Hierarchical preconditioners for high-order FEM

Sabine Le Borne[1]

The finite element discretization of partial differential equations (PDEs) requires the selection of suitable finite element spaces. While high-order finite elements often lead to solutions of higher accuracy, their associated discrete linear systems of equations are often more difficult to solve (and to set up) compared to those of lower order elements.

We will present and compare preconditioners for these types of linear systems of equations. More specifically, we will use hierarchical ($\mathcal{H}$-) matrices to build block $\mathcal{H}$-LU preconditioners. $\mathcal{H}$-matrices provide a powerful technique to compute and store approximations to dense matrices in a data-sparse format. We distinguish between blackbox $\mathcal{H}$-LU preconditioners which factor the entire stiffness matrix and hybrid methods in which only certain subblocks of the matrix are factored after some problem-specific information has been exploited. We conclude with numerical results.

1 Introduction

This contribution is concerned with preconditioning the linear systems of equations arising in high-order finite element discretizations of PDEs (Šolín et al. [2004], Deville et al. [2002]). More specifically, we will introduce and analyse hybrid blackbox hierarchical matrix techniques in which

- only the stiffness matrix is given (which excludes lower order preconditioning);
- the preconditioner might use some knowledge on expected matrix properties (e.g. sparsity structure) from the underlying problem.

Hamburg University of Technology, Schwarzenbergstr. 95E, 21073 Hamburg leborne@tuhh.de

The construction of efficient solution methods for these types of systems has been a very active recent research field. Many solution approaches are based on multigrid (Heys et al. [2005]) and/or domain decomposition approaches (Lottes and Fischer [2005]), or try to construct sparse preconditioners for the dense matrices (Austin et al. [2012]). Here, we pursue a different approach in which we propose to use hierarchical matrix techniques to construct efficient preconditioners for these systems.

This paper is organized as follows: In section 2, we will introduce a model problem and a particular type of high-order finite element discretization. While in this paper we develop preconditioners for this particular setting, the intention is in future work to extend these preconditioners to a wider range of PDEs and discretization schemes. In section 3, we introduce two preconditioning approaches. The first one is blackbox, i.e., it only requires the (sparse) matrix as input and does not make any assumptions on the origin of the matrix. The second approach also requires just the matrix as input, but in addition it "knows" that it originates from some high-order finite element discretization and incorporates this knowledge into the construction of the preconditioner. In section 4, we conclude with numerical results illustrating the performance of the proposed preconditioning approaches.

2 High-order Q_p finite elements

The three-dimensional convection-diffusion equation

$$-\epsilon \Delta u + \mathbf{b} \cdot \nabla u = f \qquad \text{in } \Omega = (0,1)^3, \tag{1}$$

$$u = x^2 + y^2 + z^2 \qquad \text{on } \partial\Omega \tag{2}$$

serves as our test problem. In particular, we consider a small viscosity $\epsilon = 10^{-3}$ and a circular convection direction $\mathbf{b}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = (0.5 - y,\, x - 0.5,\, 0.0)^T$, resulting in a convection-dominant problem.

We discretize this test problem using a finite element discretization with quadrilateral Q_p finite elements. To this end, we use a "triangulation" of a (regularly refined) quadrilateral grid (cubes), and then define the finite element space $V = Q_p$ of continuous, piecewise polynomial elements of (at most) order p in each coordinate direction. As a basis for this finite element space, we use Lagrange (tensor) basis functions $\{\phi_1, \cdots, \phi_n\}$ satisfying $\phi_i(x_j) = \delta_{ij}$. The finite element discretization results in a linear system of equations $Ax = b$ whose solution yields the finite element approximation $u_h = \sum x_i \phi_i \in V$.

The following Figure 1 illustrates the sparsity structures of matrices obtained for linear ($h = \frac{1}{16}$), quadratic ($h = \frac{1}{8}$), fourth ($h = \frac{1}{4}$) and eighth ($h = \frac{1}{2}$) order basis functions, all having size 3375×3375.

