} continuous across the entire mesh, in the direction defined by the knot-vector. rIGA introduces C^0 separators by repeating knot-points every *iblock*-elements. As described in [25], the repetition of the knots reduces the computational cost of sequential direct solvers significantly.

3. Solution and communication cost estimates

From the direct solver perspective, systems of linear equations arising from refined Isogeometric Analysis (rIGA) discretizations can be solved in two steps. In the first one, we perform static condensation of macro-elements, which are composed of a set of elements with C^{p-1} continuity basis functions on their interiors and C^0 separators on their boundaries. In the second step, we solve the remaining skeleton problem.

Having 2^s finite elements in 1D, we partition them into 2^r macro-elements, each of them having 2^{s-r} elements per spatial direction. In our simulations we use macro-elements $2^{s-r} = 8$, following the general conclusions of Figure 20 in [25], for quadratic and cubic B-splines over 32^3 and 64^3 elements mesh. For larger grids and higher B-splines orders, larger macro-elements further reduce the computational cost. For this number of elements within the macro-elements, the computational cost of static condensation is of lower order in comparison to the computational cost of the skeleton problem solution. Moreover, the static condensation scales linearly with the number of elements, when executed sequentially.

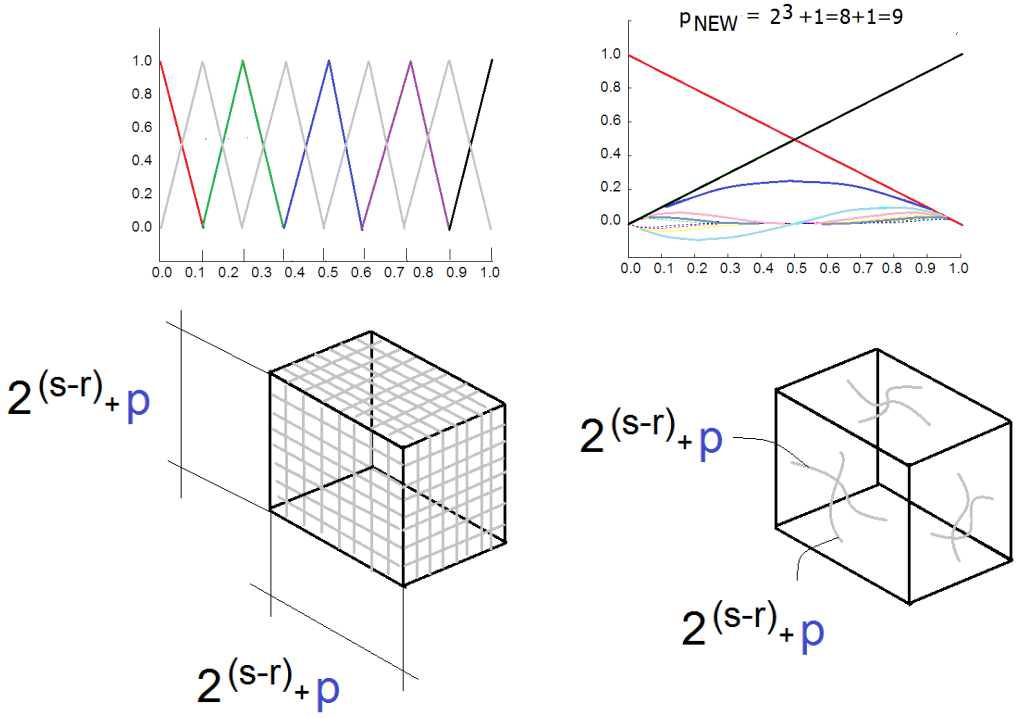

Figure 1: Equivalence between a macro-element with $2^3 = 8$ elements and linear B-splines (left panel) and a higher order element with for $p_{NEW} = 9$ (right panel).

The static condensation scales parfectly, since each macro-element can be processed independently. Thus, we focus only on the skeleton problem.

The non-zero structure of the skeleton problem is identical to that obtained from a discretization of a finite element method with the macro-element mesh and a polynomial order p^{NEW} after static condensation, with p^{NEW} being:

$$p^{NEW} = p + 2^{s-r} \tag{7}$$

per direction. This is because the number of unknowns on a face of a macro element is equal to $(2^{(s-r)} + p)^2$ where $2^{2(s-r)}$ is the number of elements on the face, and p is the underlying polynomial order. Thus, $2^{s-r} + p$ is the number of unknowns on the face per spatial direction. See Figure 1 for details.

The above observation implies that scalability properties of rIGA are those of a higher-order coarse finite element grid.

We analyzed the parallel scalability of highly-continuous IGA systems in [35], where we showed that the direct solution of IGA systems has similar parallel scalability properties to standard C^0 FE systems. Although the performance of the direct solution of IGA systems is significantly worse than that of FE systems for a fixed number of degrees of freedom. This occurs because, in the sequential version, IGA systems are more challenging to solve than FE ones due to the presence of thick separators on IGA discretizations (see [17]).

