Computing the preconditioner for the Schur complement

K. Moriya[*] T. Nodera[†]

(Received 27 October 2004; revised 11 March 2005)

Abstract

The Newton scheme is used to construct an approximate inverse preconditioner for the Schur complement. However, this scheme is very expensive because of the computation cost of the matrix-matrix product. In this paper, the computation cost of the Newton scheme is reduced by implementing the preconditioner implicitly using the matrix-vector product. We also show that such an implementation is less expensive than computing the preconditioner explicitly.

Contents

[*]Dept. of Information and Integrated Tech., Aoyama Gakuin University, Japan. mailto:moriya@it.aoyama.ac.jp

[†]Dept. of Math., Keio University, Japan. mailto:nodera@math.keio.ac.jp

See http://anziamj.austms.org.au/V46/CTAC2004/Mori for this article, © Austral. Mathematical Soc. 2005. Published May 23, 2005. ISSN 1446-8735

1 Introduction

We consider the problem of solving a large and sparse linear system of equations:

$$A\boldsymbol{x} = \boldsymbol{b}\,, \quad A \in R^{n\times n}\,, \quad \boldsymbol{x}, \boldsymbol{b} \in R^{n}\,. \tag{1}$$

In order to reduce the size of the linear system (1), we arrange a reduced linear system with a Schur complement by using an independent set. The Schur complement is composed by reordering the rows and columns of matrix A. We aim to apply an approximate inverse preconditioner to a reduced linear system with the Schur complement to improve the convergence rate. We are interested in the Newton scheme for computing the approximate inverse preconditioner since it is simple to implement. However, the matrix-matrix product is more difficult to compute as the preconditioner is less sparse. In this paper, we implement the Newton scheme without the matrix-matrix product. This implementation computes the preconditioner implicitly by using the matrix-vector product. The numerical results suggest that computing the preconditioner implicitly is more effective than computing it explicitly.

This paper is organized as follows. Section 2 provides an overview of composing the Schur complement. In Section 3, we consider the Newton scheme for the approximate inverse preconditioner and implement the preconditioner implicitly. The numerical results are reported in Section 4, and the concluding remarks are presented in Section 5.

2 Composing the Schur complement

2.1 Independent set

In this subsection, we introduce the independent set G. The indices s_i and s_j are independent of each other if the entries $a_{s_i s_j}, a_{s_j s_i} \in A$ $(s_i \neq s_j)$ are zero. All the indices belonging to G must be independent of each other. Therefore, define

$$G := \{s_1, s_2, \ldots, s_k \mid a_{s_i s_j} = 0,\ a_{s_j s_i} = 0,\ \text{for all } i, j = 1, 2, \ldots, k\}\,,$$

where k is the number of entries of G. The independent set G is constructed by using the Greedy algorithm. For more details on the implementation of the Greedy algorithm, see [7].

2.2 Reordering the rows and columns

Let $G \in \{s_1, s_2, \ldots, s_k\}$ be the independent set selected from the row and column indices of matrix A. The components of the linear system (1) are rearranged into the forms

$$PAP^T = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}, \quad P\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}, \quad P\boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix}, \tag{2}$$

where P is a suitable permutation matrix. The row and column indices of sub-matrix A_1 belong to G. Since A_1 is usually a diagonal matrix, it is not difficult to obtain A_1^{-1}. From the reordering forms (2), the following reduced linear system of size $\tilde{n} = n - k$ is generated:

$$C\boldsymbol{x}_2 = \tilde{\boldsymbol{b}}, \tag{3}$$

where

$$C = A_4 - A_3 A_1^{-1} A_2\,, \quad \tilde{\boldsymbol{b}} = \boldsymbol{b}_2 - A_3 A_1^{-1} \boldsymbol{b}_1\,.$$

The coefficient matrix C is the so-called Schur complement of sub-matrix A_1. The nonzero entries of matrix C need not be stored since the iterative solver of the linear system requires $C\boldsymbol{v}$. Therefore, only four products $A_1^{-1}\boldsymbol{v}_1$, $A_2\boldsymbol{v}_2$, $A_3\boldsymbol{v}_3$ and $A_4\boldsymbol{v}_4$ have to be computed.

3 Implementation of the preconditioner

In this section, we use the Newton scheme to develop the approximate inverse preconditioner.