The advantage of high-order elements lies in their p'th order discretization error $\|u - u_h\|_{H^1} \sim h^p$ (where $\|\cdot\|_{H^1}$ denotes the usual Sobolev H^1-norm),

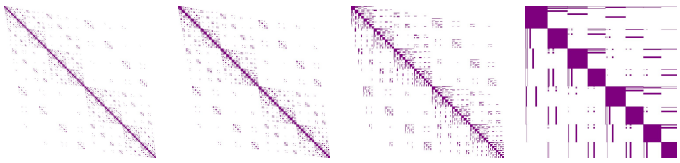

Fig. 1 Sparsity structures using linear, quadratic, fourth and eighth order polynomials, resulting in (on average) 24, 56, 186 and 850 nonzero entries per row, resp.

while their disadvantages include the linear systems to be solved to be less sparse and in general worse conditioned compared to those obtained using lower order elements.

In the following section, we introduce several block preconditioners and analyse their performance as we increase the order of discretization.

3 Block-$\mathcal{H}$-LU factorizations

The following block preconditioners are constructed through a combination of a row and column permutation of the (entire) matrix followed by an approximate LU factorization, either of the entire matrix or only of the blocks on the diagonal to be used in combination with a block Gauß-Seidel preconditioner.

3.1 Nested dissection based block structures

While high-order finite elements lead to less sparse matrices, they are usually still "sparse enough" for the well-known nested dissection ordering: Based on the matrix graph, the degrees of freedom are divided into three subsets D_1, D_2 and S. The two subsets D_1, D_2 are disconnected in the sense that $i \in D_1$ and $j \in D_2$ implies $a_{ij} = 0$ for the respective matrix entry. The subset S, the so-called "separator" which is preferably small (in the case of our three-dimensional model problem it is of order $\mathcal{O}(n^{2/3})$) consists of degrees of freedom with connections to both subsets D_1 and D_2. We recursively apply this ordering strategy to the two subsets D_1, D_2. Figure 2 shows the sparsity structures for the matrices of Figure 1 after a nested dissection reordering.

The advantage of such a reordering prior to an (approximate) LU factorization lies in the fact that an (exact) LU factorization preserves the two off-diagonal zero blocks in the 1×2 and 2×1 block positions of the 3×3 block matrix. Therefore, the LU factorizations of the first two (approximately equal sized) diagonal blocks can be computed in parallel followed by the factorization of the last, smaller block corresponding to the separator. We make

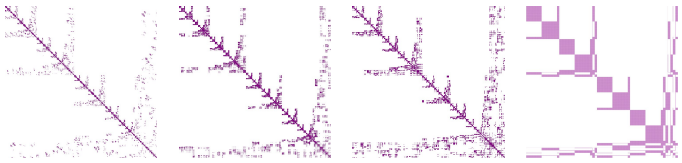

Fig. 2 Sparsity structures using linear, quadratic, fourth and eighth order polynomials after a nested dissection ordering

the following two observations with respect to the polynomial order p of the underlying finite element space:

- The relative size of the separator is independent of the polynomial order p of the finite element space (when using a Lagrange basis). This is important since it is typically the factorization of this separator block that dominates the work complexity of the factorization of the entire matrix.
- For larger p, the nested dissection "finds" the dense subblocks that correspond to so-called "bubble" functions - these are basis functions with support in a single element that can be eliminated without additional fill-in into the stiffness matrix.

The next step is the computation of an approximate LU-factorization in the hierarchical ($\mathcal{H}$-) matrix format. Hierarchical matrices have been introduced more than a decade ago (Hackbusch [1999]). They are based on a blockwise low rank approximation of off-diagonal matrix blocks. The efficient storage (of the rectangular matrix factors) of these low rank blocks reduces the storage complexity as well as the computational complexity for most matrix operations to almost optimal order (up to powers of logarithmic factors). For general details, we refer to the comprehensive lecture notes (Börm et al. [2003]), and for nested dissection based $\mathcal{H}$-LU factorization that is used in the numerical results in section 4 to (Grasedyck et al. [2009]). Without going into the details of $\mathcal{H}$-matrices, here we only exploit the fact that the accuracy of $\mathcal{H}$-LU factorizations can be controlled adaptively through a parameter $\delta_{\mathcal{H}}$. As $\delta_{\mathcal{H}} \to 0$, the $\mathcal{H}$-LU factors converge toward the exact LU factors, although at the expense of increasing storage and computational costs.