In addition to the above theoretical considerations, we know that different direct solver implementations exhibit distinct scalability properties, as shown in [34]. Therein, the authors consider specific finite differences and finite element systems and demonstrate via numerical experimentation that for those systems, the parallel scalability of solvers Watson Sparse Matrix Package (WSMP) [22] and MUMPS exhibit large discrepancies. In some cases, WSMP scales well for up to 30 times more processors than MUMPS is able. In view of these large scalability discrepancies based on the selected particular implementation, in here we avoid predicting theoretically the parallel scalability limit for a given matrix, since particular implementations behave differently. We rather focus on the impact that introducing C^0 separators to build rIGA discretizations has on the expected scalability of a direct solver. To do that, we analyze:

$$\frac{Time(IGA)}{Time(rIGA)} \tag{8}$$

We assume that our IGA/rIGA computations are executed on distributed memory parallel machine with infinite memory available on each computing node, and with the communication channels interconnecting all the nodes. These assumptions are justified for the problem sizes and number of processors we use in our experiments on Prometheus Linux cluster [36], a part of PL-grid infrastructure [13]. We denote the time of performing a single floating-point operation as t_{comp}, and the time of sending a floating-point number as t_{comm}.

The number of degrees of freedom n_{IGA} along one direction for an IGA discretization is equal to the number of 1D elements 2^s plus p additional B-splines on the boundary.

$$n_{IGA} = 2^s + p. \tag{9}$$

Let us also define the number of degrees of freedom along one axis for rIGA mesh (n_{rIGA}), which is the number of 1D macro-elements 2^r times the number of 1D elements per axis in a macro-element 2^{s-r} plus p additional B-splines on the boundary, plus the number of separators $2^r - 1$ times the number of knot repetition $(p-1)$ per separator [25].

$$n_{rIGA} = 2^r 2^{s-r} + p + (2^r - 1)(p-1) = \mathcal{O}(2^s + 2^r p). \tag{10}$$

Thus, the problem size of IGA is smaller than for rIGA, namely $n_{IGA} < n_{rIGA}$. However, the computational cost to factorize an rIGA system is smaller than the one for IGA, when the number of elements per dimension is kept constant. To see this, we observe that the most expensive part of the LU factorization of the three-dimensional problem is the solution of the dense top skeleton problem. The dimension of this problem is equal to the number of B-splines at the two-dimensional cross-section of the three-dimensional mesh. For highest-continuity IGA discretizations, this cross-section has a thickness of p (i.e., the interface has p functions with support over it). In the case of rIGA computations, the thickness is equal to 1 (i.e., as in standard FEA only one basis has support on the interface).

The cost of factorization of the dense top problem for IGA system is given by

$$\mathrm{Cost}_{IGA}(\mathrm{TopSkeleton}) = \mathcal{O}((n_{IGA}^2 p)^3) = \mathcal{O}((2^s + p)^6 p^3) = \mathrm{O}(2^{6s} p^3) \tag{11}$$

while the cost of factorization of the top dense problem for rIGA system is

$$\mathrm{Cost}_{rIGA}(\mathrm{TopSkeleton}) = \mathcal{O}((n_{rIGA}^2 1)^3 = \mathcal{O}((2^s + 2^r p)^6) = \mathcal{O}(2^{6s} + \mathrm{L.O.T.}) \tag{12}$$

Thus, we can conclude, $\mathrm{Cost}_{IGA}(\mathrm{TopSkeleton}) > \mathrm{Cost}_{rIGA}(\mathrm{TopSkeleton})$. The top dense skeleton problem results from a sequence of factorizations of the intermediate skeleton problems.

The parallel processing of the skeleton problem follows the parallel multi-frontal elimination pattern for FE and IGA grids, described in [35]. The macro-elements are merged into sets of larger macro-elements, and fully assembled unknowns from the common interfaces shared between the patches are eliminated. We repeat this process until we join all macro-elements, and end up with the top skeleton problem.

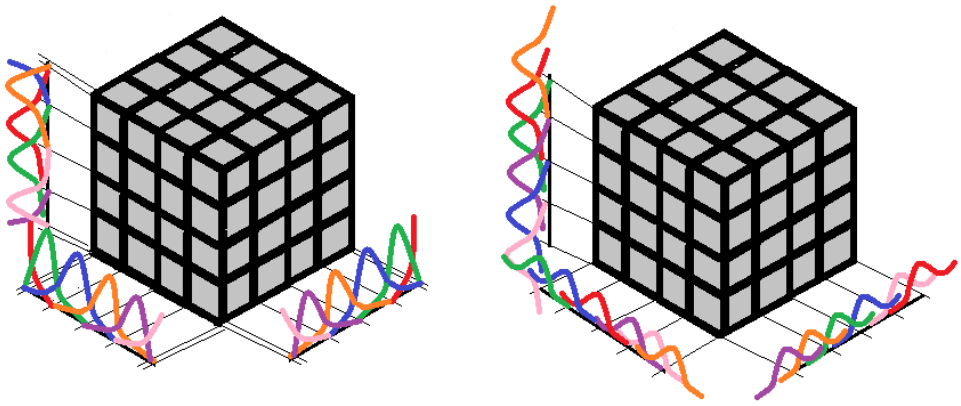

Figure 2: Single macro-element for $r = 2$, namely $2^{3r} = 2^6 = 64 = 4 * 4 * 4$ elements. The left panel denotes the rIGA case, where we reduced the continuity at each patch interface (i.e., repeated knots at the interface), C^0 hyperplanes between the macro-elements. The right panel denotes the IGA case.

