Performance of some preconditioners for the p- and hp-version of the finite element method in 3D

Chr. Badura and W. Dörfler*

1 Introduction.

The finite element method is a fundamental tool for the solution of partial differential equations. While the classical h-version approximates the solution on refined grids with a fixed piecewise polynomial degree, the p-version uses increasing piecewise polynomial degree on a fixed grid. The combination of both results in the hp-version, featuring (for correctly chosen grids) an exponential convergence of the form $E := \|u - u_h\|_{H^1} \leq C\exp(-\gamma N^\beta)$ [3] instead of just an algebraic one of the form $\|u-u_h\|_{H^1} \leq CN^{-\alpha}$ achieved by the h-version [4], see also Figures 1, 2, and Tables 1, 2.

In this paper we consider the Laplace equation

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

which yields after discretisation a symmetric positive definite stiffness matrix. Depending on the choice of the finite elements, the stiffness matrix either is sparse or consists of dense blocks of small dimensions which themselves are sparsely distributed over the matrix.

We will use the CG-algorithm to solve the linear equations. This method can be accelerated by application of a preconditioner. Depending on the sparsity pattern of the matrix various preconditioners promise good results. In this paper we want to compare some of them.

2 Design Decisions.

We consider discretisations based on cubes. For assigning the basis functions to the vertices, edges, and faces, we use the 3D serendipity space [2]. This space exceeds the space $\mathcal{P}^p$ of polynomials up to a given degree p by $3p+3$

*Fachbereich Mathematik, Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany. Email: doerfler@mathematik.uni-kl.de.

degrees of freedom [5], i. e. is just insignificantly larger since both spaces are of order $\mathcal{O}(p^3)$.

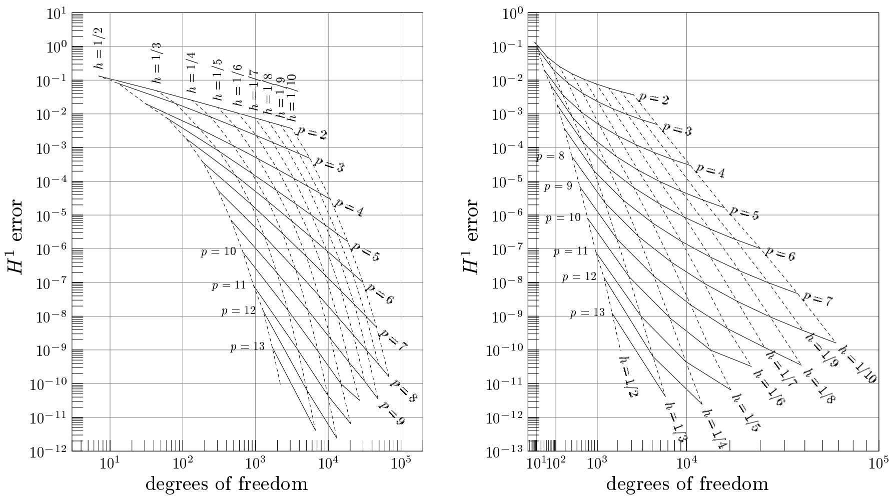


Figure 1: Uniform grid (see Figure 4): H^1-error E depending on the number of degrees of freedom. Using $\log E \times \log N$ and $\log E \times N^{1/3}$ scale. p: polynomial degree, h: grid width. $u(x) = \sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3)$.

h	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$	$p=7$	$p=8$	$p=9$	$p=10$
1/3	2.497	3.377	4.189	5.551	6.276	7.083	7.850	8.842	9.775
1/4	2.275	3.213	4.030	5.257	6.095	6.978	7.871	8.815	9.838
1/5	2.168	3.131	4.000	5.148	6.042	6.963	7.915	8.851	9.861
1/6	2.112	3.089	3.993	5.097	6.022	6.966	7.942	8.876	–
1/7	2.080	3.064	3.992	5.068	6.013	6.972	7.957	8.523	–
1/8	2.060	3.048	3.993	5.051	6.009	6.976	7.956	–	–
1/9	2.047	3.037	3.994	5.039	6.006	6.980	7.807	–	–

Table 1: Uniform grid (Figure 4): experimental order of convergence. Showing α where $E = Ch^\alpha$. $u(x) = \sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3)$.

In this work we assume that the elements are oriented parallel to the coordinate axes. Hence local adaption has to be achieved by non-conforming refinement (Figures 5 and 6). Although this yields a sparse stiffness matrix, it also results in preconditioning problems which are dealt with in section 5.

According to [10][8], we decompose the degrees of freedom (dofs) into inner dofs and boundary dofs, where the latter consist of face, edge, and vertex dofs.

3 Stiffness Matrix.

Since the elements' edges are parallel to the coordinate axes, the local stiffness matrices are sparse. Consequently, the global matrix is also sparse [5].

