Direct Solvers Performance on h-Adapted Grids

Maciej Paszynski[a], David Pardo[b,c,d], Victor M. Calo[e]

[a] *AGH University of Sciences and Technology, Krakow, Poland*
[b] *Department of Applied Mathematics, Statistics, and Operational Research at the University of the Basque Country (UPV/EHU), Bilbao, Spain*
[c] *Basque Center for Applied Mathematics (BCAM), Bilbao, Spain*
[d] *Basque Foundation for Science (IKERBASQUE), Bilbao, Spain*
[e] *King Abdullah University of Sciences and Technology, Thuwal, Saudi Arabia*

Abstract

We analyze the performance of direct solvers when applied to a system of linear equations arising from an h-adapted, C^0 finite element space. Theoretical estimates are derived for typical h-refinement patterns arising as a result of a point, edge, or face singularity as well as boundary layers. They are based on the elimination trees constructed specifically for the considered grids. Theoretical estimates are compared with experiments performed with MUMPS using the nested-dissection algorithm for construction of the elimination tree from METIS library. The numerical experiments provide the same performance for the cases where our trees are identical with those constructed by the nested-dissection algorithm, and worse performance for some cases where are trees are different. We also present numerical experiments for the cases with mixed singularities, where how to construct optimal elimination trees is unknown. In all analyzed cases, the use of h-adaptive grids significantly reduces the cost of the direct solver algorithm *per unknown* as compared to uniform grids. The theoretical estimates predict and the experimental data confirm that the computational complexity is linear for various refinement patterns. In most cases, the cost of the direct solver *per unknown* is lower when employing anisotropic refinements as opposed to isotropic ones.

Keywords: direct solver, h-version refinement, finite element method, singularities

1. Introduction

When solving an engineering problem using the finite element method, the following tasks need to be executed:

- *Geometry definition.*
- *Problem formulation* in terms of the weak variational form associated to the Partial Differential Equation (PDE) and boundary conditions governing the problem.
- *Initial mesh generation and corresponding discretization* leading to a system of linear equations.
- *Solution* of the system of linear equations.

When the error delivered by a discretization exceeds a given tolerance, an *adaptation* (or grid enrichment) step is performed in order to reduce the error in the areas exhibiting large errors. As a result, several iterations of the *solution* and *adaptation* steps may be needed to achieve the required accuracy of the numerical solution.

The refinements occurring during the adaptation process are often intended to minimize the error produced by numerical singularities or boundary layers towards a number of points, edges, and/or faces. This generates a sequence of meshes refined towards solution singularities.

Since the *solution* step is typically the most expensive one in terms of both memory and CPU time, in this paper we focus our attention on the performance of direct solvers of linear equations in the presence of adapted grids. Although direct solvers are often slower than iterative ones, they exhibit a number of advantages, and thus, their study is of great importance. Among those advantages, we emphasize that they enable for local static condensation [9] or even reutilization of partial LU factorizations under local refinements [25, 24]. Direct solvers are also convenient when solving problems with multiple right-hand-sides, since the LU factorization only needs to be computed once. They are particularly important for ill-conditioned or indefinite problems, where iterative solvers encounter serious problems with convergence. Examples of these problems are multi-physics problems or highly non-regular meshes with elements having large aspect ratios, which often result from the execution of adaptive algorithms.

The multi-frontal solver is the state-of-the-art algorithm for solving linear systems of equations [14, 17] using a direct solver. It is a generalization of the frontal solver algorithm proposed by [13, 19]. The multi-frontal algorithm constructs an assembly tree based on the analysis of the connectivity data of the computational mesh. Finite elements are joined into pairs and fully assembled unknowns are eliminated within the frontal matrices associated to multiple branches of the tree. The process is repeated until the root of the assembly tree is reached. Finally, the common interface problem is solved and partial backward substitutions are recursively called on the assembly tree.

When employing a direct (multi-frontal) solver of linear equations on a regular FE grid with the same number of elements in each spatial direction, the time complexity scales asymptotically as $\mathcal{O}(N)$ in 1D, $\mathcal{O}(N^{1.5})$ in 2D, and $\mathcal{O}(N^2)$ in 3D, where N is the number of unknowns [10]. When the number of elements is increased only in one or two spatial directions, then the corresponding time complexity scales as $\mathcal{O}(N)$ and $\mathcal{O}(N^{1.5})$, respectively.

While direct solvers have been widely studied in context of regular FE grids, their performance on h-adapted grids has been ignored. Local h-refinements are essential to solve a variety of engineering problems [20, 22, 6, 26], and different h-adaptive versions have been designed for that purpose [5, 7, 16, 8, 23]. These local h-refinements greatly reduce the number of unknowns needed to solve a given problem with a prescribed error tolerance. However, there is no previous study on how such adaptive strategies affect to the cost of solving (using a direct solver) the resulting system of linear equations. This is a critical issue since, for example, in the case of iterative solvers, it is well known that local h-refinements may quickly increase the condition number of the resulting matrix [1].

In this paper, we study the performance of direct solvers in h-adapted grids. We consider six basic refinement patterns, namely, uniform refinements, refinements towards a point, isotropic and anisotropic refinements towards an edge, as well as isotropic and anisotropic refinements towards a face. For all these cases we construct specific elimination trees that result in linear computational cost $\mathcal{O}(N)$ of the direct solver algorithm, except for the uniform case where we obtain quadratic cost $\mathcal{O}(N^2)$, and the isotropic face case, where we obtain $\mathcal{O}(N^{3/2})$ cost. We compare these experimental results with numerical experiments performed with MUMPS solver [2, 3, 4] using the nested-dissection algorithm from METIS [21] library for construction of the elimination tree. The numerical results confirm the costs obtained from theoretical analysis, except for some cases where they deliver worse performance.

Our results indicates that there is a space for improvement in the design of algorithms for construction of the elimination trees, often called the ordering algorithms, for the h-refined grids, since a state-of-the-art algorithm like nested-dissection, proven to be optimal for the uniform grid, does not provide the optimal elimination trees for some h-refined grids.

We also present the numerical results for the cases with mixed singularities, namely point and edge singularity, face and edge singularity, point and face singularity, as well as point plus edge plus face singularity. For these cases, we do not know how to construct theoretically optimal elimination trees, so we only provide the numerical experiments with MUMPS and nested-dissection algorithm from METIS library.

Using numerical results supported and interpreted via theoretical estimates for some selected cases, we prove that the use of h-adapted grids may significantly reduce the cost *per unknown* of solving the resulting system of linear equations. Moreover, we show that the use of anisotropic refinements as opposed to isotropic refinements further reduces such a cost per unknown.

Thus, this paper advocates the use of adaptive techniques to reduce the number of unknowns needed to solve a given problem as well as to significantly reduce the computational cost per unknown of solving the resulting system of equations when using a direct solver. This situation is the opposite as the one that occurs with iterative solvers, where adaptive grids may rapidly increase the condition number of the resulting system. This also suggests that in certain situations, it may be convenient to use partial LU factorization in certain parts of the domain before calling an iterative solver.

We only consider direct solvers in the sense that they provide the exact solution up to round-off error in one step. In particular, we do not analyze H-matrices based solvers such as [27, 28] or correction schemes such as [15].

