Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 1854–1861

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

Computational complexity and memory usage for multi-frontal direct solvers used in *p* finite element analysis

Victor M. Calo[a], Nathaniel O. Collier[a], David Pardo[b], Maciej R. Paszyński[c]

[a]*Department of Applied Mathematics, Computational Science, Earth Engineering, King Abdullah University of Science and Technology, Thuwal 23955-6900, Kingdom of Saudii Arabia*
[b]*Department of Applied Mathematics, Statistics, and Operational Research, Univsersity of the Basque Country and Ikerbasque, Bilbao, Spain*
[c]*Department of Computer Science, AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Kraków, Poland*

Abstract

The multi-frontal direct solver is the state of the art for the direct solution of linear systems. This paper provides computational complexity and memory usage estimates for the application of the multi-frontal direct solver algorithm on linear systems resulting from *p* finite elements. Specifically we provide the estimates for systems resulting from C^0 polynomial spaces spanned by B-splines. The structured grid and uniform polynomial order used in isogeometric meshes simplifies the analysis.

Keywords: Multi-frontal direct solver; computational complexity, memory usage

1. Introduction

Isogeometric analysis [4] is a relatively new method that has been the subject of much work in recent years. While at its core it is spline-based isoparametric finite element analysis, the spaces used possess unique refinement strategies that form spaces that are supersets of conventional finite elements. The mesh in B-spline based isogeometric analysis is always a structured grid of uniform polynomial order. This simplification allows for specialized complexity and memory usage estimates to be developed for spaces that are C^0 when the direct solver is the multi-frontal solver.

The rest of this paper is organized as follows. In section 2 we present an explanation of the multi-frontal direct solver algorithm. In sections 3 and 4 we present the computational complexity of the algorithm on a single element, and a structure grid mesh consisting of identical elements respectively. In sections 5 and 6 we present the accompanying memory usage. In section 7 we show numeric results that verify the estimates.

2. Multi-frontal direct solver algorithm

The state of the art direct solver is the multi-frontal solver proposed by [1, 2]. It is the generalization of the frontal solver algorithm proposed by [3] The *multi-frontal solver* constructs the elimination tree based on the connectivity for the degrees of freedom, browses the elimination tree from leaves up to the root node, eliminates fully assembled

Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.201

degrees of freedom and merges resulting Schur complement contributions. The procedure for two finite element mesh can be summarized as follows. It consists in creation of the two frontal matrices, each one associated with one element followed by elimination of interior and boundary degrees of freedom from each frontal matrix, leaving the not-fully assembled common edge degrees of freedom not eliminated yet. These steps are illustrated below,

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 \\ \mathbf{C}_1 & \mathbf{A}_1^s \end{bmatrix} \begin{Bmatrix} \mathbf{x}_1 \\ \mathbf{x}_1^s \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_1 \\ \mathbf{b}_1^s \end{Bmatrix} \rightarrow \begin{bmatrix} \mathbf{U}_1 & \mathbf{B}_1^* \\ \mathbf{0} & \mathbf{A}_1^{s*} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_1 \\ \mathbf{x}_1^s \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_1^* \\ \mathbf{b}_1^{s*} \end{Bmatrix} \tag{1}$$

$$\begin{bmatrix} \mathbf{A}_2 & \mathbf{B}_2 \\ \mathbf{C}_2 & \mathbf{A}_2^s \end{bmatrix} \begin{Bmatrix} \mathbf{x}_2 \\ \mathbf{x}_2^s \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_2 \\ \mathbf{b}_2^s \end{Bmatrix} \rightarrow \begin{bmatrix} \mathbf{U}_2 & \mathbf{B}_2^* \\ \mathbf{0} & \mathbf{A}_2^{s*} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_2 \\ \mathbf{x}_2^s \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_2^* \\ \mathbf{b}_2^{s*} \end{Bmatrix} \tag{2}$$