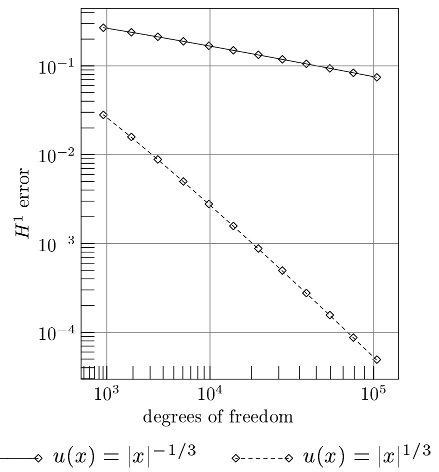


Figure 2: Geometric grid (Figure 6): H^1-error E depending on the number of degrees of freedom. Using $\log E \times N^{1/5}$ scale.

p	$u(x) = \|x\|^{-1/3}$	$u(x) = \|x\|^{1/3}$
4	0.270	0.271
5	0.243	0.250
6	0.232	0.239
7	0.228	0.234
8	0.227	0.233
9	0.227	0.232
10	0.231	0.233
11	0.228	0.234

Table 2: Geometric grid (Figure 6): experimental order of convergence. Showing β where $E = C\exp(-\gamma N^\beta)$.

Hence it can be stored in the compressed column format [6, Ch. 4].

If the elements are not parallel to the axes, the local matrices are dense so that it makes sense to store the stiffness matrix as a sparse structure consisting of dense blocks where blocks of adjacent elements overlap. The matrix and its preconditioners will strongly depend on the considered finite element method, particularly on the used grid.

The stiffness matrix is composed of local stiffness matrices A_k corresponding to the k-th element,

$$A = \sum_k N_k A_k N_k^T. \tag{2}$$

The matrix N_k maps local dofs into global dofs and hence consist only of 1's and 0's where there is just one 1 per column. Of course, N_k is not stored as dense matrix, but for each local dof just its global index is stored.

We will consider preconditioning techniques for these two different storage techniques.

4 Preconditioners.

The p- and hp-version constitutes a major improvement of the finite element method compared to the h-version. Yet the performance of the iterative solver can be improved significantly by using a preconditioner—for the possible acceleration measured in CPU time see e. g. Figure 10.

Preconditioning is based on the fact that the number of iterations needed for solving

$$Ax = b$$

depends on the condition number $\kappa(A) = \lambda_{\max}/\lambda_{\min}$. For acceleration, the modified equation

$$B^{-1}Ax = B^{-1}b$$

is solved [12], where we demand $\kappa(B^{-1}A) \ll \kappa(A)$. Clearly, the gain achieved by improving the condition number $\kappa(B^{-1}A)$ should surpass the additional computational effort needed for applying B^{-1}.

Since the stiffness matrix can either be stored as sparse matrix or as block matrix, there are several possible preconditioners. In the sequel we consider the following combinations:

sparse matrix:
- (1) without preconditioner
- (2) with diagonal scaling [14, §4.6.3]
- (3) with SSOR(ω) preconditioning where $\omega = 1$ [13, §7.6] [5]
- (4) with incomplete Cholesky decomposition [9, §2.2]

block matrix:
- (5) without preconditioner
- (6) with static condensation [2]
- (7) with static condensation and diagonal scaling
- (8) with partial orthogonalization [10]
- (9) with two-level domain decomposition [10]

The preconditioners based on the sparse matrix are well-known. For the others (5)-(9) we want to add some notes on the implementation. The data vectors v are stored globally, but for multiplication by A, we temporarily use local vectors $N_k^T v$, corresponding to the k-th element. Thus the matrix A is stored in dense blocks A_k according to (2).

4.1 Static Condensation.

The inner dofs are not coupled with dofs of any other element. Additionally, an element of polynomial degree p has $\mathcal{O}(p^3)$ inner dofs and just $\mathcal{O}(p^2)$ boundary dofs. Hence, for high degrees p it is desirable to locally eliminate inner dofs, so that the only dofs that have to be eliminated globally by the CG-method are the remaining boundary dofs. This preconditioner is called *static condensation*.

It has the advantage that it is well fitted for parallelization [1]. But as can be seen in Table 3, for the current polynomial degrees of $p = 1, \ldots, 15$, the number of inner dofs does not yet dominate the number of all dofs so that this preconditioner is less efficient than expected.

4.2 Partial Orthogonalization.

Partial orthogonalization requires the elimination of the inner dofs in advance, i. e. static condensation has to be applied first. Then the face functions, edge functions, and vertex functions are partially orthogonalized successively [10].

Therefore, all face functions, edge functions, and vertex functions need to be tied together to blocks corresponding to the faces, edges, and vertices. Since the stiffness matrix initially is decomposed into blocks per element, we need to reorder all degrees of freedom for applying the partial orthogonalization. This reordering and the partial orthogonalization are quite time consuming, but yet it yields a very efficient preconditioner, see Figures 7 – 10.

4.3 Two-level Domain Decomposition.

Although partial orthogonalization performs very well (the condition number is reduced from about 10^6 to less than 100), its condition number shows a strange peak for polynomial degrees between 3 and 6, see Figure 3a. These peaks can be cut off by application of an additional preconditioner to the partially orthogonalized stiffness matrix (3b). According to Figure 3, the condition number seems to depend polynomially on p and logarithmicly on h.

