Comput. Methods Appl. Mech. Engrg. 213–216 (2012) 353–361

Contents lists available at SciVerse ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers

Nathan Collier [a,*], David Pardo [b], Lisandro Dalcin [d], Maciej Paszynski [c], V.M. Calo [a]

[a] Applied Mathematics and Computational Science, Earth and Environmental Sciences and Engineering, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
[b] Department of Applied Mathematics, Statistics, and Operational Research, The University of the Basque Country and Ikerbasque, Bilbao, Spain
[c] AGH University of Science and Technology, Krakow, Poland
[d] Consejo Nacional de Investigaciones Científicas y Técnicas, Santa Fe, Argentina

ARTICLE INFO

Article history:
Received 25 January 2011
Received in revised form 14 October 2011
Accepted 2 November 2011
Available online 26 November 2011

Keywords:
Isogeometric analysis
Direct solvers
Multi-frontal solvers
k-Refinement
Performance

ABSTRACT

We study the performance of direct solvers on linear systems of equations resulting from isogeometric analysis. The problem of choice is the canonical Laplace equation in three dimensions. From this study we conclude that for a fixed number of unknowns and polynomial degree of approximation, a higher degree of continuity k drastically increases the CPU time and RAM needed to solve the problem when using a direct solver. This paper presents numerical results detailing the phenomenon as well as a theoretical analysis that explains the underlying cause.

1. Introduction

Isogeometric analysis has received a lot of attention in recent years, beginning with the seminal work of Hughes et al. [1]. This work was originally motivated by the desire to find a method for solving partial differential equations (PDE's) which would simplify, if not eliminate, the problem of converting geometric discretizations in the engineering design process. Tangential to the benefits of geometry/analysis unification, the method is also well suited for solving nonlinear and higher-order PDE's due to its higher-order continuity.

A wide variety of application areas have taken advantage of the strengths of isogeometric analysis. These applications include the modeling and simulation of structural vibrations, fluid–structure interaction, patient-specific arterial blood flow, complex fluid flow and turbulence, shape and topology optimization, phase field models via the Cahn–Hilliard equation, cavitation, deformation in the incompressible limit, and shell analysis [2–17]. In addition, a book [18] on the subject carefully details how to solve such problems, including the motivating philosophy which has driven this research effort. Many of these applications benefit from a more continuous basis. Due to this increased interest in isogeometric analysis in both engineering and scientific applications, it is important to fully understand the consequences of the use of a more continuous basis.

A quantification of the cost of a method is to compare a measure of accuracy with the number of degrees of freedom (cf. [19,20]). While this is an important link to establish and meaningful to demonstrate convergence, the approach neglects the cost of computing each degree of freedom by implicitly assuming that the cost is equivalent for different discretizations. The relationship between accuracy and computational cost is a composition of functions representing the relationship of the cost of solving the linear system ($\mathcal{C}$) to the degrees of freedom ($\mathcal{N}$) of a discretization, and the relationship of the degrees of freedom to the accuracy of this discretization ($\mathcal{E}$), as shown in Fig. 1. Thus, if $g : \mathcal{C} \to \mathcal{N}$ and $f : \mathcal{N} \to \mathcal{E}$, then $f \circ g : \mathcal{C} \to \mathcal{E}$ is a true measure of accuracy versus computational costs.

The relationship of error to degrees of freedom (f) is particular to each problem and depends on the parameters of the partial differential equation and/or the geometry of the domain. However, the relationship of computational cost to the degrees of freedom for B-splines discretizations (g) when using direct solvers has not yet been established, and depends solely on the structure of the resulting linear system. This structure is fully determined by the support and interaction of the basis functions, that is the connectivity of the system.

* Corresponding author.
E-mail address: nathaniel.collier@gmail.com (N. Collier).

0045-7825/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2011.11.002

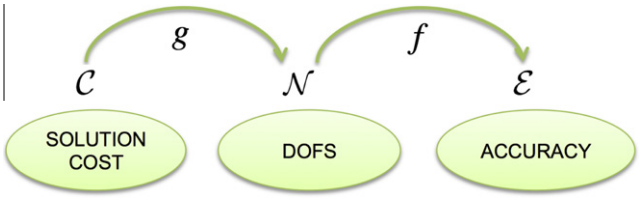


Fig. 1. The efficiency of a method relates to the number of degrees of freedom ($\mathcal{N}$) needed for a given accuracy ($\mathcal{E}$) and the solution cost ($\mathcal{C}$) for those degrees of freedom.

The main contribution of this work is to present a theoretical and numerical analysis of the relationship between the number of degrees of freedom of higher-order continuous B-spline spaces and the solution time of the linear system when using direct solvers. We chose direct solvers for four reasons. First, while implementations of direct solvers vary, the underlying algorithm is still *LU* factorization and therefore, the trends shown in this paper apply to the best existing direct solvers. Second, direct solvers are important for problems with multiple right-hand sides, such as goal-oriented adaptivity or inverse problems. Third, direct solvers are used in stiff problems where iterative solvers do not converge or are computationally more expensive. Last, direct solvers constitute the main building block of many preconditioners used within iterative solvers and as such, direct solvers are a logical first step to understand the behavior of iterative solvers on systems generated by isogeometric analysis.

Specifically, we will use MUMPS [21,22] (in-core) for most of the numerical results. At the time of this writing, MUMPS is considered to be among the fastest direct solvers for solving in core [23–25]. The software is freely available for academic use as well as included in larger scientific libraries such as PETSc [26,27]. In addition to MUMPS, we will also use the solver PARDISO [28,29] to show that our results are independent of the particular implementation of the direct solver we employ. Moreover, we will also consider different ordering algorithms (including METIS [30] and AMD [31]) to illustrate that the main trends are also independent of the selected ordering, provided that this ordering algorithm is competitive. While this work chooses to focus on isogeometric methods, the results apply to a broader class of *hpk*-finite elements, such as those proposed by Liu et al. [32–36].

2. Time and memory estimates

In this section, we compute estimates for the number of floating point operations (FLOPS) and memory required to solve the system of linear equations resulting from finite element discretizations using a direct multi-frontal solver. We assume the number of FLOPS will measure the execution time of the solver, given their strong correlation. While this analysis is particular to systems generated from finite element spaces of three spatial dimensions, the same analysis can be applied to lower spatial dimension systems as well as those generated by other problems. First, we analyze the time and memory cost of performing the Schur complement, which is the main building block in the construction of a multi-frontal solver.