Figures 3, 4 and 5 show the merging of four IGA or rIGA macro-elements and the subsequent elimination of the fully assembled unknowns.

That is, on the skeleton problem, we perform $i = 1, ..., r$ steps. At the i-th step, we merge 2^{2i} macro-elements. The fully assembled unknowns form a 3D cross, with 3 hyperplanes. Each hyperplane is a tensor product of

$$n^i_{IGA} = 2^i 2^{s-r} + p. \tag{13}$$

unknowns in two directions. That is, we have 2^i macro-elements in each direction. This formula is a restriction of Eq. (9), where we considered the hyperplane equivalent to the cross-section of the entire IGA domain. Namely, we replace the total number of elements 2^s by the number of elements resulting from merging 2^i macro-elements, equal to $2^i 2^{s-r}$.

A similar procedure is performed for IGA with C^0 separators case (rIGA), where we have hyperplanes with

$$n^i_{rIGA} = 2^i 2^{s-r} + p + (2^i - 1)(p - 1). \tag{14}$$

unknowns in two directions. This formula is also a restriction of Eq. (10), where we considered the hyperplane equivalent to the cross-section of the entire rIGA domain. Namely, we replace the total number of macro-elements 2^r by the number of merged 2^i macro-elements. The number of

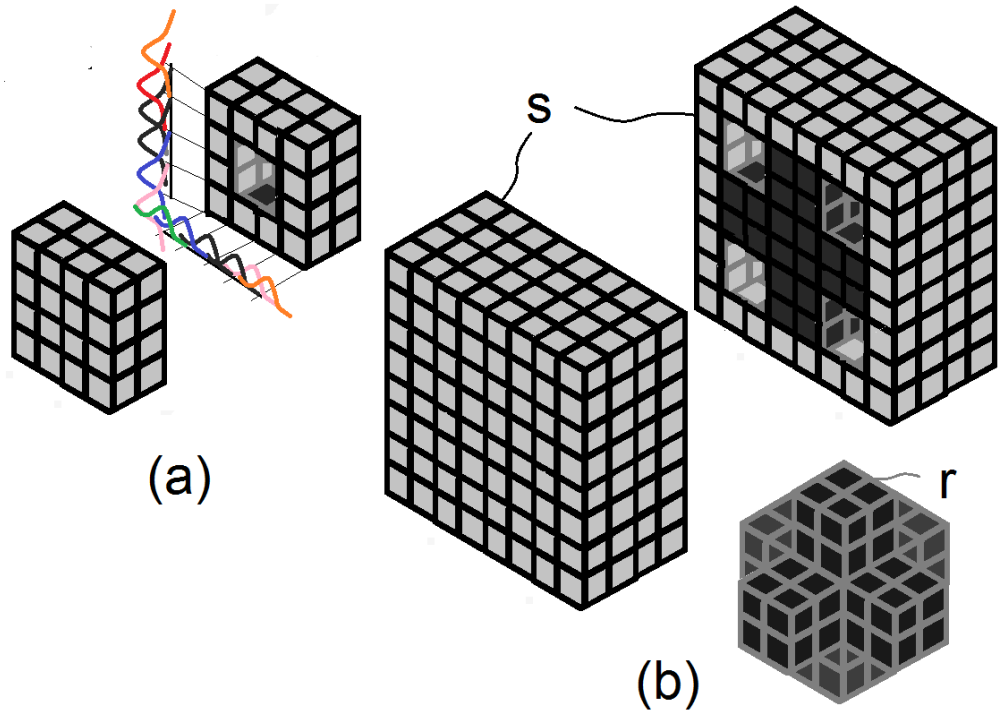

Figure 3: Panel (a) describes a cross-section of a single IGA macro-element with 64 elements and quadratic B-splines, with the 8 interior unknowns eliminated. The elements related to eliminated unknowns are denoted with dark gray color. Panel (b) depicts a cross-section of eight merged macro-elements. The macro-elements do not have interior unknowns, since they were eliminated during the local static condensation. Instead, they have a new fully assembled unknowns, depicted with the dark gray color. They form a 3D cross.