The structure of the paper is the following. We start with Section *Method and Assumptions* describing the numerical framework we used in the experiments. In the next Section, entitled *Theoretical Complexity Estimates*, we derive the computational cost estimates for different kinds of refined meshes, and we verify the estimates in Section *Numerical Results*. We summarize the main results of the paper in the *Conclusions* Section.

2. Method and Assumptions

In this Section, we describe the method and assumptions made in this study to estimate the computational complexity of a direct solver when applied to h-adapted grids.

These h-adaptive grids are typically intended to capture rapid variations in the solution (or its derivatives) in certain areas of the domain. Often, this results in refinements towards a certain entity of the domain, namely, a point, an edge, a face, or a combination of them. For simplicity, and according to the above classification, we study the following cases:

- Reference case: Uniform refinements with equal number of elements in each direction.
- Refinement patterns towards one entity: a point, an edge, or a face.
- Refinement patterns towards multiple entities: (a) a point and an edge, (b) a point and a face, (c) an edge and a face, and (d) a point, an edge, and a face.

We consider both isotropic and fully anisotropic refinements. For simplicity, the cases of partially anisotropic refinements or multiple singularities are omitted in this study.

The study combines theoretical estimates with numerical results. For the numerical results, we employ the MUMPS solver [2, 3, 4] with the ordering of unknowns provided by METIS [21], which is considered among the best existing direct solvers for finite element method (FEM).

For some cases of refinements towards a single entity, even when theoretical results are available, we provide numerical results to study the existing agreement between theory and practice. Many factors, including possible imperfections in the solver, late asymptotics, and so on may distort the matching between the theoretical estimates and the actual results.

To facilitate the discussion, we consider a three-dimensional Laplace problem with Dirichlet and Neumann boundary conditions, namely:

$$\text{Find } u = u(x,y,z) \in H^1(\Omega) \text{ such that } \Delta u = 0, \tag{1}$$

where $\Omega = (0,1)^3$, with boundary conditions

$$u(:,:,0) = 0, \tag{2}$$

$$u\left(:,:,1\right)=1, \tag{3}$$

$$\frac{\partial u}{\partial x}\left(0,:,:\right)=\frac{\partial u}{\partial x}\left(1,:,:\right)=\frac{\partial u}{\partial y}\left(:,0,:\right)=\frac{\partial u}{\partial y}\left(:,1,:\right)=0. \tag{4}$$

The weak variational formulation is obtained by taking the L^2 inner product with functions $v \in H^1\left(\Omega\right)$, integrating by parts, and performing the shift $u \in \tilde{u} + H^1\left(\Omega\right)$ where $\tilde{u}\left(x,y,z\right)=z$.

$$\text{Find } u \in V = \left\{u \in H^1\left(\Omega\right) : u\left(:,:,0\right)=u\left(:,:,1\right)=0\right\} \text{ such that} \tag{5}$$

$$b\left(u,v\right)=l\left(v\right), \forall v \in V, \tag{6}$$

$$b\left(u,v\right)=\int_\Omega \nabla u \cdot \nabla v dV, \tag{7}$$

$$l\left(v\right)=-\int_\Omega \frac{\partial v}{\partial z} dV. \tag{8}$$

The numerical experiments presented in this paper were obtained using a DELL T7500 Workstation with 96 GB RAM, equipped with Intel(R) Xeon(R) CPU E5620 operating at 2.40GHz. All results were executed sequentially in a single core.

3. Theoretical Complexity Estimates

In this Section, we derive theoretical complexity estimates for various refinement patterns: (a) uniform, (b) towards a point, (c) towards an edge, and (d) towards a face. We consider the cases of isotropic and fully anisotropic refinements.

To obtain these estimates, we first recall the complexity of a Schur complement operation. Then, we provide a general and simple framework for calculating the complexity of multi-frontal solvers given a refinement pattern and the corresponding elimination order. Finally, we derive precise estimates for each of the considered cases by selecting a particular elimination ordering. These orderings are known to be asymptotically optimal for the case of uniform refinements. Additionally, when refinements occur towards a point, or an edge, the selected orderings are also asymptotically optimal, since the resulting cost scales linearly with respect to the number of unknowns. For the case of isotropic refinements towards a face, we cannot prove theoretically that the selected ordering is asymptotically optimal, although numerical results exhibit a good matching with the theoretical estimates.

3.1. Schur complement

First, we analyze the time and memory complexity of performing the Schur complement, which is the main building block for construction of a multi-frontal solver.

Let matrix A be decomposed as:

$$A = \begin{bmatrix} \mathrm{B} & \mathrm{C} \\ \mathrm{D} & \mathrm{E} \end{bmatrix}. \tag{9}$$

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

$$A = \begin{bmatrix} \mathrm{I} & 0 \\ \mathrm{DB}^{-1} & \mathrm{I} \end{bmatrix} \begin{bmatrix} \mathrm{B} & 0 \\ 0 & \mathrm{E\text{-}DB}^{-1}\mathrm{C} \end{bmatrix} \cdot \begin{bmatrix} \mathrm{I} & \mathrm{B}^{-1}\mathrm{C} \\ 0 & \mathrm{I} \end{bmatrix}. \tag{10}$$

The term E-DB^{-1}C denoted the Schur complement.

In order to estimate the number of floating point operations (FLOPS) required to perform the above partial LU factorization, we denote by q to the dimension of squared matrix B and by r to the number of nonzero entries in each row of C and column of D. We assume that r is a constant. Then, we have:

$$\mathrm{FLOPS} = \mathcal{O}(q^3 + q^2 r + q r^2) = \mathcal{O}(q^3 + q r^2). \tag{11}$$

3.2. Multi-frontal solvers

We divide our computational domain in $N_p = N_p(0)$ patches, where each patch is composed of one or several elements. The idea of the multi-frontal solver is to join at each step i, the existing $N_p(i)$ patches into $N_p(i+1)$ new patches. The number of FLOPS needed to perform the Schur complement at the i-th step for each patch is denoted by $S(i)$. The total number of steps is denoted by s. Thus, $N_p(s) = 1$, $S(s) = 0$. Then, the total FLOPS is given by:

$$\sum_{i=0}^{s-1} N_p(i) \cdot S(i), \tag{12}$$

Following the notation of the previous subsection on the Schur complement, we define $q = q(i)$ as the number of interior unknowns of each path at the i-th step, and $r = r(i)$ as the number of interacting unknowns at the i-th step.