As example, we choose the *two-level domain decomposition* preconditioner, preconditioning each block by diagonal scaling. Since these blocks usually are singular, we have to make sure that we perform all computations in the complements of the sub-matrices' kernels [10].

Instead, diagonal scaling of the complete partially orthogonalized stiffness matrix probably would give a similar result with less computational cost, but we did not test that alternative.

4.4 Stopping Criterion.

For a fair comparison of these preconditioners the stopping criterion of the CG-method should not depend on the preconditioners. Thus we should not use the usual criterion $\|x - x_n\|_{B^{-1}A} < \varepsilon$ but instead $\|x - x_n\|_2 < \varepsilon$, which is guaranteed by

$$\frac{(B^{-1}r_j, r_j)^{1/2}}{\|x_j\|_2} < \frac{\lambda_{\min}(B^{-1}A)}{\lambda_{\max}(B^{-1})^{1/2}} \, \frac{\varepsilon}{1+\varepsilon},$$

where $r_j := b - Ax_j$ [5, Lemma 4.3].

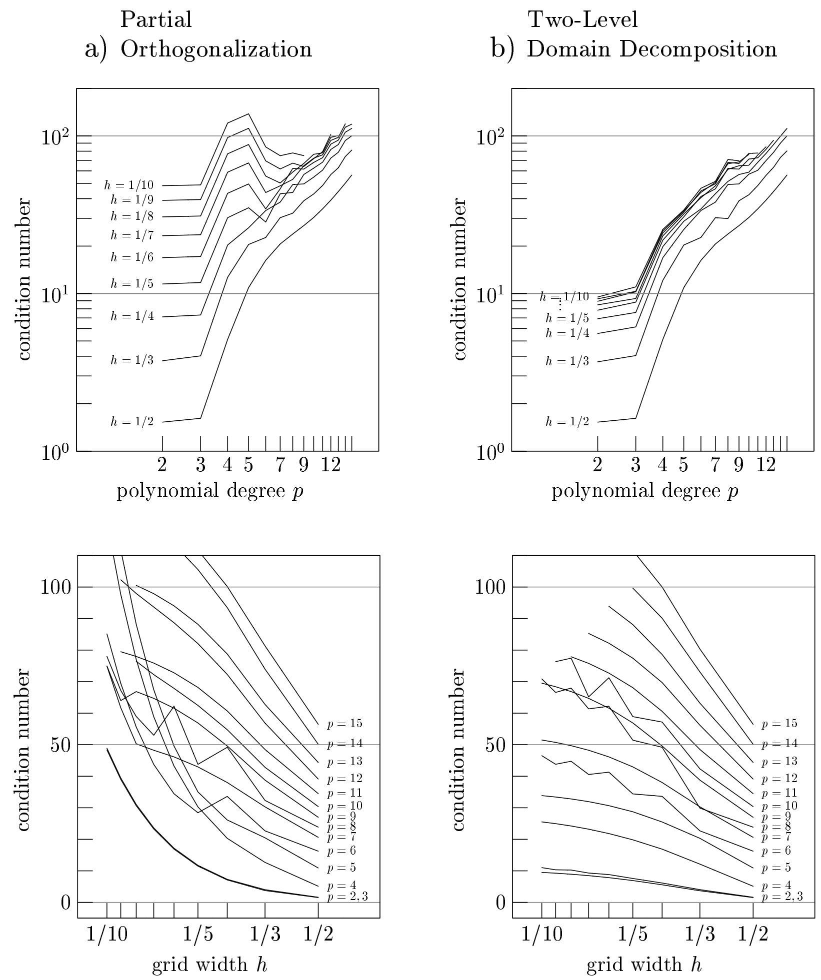


Figure 3: Uniform grid (Figure 4): condition number as function of the grid width h and of the polynomial degree p. Using $\log \kappa \times \log p$ scale and $\kappa \times \log h$ scale.

5 Non-conforming Refinement.

Since non-conforming refinement usually destroys the continuity of the ansatz space,we have to eliminate explicitly those dofs which otherwise would result in jumps of the trial functions [1][7][5].

This elimination corresponds to a matrix $P \in \mathbb{R}^{m \times n}$ where $m > n$, so that we have to solve the problem

$$P^T A P \tilde{x} = P^T b$$

instead of

$$Ax = b,$$

where $x = P\tilde{x}$.

By appending columns consisting of zeros, i. e. defining $\bar{P} := (\, P \; 0 \,) \in \mathbb{R}^{n \times n}$ and $\bar{x} := (\, \tilde{x} \; 0 \,)^T \in \mathbb{R}^n$ we obtain the same result when considering

$$\bar{P}^T A \bar{P} \bar{x} = \bar{P}^T b.$$

Note that this is more easily implemented since $\bar{P}^T A \bar{P}$ has the same dimension as A.

Usually, the matrix $\bar{P}^T A \bar{P}$ is not computed explicitly, but the action is obtained by applying $\bar{P}$, A, and $\bar{P}^T$ successively. Therefore it is nearly impossible to find a preconditioner for $\bar{P}^T A \bar{P}$ whereas it is quite easy finding one for A.