3.1 Initial guess in Newton iteration

Schulz [1] proposed the Newton formula

$$N_{l+1} = (2I - N_l C)N_l\,, \tag{4}$$

where N_l is the preconditioner of matrix C in the lth Newton iteration. The convergence of the Newton scheme depends on the initial guess N_0. The preconditioner N_l converges to matrix C^{-1} if the spectral radius $\rho(I - N_0 C)$ is less than 1.0 [8]. Ben–Israel et al. [3] proved that this preconditioner converges to C^{-1} if the Newton iteration begins with

$$N_0 = \alpha C^T\,, \tag{5}$$

and the condition

$$0 < \alpha < \frac{2}{\rho(CC^T)} \tag{6}$$

is satisfied (Householder [2, p.95]). Here, $\rho(.)$ denotes the spectral radius of the matrix. Pan et al. [6] selected $\alpha = 1/(\|C\|_1\|C\|_\infty)$. This makes

$$N_0 = \frac{C^T}{\|C\|_1\|C\|_\infty}\,, \tag{7}$$

where $\|C\|_1 = \max_j \sum_{i=1}^{\tilde{n}} |c_{ij}|$ and $\|C\|_\infty = \max_i \sum_{j=1}^{\tilde{n}} |c_{ij}|$. However, the problem with this selection is that all the nonzero entries of matrix C are required for computing the matrix norms $\|C\|_1$ and $\|C\|_\infty$. All the nonzero entries of C must be computed with the matrix-matrix product. Therefore, it is very expensive to compute the initial guess N_0 with equation (7).

On the other hand, we select the diagonal matrix

$$N_0 = \operatorname{diag}\{1/c_{11}, 1/c_{22}, \ldots, 1/c_{\tilde{n}\tilde{n}}\} \tag{8}$$

as the initial guess, where c_{ii} is the ith diagonal entry of matrix C. The advantages of this choice are that only the diagonal entries of C are needed and the number of nonzero entries is limited to $\tilde{n}$. The numerical results reported in Section 4 show that N_l performs well as the preconditioner even if the equation (8) is selected as the initial guess N_0. In general, the Newton formula (4) does not always converge to C^{-1} even if the equation (8) is selected. However, we focus on reducing the computation cost of N_0. Moreover, in Section 4, we show that the selection (7) as the initial guess N_0 is more expensive than selecting (8).

3.2 Newton scheme without the matrix-matrix product

The difficulty in using the Newton formula (4) is that the matrix-matrix product is often very expensive to compute. In order to overcome this drawback, we implement the Newton scheme without the matrix-matrix product. The main goal of this implementation is to avoid matrix-matrix products in the Newton scheme by using the matrix-vector product. The iterative solver does not need the preconditioner N_l in explicit form; Instead, it requires $N_l\boldsymbol{v}$, where $\boldsymbol{v}$ is an arbitrary vector. Therefore, we compute N_l implicitly by using $N_l\boldsymbol{v}$. By substituting the Newton formula (4) in the equation $\boldsymbol{w}_l = N_l\boldsymbol{v}$, we describe vectors $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ and $\boldsymbol{w}_3$:

$$\begin{aligned}
\boldsymbol{w}_0 &= N_0\boldsymbol{v}\,, && (9)\\
\boldsymbol{w}_1 &= (2I - N_0C)N_0\boldsymbol{v} = 2\boldsymbol{w}_0 - N_0C\boldsymbol{w}_0\,, && (10)\\
\boldsymbol{w}_2 &= (2I - N_1C)N_1\boldsymbol{v}\\
&= \{2I - (2I - N_0C)N_0C\}\boldsymbol{w}_1\\
&= 2\boldsymbol{w}_1 - (N_0C)\{2\boldsymbol{w}_1 - (N_0C)\boldsymbol{w}_1\}\,, && (11)\\
\boldsymbol{w}_3 &= (2I - N_2C)N_2\boldsymbol{v}\\
&= \{2I - (2I - N_1C)N_1C\}\boldsymbol{w}_2\\
&= [2I - \{2I - (2I - N_0C)N_0C\}(2I - N_0C)N_0C]\boldsymbol{w}_2\\
&= 2\boldsymbol{w}_2 - (N_0C)[4\boldsymbol{w}_2 - (N_0C)\times\\
&\qquad \{6\boldsymbol{w}_2 - (N_0C)\{4\boldsymbol{w}_2 - (N_0C)\boldsymbol{w}_2\}\}]\,. && (12)
\end{aligned}$$