2.1. The Schur complement

Let a dense matrix A be decomposed into four blocks as:

$$A = \begin{bmatrix} B & C \\ D & E \end{bmatrix}. \tag{1}$$

The Schur complement method consists of performing partial *LU* factorization of the square submatrix B to obtain:

$$A = \begin{bmatrix} I & 0 \\ DB^{-1} & I \end{bmatrix} \cdot \begin{bmatrix} B & 0 \\ 0 & E - DB^{-1}C \end{bmatrix} \cdot \begin{bmatrix} I & B^{-1}C \\ 0 & I \end{bmatrix}, \tag{2}$$

where I is the identity matrix. The term $E - DB^{-1}C$ is known as the Schur complement.

In order to estimate the FLOPS and memory required to perform the above partial *LU* factorization, we denote the dimension of the square matrix B by q. We denote the number of columns in C and rows in D as r, where r is assumed constant. Then, we have:

$$\begin{aligned} &\text{FLOPS} = \mathcal{O}(q^3 + q^2 r + q r^2) = \mathcal{O}(q^3 + q r^2), \\ &\text{Memory} = \mathcal{O}(q^2 + qr). \end{aligned} \tag{3}$$

The FLOPS estimate is obtained by counting the operations needed to find the *LU* factors of B, $\mathcal{O}(q^3)$. To this we add the FLOPS required to perform r back-substitutions to form $B^{-1}C$, $\mathcal{O}(rq^2)$. Finally, we add the cost of matrix multiplication of D to $B^{-1}C$, $\mathcal{O}(qr^2)$. The memory estimate is obtained by adding the memory needed to store the *LU* factors of the matrix B, $\mathcal{O}(q^2)$, to that required to store $B^{-1}C$ and DB^{-1}, $\mathcal{O}(rq)$.

In the above memory estimate, we are only concerned with the space required to store L and U, since it is well-known that the cost of storing original matrix A is always smaller than or equal to the memory required to store factors L and U. In particular, we have not included the memory required to store the Schur complement, since this is replaced in the next steps of *LU* factorization by additional Schur complement operations.

The Schur complement method is used to eliminate fully assembled degrees of freedom at each level of the elimination process. This local matrix computation needs to be reordered such that all contributions from fully assembled basis functions are placed into the submatrix B. Contributions from basis functions which interact with functions with other non-local contributions are assembled into E, and interactions between the two subgroups into matrices C and D.

2.2. The multi-frontal solver

The analysis will continue by considering two limiting cases that represent minimum and maximum continuity. The first is that of C^0 B-splines and the second is C^{p-1} B-splines, where p refers to the polynomial order of the B-splines used. In this section we will extend the analysis to include the performance of the multi-frontal solver algorithm.

We divide our computational domain in N_e clusters of elements. For the C^0 case, each cluster is simply an element, while for C^{p-1}, each cluster is a set of $p+1$ consecutive elements in each dimension. We assume for simplicity that the number of clusters in our computational domain is $(2^3)^s = 8^s$, where s is a positive integer which represents the number of levels of the multi-frontal algorithm. Notice that even if this assumption is not verified, the final result still holds true provided that the number of degrees of freedom is sufficiently large.

The idea of the multi-frontal solver is to eliminate the interior (fully assembled) unknowns of each cluster, then join 8 neighboring clusters into one to produce 8^{s-1} new clusters of elements. Subsequently the interior unknowns of each new cluster are eliminated and the remaining degrees of freedom which are not fully assembled are aggregated into new clusters. This procedure is continued recursively by joining again 8 clusters into one. The algorithm is given in Algorithm 1.

Algorithm 1: Multi-frontal algorithm

```
for i = 0 to s − 1 do
    N_c = N_c(i) = (8)^{s−i}
    if i = 0 then
        Define N_c(0) clusters
    else
        Join the old N_c(i − 1) clusters
        Eliminate interior degrees of freedom
        Define N_c(i) new clusters
    end if
end for
```

The FLOPS and memory required by Algorithm 1 can be expressed as:

$$\sum_{i=0}^{s-1} N_c(i) S(i), \tag{4}$$

where $S(i)$ is the cost (either FLOPS or memory) of performing each Schur complement at the ith level. Using the notation of the previous subsection on the Schur complement, we define $q = q(i)$ as the number of interior unknowns of each cluster at the ith step, and $r = r(i)$ as the number of interacting (not fully assembled at this level) unknowns at the ith step.

For C^0 finite elements of polynomial order p, there are $(p-1)^3$ degrees of freedom which may be eliminated at the $i = 0$ level of the multi-frontal algorithm. They can be eliminated in terms of the $\mathcal{O}(6p^2)$ degrees of freedom corresponding to basis functions which have support on the element boundaries. That is, $q(0) = \mathcal{O}(p^3)$ and $r(0) = \mathcal{O}(p^2)$. This initial step is known as static condensation. After the initial elimination of bubbles, the agglomeration and elimination proceeds in a recursive fashion. At the next level, the 12 interior faces of the newly assembled cluster of 8 elements are eliminated in terms of the remaining 24 exterior faces. Since each face contributes $\mathcal{O}(p^2)$ degrees of freedom, $q(1) = \mathcal{O}(12p^2)$ and $r(1) = \mathcal{O}(24p^2)$. From the initial 8^s element mesh, 8^{s-1} clusters will be assembled. For level i, where $i \neq 0$, we can write the number of degrees of freedom which we can eliminate as $q(i) = \mathcal{O}(3 \times 2^{2i}p^2)$ and those that remain as $r(i) = \mathcal{O}(6 \times 2^{2i}p^2)$. Here, the advantage of the multi-frontal algorithm can be seen when applied to C^0 finite elements: roughly a third of the degrees of freedom at each level of the algorithm may be eliminated. To make further analysis simpler, we drop constants and take:

$$q(i) = r(i) = \mathcal{O}(2^{2i}p^2),$$

since for large p and s these estimates are adequate.

As the continuity k of the basis is increased, two effects are seen. First, there are fewer functions fully assembled on each cluster and therefore, less to statically condense in terms of the boundary degrees of freedom. Second, there are more basis functions with support on element interfaces because more basis functions have support across these interfaces. This means that as continuity increases, the multi-frontal algorithm loses its advantage. For simplicity we will describe the estimates for spaces with maximum continuity, $k = p - 1$.

For C^{p-1} B-spline spaces, we begin with level $i = 0$ of the multi-frontal algorithm by assembling $p + 1$ elements in each direction. The local matrix of size $(2p + 1)^3$ is computed and a single degree of freedom is fully assembled. This means that a single degree of freedom, $q(0) = 1$, may be expressed in terms of the remaining, $r(0) = \mathcal{O}(8p^3)$. This grouping of $p + 1$ elements in each direction becomes a cluster on level $i = 0$.

As we merge clusters, as in the C^0 case, we eliminate the interior degrees of freedom in terms of the exterior ones. At each elimination level i, the order of magnitude of the number of exterior degrees of freedom, $r(i)$, may be estimated simply, as they are those which have support on the cluster interface. The order of magnitude of the number of degrees of freedom that remain to be eliminated, $q(i)$, are those without support on the cluster interface which have not been eliminated in any previous level.