3.2 Degree of freedom type based block structures

We now propose to begin with an ordering of the degrees of freedom according to their association with vertices (V), edges (E), faces (F) or (the interior of) cells (C). Since we use a structured grid, this ordering can be obtained directly from the sparsity pattern without access to the grid geometry (through the number of nonzero entries per matrix row). Figure 3 shows the resulting reordered sparsity structures for the matrices of Figure 1. The matrices still

include the (Dirichlet) boundary vertices (resulting in the last block on the diagonal of equal size for all four matrices, here about one third of the total number of matrix rows due to the small problem size) which will be eliminated before the iterative solution for the true degrees of freedom.

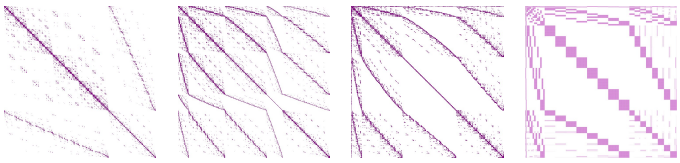

Fig. 3 Sparsity structures for linear, quadratic, fourth and eighth order polynomials after a reordering based on the types of degree of freedom (vertex, edge, face, cell or boundary)

Besides this last block of boundary vertices, the matrices for quadratic, fourth and eighth degree polynomials have an additional common structure of four blocks on the diagonal according to their vertex, edge, face and (interior) cell degrees of freedom. The block corresponding to the (interior) cell degrees of freedom is the easiest one to recognize since it is of block diagonal structure, and its relative size, as well as the size of the dense blocks on the diagonal, increases with the increase in polynomial order. In theory, this block can also be eliminated without creating any additional fill-in in the remaining matrix. This process is called "static condensation" which is known to be sometimes ill-conditioned. For smaller polynomial order p, this can also be done in practice, but for high order p, the computational costs of such an elimination increase substantially and may dominate the overall computational costs for the solution of the linear system.

The resulting block structure can be used for Jacobi or Gauß-Seidel block preconditioners. These preconditioners require (approximate) solvers for the matrix blocks on the diagonal for which we will again use $\mathcal{H}$-LU factorizations. In particular, we will compute $\mathcal{H}$-LU factorizations for the two diagonal blocks in the matrices (3) and (4).

No static condensation:

$$\begin{pmatrix} A_{VV} & & & \\ & A_{EE} & A_{EF} & A_{EC} \\ & A_{FE} & A_{FF} & A_{FC} \\ & A_{CE} & A_{CF} & A_{CC} \end{pmatrix}, \quad \begin{pmatrix} A_{VV} & A_{VE} & & \\ A_{EV} & A_{EE} & & \\ & & A_{FF} & A_{FC} \\ & & A_{CF} & A_{CC} \end{pmatrix}, \tag{3}$$

After static condensation of interior cell (C) degrees of freedom:

$$\begin{pmatrix} \overline{A_{VV}} & & & \\ & \overline{A_{EE}} & \overline{A_{EF}} & \\ & \overline{A_{FE}} & \overline{A_{FF}} & \\ & & & \end{pmatrix}, \quad \begin{pmatrix} \overline{A_{VV}} & \overline{A_{VE}} & & \\ \overline{A_{EV}} & \overline{A_{EE}} & & \\ & & \overline{A_{FF}} & \end{pmatrix}. \tag{4}$$

The bars above the blocks after static condensation indicate that the matrix entries will have changed after static condensation while the sparsity pattern is the same as before. The $\mathcal{H}$-LU factorizations of the blocks on the diagonal will once more be preceded by a nested dissection ordering (within these subblocks) as described in the previous subsection.

4 Numerical results and conclusions

In this section, we will provide a small selection of numerical results for the various preconditioners that have been introduced in subsections 3.1 and 3.2. All tests were performed on a DELL Latitude E6530 laptop (2.60GHz, 16GB). The discrete linear systems have been produced by the software package deal.II (Bangerth et al. [2013, 2007]), and we use the $\mathcal{H}$-arithmetic of the HLib package (Börm and Grasedyck).