3.3. Uniform refinements

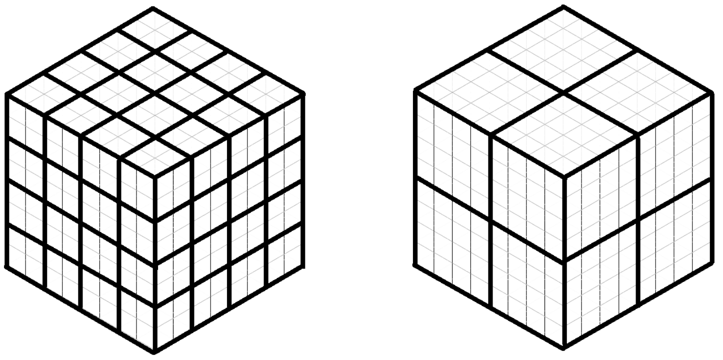
This case is illustrated in Figure 1. Each patch consists of a single element. We assume for simplicity that the number of elements N_e of our computational domain is $(2^s)^d$, where s is an integer, and d is the problem dimension. Even if this assumption is not verified, the final result still holds true. At every step, we join 2^d patches (elements) into one to produce $(2^{s-1})^d$ new patches, eliminate interior (fully assembled) unknowns of each new patch, and continue with the recursive procedure by joining again 2^d patches into one. Thus, $N_p(i) = (2^{s-i})^d$. We have $q(i) = r(i) = \mathcal{O}(2^{(d-1)i})$. Thus, $S(i) = \mathcal{O}(2^{3(d-1)i})$. We obtain:

$$\begin{aligned}
\text{FLOPS (1D)=} \quad & \sum_{i=0}^{s-1} \mathcal{O}(2^{s-i}) = \mathcal{O}(2^s) = \mathcal{O}(N_e) = \mathcal{O}(Np), \\
\text{FLOPS (2D)=} \quad & \sum_{i=0}^{s-1} \mathcal{O}(2^{2(s-i)}2^{3i}) = \mathcal{O}(2^{3s}) = \mathcal{O}(N_e^{1.5}) = \mathcal{O}(N^{1.5}p^3), \\
\text{FLOPS (3D)=} \quad & \sum_{i=0}^{s-1} \mathcal{O}(2^{3(s-i)}2^{6i}) = \mathcal{O}(2^{6s}) = \mathcal{O}(N_e^2) = \mathcal{O}(N^2p^6).
\end{aligned} \tag{13}$$

In the above formulae we included the polynomial order of approximation p assumed to be uniform over the entire grid. Following the definition of hierarchical basis functions [11] we have $N_e = Np$ in 1D, $N_e = Np^2$ in 2D and $N_e = Np^3$ in 3D. The derivation for the uniform meshes is similar to those presented in our previous papers [10].

3.4. Refinements towards a point

This case is illustrated in Figure 2. We assume that at the beginning we perform the static condensation (elimination of interior degrees of freedom inside each element). The number of interior degrees of freedom inside each element is of the order of $\mathcal{O}(p^3)$. Thus, the computational complexity of the static condensation over a single element is of the order of $\mathcal{O}((p^3)^2p^3)) = \mathcal{O}(p^9)$, and the total computational complexity of static condensation for all elements is of the order of $\mathcal{O}(p^9N_e) = \mathcal{O}(p^6N)$ since the number of elements $\mathcal{O}(Ne) = \mathcal{O}(N/p^3)$. Each patch consists of a layer of elements. At each step i, we consider one patch composed of the smallest, un-eliminated elements, which are equally sized, *i.e.*, $N_p(i) = 1 \quad \forall i$. The remaining elements are untouched at this step. The number of degrees of freedom in each patch is



(a) Step (level) 1 (b) Step (level) 2

Figure 1: Uniform Refinements. Patches to be eliminated at each step (level) are marked in dark black.

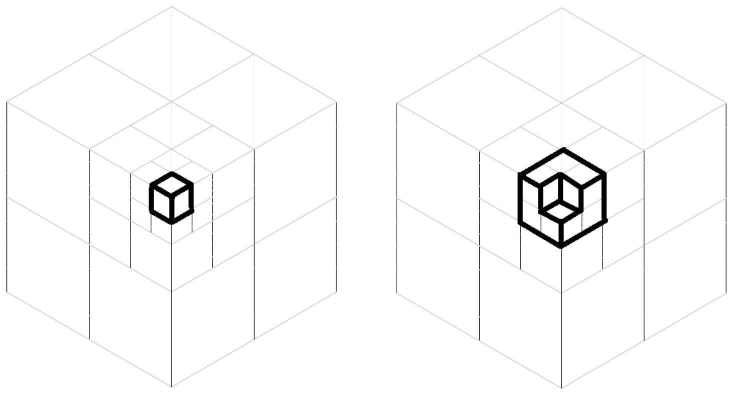
of order $O(p^3)$. The number of interface degrees of freedom in each patch is of the order of $\mathcal{O}(p^2)$. Thus, the computational complexity of the Schur complement within each patch is of the order of $\mathcal{O}((p^2)^2p^2) = O(p^6)$. The number of patches is of the order of $\mathcal{O}(s) = \mathcal{O}(N_e) = \mathcal{O}(N/p^3)$. Thus, the total computational complexity is of the order of

$$\text{FLOPS (3D)} = \mathcal{O}(p^6 s) = \mathcal{O}(p^6 N/p^3) = \mathcal{O}(Np^3). \tag{14}$$

This means that the LU factorization $\mathcal{O}\left(Np^3\right)$ is actually less expensive than static condensation of the interior degree of freedom $\mathcal{O}\left(Np^6\right)$.

3.5. Refinements towards an edge

Isotropic refinements. This case is illustrated in Figure 3. We assume that at the beginning we perform the static condensation of all interior-to-the-element degrees of freedom with a computational complexity of $\mathcal{O}(N/p^6)$. In this case we need to apply the multi-frontal elimination pattern, which constructs larger and larger patches of elements, with some edges and faces already eliminated in the previous patch. The multi-frontal solver perform $k = 1, ..., s$ steps, where s = number of layers in the mesh. The number of degrees of freedom in a patch is $\mathcal{O}(kp^2)$, where k is the step number. The number of patches in a single layer is $\mathcal{O}(2^{s-k})$. The number of interface degrees of freedom is $\mathcal{O}(kp^2)$. The computational complexity of the Schur complement of a single layer is of the order of $\mathcal{O}(2^{s-k}k^3p^6)$. For s=number



(a) Step (level) 1 (b) Step (level) 2

Figure 2: Refinements towards a point. Patches to be eliminated at each step (level) are marked in dark black.

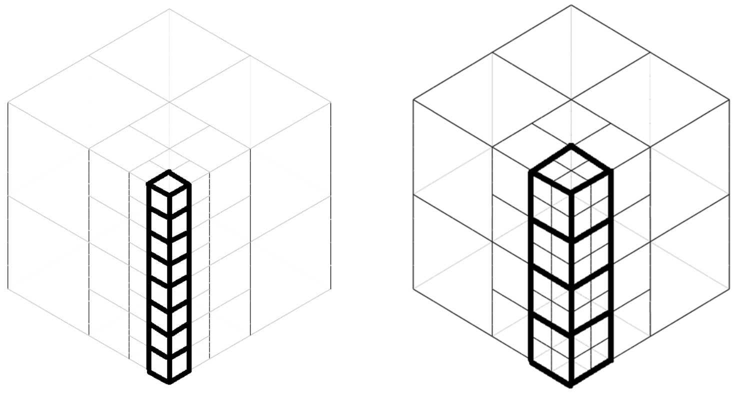
of layers, we have $N = \mathcal{O}\left(\sum_{k=1}^{s} 3 * 2^k p^3\right) = \mathcal{O}\left(\sum_{k=1}^{s} 2^k * p^3\right) = \mathcal{O}\left(p^3 2^s\right)$. Thus, the computational complexity of LU factorization is

$$\text{FLOPS (3D)}= \mathcal{O}\left(\sum_{k=1}^{s} 2^{s-k} k^3 p^6\right) = \mathcal{O}\left(p^6 2^s\right) = \mathcal{O}\left(Np^3\right). \tag{15}$$

Again, the LU factorization $\mathcal{O}\left(Np^3\right)$ is actually less expensive than static condensation $\mathcal{O}\left(Np^6\right)$.