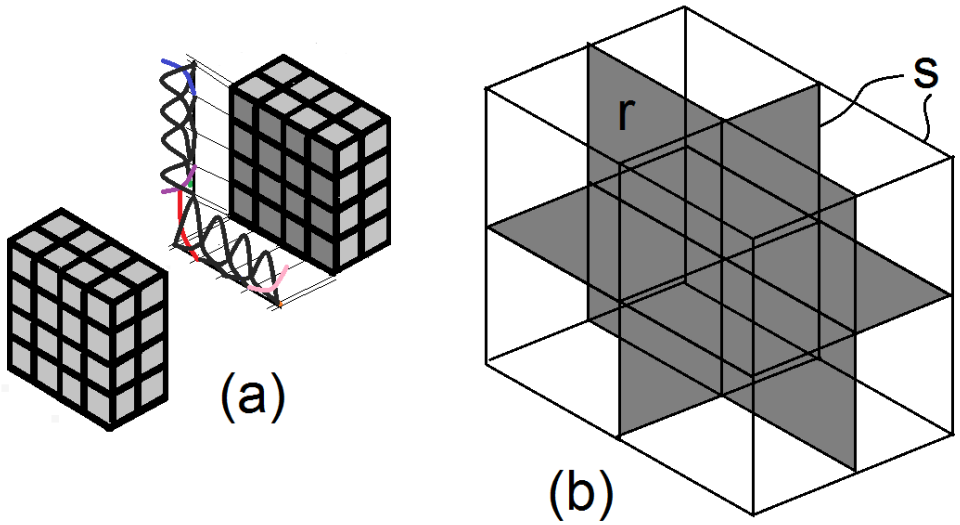

Figure 4: Panel (a) describes a cross-section of a single rIGA macro-element with 64 elements and quadratic B-splines, with the 64 interior unknowns eliminated. They elements related to the eliminated unknowns are denoted by the dark gray color, while the C^0 hyperplanes with not fully assembled unknowns are denoted by light gray color. Panel (b) depicts a cross-section of the eight merged macro-elements. The macro-elements do not have interior unknowns, since they were eliminated during static condensation. Instead, they have a new fully assembled unknowns, related to the shared C^0 separators. The hyperplanes with fully assembled unknowns are denoted by the grey color. They form a thin 3D cross. The hyperplanes with not fully assembled unknowns are transparent.

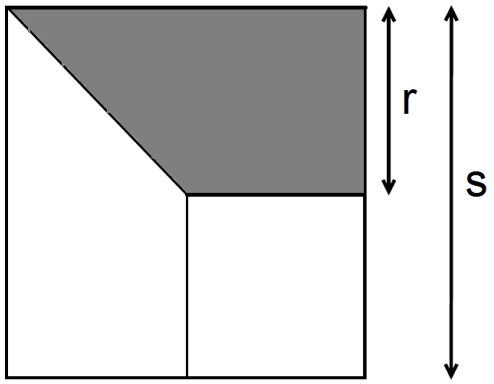

Figure 5: The process of elimination of the r fully assembled unknowns from the 3D cross, from the total of s unknowns over the local system.

degrees of freedom at the i-th step in the skeleton problem for IGA and rIGA is equal to

$$N^i_{IGA} = (n^i_{IGA})^2 p, \tag{15}$$

$$N^i_{rIGA} = (n^i_{rIGA})^2 1. \tag{16}$$

The local matrices to factor at the skeleton problem are dense. Thus, the computational cost of the i-th step of the skeleton problem for IGA and rIGA is

$$Cost^i(IGA) = (N^i_{IGA})^3, \tag{17}$$

$$Cost^i(rIGA) = (N^i_{rIGA})^3. \tag{18}$$

and there are synchronization barriers between step i and $i+1$, when the Schur complements are exchanged. The above costs are obtained under the assumption that we have enough processors to process all local problems at the given i-th step, fully in parallel. If this is not the case, then we need to map the computational problems from a given step into a set of available processors, and each processor needs to solve several of these problems. At the i-th step, we have 2^{s-r+i} local problems to eliminate. Thus, one possible distribution of the computational problems into 2^c processors is such that each processor solves $2^{c-s+r-i}$ local problems.

The total cost of IGA and rIGA processing of the skeleton problem is

$$Cost(IGA) = \sum_{i=1,\ldots,r} Cost^i(IGA) \tag{19}$$

$$Cost(rIGA) = \sum_{i=1,\dots,r} Cost^i(rIGA) \tag{20}$$

when we have enough processors to process all the computational problems fully in parallel, or

$$Cost(IGA) = \sum_{i=1,\dots,r} 2^{c-s+r-i} Cost^i(IGA) \tag{21}$$

$$Cost(rIGA) = \sum_{i=1,\dots,r} 2^{c-s+r-i} Cost^i(rIGA) \tag{22}$$

assuming we have less processors than computational problems.

If we do not have enough cores to process the local problems fully in parallel, we add the appropriate factors related to the distribution of the computations into computing nodes in front of the costs for IGA and rIGA. However, these factors are identical for both IGA and rIGA, since we distribute both computations in an identical manner.

The resulting ratios of the computational cost for different B-spline orders p, different mesh dimensions 2^{3s} and macro-element dimensions $2^{s-r} = 8$ are summarized in Table 1.

N_e	32^3	64^3	128^3	256^3
p=2	4.83	4.37	4.15	4.05
p=3	10.52	8.67	7.84	7.45

Table 1: Ratio of the computational costs of Cost(IGA)/Cost(rIGA) for quadratic and cubic B-splines. The macro-elements are of size 8^3 elements, i.e., $r = s - 3$.

Let us consider now the communication costs.

