Computers and Mathematics with Applications 70 (2015) 2331–2341

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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

Impact of element-level static condensation on iterative solver performance

D. Pardo [a,b,c,*], J. Álvarez-Aramberri [a,b,d], M. Paszynski [e], L. Dalcin [f,g], V.M. Calo [g,h]

[a] Department of Applied Mathematics, Statistics, and Operational Research, University of the Basque Country (UPV/EHU), Bilbao, Spain
[b] Basque Center for Applied Mathematics (BCAM), Bilbao, Spain
[c] IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
[d] University of Pau (UPPA), France
[e] AGH University of Science and Technology, Krakow, Poland
[f] National Scientific and Technical Research Council, Argentina
[g] Center for Numerical Porous Media, King Abdullah University of Science and Technology, Saudi Arabia
[h] Applied Mathematics & Computational Science and Earth Science & Engineering, King Abdullah University of Science and Technology, Saudi Arabia

ARTICLE INFO

Article history:
Received 30 October 2014
Received in revised form 24 August 2015
Accepted 5 September 2015
Available online 2 October 2015

Keywords:
Static condensation
Finite element method
p-FEM
Iterative solvers

ABSTRACT

This paper provides theoretical estimates that quantify and clarify the savings associated to the use of element-level static condensation as a first step of an iterative solver. These estimates are verified numerically. The numerical evidence shows that static condensation at the element level is beneficial for higher-order methods. For lower-order methods or when the number of iterations required for convergence is low, the setup cost of the elimination as well as its implementation may offset the benefits obtained during the iteration process. However, as the iteration count (e.g., above 50) or the polynomial order (e.g., above cubics) grows, the benefits of element-level static condensation are significant.

1. Introduction

Within the Finite Element (FE) community, the term static condensation of interior degrees of freedom refers to the Gaussian elimination of the element interior *bubble functions* arising from high-order discretizations [1]. Other terms such as Guyan condensation (reduction) can also be found in the literature to refer to the same set of linear algebra operations [2]. Static condensation can also be interpreted as a partial LU factorization of the interior degrees of freedom, as a first step of a specific substructuring technique, or as a partial orthogonalization of basis functions [3].

Interpreted in any of these forms, static condensation constitutes a fundamental building block for direct solvers and delivers significant performance improvements [4–6]. In high-order methods such as the *p*- and *hp*-FE methods, interior degrees of freedom are eliminated first, leading to a reduced system (called Schur complement) that is subsequently LU factorized. This static condensation step ensures the elimination of interior degrees of freedom before starting the LU factorization of the skeleton problem, thereby providing often better performance than that achieved with traditional ordering techniques, as shown in [7]. It also explains why those matrices lacking a structure that enables static condensation

* Corresponding author at: Department of Applied Mathematics, Statistics, and Operational Research, University of the Basque Country (UPV/EHU), Bilbao, Spain.
E-mail address: dzubiaur@gmail.com (D. Pardo).

http://dx.doi.org/10.1016/j.camwa.2015.09.005
0898-1221/

(e.g., higher-continuous basis functions as those used in meshless methods [8], reconstructing kernel element methods [9], and isogeometric analysis [10]) typically require a significantly larger number of floating point operations in order to be factorized [5].

While the use of static condensation in direct solvers is always beneficial [11], its use with iterative solvers is more controversial. Some authors postulate that static condensation should always be the starting point of any iterative solver, while others refrain from doing so, since it adds some complexity to the implementation. Even when static condensation is unused, most iterative solvers for *p*- and *hp*-FE methods still perform some type of elimination (or a spectrally equivalent operation) of local interior bubble functions (cf., [12–17]).

The key point, however, is to determine how profitable it is to explicitly build the Schur complement of element bubble functions (as performed in static condensation) and eliminate the corresponding unknowns from the global system before performing iterations with respect to keeping the local LU-factorized matrices as part of the preconditioner without ever computing the Schur complement. In other words, the distinguishing feature between iterative solvers that employ partial LU factorizations versus those that perform static condensation before executing an iterative solver is that the latter explicitly build the Schur complement and eliminate interior bubble functions from the global matrix rather than only evaluating their action over a given residual.

This paper provides quantitative estimates about the profitability of using static condensation before employing an iterative solver. We corroborate these estimates with numerical experiments in two and three spatial dimensions. Numerical experimentation also enlightens the behavior on the pre-asymptotic regime. As a result, we describe those situations in which the use of static condensation is most beneficial. To quantitatively compare both methods, we present floating point operations (FLOPs) estimates that also provide interesting clues for the design of optimal hybrid solvers [18].

In order to make this analysis tractable and easy to follow, we make several assumptions, which are described in Section 2 along with our model problem. Section 3 presents precise theoretical complexity estimates illustrating the advantages and limitations of using static condensation for each particular discretization. We describe the implementation details in Section 4 and we present numerical results confirming the estimates in Section 5. Section 6 describes the conclusions of our study and suggests future research lines in the topic.

2. Model problem and assumptions

Our starting point is the following algebraic system of linear equations:

$$Ax = b, \tag{1}$$

where A is a non-singular real-valued $N \times N$ sparse matrix, b is the right-hand side, and x is the solution vector.

In this work, we assume that the system matrix A is associated to a regular quadrilateral or hexahedral grid coming from a finite element discretization with uniform order of approximation p and with the same number of elements in each spatial direction. When the number of elements in each direction is substantially different, then the problem complexity reduces to that given by a lower dimensional problem.

In our estimates and computations, we avoid taking advantage of orthogonal basis functions, i.e., we consider all contributions originating from a trial and a test function with shared support as nonzero (a.k.a. "logical nonzero entry"), despite the fact that the actual values could indeed be zero. In arbitrarily mapped elements (non-affine) and/or in complex bilinear forms, logical nonzero entries are indeed different from zero.