For example, at level $i = 1$, we have approximately $(2 \cdot 2 - 1)^3 p^3 = 3^3 p^3$ degrees of freedom. We eliminate $q(1) = p^3$ degrees of freedom in terms of $r(1) = (3^3 - 1)p^3$. At level $i = 2$, we have approximately $(2 \cdot (2 \cdot 2 - 1) - 1)^3 p^3 = 5^3 p^3$ degrees of freedom, with $q(2) = (3^3 - 2^3)p^3$ and $r(2) = (5^3 - 3^3)p^3$. Similarly at level $i = 3$, we estimate the total number of degrees of freedom in the cluster as $(2 \cdot (2 \cdot (2 \cdot 2 - 1) - 1))^3 p^3 = 9^3 p^3$, with $q(3) = (7^3 - 2^3 3^3)p^3$ and $r(3) = (9^3 - 7^3)p^3$. Generalizing to the ith level, the approximate number of degrees of freedom in the cluster is:

$$\left(2^{i+1} - \sum_{k=0}^{i-1} 2^k\right)^3 p^3 = \mathcal{O}\left(\left(2^{i+1} - 2^i\right)^3 p^3\right) = \mathcal{O}\left((2^i)^3 p^3\right).$$

Thus the number of degrees of freedom to eliminate is:

$$q(i) = \left((2^i - 1)^3 - (2^i - 2)^3\right)p^3,$$

and the number of interface degrees of freedom is approximately:

$$r(i) = \left((2^i + 1)^3 - (2^i - 1)^3\right)p^3.$$

We simplify the estimation by retaining only the leading orders and dropping constants to obtain:

$$q(i) = r(i) = \mathcal{O}((2^{2i})p^3).$$

These estimates for the number of degrees of freedom which are fully assembled, $q(i)$, and the number of remaining degrees of freedom, $r(i)$, per level i of the multi-frontal process are summarized in Table 1. This table shows that for C^0 B-spline spaces, the number of degrees of freedom that remain at all levels, $r(i)$, is proportional to p^2. As the continuity increases to $k = p - 1$, this factor grows to p^3. That is, as the continuity of the basis grows the cost of elimination grows with the polynomial order.

2.3. FLOPS and memory estimates

Let N be the total number of unknowns in the original system. We use the results from Table 1 with the FLOPS and memory estimates in Eq. (3) to develop Table 2. This table describes the cost in FLOPS and memory of each level of the multi-frontal algorithm. We then compute full estimates for the entire linear system solution process.

Estimates for C^0 B-splines:

$$\begin{aligned}\text{FLOPS} &= 8^s p^9 + \sum_{i=1}^{s-1} 8^{(s-i)} 2^{6i} p^6 = \mathcal{O}\left(8^s p^9 + 2^{6s} p^6\right) \\ &= \mathcal{O}\left(N_c^3 p^9 + N_c^6 p^6\right) = \mathcal{O}\left(Np^6 + N^2\right),\end{aligned}$$

$$\begin{aligned}\text{Memory} &= 8^s p^6 + \sum_{i=1}^{s-1} 8^{(s-i)} 2^{4i} p^4 = \mathcal{O}\left(8^s p^6 + 2^{4s} p^4\right) \\ &= \mathcal{O}\left(N_c^3 p^6 + N_c^4 p^4\right) = \mathcal{O}\left(Np^3 + N^{4/3}\right).\end{aligned} \tag{5}$$

Table 1
Number of interior (q) and interacting (r) unknowns at each step of the multi-frontal solver.

	$q(0)$	$r(0)$	$q(i)$, $i \neq 0$	$r(i)$, $i \neq 0$
C^0	$\mathcal{O}(p^3)$	$\mathcal{O}(p^2)$	$\mathcal{O}(2^{2i}p^2)$	$\mathcal{O}(2^{2i}p^2)$
C^{p-1}	$\mathcal{O}(1)$	$\mathcal{O}(p^3)$	$\mathcal{O}(2^{2i}p^3)$	$\mathcal{O}(2^{2i}p^3)$

Estimates for C^{p-1} B-splines:

$$\begin{aligned}\text{FLOPS} &= 8^s p^6 + \sum_{i=1}^{s-1} 8^{(s-i)} 2^{6i} p^9 = \mathcal{O}\left(8^s p^6 + 2^{6s} p^9\right) \\ &= \mathcal{O}\left(N_c^3 p^6 + N_c^6 p^9\right) = \mathcal{O}\left(N^2 p^3\right), \\ \text{Memory} &= 8^s p^4 + \sum_{i=1}^{s-1} 8^{(s-i)} 2^{4i} p^6 = \mathcal{O}\left(8^s p^4 + 2^{4s} p^6\right) \\ &= \mathcal{O}\left(N_c^4 p^6\right) = \mathcal{O}\left(N^{4/3} p^2\right).\end{aligned} \tag{6}$$

It is interesting to note that for C^0 B-spline spaces the number of FLOPS is independent of p if the number of unknowns N is large enough (i.e. $N > \mathcal{O}(p^6)$). This is a bound which is achieved in practical simulations. The number of FLOPS of the C^{p-1} method is p^3 times more expensive than the C^0 for large N. The amount of memory required by the C^{p-1} B-spline spaces is p^2 times more expensive than that required by the C^0 spaces.

3. Model problem

The problem used for our study is the Laplace equation in three dimensions on the unit cube (Fig. 2), subject to a zero Dirichlet condition on the bottom surface, a unit Dirichlet condition on the top surface, and free (zero) Neumann conditions elsewhere. Formally, we solve:

$$\begin{cases} -\nabla \cdot (\nabla u) = 0 & \text{on } \Omega, \\ u = 0 & \text{on } \Gamma_{D0}, \\ u = 1 & \text{on } \Gamma_{D1}, \\ (\nabla u) \cdot \mathbf{n} = 0 & \text{on } \Gamma_N, \end{cases} \tag{7}$$

where $\Omega = [0,1]^3$, $\Gamma_{D0} = (:,:,0)$, $\Gamma_{D1} = (:,:,1)$, and $\Gamma_N = (0,:,:) \cup (1,:,:) \cup (:,0,:) \cup (:,1,:)$. While this is a simple scalar problem, the emphasis here is on the structure of the linear system resulting from the Galerkin approximation of the variational form. As long as the system is not singular, the specific values in the system do not influence the time and memory used by the direct solver. Thus, the results described in this paper are also applicable to more complex equations, provided the dimensionality of the system unknowns is taken into account in the extension of these estimates to vector-valued problems.