Here, $\mathbf{A}_i$ stands for this part of element local matrices which is related to the interactions of the element interior and boundary edges shape functions, $\mathbf{B}_i$ and $\mathbf{C}_i$ stand for these parts of element local matrices which are related to the interactions between the element interior shape functions and the common edge shape function, and $\mathbf{A}_i^s$ stands for this part of element local matrices which is related to the interactions between the common edge shape functions. Moreover, $\mathbf{x}_i$ and $\mathbf{b}_i$ stand for the degrees of freedom and the right-hand-side terms related to the element interior and boundary edges, while $\mathbf{x}_i^s$ and $\mathbf{b}_i^s$ stand for the degrees of freedom and the right-hand-side terms related to the common edge. Finally we proceed to the merging of the resulting Schur complement matrices and performing full forward elimination. The procedure is illustrated below

$$\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}} \tag{3}$$

$$\hat{\mathbf{A}} = \mathbf{P}_1 \mathbf{A}_1^{s*} \mathbf{P}_1^T + \mathbf{P}_2 \mathbf{A}_2^{s*} \mathbf{P}_2^T \tag{4}$$

$$\hat{\mathbf{b}} = \mathbf{P}_1 \mathbf{b}_1^{s*} \mathbf{P}_1^T + \mathbf{P}_2 \mathbf{b}_2^{s*} \mathbf{P}_2^T \tag{5}$$

Here, $\mathbf{P}_i$ stand for the permutation matrices transforming an element local ordering of the degrees of freedom located on the interface (the common edge in this example) into the global ordering on the interface.

$$\mathbf{P}_i : \mathbf{M}\left(n^i_{\text{interface}} \times n^i_{\text{interface}}\right) \rightarrow \mathbf{M}\left(n^i_{\text{interface}} \times n^i_{\text{interface}}\right) \tag{6}$$

Note that the system of equations (3-5) corresponds to the so-called Schur complement

$$\hat{\mathbf{A}} = \mathbf{A}_s - \mathbf{C}_I{}^T \mathbf{A}_I{}^{-1} \mathbf{B}_I \tag{7}$$

$$\hat{\mathbf{b}} = \mathbf{b}_s - \mathbf{C}_I{}^T \mathbf{A}_I{}^{-1} \mathbf{b}_I \tag{8}$$

of the global system

$$\begin{bmatrix} \mathbf{A}_1 & & & \mathbf{B}_1 \\ & \cdots & & \cdots \\ & & \mathbf{A}_p & \mathbf{B}_p \\ \mathbf{C}_1 & \cdots & \mathbf{C}_p & \mathbf{A}_s \end{bmatrix} \begin{Bmatrix} \mathbf{x}_1 \\ \cdots \\ \mathbf{x}_p \\ \mathbf{x}_s \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_1 \\ \cdots \\ \mathbf{b}_p \\ \mathbf{b}_s \end{Bmatrix} \tag{9}$$

which can be expressed in a concise form

$$\begin{bmatrix} \mathbf{A}_I & \mathbf{B}_I \\ \mathbf{C}_I{}^T & \mathbf{A}_s \end{bmatrix} \begin{Bmatrix} \mathbf{x}_I \\ \mathbf{x}_s \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_I \\ \mathbf{b}_s \end{Bmatrix} \tag{10}$$

In other words, the Schur complement matrix (7) and the corresponding right-hand-side vector (8) can be easily obtained by executing partial forward eliminations on sub-domains and summing up renumbered sub-matrices $\mathbf{A}_s{}^{i*}$ and sub-vectors $\mathbf{b}_s{}^{i*}$, which is expressed in (4) and (5).

The next step is to solve the global interface problem (3), and we obtain the interface problem solution $\hat{\mathbf{x}}$. Finally, we proceed to the global problem can be solved by executing partial backward substitutions, based on the solutions propagating from the previous level systems. These partial backward substitutions may be achieved by the replacement of the Schur complement contributions $\mathbf{A}_i^{s*}$ in (1) and (2) by the identity matrices, replacing the right-hand-side parts $\mathbf{b}_i^s$ by the obtained solution (remapped into the element local ordering of the interface degrees of freedom), and executing backward substitution on both systems.

$$\begin{bmatrix} \mathbf{U}_i & \mathbf{B}_i^* \\ \mathbf{0} & \mathbf{A}_s{}^{i*} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_i \\ \mathbf{x}_s{}^i \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_i^* \\ \mathbf{b}_s{}^{i*} \end{Bmatrix} \rightarrow \begin{bmatrix} \mathbf{U}_i & \mathbf{B}_i^* \\ \mathbf{0} & 1 \end{bmatrix} \begin{Bmatrix} \mathbf{x}_i \\ \mathbf{x}_s{}^i \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_i^* \\ \mathbf{P}_i^{-T} \mathbf{x}_s \mathbf{P}_i^{-1} \end{Bmatrix} \tag{11}$$

The procedure can be recursively generalized into a multiple level elimination tree for the case with many finite elements (many subdomains).