We assume that the number of iterations needed to solve a given problem before static condensation is of the same order as that needed after static condensation. A large family of iterative solvers complies with this assumption, as shown in the Appendix.

We further assume that the cost of building the preconditioner associated to the skeleton problem is negligible, since the number of unknowns in the skeleton problem is $\mathcal{O}(p)$ times smaller than those in the interiors of the elements. In the case of a multigrid solver, we also assume that the coarse-grid correction has a negligible cost, since it consists of solving a smaller-size problem.

For simplicity, we restrict our attention to boundary value problems (with Dirichlet boundary conditions) that are governed by second order partial differential equations (PDEs) of the form:

$$\begin{aligned} -\nabla \cdot (c_1 \nabla u) + c_2 \nabla u + c_3 u &= f \quad \text{in } \Omega, \\ u &= u_0 \quad \text{on } \Gamma = \partial\Omega, \end{aligned} \tag{2}$$

where c_1 is a symmetric positive definite tensor, c_2 a vector, and c_3 a scalar function, f is the right-hand side, u is the solution, u_0 is the Dirichlet data, and ∇, $\nabla\cdot$ are the gradient and divergence operators, respectively. c_1, c_2, and c_3 are bounded and spatially varying so they may also incorporate the Jacobian of a transformation from the reference elements to a deformed geometry [19,20]. We also assume that the coefficients are such that the above problem has a unique solution.

The variational formulation of problem (2) is given by (see e.g., [21]):

$$\begin{cases} \text{Find } u \in u_0 + V \text{ such that,} \\ b(u, v) = l(v) \quad \forall v \in V, \end{cases} \tag{3}$$

where the bilinear form $b(u, v)$ and the linear form $l(v)$ are defined as:

$$\begin{aligned} b(u, v) &= \int_{\Omega} c_1 \nabla u \cdot \nabla v \, d\Omega + \int_{\Omega} c_2 \nabla u v \, d\Omega + \int_{\Omega} c_3 u v \, d\Omega \\ l(v) &= \int_{\Omega} f v \, d\Omega, \end{aligned} \tag{4}$$

with v a test function belonging to space $V = H_0^1(\Omega) = \{u \in L^2(\Omega) : u|_{\Gamma} = 0, \nabla u \in \mathbf{L^2}(\Omega)\}$.

For the case of multiple equations and/or $H(\mathbf{curl})$, $H(\mathrm{div})$, or L^2 discretizations, most of the results presented here can be easily generalized using the same methodology as that shown in this paper. For complexity estimates associated to isogeometric analysis (IGA) and finite difference (FD) discretizations, we refer to Collier et al. [5,22]. In these cases, it is not possible to perform static condensation of interior unknowns. Other discretizations such as Discontinuous Galerkin (DG) or Hybridizable Discontinuous Galerkin (HDG) [23,24] can be analyzed using the same techniques employed here but taking into account the precise number of element-interior and element-boundary unknowns delivered by each formulation. For the case of time-domain problems with a single time independent matrix, they can be interpreted as a single problem with multiple right hand sides, and thus, results shown here can also be trivially extended to that situation.

Our study focuses on moderate values of p, namely, $2 < p < 15$ in 2D and $2 < p < 10$ in 3D. Nonetheless, some tables and graphics showing higher values of p have also been computed for illustration and verification purposes. For larger values of p, the costs of integration, memory storage, and solution of the system of linear equations may become prohibitively expensive, and one needs to employ specific spectral method techniques such as sum factorization [25,26] or the Spectral Galerkin method [27]. The operation count of both integration and matrix–vector multiplication reduces significantly when using either of the above techniques for high p. However, in this paper we consider moderate values of p and we avoid the use of sum factorization techniques. The analysis performed in this paper does not consider the use of sparse representations of nodes for building the preconditioners [28], which would require a separate study.

This work focuses on operation counts and does not study parallelization costs such as communication. However, under the assumption that each element is contained only in a single processor (which is typical for the moderate values of p considered here), the element-level static condensation is performed in a single processor, and no communication costs occur during such operation.

3. Theoretical complexity estimates

In this section, we derive theoretical complexity estimates to compare the number of FLOPs needed to solve a system of linear equations with and without static condensation.

The considered iterative algorithm consists of the following steps:

1. For the static-condensation case, construct the Schur complement matrix in each element.
2. Construct a block Jacobi preconditioner on the fine grid with blocks associated to the unknowns of a single element, as in [13]. For the case with static condensation, bubbles have already been eliminated in the previous step, thus, blocks are associated to the unknowns of the element skeleton.
3. Construct restriction/prolongation operators and coarse-grid solver for the case of multigrid. As described in the previous section, this step is assumed to be of lower-order complexity and thus has a negligible computational cost.
4. Iterate (matrix–vector multiplications) until a given error tolerance is reached. For the case with static condensation, iterations are performed only over the reduced skeleton system.

The computational cost associated to the use of other commonly employed preconditioners such as Incomplete LU (ILU) factorization with no fill-in – ILU(0) – is of the same order, and thus, the analysis performed here also applies to those cases.

3.1. Notation and preliminary considerations

We express the amount of FLOPs needed to solve the above system of equations using an iterative solver without static condensation as:

$$\text{FLOPs} = (n_{it} c + g) M, \tag{5}$$

where:

- M is the number of nonzero entries in the matrix A,
- n_{it} is the number of iterations, which depends on the structure of A,
- gM is the number of FLOPs required to build the preconditioner, and
- c is a constant that incorporates the number of matrix–vector multiplications that need to be performed within each iteration.

Table 1
Number of independent vertices (Nr_Vert), edges (Nr_Edg), faces (Nr_Fac), and volumes (Nr_Vol) for a single element (quadrilateral in 2D or hexahedron in 3D) with periodic boundary conditions.