When $l \geq 4$, it is possible to compute the vector $\boldsymbol{w}_l$ in the same manner as when $l \leq 3$. $\boldsymbol{w}_l$ requires only $N_0\boldsymbol{v}_1$ and $C\boldsymbol{v}_2$, and no matrix-matrix product is needed. In general, the number of matrix-vector products required by $\boldsymbol{w}_l$ is 2^l. Therefore, l should be relatively small in order to reduce the computation cost of the matrix-vector product. Compared with other preconditioners, fill-in must be considered in the incomplete LU preconditioner, and the drop-off of nonzero entries is needed in the approximate inverse computed from the MR algorithm [8]. On the other hand, regarding the implicit implementation considered in this paper, the sparsity pattern of the preconditioner N_l need not be considered since N_l is not computed explicitly. This fact is also one of the advantages of computing N_l implicitly.

4 Numerical experiments

The numerical experiments are executed on a PC with Pentium4 2.66 GHz processor and 512 MB main memory. In all examples, the Schur complement is generated from matrix A to solve the reduced linear system (3).

4.1 First example

We discuss the boundary value problem of nonlinear partial differential equations in the region $\Omega = [0,1]^3$ (see Schönauer [5]):

$$\left\{ \begin{array}{c} u_{xx} + u_{yy} + u_{zz} + D(uu_x + vu_y + wu_z) + u = f_1 \quad \text{on } \Omega\,, \\ v_{xx} + v_{yy} + v_{zz} + D(uv_x + vv_y + wv_z) + v = f_2 \quad \text{on } \Omega\,, \\ w_{xx} + w_{yy} + w_{zz} + D(uw_x + vw_y + ww_z) + w = f_3 \quad \text{on } \Omega\,. \end{array} \right. \tag{13}$$

The right hand side functions f_1, f_2 and f_3, and the boundary condition are chosen so that the exact solutions are

$$\begin{aligned} u &= \sin(\pi x)\cos(\pi y)\cos(\pi z)\,, \\ v &= \cos(\pi x)\sin(\pi y)\cos(\pi z)\,, \\ w &= \cos(\pi x)\cos(\pi y)\sin(\pi z)\,. \end{aligned}$$

The problem (13) is discretized by the seven points central difference scheme using a square $40 \times 40 \times 40$ grid, where the mesh width h is $1/41$. In the equation (13), note that D is defined as the Reynolds number. In Table 1, D is varied so that the product Dh is equal to 2^{-6}, 2^{-5}, 2^{-4} and 2^{-3}. From the discretization, a nonlinear system of size 192,000 is generated. The discretized nonlinear system is usually solved by the following Newton scheme:

$$\boldsymbol{q}_{l+1} = \boldsymbol{q}_l + J(\boldsymbol{q}_l)^{-1}\psi(\boldsymbol{q}_l)\,, \tag{14}$$

where $\boldsymbol{q}_l$, $\psi(\boldsymbol{q}_l)$ and $J(\boldsymbol{q}_l)$ are the lth approximate solution, residual vector and Jacobi matrix of $\psi(\boldsymbol{q}_l)$, respectively. In the Newton formula (14), the linear system

$$J(\boldsymbol{q}_l)\boldsymbol{x} = \psi(\boldsymbol{q}_l) \tag{15}$$

must be solved once per iteration step. Although the discretized nonlinear system can also be solved by the inexact Newton scheme [9], it is not discussed in this paper. The linear system (15) is transformed into a reduced linear system with a Schur complement of size 128,000. In this example, when it is the first step of the Newton formula (14), the reduced linear system is solved by GMRES(m) algorithm [4]. The initial guess $\boldsymbol{q}_0$ is produced using the linear polynomial connecting two grid points: $(0, jh, kh)$ and $(1, jh, kh)$. We also apply the preconditioner N_l to the reduced linear system using the implicit implementation, as mentioned in Subsection 3.2. The iteration of the GMRES(m) algorithm is terminated as soon as the stopping criterion on the residual norm,