As a remedy, we can consider the following preconditioned matrices:

$B^{-1}A$: This corresponds to a non-continuous ansatz space, but the correct preconditioner.

$B^{-1}\bar{P}^T A \bar{P}$: This corresponds to a continuous ansatz space, but the incorrect preconditioner, since B is a preconditioner for A, but not for $\bar{P}^T A \bar{P}$.

$\tilde{B}^{-1}\bar{P}^T A \bar{P}$: Here, $\tilde{B}$ means the preconditioner for $\bar{P}^T A \bar{P}$, so that this corresponds to a continuous ansatz space and the correct preconditioner. But we have mentioned before that the computation of $\tilde{B}$ may be highly inefficient if not even impossible.

As can be seen in Figure 8, there is no fundamental loss of quality of the preconditioner considering just the approximate preconditioner instead of the exact one. The only exception of that is formed by the incomplete Cholesky decomposition, which is absolutely inadequate for that case. Therefore, as simplified preconditioner we use the form $B^{-1}\bar{P}^T A \bar{P}$. Note that this problem does not occur for static condensation [5].

6 Numerical Tests.

We solve the Laplace equation (1) where we use the following values for the solution u and the domain Ω.

Model Problem 1.

$$\Omega = (0,1)^3,$$
$$u(x) = \sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3), \quad x = (x_1, x_2, x_3) \in \mathbb{R}^3.$$

We use a uniform grid (Figure 4) of grid widths $h = \frac{1}{2}, \ldots, \frac{1}{10}$ and constant polynomial degrees $p = 1, \ldots, 13$.

Model Problem 2.

$$\Omega = (-1,1)^3 \setminus [0,1)^3,$$
$$u(x) = |x|^{-1/3} \text{ and } u(x) = |x|^{1/3}.$$

The grid (Figure 6) is geometrically, non-conformingly refined near the point (0,0,0). The polynomial degree of the finest element is 1 and on the elements on the next coarser level the degree is (recursively) increased by 1 [4].

6.1 Comparison of the Preconditioners.

In Figures 7 and 8 we compare the efficiency of the various preconditioners by plotting the condition number of the preconditioned problem against the condition number of the unpreconditioned problem for model problem 1 and 2 respectively. In Figure 7 we recognize that except for partial orthogonalization the preconditioners' efficiencies are nearly independent of whether the dofs are induced by decreasing the grid width h or by increasing the polynomial degree p. And we find that each preconditioner has a similar efficiency both for the geometric and the uniform grids (see regression line in Figure 8).As we have mentioned before, for the geometric grid the incomplete Cholesky decomposition is not practicable when using the simplified preconditioner explained in section 5.

For the preconditioners' performance the CPU time needed for applying the CG-method up to the chosen stopping criterion is even more crucial than the condition numbers (Figures 9 and 10).

If the stiffness matrix is sparse, a storing convention reflecting this aspect is more efficient than storing it as block matrix (Figures 9 and 10). Hence we compare the preconditioners Diagonal Scaling, SSOR(1), and Incomplete Cholesky, which are based on a sparse structure, against the unpreconditioned problem where the matrix is also stored in a sparse structure. In contrary, the other preconditioners are compared against a matrix stored as

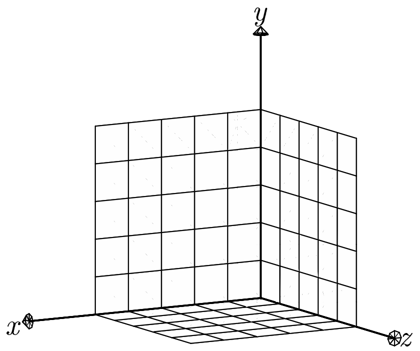


Figure 4: Uniform grid. Domain $\Omega = (0,1)^3$.

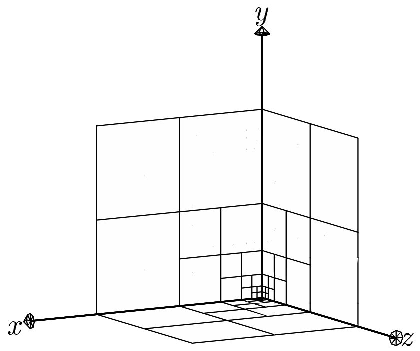


Figure 5: Geometric grid with non-conforming refinement. Domain $\Omega = (0,1)^3$.