While the computations of IGA and rIGA at the local systems processed by the skeleton problem solver are of the order of $\mathcal{O}({N^i_{IGA}}^3)$ or $\mathcal{O}({N^i_{rIGA}}^3)$, respectively, the communication is of the $\mathcal{O}({N^i_{IGA}}^2)$ or $\mathcal{O}({N^i_{rIGA}}^2)$ order. Thus, when the communication dominates the computations we have:

$$Communication^i(IGA) = (N^i_{IGA})^2 \tag{23}$$

$$\mathrm{Communication}^{i}(\mathrm{rIGA}) = (N^{i}_{\mathrm{rIGA}})^{2} \tag{24}$$

with the total communicaton costs

$$\mathrm{Communication}(\mathrm{IGA}) = \sum_{i=1,\ldots,r} \mathrm{Communicaton}^{i}(\mathrm{IGA}) \tag{25}$$

$$\mathrm{Communication}(\mathrm{rIGA}) = \sum_{i=1,\ldots,r} \mathrm{Communicaton}^{i}(\mathrm{rIGA}) \tag{26}$$

Therefore, the total ratio is given by

$$\mathrm{Communication}(\mathrm{IGA/rIGA}) = \frac{\mathrm{Communicaton}(\mathrm{IGA})}{\mathrm{Communicaton}(\mathrm{rIGA})} \tag{27}$$

The resulting ratios of the computational costs for different B-spline orders p, different mesh dimensions 2^{3s} and macro-element dimensions $2^{s-r} = 8$ are summarized in Table 2.

N_e	32^3	64^3	128^3	256^3
p=2	2.87	2.68	2.59	2.54
p=3	4.86	4.25	3.96	3.82

Table 2: Ratio of the communication costs of Communication(IGA)/Communication(rIGA) for quadratic and cubic B-splines. The macro-elements are of size 8^3 elements, i.e., $r = s - 3$.

4. Numerical experiments

We consider highly-continuous IGA discretizations solved with a parallel direct solver. We select the MUMPS solver for our numerical experiments [1, 2, 3].

Following [28], we present weak and strong scaling numerical results, and we numerically estimate the ratio between the costs of solving and IGA vs rIGA discretizations.

The numerical experiments have been performed over 128 nodes of the Prometheus cluster from ACK Cyfronet [36], a part of PL-grid infrastructure [13], equipped with 2,50 GHz processor and 128 GB RAM.

4.1. Weak scaling

We utilize MUMPS solver version 5.0.2, linked with MKL 11.3.2, Intel MPI 5.1.3, Metis 5.1.0, compiled with Intel 16.0.2, using -03 optimization flag.

We introduce the C^0 separators every eight elements along x, y and z axis. We assign four macro-elements with $8 \times 8 \times 8 = 512$ elements per processor. Figures 6 and 7 show the weak scaling of the parallel MUMPS solver for IGA and rIGA computations. In the case of the weak scalability, horizontal line denotes perfect scalability, while the trend going up means that scalability is worse. Tables 3 and 4 show the rations we compute between IGA and rIGA, to obtain a p^2 factor.

cores	IGA	rIGA	**IGA/rIGA**
32	15.11	4.54	**3.32**
64	18.11	5.98	**3.02**
128	31.61	7.75	**4.07**

Table 3: Experimental IGA/rIGA ratios for weak scaling experiments for quadratic B-splines.

cores	IGA	rIGA	**IGA/rIGA**
32	70	8	**8.57**
64	261	27	**9.66**
128	683	74	**9.22**

Table 4: Experimental IGA/rIGA ratios for weak scaling experiments for cubic B-splines.

4.2. Strong scaling

Figures 8-9 present the strong scalability results for IGA-FEM and rIGA-FEM.

We validate our theoretical estimates with the numerical evidence below. The theoretical results predict the ratio between the IGA/rIGA of order of p^2, when the computations dominates the communication, as

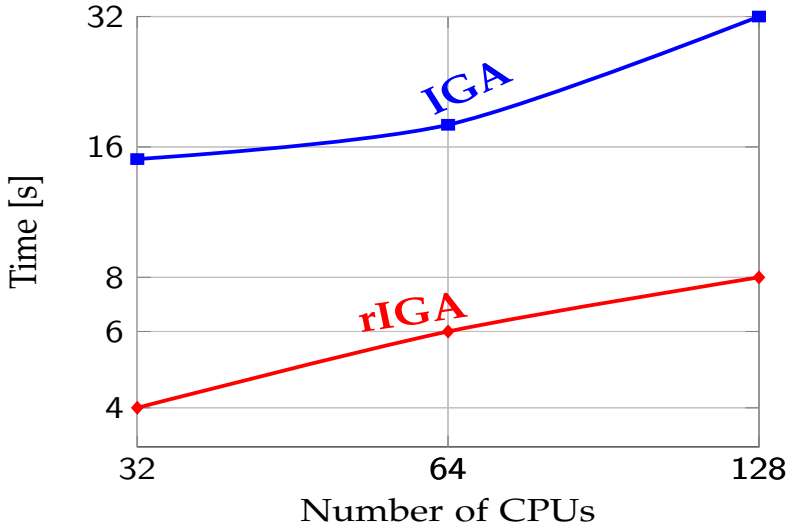

Figure 6: Weak scalling of parallel MUMPS solver for IGA-FEM and rIGA-FEM with quadratic B-splines.

well as the ratio of the order of p when the communication costs start to dominate.

Numerical results summarized in Tables 5 and 6 confirm these predictions. For number of processors less than or equal to 64, the ratio between IGA and rIGA is of order p^2. For a large number of processors such as 128, the communication dominates, and the savings ratio decreases.

cores	1	2	4	8	16	32	64	128
IGA time [s]	1062	820	386	227	141	82	47	31
rIGA time [s]	250	153	91	59	31	19	11	8
ratio IGA/rIGA	4.28	5.35	4.24	3.84	4.54	4.31	4.27	3.87

Table 5: Experimental ratios for $p = 2$, $N_e = 64^3$.