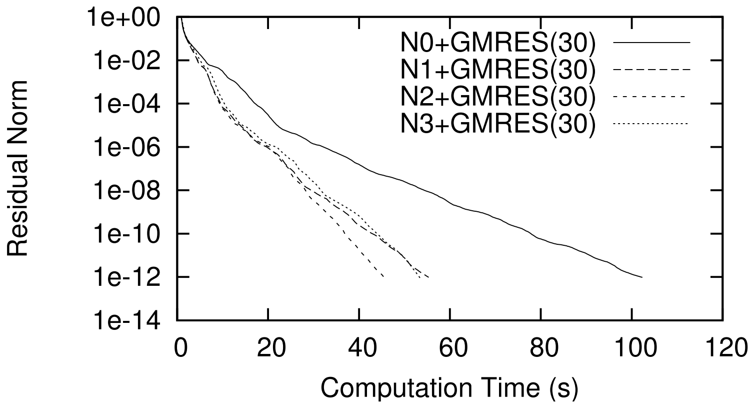
$$\|\boldsymbol{r}_i\|_2/\|\boldsymbol{r}_0\|_2 \leq 1.0 \times 10^{-12}\,, \tag{16}$$

is satisfied, where $\boldsymbol{r}_i$ is the ith residual vector of GMRES(m) algorithm. The zero vector is employed as the initial approximate solution of the reduced linear system. In the first experiment, the computation time of the initial guess N_0 is measured with respect to the cases of using the equations (7) and (8) when $Dh = 2^{-6}$. Even if the equation (7) is selected as the initial guess N_0, $N_l\boldsymbol{v}$ is not required explicitly for matrix C^T. Therefore, only the matrix norms $\|C\|_1$ and $\|C\|_\infty$ need to be computed. However, all the nonzero entries of matrix C are required for obtaining $\|C\|_1$ and $\|C\|_\infty$, as mentioned in Subsection 3.1. On the other hand, if the equation (8) is selected, only the diagonal entries of C are required.

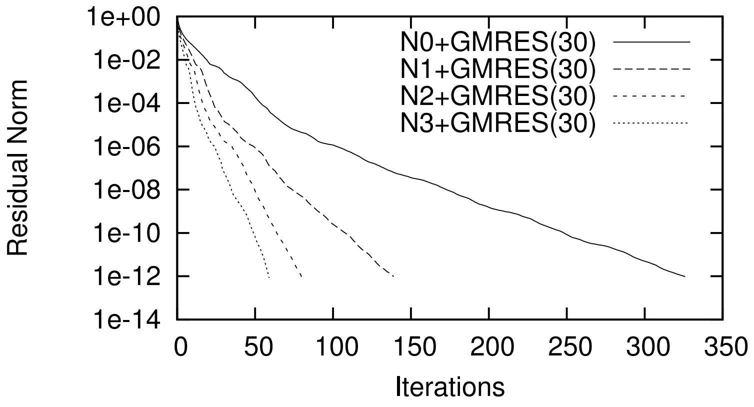
Whereas the computation time of the matrix norms $\|C\|_1$ and $\|C\|_\infty$ is 5150.0 s, that of the diagonal entries of C is 1.9 s. Therefore, the Newton formula (4) begins with the equation (8). Table 1 presents the computation time and iterations of the GMRES(m) algorithm that are required for satisfying the criterion (16). 'N_l + GMRES(m)' and 'GMRES(m)' denote the GMRES(m) algorithm that is preconditioned using N_l computed implicitly and the non-preconditioned GMRES(m) algorithm, respectively. The parameter Dh is varied from 2^{-6} to 2^{-3}. In any restart of the GMRES(m) algorithm,

TABLE 1: Example 4.1: The performance of the GMRES(m) algorithm (time: Computation time (s), iter: Iterations)