Anisotropic refinements. This case is illustrated in Figure 4. The mesh is constructed as follows. We start from a single element and break it into four elements. In the next step, we select one element adjacent to the edge, and break it again into four elements. The situation is analogous to the refinement towards a vertex in two spatial dimensions. As usual we assume that we perform static condensation with a computational complexity of $\mathcal{O}(N/p^6)$. We have a sequence of layers, each one with three elongated elements (except the first layer where we have only one elongated element) The number of degrees of freedom in each layer is of order $\mathcal{O}(p^2)$. The number of degrees of freedom located on the interface between layers is of order $\mathcal{O}(p^2)$. Thus, the computational complexity of the construction of the Schur complement within each layer is of order $\mathcal{O}((p^2)^2 p^2) = \mathcal{O}(p^6)$. Since the number of layers is $s = \mathcal{O}(N_e) = \mathcal{O}(N/p^3)$, the computational complexity of the LU factorization is

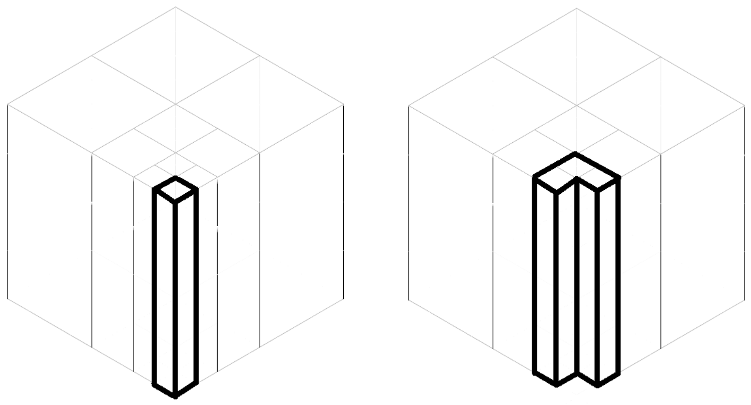
$$\text{FLOPS (3D)}= \mathcal{O}(p^6 s) = \mathcal{O}(p^6 N/p^3) = \mathcal{O}(Np^3). \tag{16}$$



(a) Step (level) 1 (b) Step (level) 2

Figure 3: Isotropic refinements towards an edge. Patches to be eliminated at each step (level) are marked in dark black.

This time LU factorization $\mathcal{O}\left(Np^3\right)$ is less expensive than static condensation $\mathcal{O}\left(Np^6\right)$.



(a) Step (level) 1 (b) Step (level) 2

Figure 4: Anisotropic refinements towards an edge. Patches to be eliminated at each step (level) are marked in dark black.

3.6. Refinements towards a face

Isotropic refinements. This case is illustrated in Figure 5. In this case we apply the multi-frontal elimination pattern again. The multi-frontal solver perform $k = 1, ..., s$ steps, where $s =$ number of layers in the mesh. The number of degrees of freedom in a patch is $\mathcal{O}\left(2^k p^2\right)$. The number of patches in a single layer is $\mathcal{O}(2^{2(s-k)})$. The number of interface degrees of freedom is $\mathcal{O}(2^k p^2)$. The computational complexity of Schur complement of a single layer is then $\mathcal{O}(2^{2(s-k)} 2^{3k} p^6)$. For s=number of layers, we have $N = \mathcal{O}\left(\sum_{k=1}^{s} 2^{2k} * p^3\right) = \mathcal{O}\left(p^3 2^{2s}\right)$. Thus, the computational complexity of LU factorization is

$$\text{FLOPS (3D)}= \mathcal{O}\left(\sum_{k=1}^{s} 2^{2(s-k)} 2^{3k} p^6\right) = \mathcal{O}\left(p^6 2^{3s}\right) = \mathcal{O}\left(N^{1.5} p^{1.5}\right). \quad (17)$$

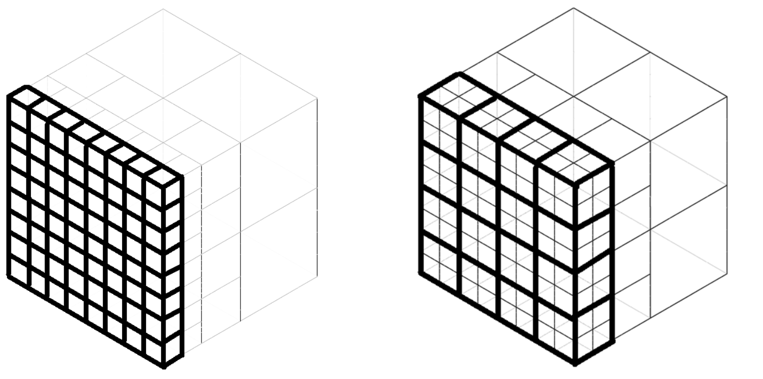

(a) Step (level) 1 (b) Step (level) 2

Figure 5: Isotropic refinements towards a face. Patches to be eliminated at each step (level) are marked in dark black.

Anisotropic refinements. This case is illustrated in Figure 6. The mesh is constructed as follows. We start from a single element and break it into two elements. In the next step, we select one element adjacent to the face and break it again into two elements. The situation is analogous to the case of performing refinements towards a point in 1D spatial dimensions. We first perform the static condensation with a computational complexity $\mathcal{O}(N/p^6)$. spatial dimensions. We have a sequence of layers, each one with one flat 2D element. The number of degrees of freedom over each layer is of the order of $\mathcal{O}(p^2)$. The number of degrees of freedom located on the interface between layers is of the order of $\mathcal{O}(p^2)$. Thus, the computational complexity of the Schur-complement construction within each layer is of order $\mathcal{O}((p^2)^2p^2) = \mathcal{O}(p^6)$. Since the number of layers is $s = \mathcal{O}(N_e) = \mathcal{O}(N/p^3)$, the computational complexity of the LU factorization is

$$\text{FLOPS (3D)} = \mathcal{O}(p^6 s) = \mathcal{O}(p^6 N/p^3) = \mathcal{O}(Np^3). \tag{18}$$

Thus, the LU factorization $\mathcal{O}(Np^3)$ is less expensive than static condensation $\mathcal{O}(Np^6)$ in this case.

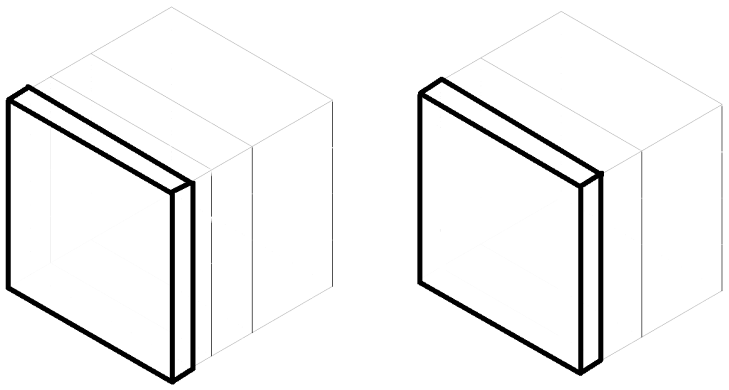

(a) Step (level) 1 (b) Step (level) 2

Figure 6: Anisotropic refinements towards a face. Patches to be eliminated at each step (level) are marked in dark black.