3.1. Discretization

We use unmapped B-splines to solve the PDE in the parametric unit cube domain. While the rational version of the basis is more general (non-uniform rational B-splines or NURBS [37,38]) and mapped geometries are a critical component to isogeometric analysis [1], neither change the structure of the matrix resulting from the Galerkin weak form, and so results for B-splines apply to these extended cases as well.

We compute solutions to the model problem on B-splines discretizations which vary in polynomial degree $p = 1,2,\ldots,8$ and continuity $k = 0,1,\ldots,p-1$ yet with almost constant overall number of degrees of freedom. We choose three different numbers of degrees of freedom to analyze: 10,000, 30,000, and 100,000. The comparison is of the same spirit as in [19], where different numbers of uniformly refined elements are used to attain a comparable number of degrees of freedom for different levels of continuity.

Table 2
FLOPS and memory estimates at each step of the multi-frontal solver.

	FLOPS $S(0)$	Memory $S(0)$	FLOPS $S(i)$, $i \neq 0$	Memory $S(i)$, $i \neq 0$
C^0	$\mathcal{O}(p^9)$	$\mathcal{O}(p^6)$	$\mathcal{O}(2^{6i}p^6)$	$\mathcal{O}(2^{4i}p^6)$
C^{p-1}	$\mathcal{O}(p^6)$	$\mathcal{O}(p^3)$	$\mathcal{O}(2^{6i}p^9)$	$\mathcal{O}(2^{4i}p^6)$

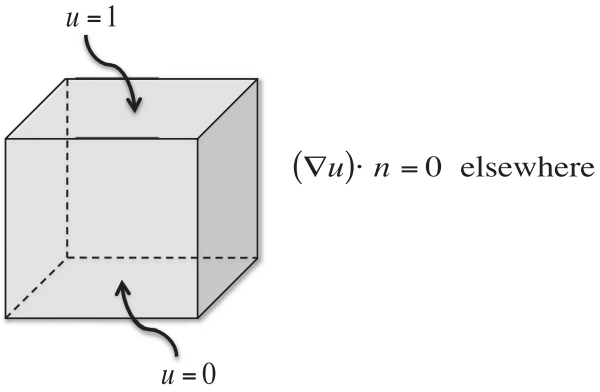


Fig. 2. Boundary conditions for the model problem used in numerical tests.

For each scenario, we record the actual number of degrees of freedom along with the solution time and memory. The actual number of degrees of freedom will not be always equal to the target number since it is not always possible to build a three dimensional grid that contains a specific number of degrees of freedom. For those cases, we selected a discretization with a number of degrees of freedom as close as possible to the target and then adjusted the solution time and memory in a post-processing step, described in detail below. We maintain the number of elements in each coordinate direction as uniform as possible. If the target number of degrees of freedom is N, and the actual number of degrees of freedom is n, then we employ the following approximations to adjust the time, t, and memory, M:

$$t(N) = t(n)\left(\frac{N}{n}\right)^2, \tag{8}$$

$$M(N) = M(n)\left(\frac{N}{n}\right). \tag{9}$$

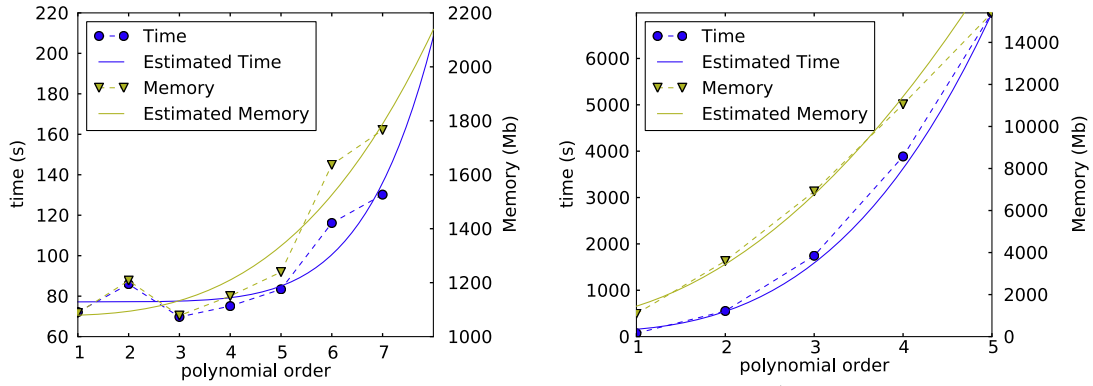
Table 3 presents sample data with adjustments. The above "correction formulas" are obtained by approximating the time of direct solvers on sparse matrices by $O(N^2)$ and the memory by $O(N)$ given that the bandwidth of each matrix are comparable. In addition, note that generally n does not vary from N more than 5%. However in a few cases the difference rose to as much as 15%. The amount of adjustment is minimized and does not influence the trends.

4. Numerical results

All computational experiments have been performed on a workstation with two quad-core Xeon X5550 processors and 24 Gb of memory running Fedora 11. The model problem was implemented using PETSc data structures. We interfaced to MUMPS through PETSc, with the option to solve an asymmetric

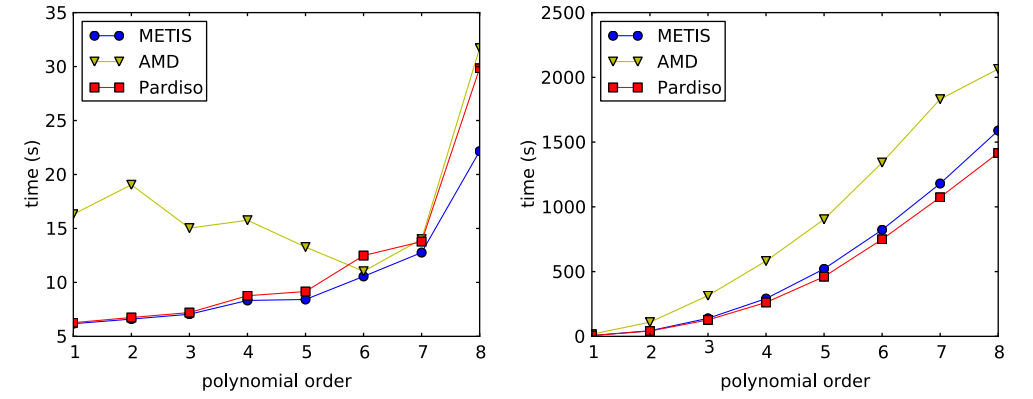
Table 3
Sample data and adjustments for $N = 100{,}000$ degrees of freedom of C^0 B-splines.

p	n	Time (s)		Memory (Mb)	
		Computed $t(n)$	Adjusted $t(N)$	Computed $M(n)$	Adjusted $M(N)$
1	99452	72.14	72.94	1087	1093
2	99405	85.93	86.96	1207	1214
3	97336	69.73	73.6	1077	1107
4	99225	75.09	76.27	1150	1159
5	97336	83.38	88.0	1239	1273
6	103243	116.16	108.98	1636	1585
7	92450	130.18	152.31	1766	1910