Denomination	Dimension (d)	Nr_Vert	Nr_Edg	Nr_Fac	Nr_Vol
2D	2	1	2	1	0
3D	3	1	3	3	1

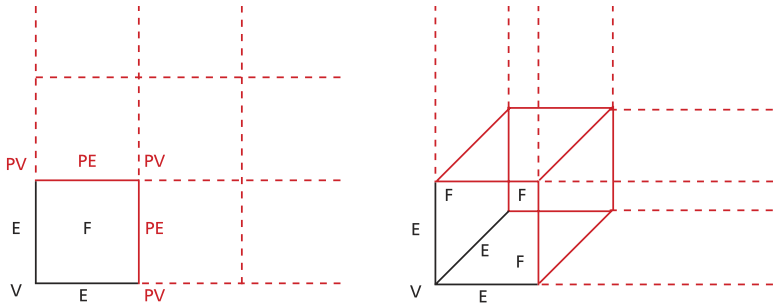

Fig. 1. Quadrilateral (left panel) and Hexahedral (right panel) in a periodic mesh. V = Vertex, E = Edge, F = Face. PV, PE, and PF correspond to vertices, edges and faces where periodic boundary conditions are imposed, and they are denoted with the red color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

When using static condensation, the total amount of FLOPs needed to solve the above system of equations is given by:

$$\text{FLOPs}_{SC} = (\tilde{n}_{it}\tilde{c} + \tilde{g})\tilde{M} + \tilde{s}M, \tag{6}$$

where:

- $\tilde{M}$ is the number of nonzero entries in the stiffness matrix after static condensation,
- $\tilde{n}_{it}$ is the number of iterations needed in the statically condensed system,
- $\tilde{g}\tilde{M}$ is the number of FLOPs required to build the preconditioner of the statically condensed system,
- $\tilde{c}$ is a constant that incorporates the number of matrix–vector multiplications performed within each iteration of the statically condensed system, and
- $\tilde{s}M$ is the number of FLOPs needed to perform static condensation over the original system (independent of $\tilde{n}_{it}$).

The above formulas depend upon the order of approximation p. In other words, g, M, $\tilde{g}$, $\tilde{M}$, and s are p dependent. Additionally, n_{it}, and $\tilde{n}_{it}$ may depend upon p, but here we assume that the preconditioner is able to keep the number of iterations almost constant irrespective of p. In fact, there exist several iterative solvers in the literature for which the number of iterations needed to converge is almost independent of p (cf., [12,13,17,15,29]).

The objective of static condensation is to reduce the total cost by reducing the value of the constant $\tilde{M}$ versus M, possibly at the cost of some small overhead $\tilde{s}M$ needed to build the Schur complement matrix. In other words, our analysis focuses on the amount of operations needed to build the Schur complement and the savings in terms of number of nonzero entries that this system may deliver during iterations (matrix–vector multiplications).

The speedup factor in terms of the number of FLOPs due to the use of static condensation is independent of the number of elements (up to boundary conditions). In order to properly normalize the results, we provide all estimates on a per-element basis. We assume that a sufficiently large number of elements is used, thus making the effect of the boundaries irrelevant.

To compute the number of independent nodes per element for a problem with infinite number of elements in each direction, we consider a single hexahedron problem with periodic boundary conditions [4,5]. The element has four different types of nodes (node_type), namely, vertices, edges, faces, and volumetric interiors. The number of nodes of each particular type – nr_nodes(node_type) – in an element with periodic boundary conditions is displayed in Table 1, and illustrated in Fig. 1. The problem size (per element) N is given by:

$$N = \sum_{\text{node\_type}} \text{nr\_nodes(node\_type)} \cdot \text{dof(node\_type)}, \tag{7}$$

where dof(node_type) is the number of degrees of freedom for each node_type. Let e be the node dimension of node_type, i.e., 0 for vertices, 1 for edges, 2 for faces, and 3 for volumes. For an $H^1(C^0)$ element of order p, the number of degrees of freedom is equal to

$$\text{dof(node\_type)} = (p-1)^e. \tag{8}$$

3.2. Without static condensation

We first consider the case of an iterative solver built without performing static condensation. Following Eq. (5), we decompose the total cost into two parts: (a) building the preconditioner, and (b) performing iterations.

Cost of building the preconditioner. For the particular case of a multigrid solver, we assume that the construction of the transfer operators, smoothers, and the coarsest grid solve are significantly cheaper operations.

Cost of iterations. The number of FLOPs needed to perform each iteration is dominated by two matrix–vector multiplications: one corresponding to the original matrix A, and the second one corresponding to the preconditioner matrix. Assuming that the number of nonzero entries in the preconditioner is smaller or proportional to that of the original matrix, we conclude that the iteration cost is proportional to the number of nonzero entries in the original matrix.

Then, for an infinitely large problem, the number of nonzero entries per element in the original system M is given by:

$$M = \sum_{\text{node\_type}} \text{nr\_nodes(node\_type)} \cdot \text{dof(node\_type)} \cdot \text{interact\_dof(node\_type)}, \tag{9}$$

where interact_dof(node_type) is the number of degrees of freedom interacting with a given node_type, and its formula is given by:

$$\text{interact\_dof}\,(\text{node\_type}) = (2p+1)^{d-e} \cdot (p+1)^{e}. \tag{10}$$

3.3. With static condensation

In this case, we need to first consider the number of FLOPs needed to perform static condensation. Additionally, we also need to consider the complexity of building the preconditioner and to perform the iterations.