4. Numerical Results

In the first part of this section, we report numerical experiments for six basic kinds of refinement patterns: uniform refinements, refinements towards a point, an edge (isotropic and anisotropic cases), and a face (isotropic and anisotropic cases). For all these cases, numerical results are compared with the theoretical estimates provided in the previous section. In the second part of this section, we consider the following mixed refinement patterns: (a) towards both a point and an edge, (b) towards a point and a face, (c) towards an edge and a face, and (d) towards a point, an edge, and a face.

All numerical results reported in this section has been performed with MUMPS solver using the nested-dissection algorithm from METIS library.

4.1. Refinements towards a single entity: a point, an edge, or a face

Following the theoretical estimates described in the previous section, we assume that the FLOPS required to solve the linear systems of equations scales as N^{α}, where α is an unknown coefficient. Thus, when displaying numerical results in the FLOPS vs $\log N$ scale, we expect to observe straight lines. This is confirmed by the numerical results.

Table 1 displays the exponents α computed using linear regression on the data collected experimentally. We consider various orders of approximation, namely, $p = 2, 4, 6$. We now examine each case in more detail.

	Uniform	Point	Edge Isotropic	Edge Anisotropic	Face Isotropic	Face Anisotropic
$p = 2$	1.86	1.09	1.10	1.07	1.47	1.01
$p = 4$	1.86	1.17	1.18	1.07	1.47	1.12
$p = 6$	1.83	1.08	1.21	1.09	1.54	1.08
Theoretical	2	1	1	1	1.5	1

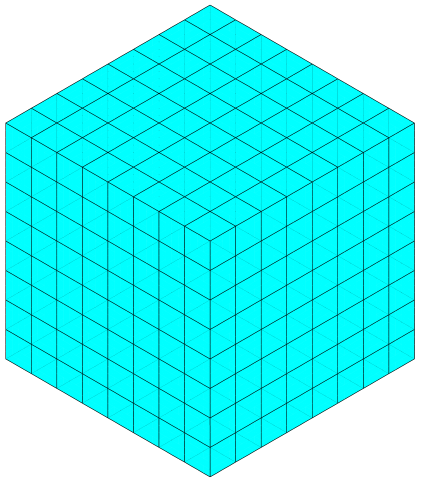
Table 1: Comparison of numerical vs. theoretical scalability exponent factors α for refinements towards a single entity.

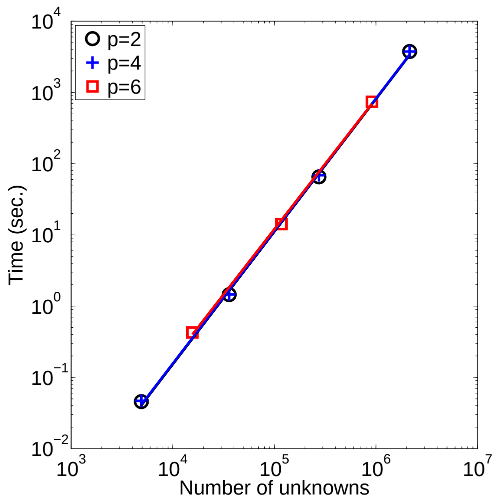
4.1.1. Uniform Refinements

Numerical experiments (see Table 1) exhibit a slightly better performance than the theoretical asymptotic scalability estimate $\mathcal{O}(N^2)$. For high p, the computational complexity in terms of FLOPS scales as $\mathcal{O}(Np^6) + \mathcal{O}(N^2)$ (see [10]). Thus, for a relatively low number of degrees of freedom and high polynomial order of approximation, the first term diminishes the exponent factor. In other words, the $\mathcal{O}(N^2)$ scalability is only obtained in the asymptotic regime, *i.e.*, for very large problems. When performing the linear regression with only the data corresponding to the three largest grids of Figure 7b, we obtain an exponent factor for $p = 2$ and $p = 4$ equal to 1.92 rather than 1.86.

4.1.2. Refinements Towards a Point

Figure 8b displays the number of FLOPS required to solve the problem as a function of N. The resulting scalability is slightly worse than that provided by the theoretical estimates. The ordering provided by METIS used in the numerical experiments is known to be asymptotically optimal


(a) Grid



(b) LU factorization time

Figure 7: Uniform refinements.

(up to a constant) for uniform grids [12], but not for highly non-uniform grids, as the ones appearing as a result of performing refinement towards a point. In any case, results are quite close to those predicted by the theory.

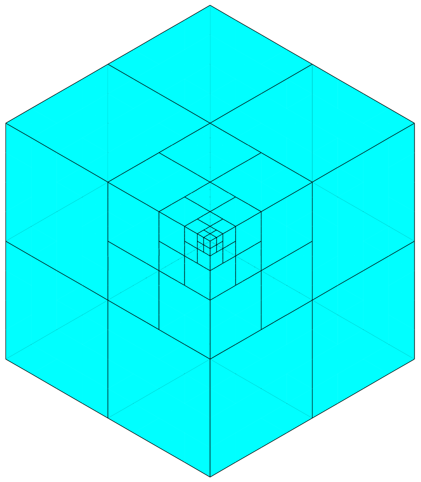
4.1.3. Refinements Towards an Edge

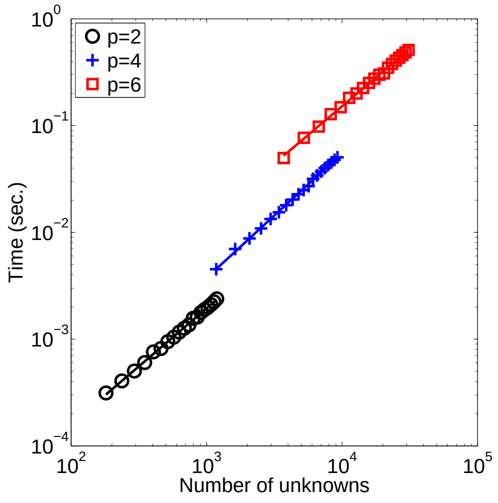
Isotropic Refinements. Numerical results displayed in Figure 9b and the third column of Table 1 show that the scalability in terms of FLOPS is worse than that predicted by the theory. However, by performing the linear regression with only the points corresponding to the three largest grids, the exponent factor is reduced to 1.01, 1.05, and 1.10 for $p = 2$, $p = 4$, and $p = 6$, respectively. This indicates that nested-dissection does not provide optimal ordering for this case.

Anisotropic Refinements. This case is similar to that consisting of performing refinements towards a point over a 2D mesh, and the corresponding computational complexity is linear $\mathcal{O}(N)$. Numerical results displayed in Figure 10b and the fourth column of Table 1 confirm the theoretical estimates.