As an iterative solver, we use a preconditioned BiCGStab method, and we stop the iteration once the residual is reduced by a factor of 10^{-6}, i.e., when $\|r_k\|_2 \leq 10^{-6}\|r_0\|_2 = 10^{-6}\|b\|_2$ since we use the starting vector $x_0 = 0$. In Table 1, we show numerical results for linear finite elements: As we increase the level of grid refinement, the number of degrees of freedom (dofs) increases by a factor of approximately 8. A comparable factor of increase can be observed in the storage and set-up times of the $\mathcal{H}$-LU factorizations which have been performed for a relative accuracy of $\delta_{\mathcal{H}} \in \{0.1, 0.01\}$. The convergence rates of the preconditioned BiCGStab method deteriorate only modestly as the problem size increases, i.e, we obtain a robust solution method (the BiCGStab time for the problem size on level 7 for the higher $\mathcal{H}$-accuracy $\delta_{\mathcal{H}} = 0.01$ is slowed down by swapping and hence put into parentheses).

Table 1 $\mathcal{H}$-LU: Time (seconds) and storage (MB), linear elements

dofs	3,375	$29 \cdot 10^3$	$250 \cdot 10^3$	$2 \cdot 10^6$	3,375	$29 \cdot 10^3$	$250 \cdot 10^3$	$2 \cdot 10^6$
level	4	5	6	7	4	5	6	7
	$\mathcal{H}$-accuracy $\delta_{\mathcal{H}} = 0.1$				$\mathcal{H}$-accuracy $\delta_{\mathcal{H}} = 0.01$			
Storage (MB)	7.1	100	969	8611	10	151	1556	13749
Set-up time (s)	0.3	6.6	91	796	0.7	17	271	2494
BiCGStab steps	5	7	10	23	2	3	4	4
time (s)	0.0	0.2	3.2	67.5	0.0	0.1	1.7	(30.0)

Table 2 shows respective results for finite element spaces using polynomial degrees $p = 4$, 6 and 9. We had to use higher accuracies for the $\mathcal{H}$-LU factorizations (now $\delta_{\mathcal{H}} \in \{10^{-1}, 10^{-2}\}$ for $p = 4, 6$ and even $\delta_{\mathcal{H}} \in \{10^{-4}, 10^{-5}\}$ for $p = 9$) in order to obtain convergence in the BiCGStab iteration, which is no surprise in view of the worse conditioning of the matrices for high-

Table 2 $\mathcal{H}$-LU: Time (sec) and storage (MB), polynomial degrees 4, 6, and 9

level	3	4	2	3		1	2
$\mathcal{H}$-acc	degree 4		degree 6		$\mathcal{H}$-acc	degree 9	
	Storage (MB)						
10^{-1}	117	1201	32	376	10^{-4}	33	485
10^{-2}	181	1977	61	733	10^{-5}	36	543
stiffness matrix	82	654	82	654		90	716
$\mathcal{H}$-acc	Set-up time (s)						
10^{-1}	10.3	130	1.6	30.8	10^{-4}	3.5	110
10^{-2}	31	438	5.7	123	10^{-5}	3.7	142
$\mathcal{H}$-acc	BiCGStab (steps/time)						
10^{-1}	22/1.0	20/9.3	div	div	10^{-4}	4/0.1	8/1.7
10^{-2}	3/0.2	4/2.7	8/0.2	10/2.9	10^{-5}	2/0.04	3/0.7

order elements. Once more, for an "accurate enough" $\mathcal{H}$-LU factorization as a preconditioner, we obtain an almost optimal iterative method.

Table 3 shows a comparison of the preconditioners introduced in subsection 3.1 versus those of subsection 3.2, here shown for elements of polynomial degree 4 on a level-4 grid, resulting in a total of about 250,000 dofs. As we "drop" certain off-diagonal blocks from the matrix before its factorization, the required storage and set-up times are reduced. The preconditioned BiCGStab iteration now requires additional steps and hence more time, but the savings in set-up time appear to be more significant than the disadvantage in iteration time (as long as we still have a convergent method).

Table 3 $\mathcal{H}$-LU: Time (sec) and storage (MB), polynomial degree 4, $\approx$ 250,000 dofs

prec	$\begin{pmatrix} vv & ve & vf & vc \\ ev & ee & ef & ec \\ fv & fe & ff & fc \\ cv & ce & cf & cc \end{pmatrix}$	$\begin{pmatrix} vv & & & \\ & ee & ef & ec \\ & fe & ff & fc \\ & ce & cf & cc \end{pmatrix}$	$\begin{pmatrix} vv & ve & & \\ ev & ee & & \\ & & ff & fc \\ & & cf & cc \end{pmatrix}$	$\begin{pmatrix} vv & & & \\ & ee & & \\ & & ff & fc \\ & & cf & cc \end{pmatrix}$
$\mathcal{H}$-acc	Storage (MB)			
10^{-1}	1201	1184	1008	1001
10^{-2}	1977	1902	1516	1486
$\mathcal{H}$-acc	Set-up time (s)			
10^{-1}	129	123	72	70
10^{-2}	438	393	215	207
$\mathcal{H}$-acc	BiCGStab (steps/time)			
10^{-1}	20/9.3	26/12.0	56/22.9	59/24.1
10^{-2}	4/2.7	8/5.2	47/25.6	49/26.2