Cost of static condensation. Let matrix A be decomposed as:

$$A = \begin{bmatrix} A_{II} & A_{IS} \\ A_{SI} & A_{SS} \end{bmatrix}, \tag{11}$$

where subscript I corresponds to the interior bubble degrees of freedom to be eliminated from the system by static condensation and S to the remaining skeleton unknowns. Performing the partial LU factorization of the square submatrix A_{II}, we obtain:

$$A = \begin{bmatrix} I & 0 \\ A_{SI}A_{II}^{-1} & I \end{bmatrix} \cdot \begin{bmatrix} A_{II} & 0 \\ 0 & A_{SS} - A_{SI}A_{II}^{-1}A_{IS} \end{bmatrix} \cdot \begin{bmatrix} I & A_{II}^{-1}A_{IS} \\ 0 & I \end{bmatrix}, \tag{12}$$

where $S = A_{SS} - A_{SI}A_{II}^{-1}A_{IS}$ is the Schur complement. In practice, a full inversion of A_{II} should be avoided and replaced by the corresponding LU factorization. First, because round-off errors are less relevant when performing an LU factorization. Second, because the number of FLOPs also diminishes, specially, for high values of p. Although both operations (LU factorization and inversion) exhibit the same scaling in terms of p, the corresponding constants are different. More precisely, inverting the matrix is about three times more expensive than performing its LU factorization.[1]

Taking the above observation into account, we conclude that the cost of building the Schur complement (per element) is the sum of the costs associated to: (a) performing the *LU* factorization of matrix A_{II}, (b) computing the action $A_{II}^{-1}A_{IS}$ by using backward and forward substitutions, and (c) performing a matrix–matrix multiplication. The cost of performing the *LU* factorization of matrix A_{II} is of order $\mathcal{O}(p^{3d})$, while the cost of performing backward and forward substitutions is of order $\mathcal{O}(p^{3d-1})$. The remaining operations are of lower order, thus their contribution to the total number of FLOPs is negligible.

Cost of building the preconditioner. As stated in previous sections, we assume that the cost of building the preconditioner for the skeleton problem is negligible, since the number of unknowns of this problem is $\mathcal{O}(p)$ times smaller than that of the original problem.

Cost of iterations. As in the previous case, we assume that the number of FLOPs is proportional to the number of nonzero entries in the Schur complement system S.

If we denote the nodes of type "skeleton" (all except the volumetric interior ones) by –node_ske–, then, for an infinitely large problem the number of nonzero entries per element in the statically condensed system $\tilde{M}$ is given by:

$$\tilde{M} = \sum_{\text{node\_ske}} \text{nr\_nodes(node\_ske)} \times \text{dof(node\_ske)} \times [\text{interact\_dof}\,(\text{node\_ske}) - 2^{d-e}(p-1)^{d}]. \tag{13}$$

[1] Computing the inverse of a matrix A of size $n \times n$ with LAPACK [30] requires two steps: first, compute the factorization $A = LU$ (by calling DGETRF), next compute A^{-1} out of the LU factors (by calling DGETRI). The LU factorization of a matrix of size $n \times n$ has a flop count of $\mathcal{O}(\frac{2}{3}n^3)$ and the computation of the inverse out for the LU factors has a flop count of $\mathcal{O}(\frac{4}{3}n^3)$ (see [31, pp. 120]). Therefore, the computation of the matrix inverse has a total flop count of $\mathcal{O}(2n^3)$, which is three times the one required to compute the LU factors.

Table 2
2D results for a problem with periodic boundary conditions (infinitely large problem). Number of nonzero entries per element with and without static condensation.

p	1	2	3	4	5	10	25	100	1000
$M/(p+1)^3p$	1.12	1.19	1.17	1.15	1.13	1.08	1.04	1.01	1.00
$\tilde{M}/14p^2$	0.64	0.84	0.90	0.92	0.94	0.97	0.99	1.00	1.00
$M/\tilde{M}$	1	1.36	1.99	2.78	3.72	10.6	52.7	745	71735

Table 3
3D results for a problem with periodic boundary conditions (infinitely large problem). Number of nonzero entries per element with and without static condensation.

p	1	2	3	4	5	10	25	100	1000
$M/((p+1)^5p)$	0.84	1.05	1.10	1.11	1.10	1.07	1.04	1.00	1.00
$\tilde{M}/(33p^4)$	0.82	0.87	0.90	0.92	0.94	0.97	0.99	1.00	1.00
$M/\tilde{M}$	1	1.11	1.40	1.77	2.22	5.42	24.2	322	30496

In the above formula, the term "interact_dof(node_ske) $-\ 2^{d-e}(p-1)^d$" corresponds to all interacting nodes of the given "node_ske" minus those corresponding to volumetric interior nodes already eliminated by the static condensation.

A simple asymptotic expansion shows that $M = \mathcal{O}(p^{2d})$, while $\tilde{M} = \mathcal{O}(p^{2(d-1)})$, which implies that the number of nonzero entries in the statically condensed system is $\mathcal{O}(p^2)$ times smaller than in the original system. Indeed, we can further derive the following approximations for the number of nonzero entries per element, whose accuracy are confirmed by Tables 2 and 3:

$$M(\text{2D}) \approx (p+1)^3p \qquad M(\text{3D}) \approx (p+1)^5p \tag{14}$$

$$\tilde{M}(\text{2D}) \approx 14p^2 \qquad \tilde{M}(\text{3D}) \approx 33p^4. \tag{15}$$

3.4. Discussion and final estimates

Taking into account all the above estimates, the number of FLOPs needed to solve the original and the statically condensed systems are respectively of the order:

$$\begin{aligned} \text{FLOPs} &= \mathcal{O}(n_{it}p^{2d}) + \mathcal{O}(p^{3d}), \\ \text{FLOPs}_{SC} &= \mathcal{O}(n_{it}p^{2(d-1)}) + \mathcal{O}(p^{3d}) + \mathcal{O}(p^{3d-1}), \end{aligned} \tag{16}$$

per element.

Although $\mathcal{O}(p^{3d-1})$ can be considered as a lower order term in the above estimates, it sometimes becomes significant for low and moderate values of p, since the constant in front of it is significantly higher than that of the higher order term $\mathcal{O}(p^{3d})$.