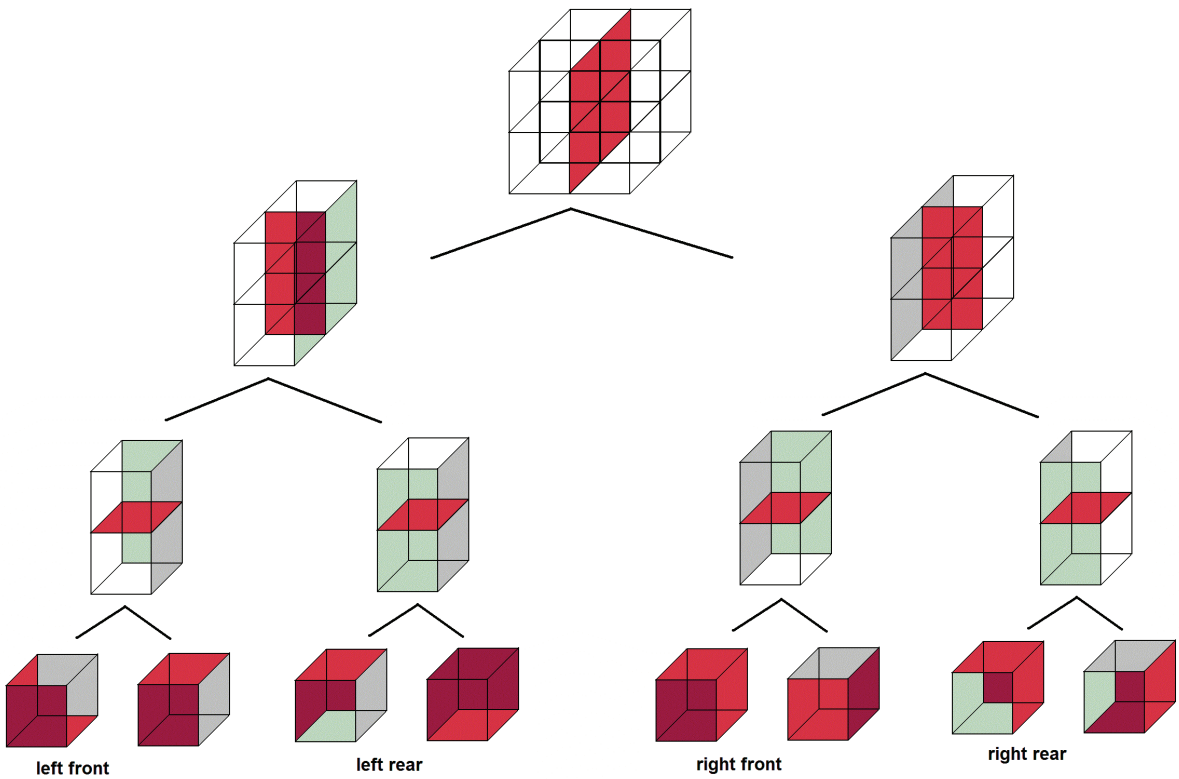

Fig. 1. The four levels of the elimination tree for a cube-shaped mesh with eight finite elements

Constructing so-called elimination tree, presented in Figure 1 for a cube-shaped finite element mesh with eight elements in it, generalizes the elimination tree to a multiple elements problem. The leaves of refinement tree act as the active finite elements. The multi-frontal direct solver algorithm browses the elimination trees, starting from the level of active elements, through the level of parent nodes, up to the root. There are basis functions associated with element vertices, edges, faces and interiors. The frontal solver creates the frontal matrices at leaf nodes. The unknowns (i.e., degrees of freedom) correspond to coefficient of basis functions related to each element. At each level of the elimination tree the fully assembled rows (and fully assembled degrees of freedom) can be eliminated from the frontal matrices. Each row has a corresponding basis function assigned. A degree of freedom from a frontal matrix is fully assembled, if the corresponding vertex, edge, or face, respectively, has been fully assembled. For example, on the level of leaves, only degrees of freedom (and their rows in the matrix) that correspond to basis functions related with element interior and boundary can be eliminated. At the parent level, only the degrees of

freedom corresponding to basis functions related the just fully aggregated common face, can be eliminated. These faces are denoted by the red color on the parent level. The procedure is recursively repeated until we reach the root of the elimination tree. At the root of the tree we can perform the full forward elimination since the top interface problem matrix corresponds to the just fully assembled cross-section of the mesh, denoted by the red color at the root node. Finally, the backward substitutions are recursively executed from the root node down to the leaves.