Finally, in Table 4 we show results where the matrix size has been reduced by eliminating the "bubble" dofs through static condensation before the system is solved iteratively. For 4th order polynomials on a level-4 grid (about 250,000 dofs before static condensation), the "naive" static condensation took 120 seconds, which is significantly more than the savings obtained through now solving a smaller system so that it is not recommended unless a more efficient implementation of static condensation can be found.

Table 4 BiCGStab: steps/time for Method "static condensation" (top) and Method "no static condensation" (bottom), "br" denotes a BiCGStab breakdown

prec	$\begin{pmatrix} vv & ve & vf \\ ev & ee & ef \\ fv & fe & ff \end{pmatrix}$	$\begin{pmatrix} vv & & \\ & ee & ef \\ & fe & ff \end{pmatrix}$	$\begin{pmatrix} vv & ve & \\ ev & ee & \\ & & ff \end{pmatrix}$	$\begin{pmatrix} vv & & \\ & ee & \\ & & ff \end{pmatrix}$
10^{-1}	17/4.8	19/5.2	49/br	53/12.6
10^{-2}	4/1.9	8/3.5	42/15.6	44/15.7
prec	$\begin{pmatrix} vv & ve & vf & vc \\ ev & ee & ef & ec \\ fv & fe & ff & fc \\ cv & ce & cf & cc \end{pmatrix}$	$\begin{pmatrix} vv & & & \\ & ee & ef & ec \\ & fe & ff & fc \\ & ce & cf & cc \end{pmatrix}$	$\begin{pmatrix} vv & ve & & \\ ev & ee & & \\ & & ff & fc \\ & & cf & cc \end{pmatrix}$	$\begin{pmatrix} vv & & & \\ & ee & & \\ & & ff & fc \\ & & cf & cc \end{pmatrix}$
10^{-1}	20/9.3	26/12.0	56/22.9	59/24.1
10^{-2}	4/2.7	8/5.2	47/25.6	49/26.2

References

T. M. Austin, M. Brezina, B. Jamroz, C. Jhurani, T. A. Manteuffel, and J. Ruge. Semi-automatic sparse preconditioners for high-order finite element methods on non-uniform meshes. *J. Comput. Physics*, pages 4694–4708, 2012.

W. Bangerth, R. Hartmann, and G. Kanschat. deal.II – a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4): 24/1–24/27, 2007.

W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and T. D. Young. The `deal.ii` library, version 8.1. *arXiv preprint* `http://arxiv.org/abs/1312.2266v4`, 2013.

S. Börm and L. Grasedyck. HLIB version 1.3. Available at www.hlib.org.

S. Börm, L. Grasedyck, and W. Hackbusch. Hierarchical matrices, 2003. Lecture Notes No. 21, Max-Planck-Institute for Mathematics in the Sciences, Leipzig, Germany.

M. O. Deville, P. F. Fischer, and E. H. Mund. *High-order methods for incompressible fluid flow*, volume 9 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 2002.

L. Grasedyck, R. Kriemann, and S. Le Borne. Domain decomposition based $\mathcal{H}$-LU preconditioning. *Numerische Mathematik*, 112:565–600, 2009.

W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62:89–108, 1999.

J. J. Heys, T. A. Manteuffel, S. F. McCormick, and L. N. Olson. Algebraic multigrid for higher-order finite elements. *J. Comput. Phys.*, 204(2):520–532, 2005.

J. Lottes and P. Fischer. Hybrid multigrid/Schwarz algorithms for the spectral element method. *J. Sci. Comp.*, 24:45–78, 2005.

P. Šolín, K. Segeth, and I. Doležel. *Higher-order finite element methods*. Studies in Advanced Mathematics. Chapman & Hall/CRC, Boca Raton, London, 2004.