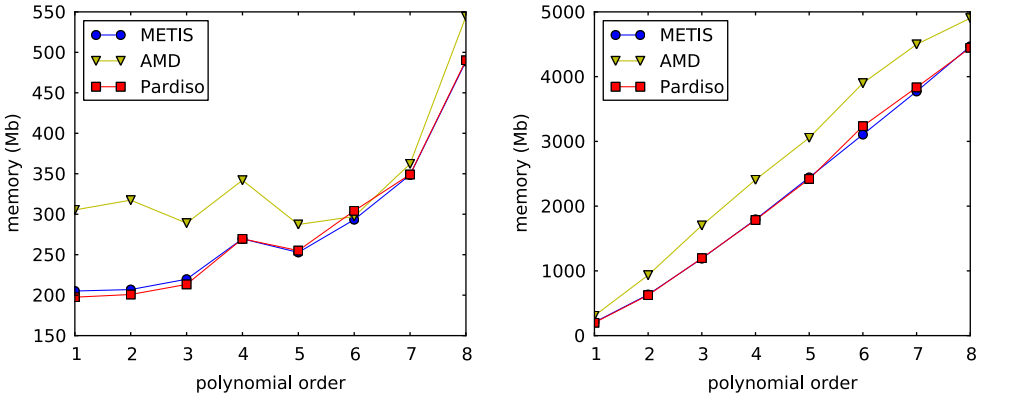


(a) C^0 B-spline spaces (b) C^{p-1} B-spline spaces

Fig. 3. Time and memory for 100,000 degrees of freedom systems along with estimates.



(a) Solution time for C^0 B-spline spaces (b) Solution time for C^{p-1} B-spline spaces



(c) Memory usage for C^0 B-spline spaces (d) Memory usage for C^{p-1} B-spline spaces

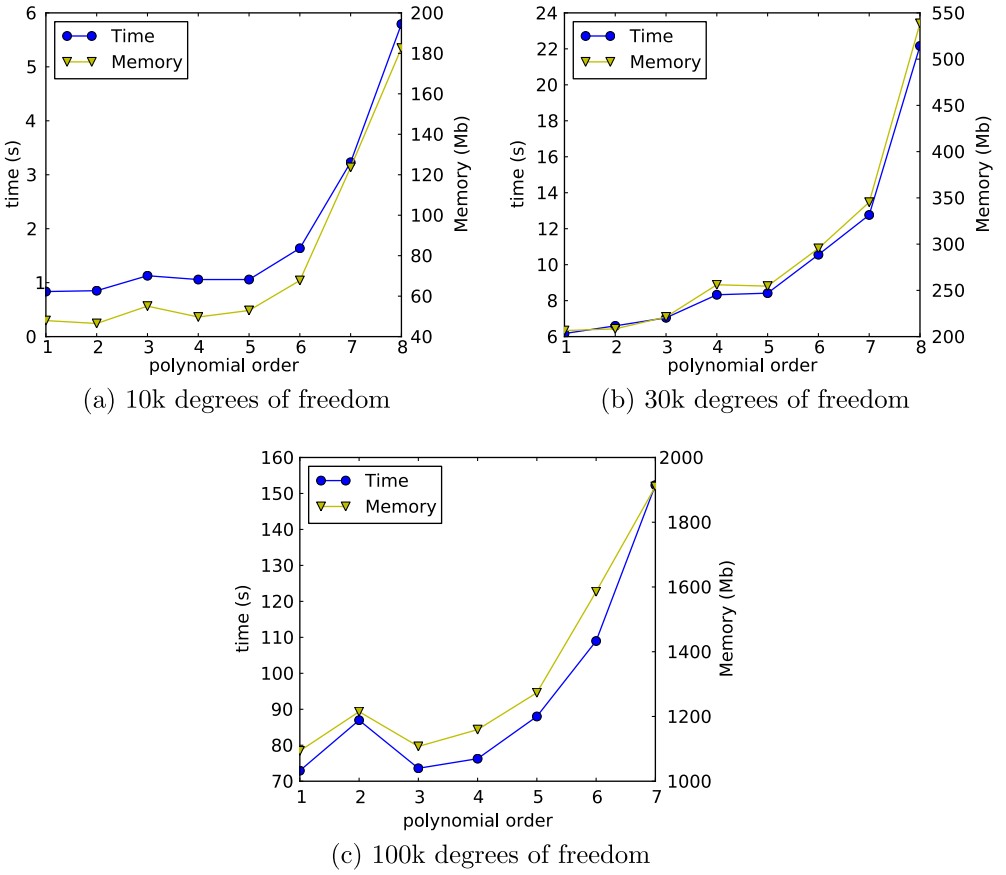
Fig. 4. Comparison of time and memory usage for different solvers and orderings for systems containing 30,000 degrees of freedom.

system. Since PETSc does not have an interface to PARDISO, we extracted the matrix in compressed row storage (CRS) format from PETSc and called the PARDISO solver directly.

For all numerical results, we relate the FLOPS estimates to the computational time measured for the solution of the linear system. The memory estimates we relate to the number of nonzero entries in the *LU* factors. We report the memory required to store an integer and a double precision number for each nonzero entry. Note that this is a conservative quantification of memory usage. In general, solvers will require the use of additional memory. However, we chose to report the memory required to store the *LU* factors as it is most closely related to the estimates derived in Section 2.

4.1. Validation of the estimates

We used the data we obtained in the case of 100,000 degrees of freedom for both C^0 and C^{p-1} spaces to determine constants for the estimates given in Eqs. (5) and (6). These constants are calculated by forming the normal equations and solving in the least square sense, minimizing the square of the residuals. The results are plot-



(a) 10k degrees of freedom

(b) 30k degrees of freedom

(c) 100k degrees of freedom

Fig. 5. Solution time and memory for C^0 B-spline spaces.

ted in Fig. 3 and show good agreement between the theory and the numerical results.

4.2. Insensitivity to solver or ordering

While in this paper we are principally showing results using MUMPS as a solver and METIS to provide the optimal ordering to the system, we would like to emphasize that the trends we report here are independent of the solver and ordering algorithms. We compute solutions to the model problem using MUMPS with METIS and AMD ordering as well as with PARDISO using METIS ordering. The results for 30,000 degrees of freedom for both C^0 and C^{p-1} spaces are shown in Fig. 4.

Note that results will vary with the ordering and/or solver choice. In particular, the AMD ordering is not as competitive for low p in the C^0 results nor for high p is the C^{p-1} results. However, it is important to note that trends do not vary with the choice of solver and ordering. A well-chosen ordering algorithm (such as METIS or AMD) automatically produces a similar effect as static condensation, leading to the performance results that conform to the estimates given in our theoretical estimates. In other words, while different solvers (implementations and orderings) yield different performance results and memory requirements, the base trend described in this work prevails.