5. Conclusions

We analyze the parallel scalability of direct solvers when dealing with refined isogeometric analysis (rIGA) discretizations in comparison with maximum continuity isogeometric analysis (IGA) ones. We show that

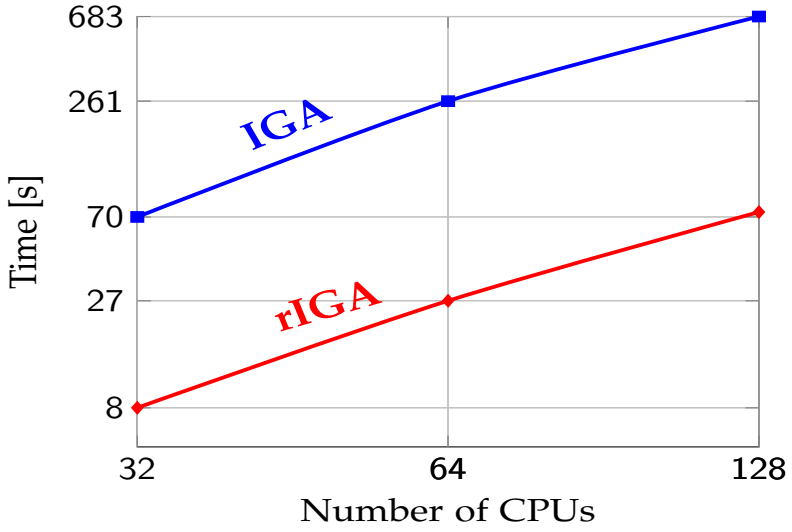

Figure 7: Weak scalling of parallel MUMPS solver for IGA-FEM and rIGA-FEM with cubic B-splines.

cores	1	2	4	8	16	32	64	128
IGA time [s]	5644	3725	2616	1325	885	439	261	156
rIGA time [s]	656	358	206	151	84	42	27	20
ratio IGA/rIGA	8.6	10.4	12.69	8.77	10.53	10.45	9.66	7.8

Table 6: Experimental ratios for $p = 3$, $N_e = 64^3$. For 64 cores we have the number of subdomains equal to the number of macro-elements, and thus the ratio is closer to predicted $p^3 = 9$.

the introduction of C^0 separators to construct the rIGA systems significantly reduces the execution time of the parallel direct solver. We derive the theoretical ratios between the parallel direct solver cost of rIGA and IGA systems and verify them experimentally. We analyze the formulas by substituting the B-spline order p the number of elements 2^{3s} and the dimensions of the macro-elements $2^{3(s-r)}$ to conclude that they indicate a reduction factor approximately p^2 when the number of available cores is limited with respect to the problem size and FLOPS dominate computations. As the number of cores augments for a fixed problem size and communication costs become dominant, the rIGA gain factor reduces to approximately p. The reduction of the computational time should not de-

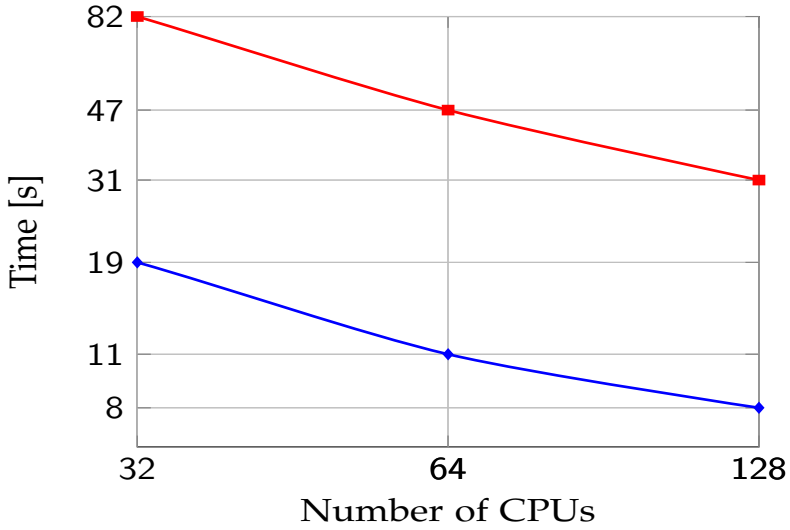

Figure 8: Scalability of parallel MUMPS solver for IGA-FEM and rIGA-FEM with quadratic B-splines for $p = 2$ and $N_e = 96^3$.

pend significantly on the implementation of the direct solver algorithm. Thus, we expect a similar reduction of the computational cost when working with other direct solvers, like e.g. parallel SuperLU solver [37, 23], parallel PaStiX solver [29], or other available solvers e.g. through PETSc interface [4, 5, 6].

6. Acknowledgments

The work of Maciej Paszyński have been supported by Polish National Science Centre grant no. DEC-2016/21/B/ST6/01539.

This research was supported in part by PL-Grid Infrastructure. The visit of Victor Trujillo in Krakow and Maciej Paszyński in Valparaiso was supported by Chilean CONICYT project PAI80160025.