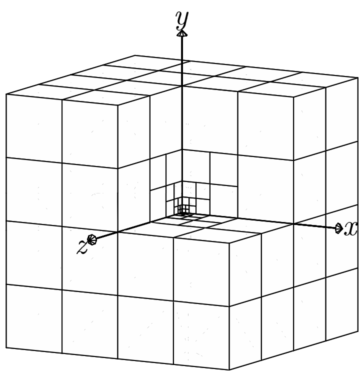
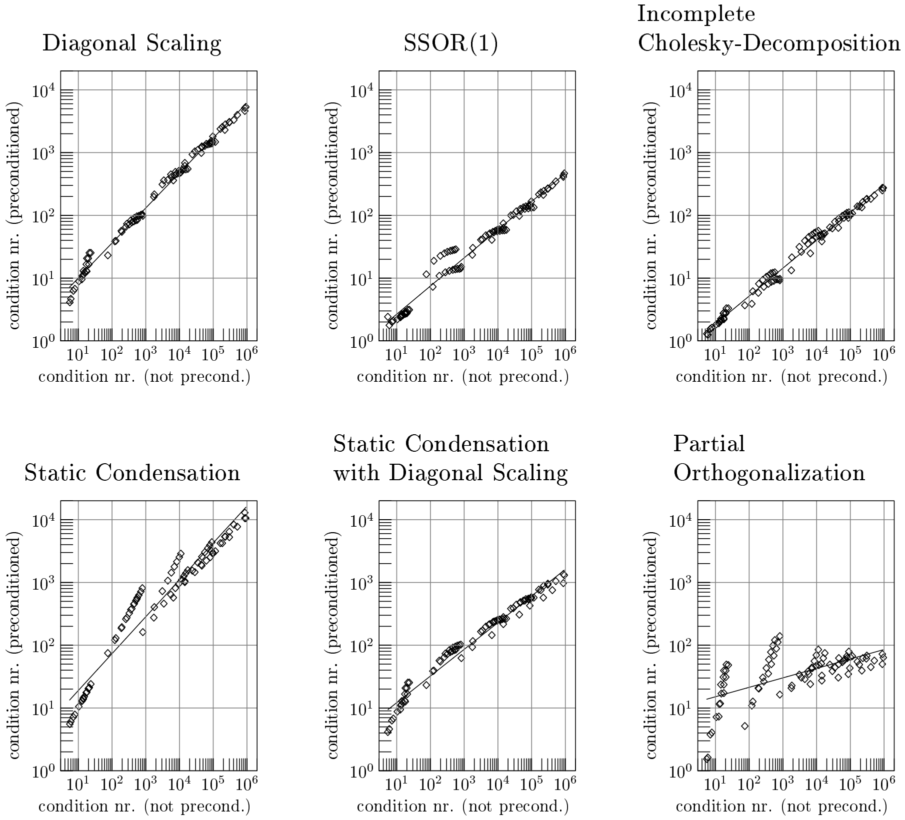
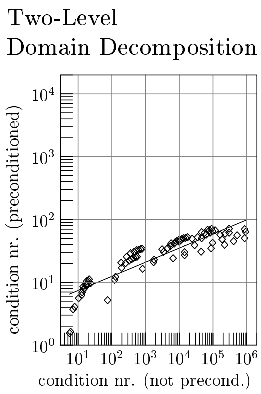


Figure 6: Geometric grid with non-conforming refinement. Domain $\Omega = (-1,1)^3 \setminus [0,1)^3$.

$p_{\max}$	N	N_B
2	341	341
3	906	906
4	1909	1909
5	3545	3545
6	6058	6009
7	9741	9496
8	14936	14201
9	22034	20319
10	31475	28045
11	43748	37574
12	59391	49101
13	78991	62821
14	103184	78929
15	132655	97620

Table 3: Geometric grid (Figure 6): Number of all dofs N *and of all boundary dofs* N_B.





The added line was computed by linear regression and has the form $b\kappa^a$ where

	a	b
Diagonal Scaling	0.558	2.775
SSOR(1)	0.448	0.948
Incomplete Cholesky Decomposition	0.444	0.669
Static Condensation	0.588	4.901
Static Cond. with Diagonal Scaling	0.425	4.567
Partial Orthogonalization	0.151	10.670
Two-Level Domain Decomposition	0.225	4.410

Figure 7: Efficiency of the preconditioner for the uniform grid (Figure 4): The condition number of the preconditioned problem against the condition number of the unpreconditioned problem.