4.3. Discussion of results

First we consider the performance of direct solvers on systems generated from C^0 B-splines, which is a particular case that coincides with traditional *hp*-finite element spaces [39]. Fig. 5 illustrates the increase in solution time and memory required as the polynomial degree increases. These results are obtained using solver MUMPS 4.9.2 with METIS ordering.

An interesting trend is the initial constant solution time required to perform the factorization, particularly noticeable for $p = 1, 2, 3, 4$, and 5. This is due to the structure of the matrix. A particular element should have its non-shared degrees of freedom statically condensed. This operation occurs on a small dense block for all the blocks of the overall matrix. Then, the remaining reduced problem (termed the skeleton problem) is solved. As the polynomial order is increased, the size of these blocks increases but not their number. Furthermore, if the number of unknowns N remains constant as we increase p, then the cost of solving the skeleton problem remains constant, which explains why the computational cost is independent of p for moderately low p. Once $N \approx p^6$, the solution time and memory usage show a dependence as predicted by our estimates.

Also note that as the linear system increases in size, the added cost of p-refinements amortizes. Specifically, observe that in the 10,000 degrees of freedom case, the ratio of maximum to minimum solver time is approximately 9 while in the 100,000 degrees of freedom case, it is 2. All of these observations are consistent with the performance of direct solvers on systems generated with standard finite element technologies and with the theoretical estimates given in Section 2.

As the continuity order increases, the block structure of the matrix abates due to the increasing overlap of basis functions. While it is true that the maximum bandwidth does not increase as continuity increases, the minimum bandwidth approaches the maximum as continuity reaches its maximum order of $p - 1$. This leads to a loss of the direct solver efficiency since less of each block, if any at all, can be statically condensed. This increases the size of the

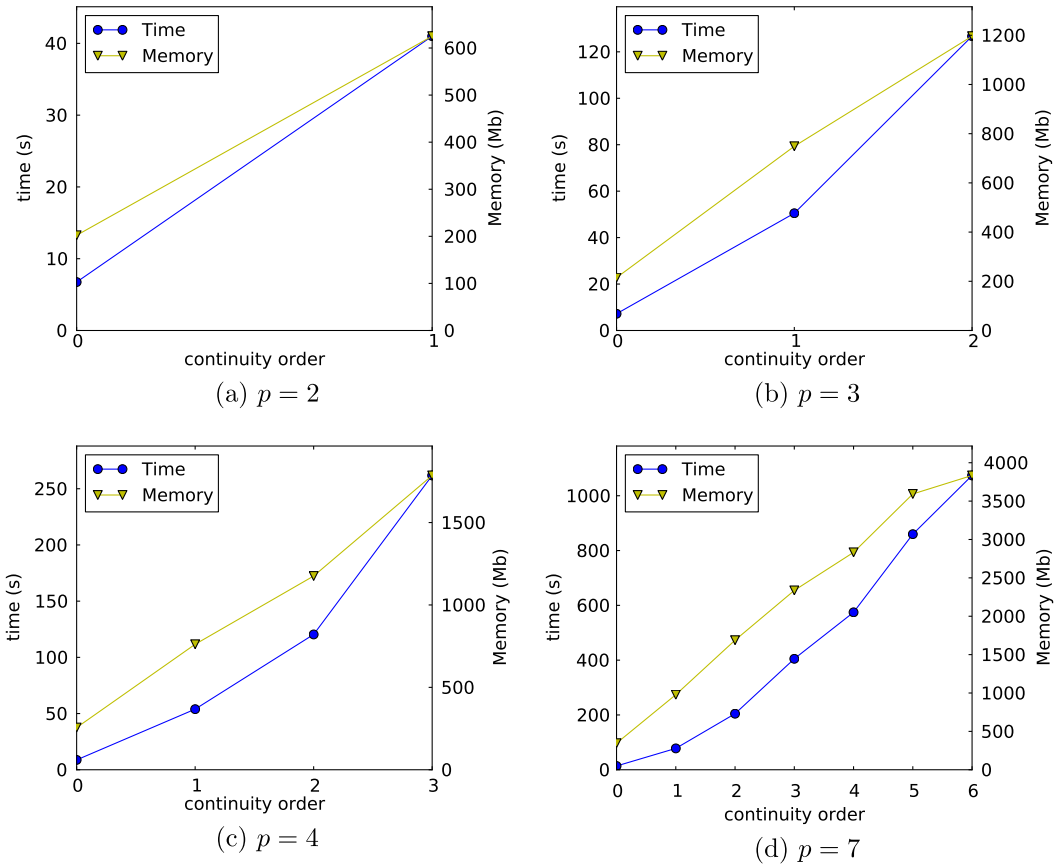


Fig. 6. Solution time and memory growth for 30,000 degrees of freedom of C^k B-spline spaces.

skeleton problem and consumes orders of magnitude more time and memory.

The numerical results support this conclusion, as shown in Fig. 6. Each sub-figure shows the time and memory used by MUMPS as a function of the continuity level for a constant polynomial degree. It can be observed from this figure that their is a strong correlation between increase in computational cost (in terms of both time and memory) and increase in continuity. Additionally, the figure confirms that C^0 and C^{p-1} B-splines constitute limiting cases of the performance of the solver algorithm.

4.4. The cost of k-refinements

As the reference literature concerning isogeometric analysis describes, the B-spline refinement space is richer than in standard finite elements, in particular the notion of *k*-refinement was introduced. This refinement strategy combines notions of degree elevation and knot insertion to refine spaces in such a way that they are globally C^{p-1} [1,18,37,38]. Typical *p*-refinements add many degrees of freedom to the overall system, while *k*-refinements add few yet benefit from the increased approximability provided by the higher polynomial order. While the economy is clear in terms of degrees of freedom, this ignores the cost involved in resolving the resultant linear systems.

To emphasize the importance of this issue, consider a space of C^0 linear B-splines containing 30,000 degrees of freedom. In the numerical tests performed, the time required to solve the resulting linear system is 5 s. Using the *k*-refinement methodology, we can refine this space to be C^1 quadratics. A conservative estimate for the number of degrees of freedom is still 30,000. Note that the time to solve this refined space is now 40 s. A further application of the *k*-refinement procedure takes 120 s to solve, 24 times more than the original. This increase in required resources is significant as shown in the comparison of Fig. 7. Note that this increase in costs completely ignores the increase in assembly cost due to quadrature.