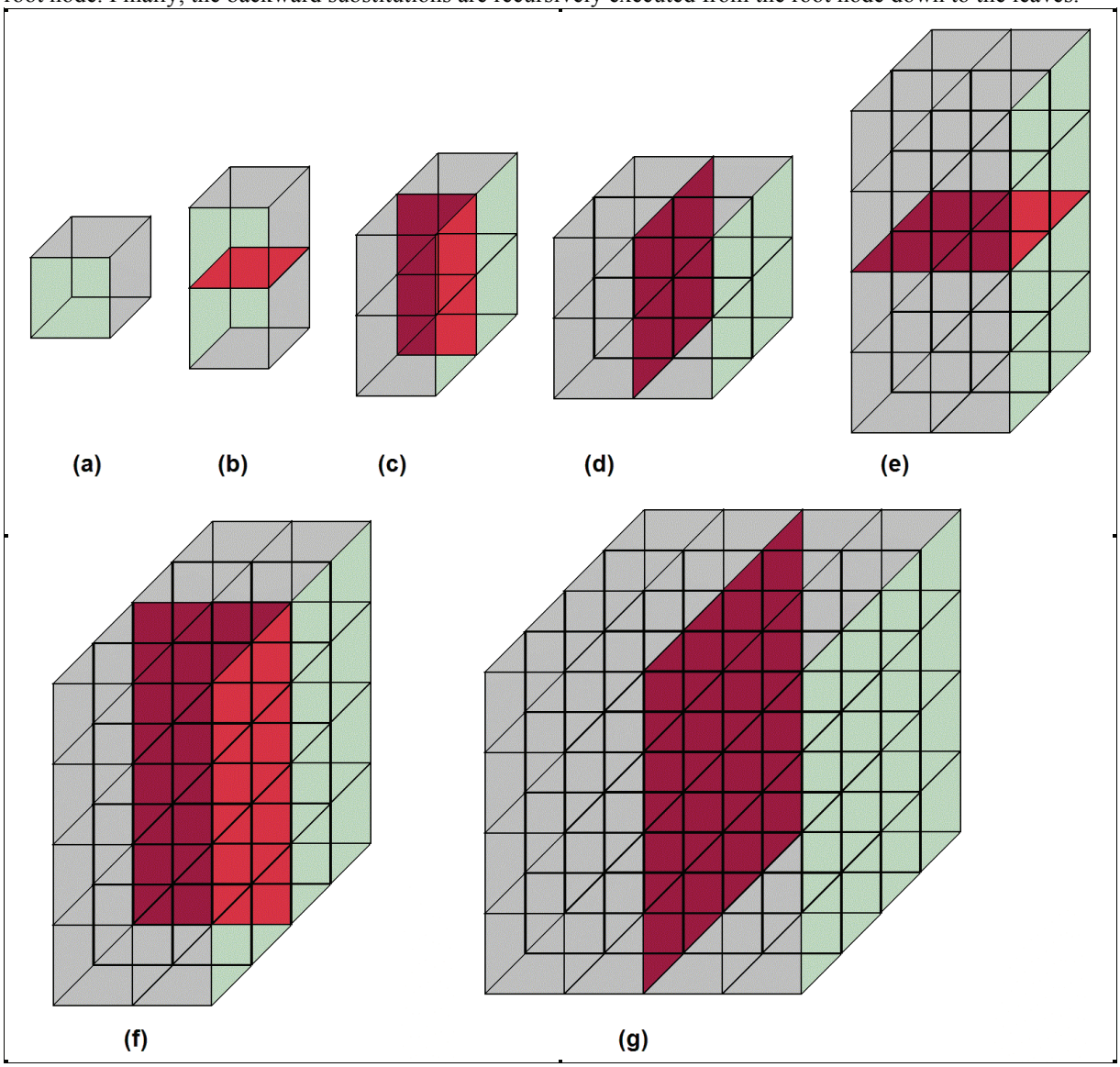

Fig. 2. The general case considered in the computational complexity and memory usage estimations **(a)** an interior of an element is eliminated **(b)** a common face shared between to adjacent elements is eliminated (interiors have been already eliminated in previous step) **(c)** two common faces and one common edge shared between two sets of two elements are eliminated (the remaining two interior faces have been already eliminated in step b) **(d)** four common faces and four common internal edges shared between two sets of four elements are eliminated (the remaining internal faces and internal edges have been already eliminated in steps b and c) **(e)** four faces and four internal edges shared between two small cubes are eliminated (the internal faces of the cubes have been already eliminated in previous steps) **(f)** eight common faces and ten common edges are eliminated (the two parts of the domain are joined, having all internal faces, edges and interiors eliminated in previous steps) **(f)** sixteen common faces as well as twenty four common edges are eliminated (the two parts of the domain are joined, having all internal faces, edges and interiors eliminated in previous steps)

3. Computational complexity of the sequential solver executed over a single *hp* finite element

Let us first estimate the number of operations performed by a sequential multi-frontal solver during forward elimination over one p finite element. This corresponds to the leaves of the elimination tree presented in Fig. 1 or to the panel a) at Fig. 2. In this estimation we assume that degrees of freedom located on the boundary have not been eliminated yet. In other words we consider a general element located inside some large domain, not the simple eight finite elements mesh like the one presented in Fig. 1. The order of approximation in the interior of the element is assumed to be equal to (p_1, p_2, p_3). Orders of approximation on element faces and edges are assumed to be equal to the corresponding orders in the interior. From this assumption it follows that there are two faces with orders (p_1, p_2), two faces with orders (p_1, p_3) and two faces with orders (p_2, p_3), as well as four edges with order p_1, four edges with order p_2 and four edges with order p_3.