Algorithm	Dh							
	2^{-6}		2^{-5}		2^{-4}		2^{-3}	
	time	iter	time	iter	time	iter	time	iter
GMRES(20)	116.0	415	122.0	437	127.0	453	132.0	474
N_0+GMRES(20)	117.0	414	121.0	432	123.0	440	134.0	474
N_1+GMRES(20)	60.0	163	56.0	152	54.0	147	62.0	170
N_2+GMRES(20)	54.0	101	55.0	105	55.0	104	57.0	108
N_3+GMRES(20)	54.0	61	55.0	64	60.0	69	63.0	73
GMRES(30)	107.0	339	101.0	330	112.0	355	125.0	395
N_0+GMRES(30)	102.0	326	104.0	330	112.0	355	122.0	389
N_1+GMRES(30)	55.0	139	55.0	139	55.0	139	54.0	137
N_2+GMRES(30)	45.0	80	49.0	87	54.0	96	55.0	100
N_3+GMRES(30)	54.0	59	54.0	60	56.0	62	58.0	64
GMRES(40)	115.0	329	117.0	335	105.0	302	125.0	354
N_0+GMRES(40)	107.0	308	100.0	287	104.0	300	120.0	347
N_1+GMRES(40)	53.0	122	54.0	124	59.0	138	62.0	145
N_2+GMRES(40)	50.0	85	52.0	87	53.0	90	55.0	93
N_3+GMRES(40)	51.0	56	54.0	58	57.0	62	60.0	65



(a)



(b)

Figure 1: Example 4.1: The convergence behavior of the preconditioned GMRES(30) algorithm, ($Dh = 2^{-6}$): (a) Residual norm versus Time (s); (b) Residual norm versus Iterations.

the N_0+GMRES(m) and GMRES(m) algorithms do not converge within 100.0 s. On the other hand, the N_1+GMRES(m), N_2+GMRES(m) and N_3+GMRES(m) algorithms converge in 45.0 to 77.0 s. Figure 1 illustrates the behavior of the residual norm of the preconditioned GMRES(30) algorithm when $Dh = 2^{-6}$. While the iterations of the N_2 + GMRES(30) algorithm cost more than those of the N_3 + GMRES(30) algorithm, the computation time of the former is less expensive than that of the latter. Also the convergence behaviors of the N_1 + GMRES(30) and the N_3 + GMRES(30) algorithms are almost the same. Moreover, the computation cost of N_l in explicit form is measured in order to compare it with $N_l\boldsymbol{v}$ when $l = 1, 2, 3$. The computation time of N_1 in explicit form is 5383.0 s. Obviously, the explicit computation of N_1 is more expensive than the computation of the GMRES(m) algorithm preconditioned implicitly. It is impossible to compute N_2 and N_3 explicitly due to insufficient strategies. Therefore, the explicit computation of N_l is not effective. The time of the convergence of all the GMRES(m) algorithms preconditioned with the equation (8) in Table 1 is less than the time required for obtaining N_0 from the equation (7), because the computation time of $\|C\|_1$ and $\|C\|_\infty$ is 5150.0 s. The computation of N_0 with the equation (8) is included in the computation time in Table 1. Therefore, it is more effective to select the equation (8) instead of the equation (7) in Example 4.1. From these reasons, we do not discuss the convergence of the GMRES(m) algorithm that is preconditioned using the equation (7).

4.2 Second example

In this example, we discuss three types of linear systems with coefficient matrices in Florida Sparse Matrix Collection [10]. Their coefficient matrices are termed 'EPB3', 'LUNG2' and 'TORSO2'. The size and nonzero entries of these three matrices are listed in Table 2. For more details on these matrices, see [10]. In all linear systems, the right hand side is determined so that all the entries of the exact solution are 1.0. Similar to Example 4.1, a reduced linear system is produced and solved by the GMRES(m) algorithm, where the size

Table 2: Example 4.2: The characteristics of the coefficient matrices

Matrix name	Size	Nonzero entries	Size of Schur complement
EPB3	84617	463625	63318
LUNG2	109460	492564	82104
TORSO2	115967	1033473	86658

of the Schur complement is given in Table 2. In this example, the computation time of the initial guess N_0 is measured. Regarding 'EPB3', 'LUNG2' and 'TORSO2', the computation times of the diagonal entries of matrix C are 0.26, 0.31 and 0.43 s, respectively. On the other hand, the computation times of the matrix norms $\|C\|_1$ and $\|C\|_\infty$ are 1089.0, 1729.0 and 3097.0 s, respectively. Therefore, we select the equation (8) as the initial guess N_0. Table 3 shows the computation time and the iterations of the GMRES(m) algorithm. The preconditioner N_l is computed implicitly using the matrix-vector product. In all linear systems, the preconditioned GMRES(m) algorithm works better than the non-preconditioned GMRES(m) algorithm. Particularly in the case of 'EPB3', the preconditioner N_l makes the convergence faster as l is increased. Moreover, the computation time of the preconditioner N_l in explicit form is measured. With regard to 'EPB', 'TORSO2' and 'LUNG2', the computation times of N_1 are 936.0, 1504.0 and 2659.0 s, respectively. In all these cases, preconditioners N_2 and N_3 cannot be computed since the strategies are inadequate. Therefore, computing N_l explicitly is more expensive than computing it implicitly using the matrix-vector product. In Table 3, some of GMRES(m) algorithms that are preconditioned with N_l using the equation (8) converge for 'EPB'. Regarding the case of 'EPB', we compute N_1, N_2 and N_3 implicitly based on the equation (7) and apply it to the GMRES(m) algorithm for comparison with the equation (8). However, in any restart of the GMRES(m) algorithms that are preconditioned with N_1, N_2 and N_3 using the equation (7), the residual norm could not converge within 15 min. Therefore, the equation (8) is a better selection than the equation (7). It is also