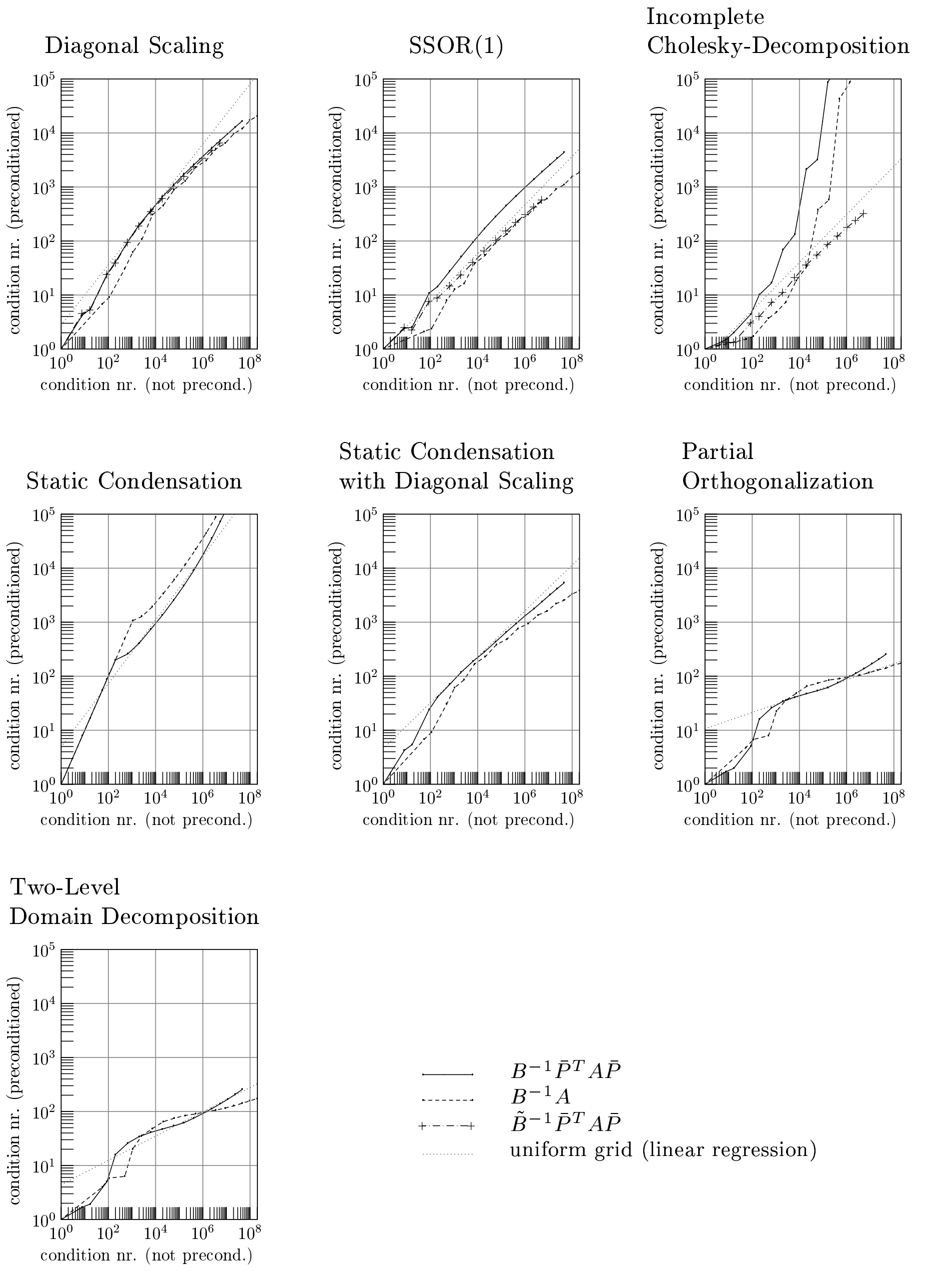


Figure 8: Geometric grid (Figure 5): Efficiency of the preconditioners with different handling of non-conforming elements.

dense blocks (although in our examples the stiffness matrix always is sparse and hence could be stored in a sparse structure).

There is a significant difference between the uniform and the geometric grid. For the uniform grid, mainly diagonal scaling and static condensation reduce the needed CPU time, and for very large problems we could also expect an improvement with partial orthogonalization and two-level domain decomposition. For the geometric grid, partial orthogonalization and two-level domain decomposition clearly give the best result, followed by diagonal scaling.

Only for large problems partial orthogonalization and two-level domain decomposition excel the unpreconditioned problem based on a sparse structure of the matrix. So, if the stiffness matrix is sparse, for a medium sized problem a matrix stored in a sparse structure, preconditioned with the easily implemented diagonal scaling is preferred for these examples.

6.2 Convergence.

Next, we plot the H^1-error E vs. the number of degrees of freedom N (Figures 1 and 2) and compute the resulting orders of convergence (Tables 1 and 2). Note that for a uniform grid the order of convergence (the α in $E = Ch^\alpha$) should coincide with the polynomial degree p [11]. For a geometric grid, a dependency of the form $E \leq C\exp(-\gamma N^\beta)$ with $\beta > \frac{1}{5}$ is expected [3].

Eventually, for model problem 2 we plot the H^1-error against the CPU time t (Figure 11) and choose such a scale that an exponential dependency of the kind $E = C\exp(-\gamma t^\beta)$ should approximately result in a straight line.