The total number of degrees of freedom (d.o.f.) in such an element is equal to

$$nrdof = (p_1 + 1)(p_2 + 1)(p_3 + 1) = O(p_1 p_2 p_3)$$

To estimate the efficiency of the sequential solver, we assume that $p_1=p_2=p_3=p$ or we can take $p = \max\{ p_1, p_2, p_3 \}$. Thus, the number of d.o.f. is given by

$$nrdof = (p+1)^3 = O(p^3)$$

the number of interior d.o.f. is given by

$$interior\ nrdof = (p-1)^3 = O(p^3)$$

and the number of interface d.o.f. is given by

$$interface\ nrdof = 6p^2 + p + 1 = O(p^2)$$

Since the element stiffness matrix is dense, it follows that

$$computational\ complexity\ over\ 1\ element = (number\ of\ d.o.f.)^3$$

Thus, the computational complexity of the sequential multi-frontal solver executed on a single p element is given by

$$computational\ complexity\ over\ 1\ element = \mathrm{nrdof}^3 = O(p^9).$$

4. Computational complexity of the sequential multi-frontal solver executed over a cube formed by N_e identical *p* finite elements

The computational domain is now assumed to be a cube with $N_e=n\times n\times n$ identical hp finite elements, where n is the number of elements along a single axis. Orders of approximation are assumed to be equal to p on all elements edges, faces, and interior. In other words the number of degrees of freedom is $N=n^3p^3$. For simplicity, the number of elements n in a single direction is assumed to be equal to a power of 2.

The multi-frontal direct solver algorithm starts with leaves, Figure 2(a). It constructs the element matrices with $O(p^3)$ degrees of freedom, and eliminates internal degrees of freedom, with computational cost $O(p^9)$ (as shown in section 3). The computational complexity of elimination of the internal d.o.f. is $n^3p^6(p^3-p^2) = O(n^3p^9)$, since there are n^3 such elements.

In the next step, two elements with interiors already eliminated are joined and the degrees of freedom associated with the common face are eliminated. This corresponds to Figure 2(b). The number of degrees of freedom in the system is $(4*2+2+1)p^2$ since there are 4*2 external faces, one top and one bottom faces, as well as one internal face. The number of degrees of freedom to be eliminated is just p^2, since only the common face is eliminated. Thus the computational complexity is $O((n^3/2)(((4*2+2+1)p^2)^2)p^2)$.