Table 3: Example 4.2: The performance of the GMRES(m) algorithm (time: Computation time (s), iter: Iterations)

Algorithm	Matrix name					
	EPB3		LUNG2		TORSO2	
	time	iter	time	iter	time	iter
GMRES(20)	–	–	–	–	8.0	66
N_0+GMRES(20)	–	–	120.0	1305	5.0	35
N_1+GMRES(20)	–	–	57.0	536	3.0	20
N_2+GMRES(20)	–	–	43.0	301	3.0	13
N_3+GMRES(20)	185.0	1063	42.0	190	3.0	9
GMRES(30)	–	–	–	–	9.0	64
N_0+GMRES(30)	–	–	121.0	1161	5.0	35
N_1+GMRES(30)	–	–	57.0	457	3.0	20
N_2+GMRES(30)	445.0	3514	43.0	263	3.0	13
N_3+GMRES(30)	226.0	1203	41.0	168	3.0	9
GMRES(40)	–	–	–	–	9.0	65
N_0+GMRES(40)	–	–	132.0	1070	5.0	34
N_1+GMRES(40)	674.0	6084	60.0	424	3.0	20
N_2+GMRES(40)	278.0	1962	44.0	244	3.0	13
N_3+GMRES(40)	195.0	963	41.0	159	3.0	9

(–): the algorithm could not converge within 15 min.

suggested that the good approximate preconditioner is not obtained within three iterations of the Newton scheme even if the equation (7) is selected as the initial guess.

5 Concluding remarks

We implemented the preconditioner N_l of the reduced linear system implicitly by using the matrix-vector product. By implementing it implicitly, no matrix-matrix product is needed. Moreover, we also selected the diagonal matrix as the initial guess of the Newton iteration in order to make it easy to compute. The numerical results show that the computation of N_1 costs more than that of $N_1\boldsymbol{v}$. The preconditioner N_l in explicit form cannot be obtained if $l \geq 2$. Therefore, the implicit implementation considered in this paper is a better alternative to computing the preconditioner N_l explicitly.

References

[1] G. Shulz. Iterative Berechnung der Reziproken Matrix, *Z. Angew. Math. Mech.*, 13: 57–59, 1933. C397

[2] A. S. Householder. *The Theory of Matrices in Numerical Analysis*, Dover, New York, 1964. C398

[3] A. Ben-Israel and D. Cohen. On Iterative Computation of Generalized Inverses and Associated Projections, *SIAM J. Numer. Anal.*, 3: 410–419, 1966. C397

[4] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, *SIAM J. Sci. Stat. Comput.*, 7: 856–869, 1986. C401

[5] W. Schönauer. *Scientific Computing on Vector Computers*, North Holland, 1987. C400

[6] V. Pan and R. Schreiber. An Improved Newton Iteration for the Generalized Inverse of a Matrix with Applications, *SIAM J. Sci. Stat. Comp.*, 12(5): 1109–1130, 1991. C398

[7] Y. Saad. *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996. C396

[8] T. Huckel. Approximate Sparsity Patterns for the Inverse of a Matrix and Preconditioning, *Appl. Numer. Math.*, 30: 291–303, 1999. C397, C399

[9] X. C. Cai and D. E. Keyes. Nonlinearly Preconditioned Inexact Newton Algorithms, *SIAM J. Sci. Comput.*, 41(1): 183–200, 2002. C401

[10] University of Florida Sparse Matrix Collection. [Online] http://www.cise.ufl.edu/research/sparse/matrices. C404