Electronic Transactions on Numerical Analysis.
Volume 23, pp. 158-179, 2006.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

ON FAST FACTORIZATION PIVOTING METHODS FOR SPARSE SYMMETRIC INDEFINITE SYSTEMS*

OLAF SCHENK[†] AND KLAUS GÄRTNER[‡]

Abstract. This paper discusses new pivoting factorization methods for solving sparse symmetric indefinite systems. As opposed to many existing pivoting methods, our Supernode–Bunch–Kaufman (SBK) pivoting method dynamically selects 1×1 and 2×2 pivots and may be supplemented by pivot perturbation techniques. We demonstrate the effectiveness and the numerical accuracy of this algorithm and also show that a high performance implementation is feasible. We will also show that symmetric maximum-weighted matching strategies add an additional level of reliability to SBK. These techniques can be seen as a complement to the alternative idea of using more complete pivoting techniques during the numerical factorization. Numerical experiments validate these conclusions.

Key words. direct solver, pivoting, sparse matrices, graph algorithms, symmetric indefinite matrix, interior point optimization

AMS subject classifications. 65F05, 65F50, 05C85

1. Introduction. We consider the direct solution of symmetric indefinite linear system $A\boldsymbol{x} = \boldsymbol{b}$, with

$$A = P_{Fill} P_{\mathcal{S}} (\, LDL^T \; + \; E) P_{\mathcal{S}}^T \; P_{Fill}^T, \tag{1.1}$$

where D is a diagonal matrix with 1×1 and 2×2 pivot blocks, L is a sparse lower triangular matrix, and E is a symmetric indefinite diagonal matrix that reflects small half-machine precision perturbations, which might be necessary to tackle the problem of tiny pivots. $P_{\mathcal{S}}$ is a reordering that is based on a symmetric weighted matching $\mathcal{S}$ of the matrix A, and tries to move the largest off-diagonal elements directly alongside the diagonal in order to form good initial 1×1 or 2×2 diagonal block pivots. P_{Fill} is a fill reducing reordering which honors the structure of $P_{\mathcal{S}}$.

We will present three new variations of a direct factorization scheme to tackle the issue of indefiniteness in sparse symmetric linear systems. These methods restrict the pivoting search, to stay as long as possible within predefined data structures for efficient Level-3 BLAS factorization and parallelization. On the other hand, the imposed pivoting restrictions can be reduced in several steps by taking the matching permutation $P_{\mathcal{S}}$ into account. The first algorithm uses Supernode–Bunch–Kaufman (SBK) pivoting and dynamically selects 1×1 and 2×2 pivots. It is supplemented by pivot perturbation techniques. It uses no more storage than a sparse Cholesky factorization of a positive definite matrix with the same sparsity structure due to restricting the pivoting to interchanges within the diagonal block associated to a single supernode. The coefficient matrix is perturbed whenever numerically acceptable 1×1 and 2×2 pivots cannot be found within the diagonal block. One or two steps of iterative refinement may be required to correct the effect of the perturbations. We will demonstrate that this restricting notion of pivoting with iterative refinement is effective for highly indefinite symmetric systems. Furthermore the accuracy of this method is for a large set of matrices from different applications areas as accurate as a direct factorization method that uses complete sparse pivoting techniques. In addition, we will discuss two preprocessing algorithms to identify large entries in the coefficient matrix A that, if permuted close to the diagonal, permit

*Received August 27, 2004. Accepted for publication July 20, 2006. Recommended by J. Gilbert.
[†]Department of Computer Science, University Basel, Klingelbergstrasse 50, CH-4056 Basel, Switzerland (`olaf.schenk@unibas.ch`).
[‡]Weierstrass Institute for Applied Analysis and Stochastics, Mohrenstr. 39, D-10117 Berlin, Germany (`gaertner@wias-berlin.de`).

the factorization process to identify more acceptable pivots and proceed with fewer pivot perturbations. The methods are based on maximum weighted matchings and improve the quality of the factor in a complementary way to the alternative idea of using more complete pivoting techniques.

Sparse symmetric indefinite linear systems arise in numerous areas, e.g. incompressible flow problems, augmented systems due to linear and nonlinear optimization, electromagnetic scattering, and eigenvalue problems. Here, the saddle point problems are often especially hard to solve [16]. In general, some kind of pivoting techniques must be applied to solve these systems accurately and the challenge is to achieve both numerical stability and sparsity of the factors.