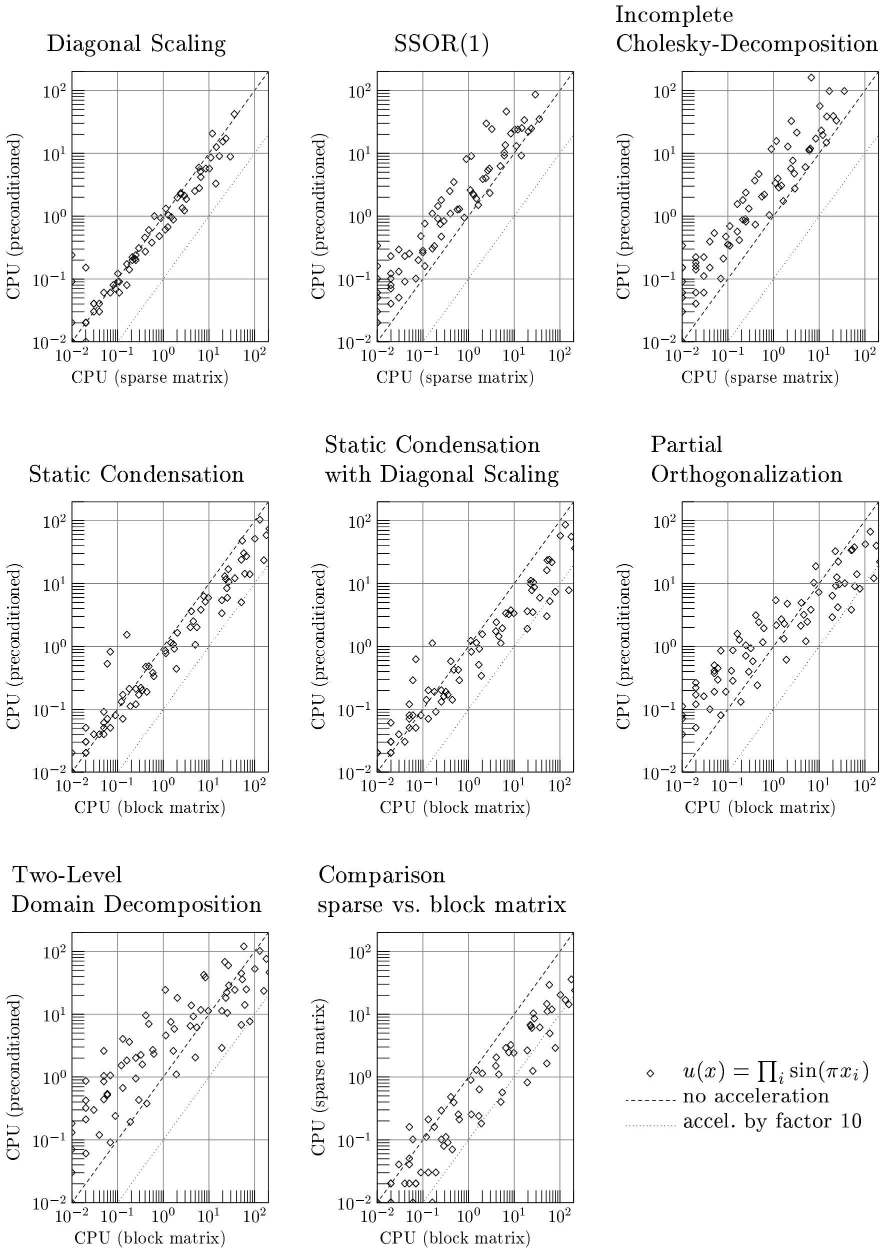


Figure 9: Uniform grid (Figure 4): Performance of the preconditioners: CPU time for solving the preconditioned system compared to the CPU time for solving the unpreconditioned system using a block/sparse matrix. The CPU time is measured in seconds.

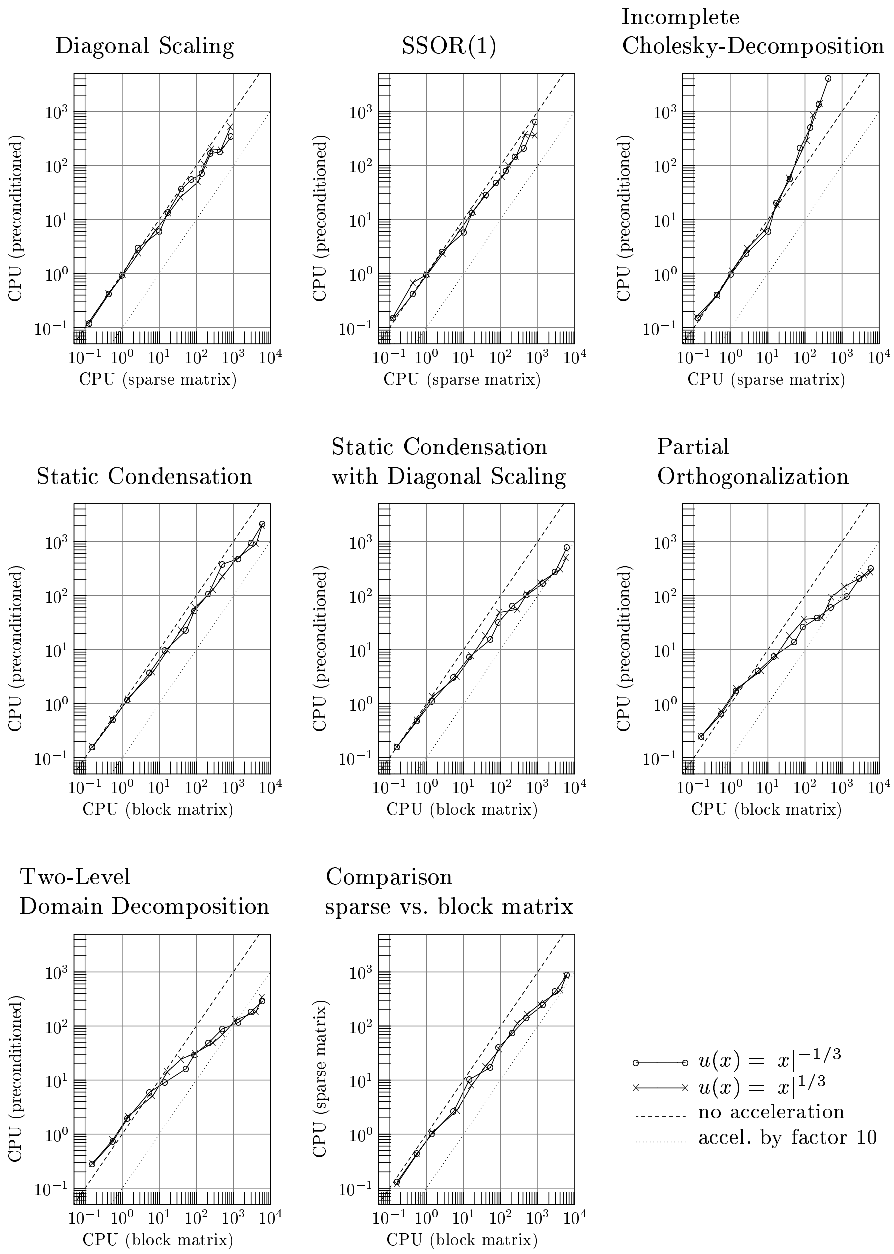
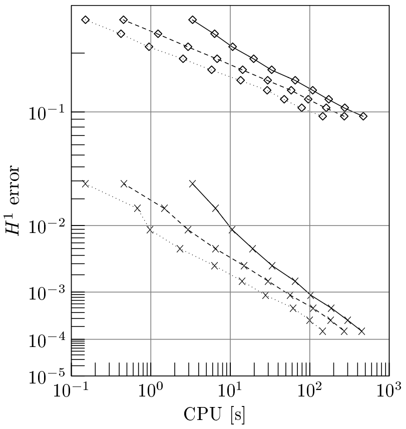