4.1.4. Refinements Towards a Face

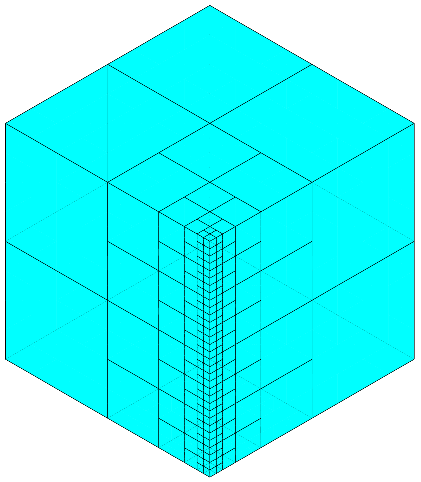
Isotropic Refinements. For this case, we could not prove that our theoretical scalability estimates are optimal, since there is always a possibility of find-

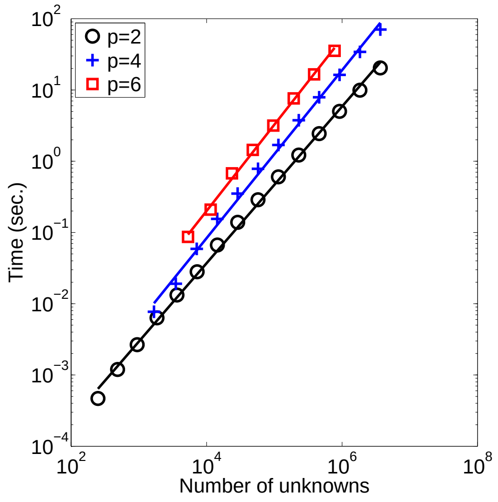

(a) Grid



(b) LU factorization time

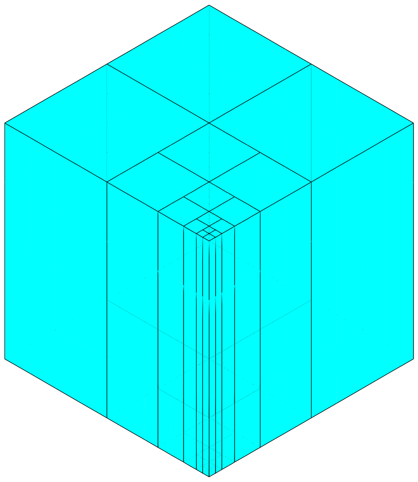
Figure 8: Refinements towards a point.


(a) Grid

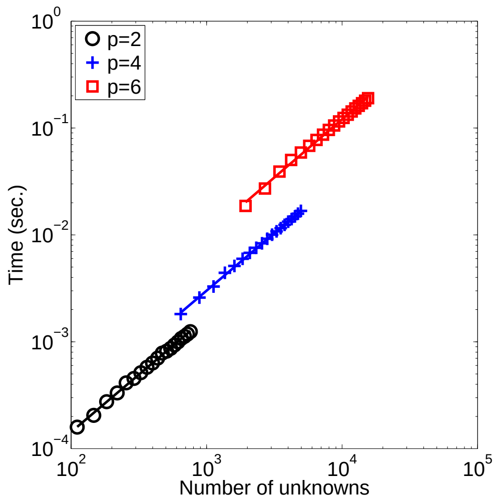


(b) LU factorization time

Figure 9: Isotropic refinements towards an edge.



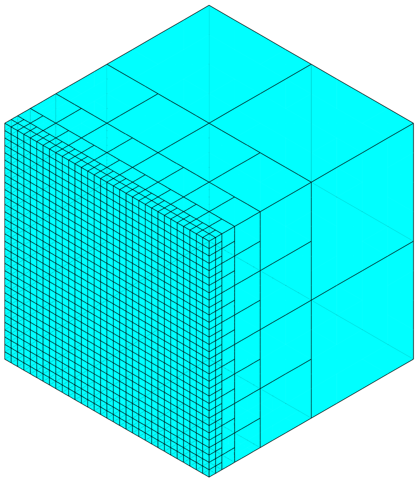
(a) Grid



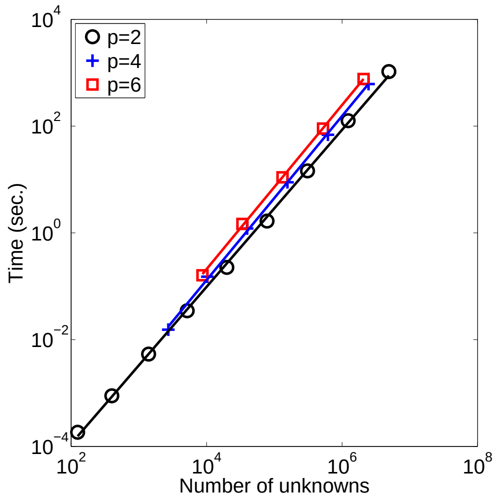
(b) LU factorization time

Figure 10: Anisotropic refinements towards an edge.

ing a better elimination ordering. However, numerical results displayed in Figure 11b and the fifth column of Table 1 agree well with the theoretical estimates.



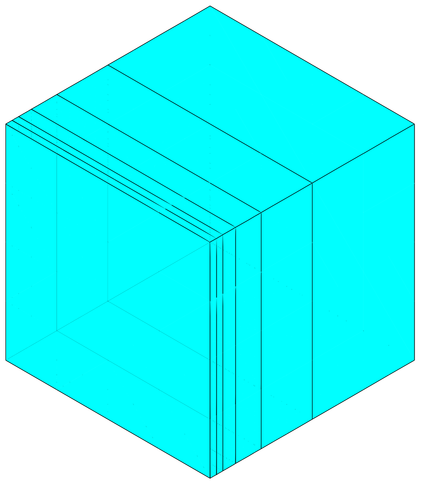
(a) Grid

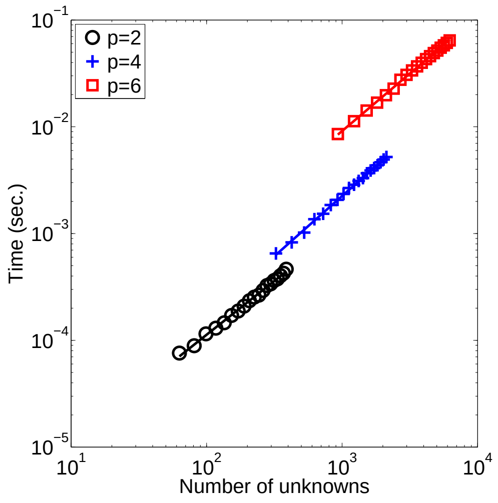


(b) LU factorization time

Figure 11: Isotropic refinements towards a face.

Anisotropic Refinements. This case is equivalent to that consisting of performing refinements over a 1D mesh, where we can easily construct an elimination tree with linear computational complexity. Numerical results displayed in Figure 12b and the sixth column of Table 1 confirm the theoretical estimates, especially, for low p where the asymptotic regime is attained at an earlier stage.


(a) Grid



(b) LU factorization time

Figure 12: Anisotropic refinements towards a face.

4.2. Refinements towards several singularity

Assuming again a FLOPS scalability of the form N^{α}, where α is an unknown coefficient, Table 2 displays the exponents α computed by linear regression over the numerical results for several mixed refinement patterns. Again, we consider the following orders of approximation: $p = 2, 4, 6$.