In the next step, Figure 2(c), two sets of two elements are joined and the two common faces are eliminated. The total number of degrees of freedom is equal to $(4*2+4*2+2)p^2$ since there are 4*2 external faces, front and the rear with 4

faces, as well as two internal faces. The number of degrees of freedom to be eliminated is just $2p^2$, since only the two common faces are eliminated. Thus, the computational complexity is $O((n^3/2^2)(((4*2+4*2+2)p^2)^2)2p^2)$.

In the next step illustrated in Figure 2(d), two sets of four elements are joint, and the four common faces are eliminated. The total number of degrees of freedom is equal to $(4*4+4*4+4)p^2$ since there are 4*4 external faces, front and the rear with 4 faces, as well as 4 internal faces. The number of degrees of freedom to be eliminated is just $4p^2$, since only the four common faces are eliminated. The computational complexity of elimination of common d.o.f. over such a set of 2*2*2 elements is $(n^3/2^3)(((6*4+4)p^2)^2 4p^2)= O((n^3/2^3)(2^6p^6))$, since $(6*4+4)p^2$ is the number of d.o.f. on external faces on a box of 2*2*2 elements, plus 2*2=4 common faces, and common faces are eliminated. There are $n^3/2^3$ such sets of elements.

The procedure continues in steps corresponding to panels e), f) and g) in Figure 2. Notice that the computational complexity of steps (b) and (c) is not larger than the computational complexity of step (d). The same applies for all the following of steps, that is, the computational complexity of steps (e) and (f) is not larger than the computational complexity of step (g). Thus, without loss of generality, to estimate the computational complexity, we can focus on a sequence of cube-shaped meshes with $2^k*2^k*2^k$ elements, with a cross-section of internal faces to be eliminated. Thus, in our estimates we will sum up computational complexities of step (a), (d) then g) and the following cube-shape domain based steps.

Now we focus on the next step, corresponding to a set of 4*4*4 elements. The computational complexity of elimination of d.o.f. corresponding to the internal faces denoted in panel (g) by the red color is

$$(n^3/2^6)((6*16+16)p^2)^2 16p^2)=O((n^3/2^6)(2^{12}+2^{12})p^6),$$

since $(6*16)p^2$ is the number of d.o.f. on faces of a box of 4*4*4 elements, plus 4*4=16 common faces, and the common faces are eliminated. There are $n^3/2^6$ such sets of elements. Thus, total computational complexity can be estimated by the following sum

$$\textit{comp. complexity for } n^3 \textit{ elements} = n^3p^9 + \sum_{k=1}^{\log_2 n} \left(n^3/2^{3k}\right) p^6 2^{6k} = O(n^3p^9+n^6p^6) = O(N_e p^9+N_e^2 p^6) = O(N p^6+N^2)$$

5. Memory usage of the sequential solver executed over a single *p* finite element

The memory usage of the solver is equal to the square of the number of degrees of freedom over a single *p* finite element. Since nrdof = $O(p^3)$ we get

$$\textit{memory usage over 1 element} = \text{nrdof}^2 = O(p^6).$$

6. Memory usage of the sequential multi-frontal solver executed over a cube formed by N_e identical *p* finite elements

The memory usage of the sequential in-core solver is equal to the sum of sizes of all the frontal matrices constructed by the algorithm. We perform our estimations of memory usage following the template presented for the computational complexity estimates. In other words, we consider here a sequence of cube shaped meshes with $2^k*2^k*2^k$ elements, with a cross-section of internal faces to be eliminated.

It is assumed that the frontal solve algorithm is eliminating internal d.o.f. of elements, then it is joining elements into sets of 2*2*2=8 elements, eliminating common fully assembled d.o.f. located on a single cross-section, then it is joining elements into sets of 4*4*4=64 elements, eliminating common fully assembled d.o.f. from the cross-section, and the process is repeated until to the entire mesh is reduced to one global cross-section problem.