In any case, we observe that as $p \longrightarrow \infty$, the number of FLOPs is the same regardless of the use of static condensation. Thus, for sufficiently high p, perhaps the additional burden of implementing static condensation should be avoided, and the main effort should be focused on performing some type of incomplete LU factorization (or some alternative iterative solver) on the bubble degrees of freedom.

On the other hand, for a fixed value of p, we see that as $n_{it} \longrightarrow \infty$, the use of static condensation is always recommended. To determine the exact regions for which the use of static condensation brings any benefit, we complement the above theoretical estimates with numerical experimentation. These simulation results allow us to estimate the relative significance of the different contributions of each term in (16).

4. Implementation

To provide further insight to the above theoretical estimates, we built a FE code in one-, two-, and three-spatial dimensions. The method is implemented in MATLAB, and supports any uniform order of approximation p. For its construction, we have assumed a tensor product structure for the basis functions and the corresponding discretization. To simplify the implementation, the size of all elements in each direction is selected to be equal to one, and we have chosen $c_1 = 1$, $c_2 = 0$, and $c_3 = 0$ everywhere, which produces a constant Jacobian (in fact, equal to one) and facilitates the decomposition of the bilinear form as the sum of tensor products of one-dimensional problems. Using this identity map for the geometry does not affect the nonzero pattern of the system, while keeping the implementation simple.

Table 4
2D numerical results for a problem with a number of elements equal to 1000^2, 333^2, 200^2, 100^2, 40^2 and 20^2 for p equal to 1, 3, 5, 10, 25 and 40, respectively. Number of nonzero entries per element with and without static condensation.

p	$p=1$	$p=3$	$p=5$	$p=10$	$p=25$	$p=40$
$M/(p+1)^3p$	1.12	1.17	1.13	1.08	1.03	1.02
$\tilde{M}/14p^2$	0.64	0.89	0.94	0.96	0.97	0.96
$M/\tilde{M}$	1	1.99	3.73	10.64	53.45	130.98

Table 5
3D numerical results for a problem with a number of elements equal to 100^3, 33^3, 20^3 and 8^3 for p equal to 1, 3, 5, and 10, respectively. Number of nonzero entries per element with and without static condensation.

p	$p=1$	$p=3$	$p=5$	$p=10$
$M/(p+1)^5p$	0.84	1.07	1.06	1.01
$\tilde{M}/(33p^4)$	0.81	0.87	0.89	0.86
$M/\tilde{M}$	1	1.40	2.24	5.74

Table 6
2D numerical results showing how the number of nonzero entries, with and without static condensation, converges to the theoretical estimates when the number of elements (N_e) increases.

$p=5$	$N_e=5^2$	$N_e=25^2$	$N_e=50^2$	$N_e=100^2$	$N_e=200^2$	Estimate
$M/(p+1)^3p$	1.02	1.11	1.12	1.13	1.13	1.13
$\tilde{M}/14p^2$	0.81	0.91	0.93	0.93	0.94	0.94
$M/\tilde{M}$	3.91	3.76	3.74	3.73	3.73	3.72

Table 7
3D numerical results showing how the number of nonzero entries, with and without static condensation, converges to the theoretical estimates when the number of elements (N_e) increase.

$p=5$	$N_e=3^3$	$N_e=5^3$	$N_e=10^3$	$N_e=20^3$	Estimate
$M/(p+1)^5p$	0.84	0.94	1.02	1.06	1.10
$\tilde{M}/(33p^4)$	0.66	0.76	0.85	0.89	0.94
$M/\tilde{M}$	2.41	2.33	2.27	2.24	2.22

5. Numerical results

In this section, we solve the Laplace equation (2) with Dirichlet boundary conditions using the FE method, as described in the previous section. Numerical results are obtained in a workstation equipped with 94 GB RAM, and an eight-core Intel Xeon E5620 processor (12 Mb cache, 2.40 GHz).

5.1. Number of nonzero entries

We first report the number of nonzero entries per element for the condensed and non-condensed systems. In Table 4, we observe that the numerical results for 2D match with the estimates of Table 2. For example, for $p = 5$, the number of nonzero entries for the statically condensed and non-condensed systems computed numerically coincide with the theoretical estimates (1.13 and 0.94, respectively). Their ratio is also the same (3.73 for the numerical value and 3.72 for the estimate).

In the 3D case (Table 5), the agreement with the estimates (Table 3) is more visible for low p. Estimates correspond to the case of an infinitely large problem. Therefore, when we solve the Laplace equation with Dirichlet boundary conditions, in order to obtain a perfect match between the numerical results and the estimates, we need to consider a sufficiently large problem. For small polynomial orders we are able to solve large problems. This is reflected in the existing good agreement between numerical results and theoretical estimates (e.g., for $p = 3$). However, an increase in p limits us from considering as many elements as needed to observe a perfect agreement (e.g., $p = 10$).

Tables 6 and 7 illustrate that when the problem size increases, the numerical results converge to the theoretical estimates, as predicted.

5.2. Time comparison

Assuming that the amount of time needed to execute an algorithm is proportional to the number of FLOPs, we can reinterpret Eqs. (5) and (6) in terms of time instead of FLOPs by rewriting them in the following way:

$$TIME = T(A \cdot \overline{v})n_{it} + T(A_{II}), \tag{17}$$

Table 8
2D numerical results for a problem with number of elements equal to 333^2, 200^2, 100^2, 40^2 20^2 and 2^2 for p equal to 3, 5, 10, 25 40 and 100, respectively. Times per element with and without static condensation. Times have been computed 100 times and averaged.