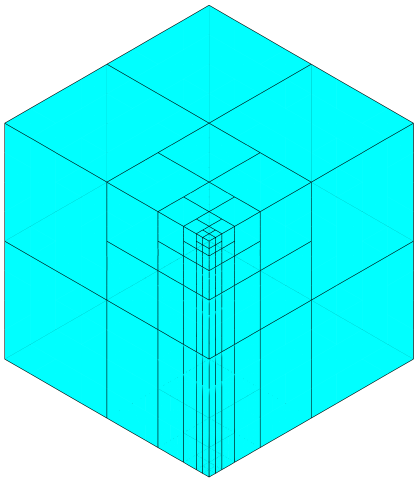
	Point + Edge	Point + Face	Edge + Face	Point + Edge + Face
$p = 2$	1.33	1.46	1.24	1.57
$p = 4$	1.45	1.60	1.35	1.75
$p = 6$	1.39	1.56	1.23	1.65

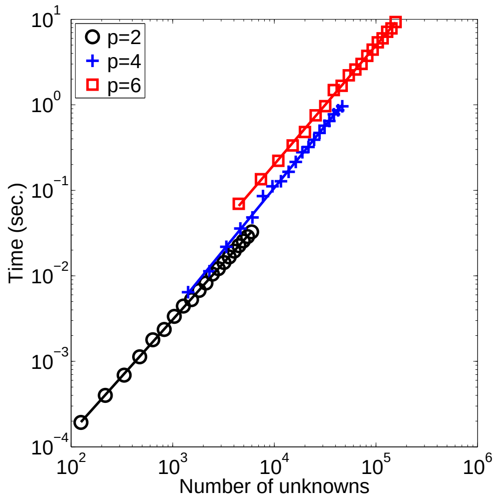
Table 2: Numerical scalability exponent factors α for refinements towards multiple entities.

Theoretical estimates for these mixed refinement patterns are unavailable. Moreover, one cannot estimate the FLOPS for mixed refinement patterns based on the FLOPS corresponding to refinements towards single entities. Thus, even when refinements towards a point or an edge scale both linearly, the combination of both types of refinements results in a significantly worse scaling. This can be seen theoretically, since the elimination trees used when refining towards single entities cannot be employed when refining towards multiple entities. Numerical results confirm that refinements towards multiple entities scale very differently than those towards single entities. For example, for the case of refinements towards a point and an edge, the resulting scalability exponent factor α (see Figure 13b and first column of Table 2) is close to 1.4. These results are worse for the cases of: (a) refinements towards a point and a face (see Figure 14b and second column of Table 2), where we obtain $\alpha \approx 1.5$, and (b) refinements towards a point, an edge, and a face (see Figure 16b and fourth column of Table 2), where we obtain $\alpha \approx 1.6$. The best results are observed for the case of refinements towards an edge and a face (see Figure 15b and third column of Table 2), where we obtain $\alpha \approx 1.25$. The main observation to be made from these numerical experiments is that the scalability of the solver with respect to the total number of degrees for anisotropic refinements is better than that for uniform refinements in all cases.

5. Conclusions

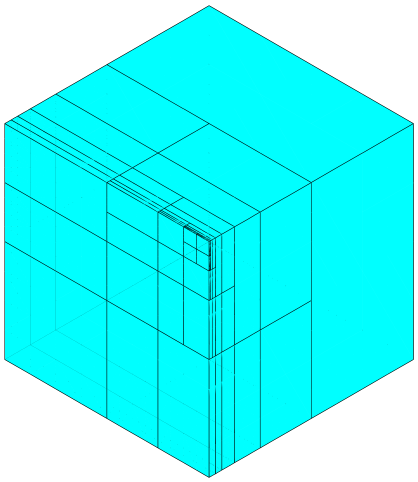
We considered six basic refinement patters for a hexahedral element, namely, uniform refinements, refinements towards a point, isotropic and anisotropic refinements towards an edge, and isotropic and anisotropic refinements towards a face. We showed theoretically that in all cases except the ones corresponding to uniform refinements and face anisotropic refinements, the FLOPS required to solve the corresponding system of linear equations grows linearly with respect to the mesh size. For the theoretical estimates we constructed elimination trees specific to particular h-refined grids. Theoretical results have been compared to numerical experiments performed with MUMPS and the nested-dissection algorithm for construction of the elimination tree provided by METIS library. For all cases our theoretical analysis provides computational costs that are identical or slightly better than the numerical experiments. The first conclusion is that there is a space for design of better algorithms for construction of elimination trees and orderings

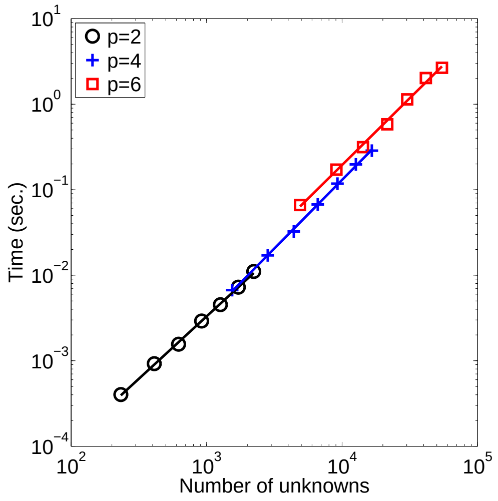

(a) Grid



(b) LU factorization time

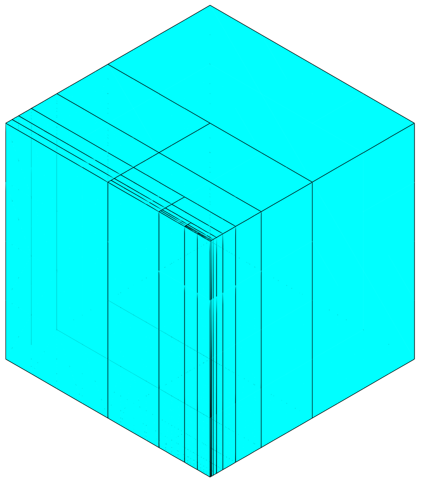
Figure 13: Anisotropic refinements towards a point and an edge.

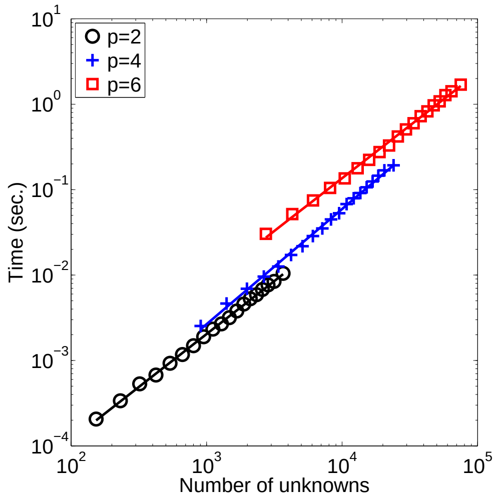

(a) Grid



(b) LU factorization time

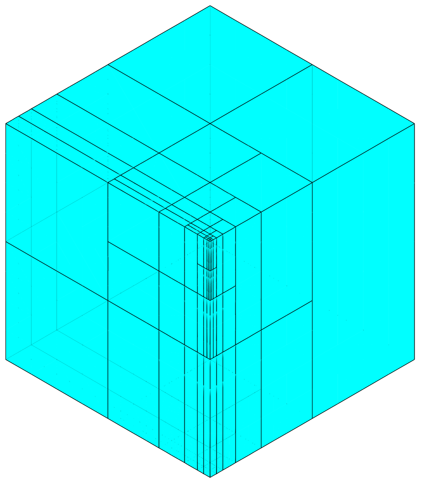
Figure 14: Anisotropic refinements towards a point and a face.