The in-core multi-frontal solver stores frontal matrices for backward substitution, so we need to multiply the memory usage by the number of elements. The front size associated with a single element is equal to the number of element d.o.f. squared, which is $O(p^6)$. Thus, the size of the all frontal matrices for all n^3 elements is $O(n^3p^6)$

When we join elements into sets of 2*2*2=8 elements, the size of the 8 elements wire-frame problem is $(6*4+4)p^2$, since $6*4p^2$ is the number of d.o.f. on external faces on a box of 2*2*2 elements, plus 2*2 p^2=4 p^2 degrees of freedom over the cross-section common faces. Thus the size of the all frontal matrices for all $n^3/2^3$ sets of 8 elements is $n^3/2^3\ ((6*4+4)p^2)^2 = O((n^3/2^3)2^4*p^4)$.

When we join elements into sets of 4*4*4=64 elements, the size of the 64 elements wire-frame problem is $(6*16+16)p^2$ since $6*16p^2$ is the number of d.o.f. on faces of a box of 4*4*4 elements, plus 4*4 p^2=16 p^2 degrees of freedom located on the cross-section common faces. Thus the size of the all frontal matrices for all $n^3/2^6$ sets of 64 elements is $n^3/2^6((6*16+16)p^2)^2 = O((n^3/2)2^8p^4)$
Thus,

$$\textit{memory usage over } n^3 \textit{ elements} = n^3p^6 + \sum_{k=1}^{\log_2 n} \left(n^3/2^{3k}\right) p^4 2^{4k} = \mathrm{O}(n^3p^6+n^4p^4)=\mathrm{O}(N_e\,p^6+N_e^{4/3}\,p^4) = O(Np^3+N^{4/3})$$