David Pardo has received funding from the Projects of the Spanish Ministry of Economy and Competitiveness with reference MTM2016-76329-R (AEI/FEDER, EU), and MTM2016-81697-ERC/AEI, the BCAM "Severo Ochoa" 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)".

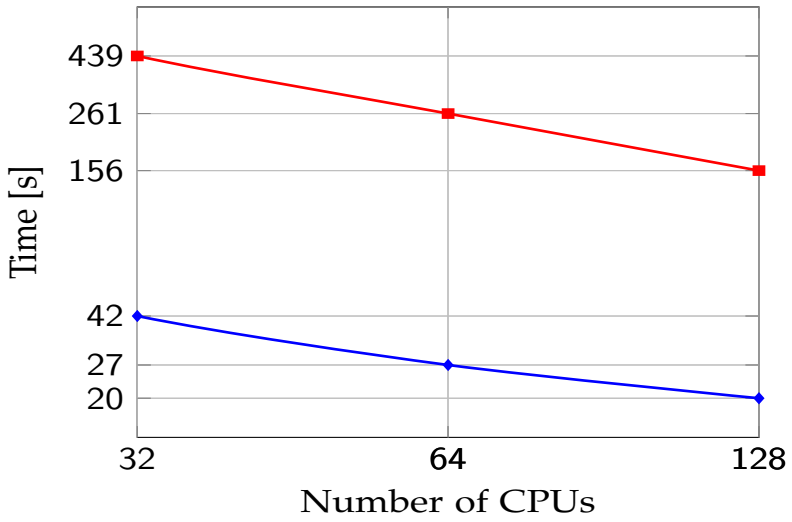

Figure 9: Scalability of parallel MUMPS solver for IGA-FEM and rIGA-FEM with cubic B-splines for $p = 3$ and $N_e = 64^3$.

This publication was also made possible in part by the CSIRO Professorial Chair in Computational Geoscience at Curtin University and the Deep Earth Imaging Enterprise Future Science Platforms of the Commonwealth Scientific Industrial Research Organisation, CSIRO, of Australia. Additional support was provided by the European Union's Horizon 2020 Research and Innovation Program of the Marie Skłodowska-Curie grant agreement No. 644202, the Mega-grant of the Russian Federation Government (N 14.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.

7. Bibliography

[1] P. R. Amestoy , I. S. Duff, *Multifrontal parallel distributed symmetric and unsymmetric solvers*, Computer Methods in Applied Mechanics and Engineering, 184 (2000) 501-520.

[2] P. R. Amestoy, I. S. Duff, J. Koster, J.Y. L'Excellent, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM Journal of Matrix Analysis and Applications, 1(23) (2001) 15-41.

[3] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, S. Pralet, *Hybrid scheduling for the parallel solution of linear systems*, Computer Methods in Applied Mechanics and Engineering, 2(32) (2001) 136-156.

[4] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, K. Rupp, B. F. Smith, H. Zhang, *PETSc Web Page*, *http://www.mcs.anl.gov/petsc* (2014)

[5] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, K. Rupp, B. F. Smith, H. Zhang, *PETSc User Manual*, Argonne National Laboratory ANL-95/11 - Revision 3.4 (2013)

[6] S. Balay, W. D. Gropp, L. Curfman McInnes, B. F. Smith, *Efficient Management of Parallelism in Object Oriented Numerical Software Libraries, Modern Software Tools in Scientific Computing*, Editors E. Arge, A. M. Bruaset and H. P. Langtangen (1997) Birkhä user Press.

[7] M. Barton, V. M. Calo, *Gauss-Galerkin quadrature rules for quadratic and cubic spline spaces and their application to isogeometric analysis*, Computer-Aided Design, 82 (2017) 57-67.

[8] Y. Bazilevs, L. Beirao da Veiga, J.A. Cottrell, T.J.R. Hughes, and G. Sangalli, *Isogeometric analysis: Approximation, stability and error estimates for h-refined meshes*, Mathematical Methods and Models in Applied Sciences, 16 (2006) 1031–1090.

[9] Y. Bazilevs, V.M. Calo, J.A. Cottrell, T.J.R. 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 (2007) 173-201.

[10] Y. Bazilevs, V.M. Calo, Y. Zhang, T.J.R. Hughes: *Isogeometric fluid-structure interaction analysis with applications to arterial blood flow*, Computational Mechanics 38(4-5) (2006) 310-322.

[11] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, S. Lipton, M. A. Scott, T. W. Sederberg, *Isogeometric analysis using T-splines*, Computer Methods in Applied Mechanics and Engineering, 199 (2010) 229-263.

[12] D.J. Benson, Y. Bazilevs, M.-C. Hsu and T.J.R. Hughes, *A large-deformation, rotation-free isogeometric shell*, Computer Methods in Applied Mechanics and Engineering, 200 (2011) 1367-1378.

[13] M. Bubak, J. Kitowski, K. Wiatr, *E-Science on Distributed Computing Infrastructure*, Achievements of PL-Grid Plus Domain-specific Services and Tools, Volume 8500 (2014) Springer

[14] V.M. Calo, N. Brasher, Y. Bazilevs, T.J.R. Hughes, *Multiphysics Model for Blood Flow and Drug Transport with Application to Patient-Specific Coronary Artery Flow*, Computational Mechanics, 43(1) (2008) 161–177.