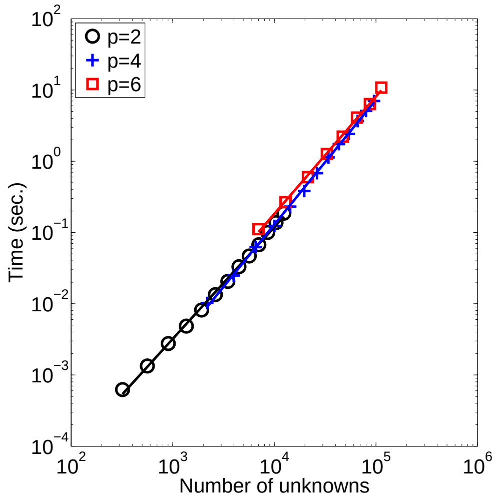

(a) Grid



(b) LU factorization time

Figure 15: Anisotropic refinements towards an edge and a face.


(a) Grid



(b) LU factorization time

Figure 16: Anisotropic refinements towards a point, an edge, and a face.

for h-refined grids.

We considered mixed refinement patterns, namely, refinements towards: (a) a point and an edge, (b) a point and a face, (c) an edge and a face, and (d) a point, an edge, and a face. In all cases, we observe a much better scalability than that obtained for uniform refinements.

The main conclusion of this work is that the use of h-adapted grids not only decreases the total number of unknowns required, but it also significantly reduces the cost of solving the algebraic system of linear equations per unknown. Moreover, the use of anisotropic refinements further reduces this cost per unknown (with few exceptions). This work suggests that direct solvers may be the fastest option for solving problems with highly h-adapted grids, or at least, for solving those parts of the computational mesh where heavy refinements towards a point, an edge, or a face occur.

Future work will involve the development of the reutilization solver for all kind of refinements, based on the ideas already developed for point singularities [24].

Acknowledgment

The work of the first and author (MP) was supported by Polish National Science Center grant no. UMO-2012/06/M/ST1/00363. The second author (DP) has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 644602, the Project of the Spanish Ministry of Economy and Competitiveness with reference MTM2013-40824-P, the BCAM Severo Ochoa accreditation of excellence SEV-2013-0323, the CYTED 2011 project 712RT0449, 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 (M^{2}SI).

References

[1] Julen Alvarez-Aramberri, David Pardo, Maciej Paszyński, Lisandro Dalcin, and Victor Manuel Calo. On round-off error for adaptive finite element methods. *Procedia Computer Science*, 9:1474-1483, 2012.

[2] Patrick R Amestoy, Iain S Duff, and Jean-Yves L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering*, 184(2):501-520, 2000.

[3] Patrick R Amestoy, Iain S Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15-41, 2001.

[4] Patrick R Amestoy, Abdou Guermouche, Jean-Yves L'Excellent, and Stephane Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136-156, 2006.

[5] Ivo Babuska and Werner C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM Journal of Numerical Analysis*, 15(4):736-754, 1978.

[6] Gang Bao, Guanghui Hu, and Du Liu. An h-adaptive finite element solver for the calculations of the electronic structures. *Journal of Com- putational Physics*, 231(14):4967-4979, 2012.

[7] Roland Becker, Hartmut Kapp, and Rolf Rannacher. Adaptive finite element methods for optimal control of partial differential equations: Basic concept. *SIAM Journal on Control and Optimisation*, 39(1):113-132, 2000.

[8] Ted Belytschko and Mazen Tabbar. H-adaptive finite element methods for dynamic problems, with emphasis on localization. *International Journal for Numerical Methods in Engineering*, 36(24):4245-4265, 1993.

[9] Paolo Bientinesi, Victor Eijkhout, Kyungjoo Kim, Jason Kurtz, and Robert van de Geijn. Sparse direct factorizations through unassembled hyper-matrices. *Computer Methods in Applied Mechanics and Engineering*, 199:430-438, 2010.

[10] Nathan Collier, David Pardo, Lisandro Dalcin, Maciej Paszyński, and Victor Manuel Calo. The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers. *Computer Methods in Applied Mechanics and Engineering*, 213-216(0):353-361, 2012.

[11] Leszek Demkowicy, Jason Kurtz, David Pardo, Maciej Paszyński, Waldemar Rachowicz, and Adam Zdunek. *Computing with hp-Adaptive Finite Elements, Vol. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications*. Chapman and Hall/Crc Applied Mathematics and Nonlinear Science, 2007.

[12] W.W. Donaldson and R.R. Meyer. A dynamic-programming heuristic for regular grid-graph partitioning. *Technical Report, Department of Computer Sciences, University of Wisconsin at Madison*, 00-06, 2000.

[13] Iain S Duff and John K Reid. The multifrontal solution of indefinite sparse symmetric linear. *ACM Transaction on Mathematical Software*, 9(3):302-325, 1983.

[14] Iain S Duff and John K Reid. The multifrontal solution of unsymmetric sets of linear equations. *SIAM Journal of Scientific Statistics*, 5(3):633-641, 1984.

[15] Alok Dutt, Leslie Greengard, and Rokhlin Vladimir. Spectral deferred correction methods for ordinary differential equations. *BIT*, 40:241-266, 2000.

[16] Kenneth Errikson and Claes Johnson. Adaptive finite element methods for parabolic problems I: A linear model problem. *SIAM Journal on Numerical Analysis*, 28(1):43-77, 1991.

[17] Po Geng, John Tinsley Oden, and Robert A Van de Geijn. A parallel multifrontal algorithm and its implementation. *Computer Methods in Applied Mechanics and Engineering*, 149(1):289-301, 1997.

[18] Johan Helsing and Rikard Ojala. Corner singularities for elliptic problems: Integral equations, graded meshes, quadrature, and compressed inverse preconditioning. *International Journal of Computational Physics*, 227(20):8820-8840, 2008.

[19] Bruce M Irons. A frontal solution program for finite element analysis. *International Journal of Numerical Methods in Engineering*, 2(1):5-32, 1970.

[20] Mina Kardani,Majidreza Nazem, Andrew J. Abbo, Daicho Sheng, and Scott W. Sloan. Refined h-adaptive finite element procedure for large deformation geotechnical problems. *Computational Mechanics*, 49(1):21-33, 2012.

[21] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359-392, 1999.

[22] Antti Niemi, Ivo Babuska, Juhani Pitkaranta, and Leszek Demkowicz. Finite element analysis of the Girkmann problem using the modern hp-version and the classical h-version. *Engineering with Computers*, 28(2):123-134, 2012.

[23] R. H. Nochetto, K. G. Siebert, and A. Veeser. *Multiscale, nonlinear and adaptive approximation*, Springer, 409-542, 2009.

[24] Maciej Paszyński, David Pardo, and Victor Manuel Calo. A direct solver with reutilization of LUfactorizations for h-adaptive finite element grids with point singularities. *Computers & Mathematics with Applications*, 65(8):1140-1151, 2013.

[25] Maciej Paszyński and Robert Schaefer. Reutilization of partial LU factorization for self-adaptive hp finite element method. Lecture Notes in Computer Science, 5101:965-974, 2008.

[26] Sanjaya K. Patro and Bosch Harold Selvam, Panneer R. Adaptive h-finite element modeling of wind flow around bridges. *Engineering Structures*, 48:569-577, 2013.

[27] Phillipe G. Schmitz and Lexing Ying. A fast direct solver for elliptic problems on general meshes in 2d. *Journal of Computational Physics*, 231:1314-1338, 2012.

[28] Phillipe G. Schmitz and Lexing Ying. A fast multifrontal solver for 3d elliptic problems using hierarchical matrices. *Journal of Computational Physics*, 2013, submitted.