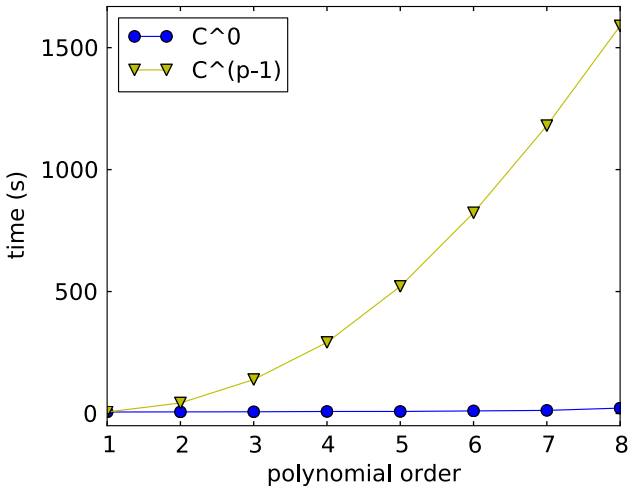


Fig. 7. A comparison of solution time for 30,000 degrees of freedom of C^0 and C^{p-1} B-spline spaces.

Fig. 8 is a compilation of all data recorded where the spaces are always C^{p-1}. Thus each sub-figure represents a *k*-refinement as *p* is increased. As the figure shows, the costs associated with using these function spaces can be as much as 100 times more than their C^0 counterparts. The costs of solving the resulting system dwarfs the enormous savings in degrees of freedom. This figure

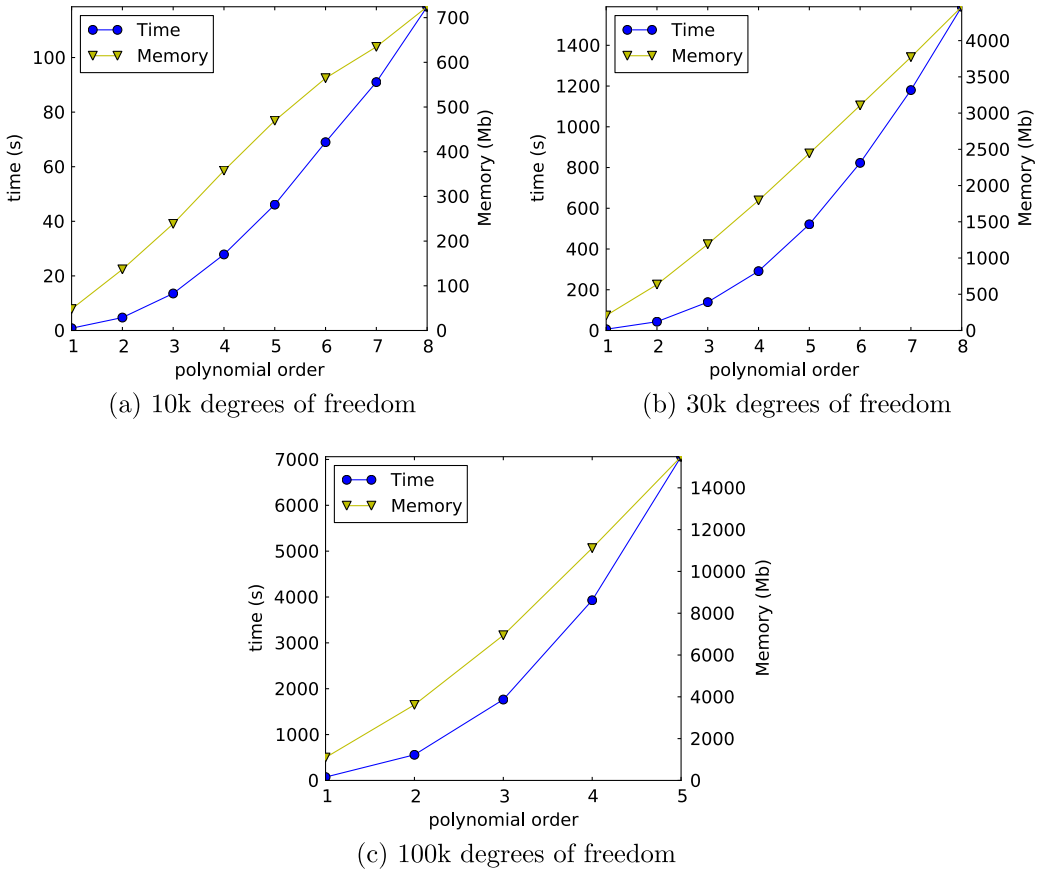


Fig. 8. Solution time and memory for C^{p-1} B-spline spaces.

demonstrates an important truth about *k*-refinements and direct solvers: all degrees of freedom are not alike.

This result suggests that the choice of refinement strategy is more complex than a simple degree of freedom count–one must include the cost of solving the linear systems. It may be that lower continuity, *hp*-refinements are preferred because, while there are many more degrees of freedom (or commensurate number of degrees of freedom but with reduced support), a direct solver may resolve the resulting linear systems more efficiently. This is, however, a question which is particular to the specific problem and beyond the scope of this paper.

5. Conclusions

Isogeometric analysis is an important and powerful analysis tool in the solution of a wide variety of outstanding problems in the scientific literature. This strength partially comes from the continuity of the basis, which may be trivially constructed for any spatial dimension. However this advantage comes at a cost which is not seen until looking in detail at the algorithms used to solve the resulting linear systems. In this case, we have shown that direct solvers require orders of magnitude more time and memory as continuity increases.

Direct solvers have degraded performance on any method where the support of the element basis functions extends across element boundaries, such as in higher continuous NURBS spaces. It is this feature which increases the average bandwidth of the matrix and causes the local element contributions to overlap more.

The performance of iterative solvers on these linear systems is an important extension to this work. Due to the fundamental algorithmic differences in iterative solvers, it is unclear how the higher continuity will affect the performance of this class of techniques. This is an important question which we will address in a future paper.

Acknowledgements

We would like to thank the anonymous reviewers for their feedback which significantly improved the quality of the paper. The second author was partially funded by the Project of the Spanish Ministry of Sciences and Innovation MTM2010-16511. The work of the third author has been partially supported by ANPCyT Argentina Grant PICT-1141/2007. The work of the fourth author was partially supported by Polish Ministry of Science and Higher Education Grant No. NN 519 447739.