1.1. Previous Work. Extensive investigation on pivoting techniques for symmetric indefinite direct solvers have been done by numerous researchers. There are three well known algorithms for solving dense symmetric indefinite linear systems: the Bunch-Kaufman algorithm [6], the Bunch-Parlett algorithm [7], and Aasen's method [1]. Furthermore, the Duff-Reid algorithm [15] based on threshold pivoting techniques is an efficient method that is frequently used for sparse symmetric indefinite systems. The primary advantage of this threshold method is that it allows to bound the element growth, although this may generate a higher fill-in during the elimination process. The most recent algorithmic paper on pivot search and pivot-admissibility is [3] and the authors proposed a bounded Bunch-Kaufman pivot selection for bounding the numerical values of the factor. In this paper, we will use an alternative pivoting strategy for sparse symmetric indefinite systems. Instead of using the Duff-Reid algorithm, we propose to restrict the pivoting to interchanges within the dense diagonal block corresponding to a single supernode. This allows us to use one of the three dense pivoting techniques. Our pivoting technique always applies the dense Bunch–Kaufman pivoting selection since it is also part of LAPACK. However, from the matching point of view, a restricted version of the Bunch-Parlett algorithm might be an interesting candidate for future research. For further details on symmetric pivoting techniques, stability, and accuracy issues see [10, 22].

In [24] the pivoting approach of SUPERLU is described for the factorization of sparse nonsymmetric systems. This method needs an additional preprocessing step based on weighted matchings [12, 25] which reduces the need for partial pivoting thereby, speeding up the solution process. More recently, in [13, 26] scalings and ordering strategies based on symmetric weighted matchings have been investigated and improvements are reported for symmetric indefinite systems.

1.2. Contributions of the paper. We conduct an extensive study on the use of Supernode–Bunch–Kaufman pivoting using 1×1 and 2×2 pivots within static data structures and supplemented by perturbation techniques for the factorization of sparse symmetric indefinite systems. To the best of our knowledge, this pivoting techniques was not used until now. Because of its pivoting restriction, we consider two strategies based on maximum weighted matchings for the computation of a permutation that identifies large entries in the coefficient matrix. These are placed close to the diagonal and permit the factorization to choose more acceptable pivots. The use of weighted matchings in combination with our pivoting method is new — other techniques in combination with other pivoting methods have recently been proposed in [11] and explored in [13, 14, 26].

While we do not claim that this approach to factor symmetric indefinite systems will always work, we do hope that the results in this paper will contribute to further acceleration for direct solvers for sparse indefinite systems. We also hope that the combination of our pivoting methods and symmetric weighted matchings will find widespread use in the area of hard to solve symmetric indefinite linear systems. The implementation of the method

ETNA
Kent State University
etna@mcs.kent.edu

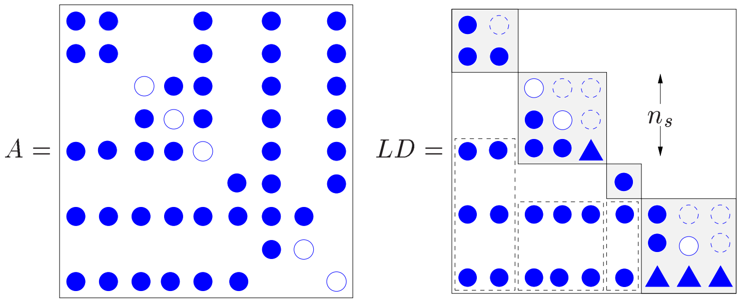


FIG. 2.1. *Illustration of a supernodal decomposition of the factor for a symmetric indefinite matrix A.The triangles indicate fill due to external updates.*

is reliable and performs well. On a 2.4 GHz Intel 32-bit processor, it factors an indefinite symmetric Karush-Kuhn-Tucker (KKT) optimization example with about 2 million rows and columns in less than one minute, producing a factor with about 1.4×10^8 nonzeros.

However, for all methods based on a pivoting restriction and symmetric matching techniques, it seems unlikely to prove backward stability without additional assumptions. This is clearly a disadvantage of the approach, but this may be overcome at least for special application classes in the future.

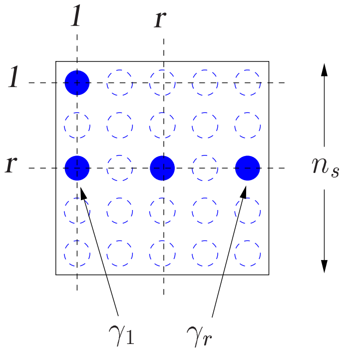
The paper is organized as follows. Section 2 provides some background on symmetric indefinite factorizations and presents our formulation of the factorization using 1×1 and 2×2 pivots. We follow this with a discussion of additional preprocessing methods to improve the numerical accuracy based on symmetric weighted matchings in Sections 3 and 4. Section 5 presents our numerical results for sparse matrices from a wide range of applications. Finally, Section 6 presents our conclusions.