[15] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, *Isogeometric Analysis: Toward Unification of CAD and FEA* John Wiley and Sons, (2009)

[16] K. Chang, T.J.R. Hughes, V.M. Calo, *Isogeometric variational multiscale large-eddy simulation of fully-developed turbulent flow over a wavy wall*, Computers and Fluids, 68 (2012) 94-104.

[17] N. Collier, L. Dalcin, V. M. 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

[18] N. Collier, L. Dalcin, D. Pardo, V. M. Calo, *The Cost of Continuity: Performance of Iterative Solvers on Isogeometric Finite Elements*, SIAM Journal Scientific Computing, 35(2) (2012) A767-A784.

[19] N.O. Collier, D. Pardo, M. Paszyński, L. Dalcín, V. M. 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 (2012) 353-361.

[20] L. Dedè,T.J.R. Hughes, S. Lipton, V.M. Calo, *Structural topology optimization with isogeometric analysis in a phase field approach*, USNC-TAM2010, 16th US National Congress of Theoretical and Applied Mechanics, At State College, PA, USA, June (2010)

[21] L. Dedè, M. J. Borden, T.J.R. Hughes, *Isogeometric analysis for topology optimization with a phase field model*, ICES REPORT 11-29, The Institute for Computational Engineering and Sciences, The University of Texas at Austin (2011).

[22] A. Gupta, M. Joshi, V. Kumar, *WSMP: a high performance shared and distributed memory parallel symmetric sparse linear solver* (1998) in B. Kagstrom, J. J. Dongarra, E. Elmroth, J. Wasniewski (Eds.), PARA'98 Workshop on Applied Parallel Computing in Large Scale Scientific and Industrial Problems, IBMT, J. Watson Research Center, Springer Verlag, Yorktown Heights, New York, Also available as RC21886.

[23] X.S. Li, J.W. Demmel, J.R. Gilbert, iL. Grigori, M. Shao, I. Yamazaki, *SuperLU Users' Guide*, Lawrence Berkeley National Laboratory, LBNL-44289 *http://crd.lbl.gov/ xiaoye/SuperLU/* (1999).

[24] R. Duddu, L. Lavier, T.J.R. Hughes, V.M. Calo, *A finite strain Eulerian formulation for compressible and nearly incompressible hyper-elasticity using high-order NURBS elements*, International Journal of Numerical Methods in Engineering, 89(6) (2012) 762-785.

[25] D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, V. Calo, *The value of continuity: Refined isogeometric analysis and fast direct solvers*, Computer Methods in Applied Mechanics and Engineering, 316 (2016) 586-605.

[26] H. Gómez, V.M. Calo, Y. Bazilevs, T.J.R. Hughes, *Isogeometric analysis of the Cahn-Hilliard phase-field model*, Computer Methods in Applied Mechanics and Engineering 197 (2008) 4333–4352.

[27] H. Gómez, T.J.R. Hughes, X. Nogueira, V.M. Calo, *Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations*. Computer Methods in Applied Mechanics and Engineering 199 (2010) 1828-1840.

[28] M. T. Heath, *A tale of two laws*, International Journal of High Performance Computing Applications, 29(3) (2015) 320-330.

[29] P. Hénon, P. Ramet, J. Roman, *PaStiX: A High-Performance Parallel Direct Solver for Sparse Symmetric Definite Systems*, Parallel Computing, 28(2) (2002) 301-321.

[30] S. Hossain, S.F.A. Hossainy, Y. Bazilevs, V.M. Calo, T.J.R. Hughes, *Mathematical modeling of coupled drug and drug-encapsulated nanoparticle transport in patient-specific coronary artery walls*, Computational Mechanics, 49(2) (2011) 213-242.

[31] M.-C. Hsu, I. Akkerman, Y. Bazilevs, *High-performance computing of wind turbine aerodynamics using isogeometric analysis*, Computers and Fluids, 49(1) (2011) 93-100.

[32] T .J. R. 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(39) (2005) 4135-4195

[33] M. Łoś, M. Paszyński, A. Kłusek, W. Dzwinel, *Application of fast isogeometric L2 projection solver for tumor growth simulations*, Computer Methods in Applied Mechanics and Engineering 316 (2017) 1257–1269.

[34] V. Puzyrev, S. Koric, S. Wilkin, *Evaluation of parallel direct sparse linear solvers in electromagnetic geophysical problems*, Computers & Geosciences, 89 (2016) 79-87.

[35] M. Woźniak, K. Kuźnik, M. Paszynski, D. Pardo, V.M. Calo: *Computational cost of isogeometric multi-frontal solvers on distributed memory parallel machines*, Computer Methods in Applied Mechanics and Engineering, 284 (2015) 971-987.

[36] http://www.cyfronet.krakow.pl/computers/15226,artykul,prometheus.html (accessed on June 2017)

[37] Xiaoye S. Li, *An Overview of SuperLU: Algorithms, Implementation, and User Interface*, TOMS Transactions on Mathematical Software, 31(3) (2005) 302-325.