p	$p=3$	$p=5$	$p=10$	$p=25$	$p=40$	$p=100$
$T(A_{SS})$	1.74e−5	3.92e−5	7.02e−4	2.31e−2	2.23e−1	16.94
$T(A_{II})$	2.95e−6	1.08e−5	2.09e−4	1.18e−2	1.55e−1	14.79
$T(A_{SS}\cdot\overline{v})$	3.13e−7	8.63e−7	3.26e−6	2.42e−5	5.11e−5	2.13e−4
$T(A\cdot\overline{v})$	6.21e−7	3.56e−6	3.51e−5	1.05e−3	6.63e−3	2.55e−1
$T(A_{SS})/T(A_{II})$	5.90	3.63	3.36	1.96	1.44	1.15
$T(A\cdot\overline{v})/T(A_{SS}\cdot\overline{v})$	1.98	4.13	10.77	43.39	129.75	1197.2

Table 9
3D numerical results for a problem with number of elements equal to 33^3, 20^3 8^3 2^3 and 1 for p equal to 3, 5 10, 15 and 25, respectively. Times per element with and without static condensation. Times have been computed 100 times and averaged.

p	$p=3$	$p=5$	$p=10$	$p=15$	$p=25$
$T(A_{SS})$	4.34e−5	7.48e−4	7.60e−2	1.49	77.76
$T(A_{II})$	4.13e−6	8.76e−5	2.09e−2	5.30e−1	36.85
$T(A_{SS}\cdot\overline{v})$	5.58e−6	4.19e−5	7.11e−4	2.34e−3	1.02e−2
$T(A\cdot\overline{v})$	7.89e−6	9.70e−5	4.10e−3	3.30e−2	6.23e−1
$T(A_{SS})/T(A_{II})$	10.51	8.54	3.64	2.81	2.11
$T(A\cdot\overline{v})/T(A_{SS}\cdot\overline{v})$	1.41	2.32	5.77	14.10	61.08

and

$$TIME_{SC} = T(A_{SS}\cdot\overline{v})n_{it} + T(A_{SS}), \tag{18}$$

where $TIME$ and $TIME_{SC}$ are the total times without and with static condensation, respectively. Here, n_{it} is again the number of iterations the iterative solver requires to converge, $T(A_{SS})$ is the time required to compute the Schur complement, $T(A_{II})$ corresponds to the time devoted to perform the LU factorization of A_{II}, and $T(A\cdot\overline{v})$ and $T(A_{SS}\cdot\overline{v})$ are, respectively, the time required to multiply the non-condensed stiffness matrix A and the statically condensed stiffness matrix A_{SS} by a vector. As before, we assume that the time needed to build the preconditioner in the condensed system is negligible.

Since these times scale linearly with the number of elements, all numerical results reported in Tables 8 and 9 are given in terms of time (seconds) per element.

According to Section 3.3, for a sufficiently large p, the dominant part in terms of number of FLOPs is the LU factorization of A_{II}, regardless the decision of using static condensation or not. When the Schur complement is explicitly computed, additional backward and forward substitutions are required. Since the ratio between interior and skeleton nodes tends to infinity as p grows, the cost of these forward and backward substitutions becomes negligible for sufficiently large p, and the quotient $T(A_{SS})/T(A_{II})$ approaches 1. However, for small values of p, the situation may vary considerably, since the number of interior and skeleton nodes are comparable. This is numerically observed in Tables 8 and 9, where we also appreciate that the ratio $T(A_{SS})/T(A_{II})$ tends to 1 for large p, as expected.

We also know that the time required to multiply a matrix times a vector depends on the number of nonzeros in the matrix. Therefore, we expect to observe that the ratios $T(A\cdot\overline{v})/T(A_{SS}\cdot\overline{v})$ and $M/\tilde{M}$ behave in a similar fashion. This is effectively observed for example by comparing the last row of Table 9 (with values equal to 1.41, 2.32 and 5.77) with the last row of Table 5 (with values equal to 1.40, 2.24 and 5.74, respectively).

Fig. 2 displays the ratio between the time required to solve a 2D problem without static condensation against that needed to solve the same problem with static condensation. As predicted by the theory, the use of static condensation is always recommended for a sufficiently large number of iterations. As the number of iterations grows, the cost of matrix–vector multiplications becomes dominant, and the savings are of the order of p^2, which correspond to $T(A\cdot\overline{v})/T(A_{SS}\cdot\overline{v})$ row in Tables 8 and 9. In the pre-asymptotic regime, these savings become substantial when the number of iterations is above 100. On the other hand, when the number of iterations is below 10, the use of static condensation provides no advantage. We also observe that as the value of p increases, the number of iterations needed for static condensation to be beneficial diminishes. Similar conclusions can be depicted from the 3D results displayed in Fig. 3.

6. Conclusions

The use of static condensation in iterative solvers is controversial. That is, some authors defend its use as a starting point of any iterative solver, while others do not. The key point is to determine whether computing the Schur complement instead of keeping the local LU-factorized matrices as a preconditioner is profitable or not.

In the present work, we provide quantitative theoretical estimates to determine the situations where it is recommended to use the static condensation, and illustrated these estimates with numerical experiments. To make the problem tractable, we have made several assumptions on the mesh regularity and on the behavior of the iterative solver so the number of iterations becomes independent of the mesh size. Under these assumptions, we show that the use of static condensation

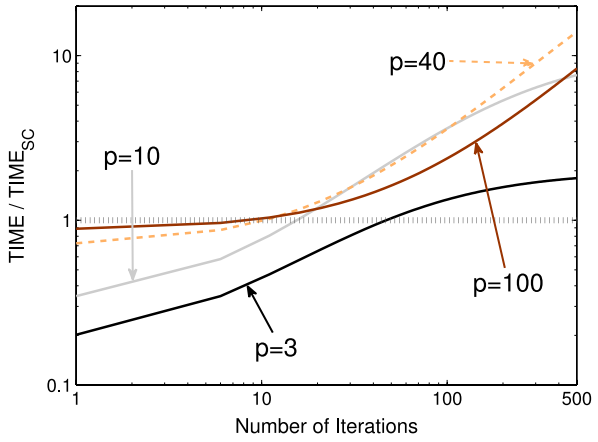

Fig. 2. Ratio between the time needed to solve a 2D problem without static condensation vs. that using static condensation.