Figure 10: Geometric grid (Figure 6): Performance of the preconditioners: CPU time for solving the preconditioned system compared to the CPU time for solving the unpreconditioned system using a block/sparse matrix. The CPU time is measured in seconds.

CPU time of various parts of the program.



—— CPU time of the complete program.

-------- CPU time needed for setting up the stiffness matrix, the right hand side, and for solving the system of linear equations (i. e. without computation of boundary conditions).

.... CPU time of the solver.

$\diamond$ $u(x) := |x|^{-1/3}$
$\times$ $u(x) := |x|^{1/3}$

Figure 11: Geometric grid (Figure 6): H^1-error E using $SSOR(1)$ as preconditioner against CPU time. Using $-\log(-\log(\|E\|_{H^1})) \times \log(t)$ scale.

References

[1] M. Ainsworth and B. Senior. Aspects of an adaptive *hp*-finite element method: Adaptive strategy, conforming approximation and efficient solvers. *Comput. Methods Appl. Mech. Eng.*, 150:65–87, 1997.

[2] I. Babuška, M. Griebel, and J. Pitkäranta. The problem of selecting the shape functions for a *p*-type finite element. *Int. J. Numer. Methods Eng.*, 28:1891–1908, 1989.

[3] I. Babuška and B. Q. Guo. Approximation propertiers of the *h-p* version of the finite element method. *Comput. Methods Appl Mech. Eng.*, 133:319–346, 1996.

[4] I. Babuška and M. Suri. The *p* and *h-p* version of the finite element method, basic principles and properties. *SIAM Review*, 36:578–632, 1994.

[5] C. Badura. Implementation einer vorkonditionierten *hp*-Version der Finiten Elemente Methode in 3D, 1999. Diploma thesis. Institut für Angewandte Mathematik, Universität Freiburg.

[6] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods.* SIAM, Philadelphia, PA, 1993.

[7] L. Demkowicz, K. Gerdes, C. Schwab, A. Bajer, and T. Walsh. HP90: A general and flexible Fortran 90 *hp*-FE code. *Comput. Vis. Sci.*, 1:145–163, 1998.

[8] M. Dryja, B. F. Smith, and O. B. Widlund. Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions. *SIAM J. Numer. Anal.*, 31:1662–1694, 1994.

[9] G. Hämmerlin and K.-H. Hoffmann. *Numerische Mathematik.* Springer, Berlin, 1994.

[10] J. Mandel. Two-level domain decomposition preconditioning for the p-version finite element method in three dimensions. *Int. J. Numer. Methods Eng.*, 29:1095–1108, 1990.

[11] R. Muñoz-Sola. Polynomial liftings on a tetrahedron and applications to the h-p version of the finite element method in three dimensions. *SIAM J. Numer. Anal.*, 34:282–314, 1997.

[12] Y. Saad. *Iterative methods for sparse linear systems.* PWS, Boston, 1996.

[13] R. Schaback and H. Werner. *Numerische Mathematik.* Springer, Berlin, 1992.

[14] H. R. Schwarz. *Methode der finiten Elemente.* Teubner, Stuttgart, 1991.