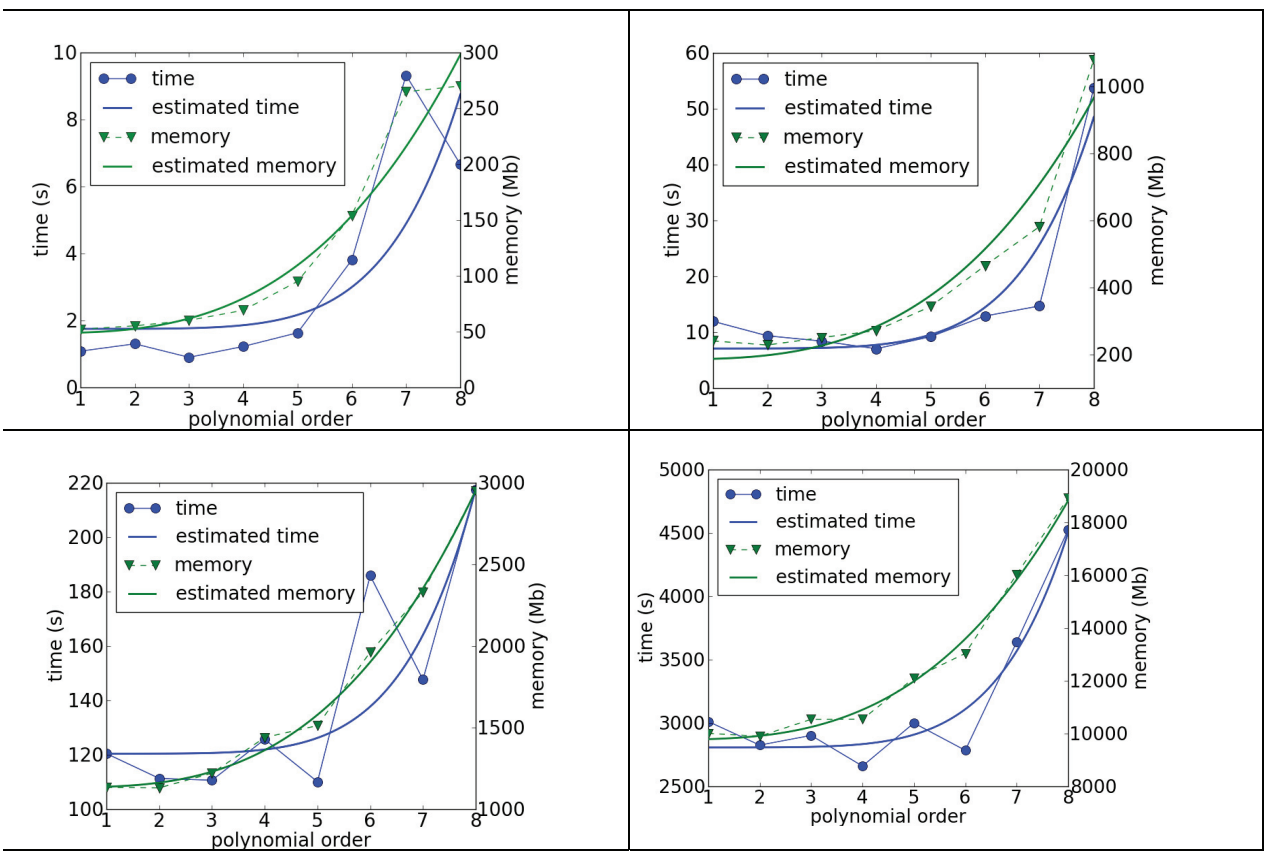

Figure 3. Plots of numerical tests run which verify the computational complexity and memory usage estimates given proposed herein. Each subplot shows the time and memory utilized as well as the statistical best fit of the data to this estimate. Each subplot represents a different level of degrees of freedom: (left top) 10,000 (right top) 30,000 (left bottom) 100,000, and (right bottom) 500,000.

7. Numerical results

In order to verify the validity of the estimates presented in sections 3-6, we setup the solution of the canonical Laplace problem on the unit cube, subject to a zero Dirichlet condition on the bottom surface and a unit Dirichlet condition on the top. While a simple, scalar problem, the idea is compare the time and memory needed to solve the resulting linear system while varying the polynomial order, number of elements, and number of degrees of freedom.

We develop a series of meshes that contain 10,000, 30,000, 100,000, and 500,000 degrees of freedom. Within each level we develop another series of meshes by varying the polynomial order from 1 to 8. The overall degrees of freedom are kept constant by generating the meshes with varying numbers of elements. The resulting linear systems were solved in core using MUMPS 4.9.2 [6,7] on a workstation with 2 quad-core Xeon X5550 processors and 24 Gb of memory.

Figure 3 presents the results of these numerical experiments. In each subfigure the time required to solve is shown in a blue solid line and the memory shown in a green dotted line. The actual data is plotted using markers and connected with a lighter line width. The important trend to note is how as the polynomial order grows in each case, the time to solve does not grow until a threshold is reached. In the 500,000 degree of freedom case p=6 is as expensive (or even slightly less expensive) to solve in terms of time as p=1. The darker line widths represent a statistical fitting of the data to the estimates. In each case, the correlation constant was between 0.8 and 0.99 indicating excellent statistical agreement.

8. Conclusions

We have presented computational complexity and memory usage estimates for the multi-frontal solver algorithm applied to the solution of linear systems resulting for structured grids with uniform polynomial order for spaces that are C^0, this kind of meshes correspond to C^0 tensor-product B-spline meshes, such as those used in isogeometric analysis. While the estimates were developed with this in mind, they apply to *p*-finite elements where the mesh is a structured grid. The results explain the interesting effect of systems resulting from higher order polynomial not costing more time to solve, reinforcing the power of the multi-frontal algorithm. Future work will expand the estimates to more appropriate C^k spaces that are of interest in isogeometric analysis, in particular, we will focus on C^{p-1} spaces, which are the maximum continuity spaces used in what is dubbed as *k*-refinement in isogeometric analysis literature.

Acknowledgements

DP has been partially supported by the Spanish Ministry of Sciences and Innovation Grant MTM2010-16511. MRP has been partially supported by the Polish MNiSW grant no. NN 519 405737 and NN519 447 739.

References

1. Duff I. S., Reid J. K., 1983: *The multifrontal solution of indefinite sparse symmetric linear systems*. ACM Transactions on Mathematical Software, 9, 302-325

2. Duff I. S., Reid J. K., 1984: *The multifrontal solution of unsymmetric sets of linear systems*. SIAM Journal on Scientific and Statistical Computing, 5, 633-641

3. Irons B., 1970: *A frontal solution program for finite-element analysis*. International Journal of Numerical Methods in Engineering, 2, 5-32

4. T. J. R. Hughes, J. Cottrell, Y. Basilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering. 194 (2005) 4135-4195.

5. J. Cottrell, T. J. R. Hughes, Y. Basilevs, Isogeometric analysis: Toward Unification of CAD and FEA, John Wiley and Sons, 2009.

6. 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. 23 (2001) 15-41.

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