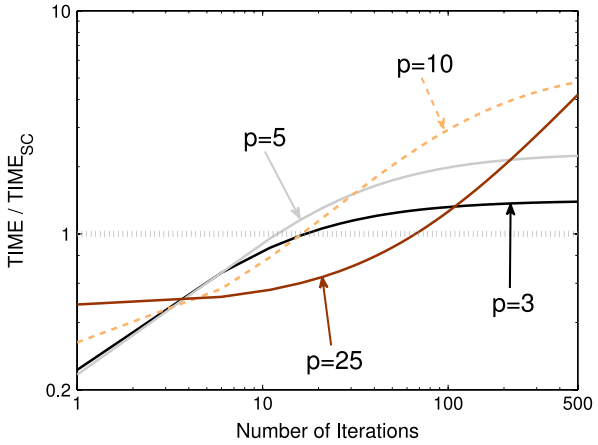

Fig. 3. Ratio between the time needed to solve a 3D problem without static condensation vs. that using static condensation.

is most beneficial when the number of iterations is sufficiently large, assuming that $p > 3$. However, when the number of iterations is below 10, the use of static condensation may be counterproductive.

Therefore, a more nuanced view point is necessary. Instead of using always this technique as a starting point for an iterative solver, we recommend to use the tables provided in this paper to analyze the profitability of this technique when dealing with high-order methods. In particular, if the iterative solver converges in over 50 iterations, the use of static condensation is highly recommended.

This work also provides a first step towards obtaining rigorous computational complexity estimates for hybrid solvers. Based on existing estimates for uniform and highly refined grids, the construction of further Schur complements (corresponding to a group of elements) before employing an iterative solver seems inadequate for uniform grids, however, quite profitable and advantageous for certain types of highly refined grids. This will be analyzed in detail in a future contribution.

Acknowledgments

This work was partially supported by 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, the Basque Government Consolidated Research Group Grant IT649-13 on "Mathematical Modeling, Simulation, and Industrial Applications (M2SI)", the Polish National Science Center grant no. DEC-2012/06/M/ST1/00363 and the Center for Numerical Porous Media at King Abdullah University of Science and Technology (KAUST). Lisandro Dalcin was partially supported by Agencia Nacional de Promoción Científica y Tecnológica grants PICT 2014–2660 and PICT-E 2014–0191. This publication also was made possible by a National Priorities Research Program grant 7-1482-1-278 from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors.

Table A.10
Number of iterations required to reduce the residual by five orders of magnitude without (first number within each cell) and with (second number within each cell) the use of static condensation for a 2D Laplace problem using an overlapping Block-Jacobi smoother [33].

Order of approximation	16×16 elements	32×32 elements	64×64 elements
$p = 2$	23/23	43/43	87/85
$p = 4$	22/22	43/43	87/86
$p = 6$	22/22	44/43	87/86
$p = 8$	23/23	44/44	88/86

Appendix. Effect of static condensation on the required number of iterations

In general, the use of element-level static condensation reduces the condition number of the original matrix in terms of polynomial order of approximation p, although not in terms of element size h (see, e.g., [3]). However, this positive effect on the condition number of the unpreconditioned stiffness matrix often has little relevance, since the actual objective is to reduce the condition number of the *preconditioned* system.

For poorly designed solvers (e.g., those using no preconditioner or only a diagonal preconditioner), the use of static condensation may significantly reduce the number of iterations required to achieve a given tolerance error. In such cases, our results need to be modified to account for the savings on the number of iterations produced by the use of static condensation.

However, for well-designed solvers that incorporate proper preconditioners in terms of p, the savings on the number of iterations produced by the use of static condensation may vanish. For example, any ILU preconditioner (when properly ordered by considering first the element interior unknowns) provides a preconditioned system whose condition number is independent of the use of static condensation, since one obtains the same algebraic system [3] for both cases. Indeed, such preconditioner is optimal for dealing with the element interior unknowns, since it is equivalent to a full LU factorization on the element interiors, because the solution of interior unknowns introduces no additional fill in.

Another prominent family of preconditioners that require the same number of iterations to converge irrespectively of the use (or not) of static condensation when combined with CG or GMRES is an overlapping block Jacobi smoother with blocks determined by those basis functions whose support is fully contained in the support of a given vertex basis function (see, for instance, [32,33, page 84]). Numerical experiments confirm this result, as illustrated on Table A.10. In this table, the number of iterations required to converge without and with static condensation are almost identical in all cases. These numerical results also confirm that static condensation only improves the condition number with respect to p, and not h. However, an adequate preconditioner in p will achieve a similar effect.

When the above preconditioners are combined with a multigrid solver, the convergence properties also remain independent with respect to the use or not of static condensation.