References

[1] T.J.R. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (2005) 4135–4195.
[2] Y. Bazilevs, V.M. Calo, Y. Zhang, T.J.R. Hughes, Isogeometric fluid–structure interaction analysis with applications to arterial blood flow, Comput. Mech. 38 (2006) 310.
[3] 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, Comput. Methods Appl. Mech. Engrg. 197 (2007) 173–201.
[4] Y. Bazilevs, V.M. Calo, T.J.R. Hughes, Y. Zhang, Isogeometric fluid–structure interaction: theory, algorithms, and computations, Comput. Mech. 43 (2008) 3–37.
[5] J. Cottrell, A. Reali, Y. Bazilevs, T.J.R. Hughes, Isogeometric analysis of structural vibrations, Comput. Methods Appl. Mech. Engrg. 195 (2006) 5257.
[6] H. Gomez, V.M. Calo, Y. Bazilevs, T.J.R. Hughes, Isogeometric analysis of the Cahn–Hilliard phase-field model, Comput. Methods Appl. Mech. Engrg. 197 (2008) 4333–4352.
[7] H. Gomez, T.J.R. Hughes, X. Nogueira, V.M. Calo, Isogeometric analysis of the isothermal Navier–Stokes–Korteweg equations, Comput. Methods Appl. Mech. Engrg. 199 (2010) 1828.

[8] L. Dedè, T.J.R. Hughes, S. Lipton, V.M. Calo, Structural topology optimization with isogeometric analysis in a phase field approach, in: USNCTAM 2010, 16th US National Congress of Theoretical and Applied Mechanics.
[9] 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.
[10] W.A. Wall, M.A. Frenzel, C. Cyron, Isogeometric structural shape optimization, Comput. Methods Appl. Mech. Engrg. 197 (2008) 2976–2988.
[11] Y. Zhang, Y. Bazilevs, S. Goswami, C.L. Bajaj, T.J.R. Hughes, Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow, Comput. Methods Appl. Mech. Engrg. 196 (2007) 2943–2959.
[12] T. Elguedj, Y. Bazilevs, V.M. Calo, T.J.R. Hughes, $\bar{B}$ and $\bar{F}$ projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements, Comput. Methods Appl. Mech. Engrg. 197 (2008) 2732–2762.
[13] Y. Bazilevs, M.-C. Hsu, J. Kiendl, R. Wchner, K.-U. Bletzinger, 3d simulation of wind turbine rotors at full scale. Part II: fluid structure interaction modeling with composite blades, Int. J. Numer. Methods Fluids 65 (2011) 236–253.
[14] J. Kiendl, K.-U. Bletzinger, J. Linhard, R. Wchner, Isogeometric shell analysis with kirchhoff–love elements, Comput. Methods Appl. Mech. Engrg. 198 (2009) 3902–3914.
[15] R. Duddu, L.L. Lavier, T.J.R. Hughes, V.M. Calo, A finite strain eulerian formulation for compressible and nearly incompressible hyperelasticity using high-order B-spline finite elements, Int. J. Numer. Methods Engrg., in press, doi:10.1002/nme.3262.
[16] 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, Comput. Mech. 43 (2008) 161–177.
[17] 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, Comput. Mech. (2011), doi:10.1007/s00466-011-0633-2.
[18] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Unification of CAD and FEA, John Wiley and Sons, 2009.
[19] I. Akkerman, Y. Bazilevs, V.M. Calo, T.J.R. Hughes, S. Hulshoff, The role of continuity in residual-based variational multiscale modeling of turbulence, Comput. Mech. 41 (2008) 371–378.
[20] J. Cottrell, T. Hughes, A. Reali, Studies of refinement and continuity in isogeometric structural analysis, Comput. Methods Appl. Mech. Engrg. 196 (2007) 4160–4183.
[21] P.R. Amestoy, I.S. Duff, J. Koster, J.-Y. L'Excellent, A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM J. Matrix Anal. Appl. 23 (2001) 15–41.
[22] P.R. Amestoy, A. Guermouche, J.-Y. L'Excellent, S. Pralet, Hybrid scheduling for the parallel solution of linear systems, Parallel Comput. 32 (2006) 136–156.
[23] D. Pardo, M.J. Nam, C. Torres-Verdin, M. Hoversten, I. Garay, Simulation of marine controlled source electromagnetic (CSEM) measurements using a parallel fourier hp-finite element method, Comput. Geosci. 15 (2011) 53–67.
[24] P.R. Amestoy, I.S. Duff, J.-Y. L'excellent, X.S. Li, Analysis and comparison of two general sparse solvers for distributed memory computers, ACM Trans. Math. Softw. 27 (2001) 388–421.
[25] O. Schenk, K. Gärtner, Solving unsymmetric sparse systems of linear equations with PARDISO, Future Gener. Comput. Syst. 20 (2004) 475–487 (Selected numerical algorithms).
[26] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, PETSc Web page. <http://www.mcs.anl.gov/petsc>, 2010.
[27] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, PETSc users manual, Technical Report ANL-95/11 – Revision 3.0.0, Argonne National Laboratory, 2008.
[28] O. Schenk, A. Wächter, M. Hagemann, Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization, Comput. Optim. Appl. 36 (2007) 321–341.
[29] O. Schenk, M. Bollhöfer, R.A. Römer, On large scale diagonalization techniques for the Anderson model of localization, SIAM Rev. 50 (2008) 91–112 (SIGEST Paper).
[30] G. Karypis, V. Kumar, Parallel multilevel *k*-way partitioning scheme for irregular graphs, in: Proceedings of the 1996 ACM/IEEE Conference on Supercomputing, p. 35.
[31] P. Amestoy, T.A. Davis, I.S. Duff, An approximate minimum degree ordering algorithm, SIAM J. Matrix Anal. Appl. 17 (1996) 886–905.
[32] W.K. Liu, W. Han, H. Lu, S. Li, J. Cao, Reproducing kernel element method: part I. Theoretical formulation, Comput. Methods Appl. Mech. Engrg. 193 (2004) 933–951.
[33] S. Li, H. Lu, W. Han, W.K. Liu, D.C. Simkins Jr., Reproducing kernel element method, part II. Global conforming I^m/C^n hierarchy, Comput. Methods Appl. Mech. Engrg. 193 (2004) 953–987.
[34] H. Lu, S. Li, D.C. Simkins Jr., W.K. Liu, J. Cao, Reproducing kernel element method, part III. Generalized enrichment and applications, Comput. Methods Appl. Mech. Engrg. 193 (2004) 989–1011.
[35] D.C. Simkins, S. Li, H. Lu, W.K. Liu, Reproducing kernel element method, part IV. Globally compatible $C^n(n \geqslant 1)$ triangular hierarchy, Comput. Methods Appl. Mech. Engrg. 193 (2004) 1013–1034.
[36] N. Collier, D. Simkins, The quasi-uniformity condition for reproducing kernel element method meshes, Comput. Mech. 44 (2009) 333–342.
[37] L. Piegl, W. Tiller, The NURBS Book, Monographs in Visual Communication, Springer, New York, 1995.
[38] G. Farin, Curves and Surfaces for CAGD: A Practical Guide, fifth ed., Morgan Kaufmann, 2002.
[39] L. Demkowicz, Computing with *hp*-adaptive finite elements, One and Two Dimensional Elliptic and Maxwell Problems, vol. 1, Chapman & Hall/CRC, 2007.