References

[1] E.L. Wilson, The static condensation algorithm, Internat. J. Numer. Methods Engrg. 8 (1) (1974) 198–203.
[2] R.J. Guyan, Reduction of stiffness and mass matrices, AIAA J. 3 (2) (1965) 380–380.
[3] T. Vejchodsky, P. Solin, Static condensation, partial orthogonalization of basis functions, and ilu preconditioning in the hp-FEM, J. Comput. Appl. Math. 218 (1) (2008) 192–200.
[4] V.M. Calo, N.O. Collier, D. Pardo, M.R. Paszyński, Computational complexity and memory usage for multi-frontal direct solvers used in p finite element analysis, Procedia Comput. Sci. 4 (2011) 1854–1861.
[5] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, V. Calo, The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers, Comput. Methods Appl. Mech. Engrg. 213–216 (2012) 353–361.
[6] N. Collier, L. Dalcin, V. Calo, On the computational efficiency of isogeometric methods for smooth elliptic problems using direct solvers, Internat. J. Numer. Methods Engrg. 100 (8) (2014) 620–632.
[7] P. Bientinesi, V. Eijkhout, K. Kim, J. Kurtz, R. Van De Geijn, Sparse direct factorizations through unassembled hyper-matrices, Comput. Methods Appl. Mech. Engrg. 199 (9) (2010) 430–438.
[8] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, P. Krysl, Meshless methods: an overview and recent developments, Comput. Methods Appl. Mech. Engrg. 139 (1) (1996) 3–47.
[9] N. Collier, D.C. Simkins Jr., The quasi-uniformity condition for reproducing kernel element method meshes, Comput. Mech. 44 (3) (2009) 333–342.
[10] T.J. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, nurbs, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39) (2005) 4135–4195.
[11] P. Bientinesi, V. Eijkhout, K. Kim, J. Kurtz, R. van de Geijn, Sparse direct factorizations through unassembled hyper-matrices, Comput. Methods Appl. Mech. Engrg. 199 (9–12) (2010) 430–438.
[12] M. Ainsworth, A preconditioner based on domain decomposition for h-p finite-element approximation on quasi-uniform meshes, SIAM J. Numer. Anal. 33 (4) (1996) 1358–1376.
[13] D. Pardo, L. Demkowicz, Integration of *hp*-adaptivity with a two grid solver for elliptic problems, Comput. Methods Appl. Mech. Engrg. 195 (2006) 674–710.
[14] D. Pardo, L. Demkowicz, J. Gopalakrishnan, Integration of *hp*-adaptivity and a two grid solver for electromagnetic problems, Comput. Methods Appl. Mech. Engrg. 195 (2006) 2533–2573.
[15] S. Beuchler, Inexact additive Schwarz solvers for hp-fem discretizations in three dimensions, in: T. Apel, O. Steinbach (Eds.), Advanced Finite Element Methods and Applications, in: Lecture Notes in Applied and Computational Mechanics, vol. 66, Springer, Berlin, Heidelberg, 2013, pp. 91–108.
[16] D.N. Arnold, R.S. Falk, R. Winther, Multigrid in H(div) and H(curl), Numer. Math. 85 (2) (2000) 197–217.
[17] J. Schoberl, J.M. Melenk, C. Pechstein, S. Zaglmayr, Additive Schwarz preconditioning for p-version triangular and tetrahedral finite elements, IMA J. Numer. Anal. 28 (1) (2008) 1–24.
[18] F. Sourbier, A. Haidar, L. Giraud, H. Ben-Hadj-Ali, S. Operto, J. Virieux, Three-dimensional parallel frequency-domain visco-acoustic wave modelling based on a hybrid direct/iterative solver, Geophys. Prospect. 59 (5) (2011) 834–856.

[19] D. Pardo, V. Calo, C. Torres-Verdin, M. Nam, Fourier series expansion in a non-orthogonal system of coordinates for the simulation of 3D-DC borehole resistivity measurements, Comput. Methods Appl. Mech. Engrg. 197 (21) (2008) 1906–1925.
[20] D. Pardo, C. Torres-Verdín, M. Nam, M. Paszynski, V. Calo, Fourier series expansion in a non-orthogonal system of coordinates for the simulation of 3D alternating current borehole resistivity measurements, Comput. Methods Appl. Mech. Engrg. 197 (45) (2008) 3836–3849.
[21] T.J. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Courier Corporation, 2012.
[22] N. Collier, L. Dalcin, D. Pardo, V. Calo, The cost of continuity: performance of iterative solvers on isogeometric finite elements, SIAM J. Sci. Comput. 35 (2) (2013) A767–A784.
[23] N. Nguyen, J. Peraire, B. Cockburn, Hybridizable discontinuous Galerkin methods, in: Spectral and High Order Methods for Partial Differential Equations, Springer, 2011, pp. 63–84.
[24] R.M. Kirby, S.J. Sherwin, B. Cockburn, To CG or to HDG: a comparative study, J. Sci. Comput. 51 (1) (2012) 183–212.
[25] P.E. Vos, S.J. Sherwin, R.M. Kirby, From h to p efficiently: Implementing finite and spectral hp element methods to achieve optimal performance for low-and high-order discretisations, J. Comput. Phys. 229 (13) (2010) 5161–5181.
[26] S.A. Orszag, Spectral methods for problems in complex geometries, J. Comput. Phys. 37 (1) (1980) 70–92.
[27] J.M. Melenk, K. Gerdes, C. Schwab, Fully discrete hp-finite elements: fast quadrature, Comput. Methods Appl. Mech. Engrg. 190 (32) (2001) 4339–4364.
[28] T.M. Austin, M. Brezina, B. Jamroz, C. Jhurani, T.A. Manteuffel, J. Ruge, Semi-automatic sparse preconditioners for high-order finite element methods on non-uniform meshes, J. Comput. Phys. 231 (14) (2012) 4694–4708.
[29] L. Gao, Kronecker products on preconditioning (Ph.D. thesis), King Abdullah University of Sciences and Technology (KAUST), 2013.
[30] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, third ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[31] S. Blackford, J. Dongarra, LAPACK working note 41—installation guide for LAPACK. Tech. Rep, University of Tennessee, 1999, URL: http://www.netlib.org/lapack/lawnspdf/lawn41.pdf.
[32] J. Schoberl, J.M. Melenk, C.G.A. Pechstein, S.C. Zaglmayr, Schwarz preconditioning for high order simplicial finite elements, in: O.B. Widlund, D.E. Keyes (Eds.), Domain Decomposition Methods in Science and Engineering XVI, in: Lecture Notes in Computational Science and Engineering, vol. 55, Springer, Berlin, Heidelberg, 2007, pp. 139–150.
[33] D. Pardo, Integration of *hp*-adaptivity with a two grid solver: applications to electromagnetics (Ph.D. thesis), The University of Texas at Austin, 2004.