2. The SBK algorithm. Virtually all modern sparse factorization codes rely heavily on a supernodal decomposition of the factor L to efficiently utilize the memory hierarchies in the hardware as shown in Figure 2.1. The filled circles correspond to elements that are nonzero in the coefficient matrix A, non-filled circles indicate elements that are zero and explicitly stored, the dotted circles indicate zeros in the factor that are stored in the supernode structure but not accessed during numerical factorization, and triangles represent fill elements. The filled boxes indicate diagonal supernodes of different sizes n_s where Supernode–Bunch–Kaufman pivoting is performed and the dotted boxes indicate the remaining part of the supernodes. The factor is decomposed into dense diagonal blocks of size n_s and into the corresponding subdiagonal blocks such that the rows in the subdiagonals are either completely zero or dense. There are two main approaches in building these supernodes. In the first approach, consecutive rows and columns with the identical structure in the factor L are treated as one supernode. These supernodes are so crucial to high performance in sparse matrix factorization that the criterion for the inclusion of rows and columns in the same supernode can be relaxed [2] to increase the size of the supernodes. This is the second approach called supernode amalgamation. In this approach, consecutive rows and columns with nearly the same but not identical structures are included in the same supernode, and artificial nonzero entries with a numerical value of zero are added to maintain identical row and column structures for all members of a supernode. The rationale is that the slight increase in the number of nonzeros and floating-point operations involved in the factorization can be compensated by a higher factorization speed. In this paper, we will always use a supernodal decomposition with the extension that we explicitly add zeros in the upper triangular part of L in order to form a rectangular supernode representation that can be factored using optimized LAPACK routines.

ETNA
Kent State University
etna@mcs.kent.edu

ALGORITHM 2.1. *Supernode Bunch-Kaufman pivot selection with half-machine precision perturbation.*

1. $\gamma_1 := |a_{r1}| = \max_{k=2,\ldots,n_s} |a_{k1}|$
   with diagonal block of size n_s
2. $\gamma_r \geq \gamma_1$ *is the magnitude of the largest off-diagonal in the* $r-$*row of the block*
3. ***if*** $\max(|a_{11}|, \gamma_1) \leq \epsilon \cdot ||A||_1$:
4. *use pivot perturbation:*
   $\tilde{a}_{11} = sign(a_{11}) \cdot \epsilon \cdot ||A||_1$
5. *use perturbed* $\tilde{a}_{11}$ *as a* 1×1 *pivot.*
6. ***else if*** $|a_{11}| \geq \alpha\gamma_1$:
7. *use* a_{11} *as a* 1×1 *pivot.*
8. ***else if*** $|a_{11}| \cdot \gamma_r \geq \alpha\gamma_1^2$:
9. *use* $|a_{11}|$ *as a* 1×1 *pivot.*
10. ***else if*** $|a_{rr}| \geq \alpha\gamma_r$:
11. *use* $|a_{rr}|$ *as a* 1×1 *pivot.*
12. ***else***
13. *use* $\begin{pmatrix} a_{11} & a_{r1} \\ a_{r1} & a_{rr} \end{pmatrix}$ *as a* 2×2 *pivot*

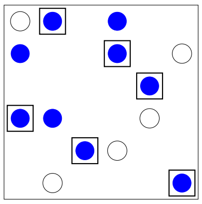


We will use a Level-3 BLAS left-looking factorization as described in [27, 28]. An interchange among the rows and columns of a supernode of diagonal size n_s, referred to as Supernode–Bunch–Kaufman pivoting, has no effect on the overall fill-in and this is the mechanism for finding a suitable pivot in our SBK method. However, there is no guarantee that the numerical factorization algorithm would always succeed in finding a suitable pivot within a diagonal block related to a supernode. When the algorithm reaches a point where it cannot factor the supernode based on the previously described 1×1 and 2×2 pivoting, it uses a pivot perturbation strategy similar to [24]. The magnitude of the potential pivot is tested against a constant threshold of $\epsilon \cdot ||A||_1$, where ϵ is a half-machine precision perturbation. Therefore, any tiny pivots encountered during elimination are set to $\text{sign}(a_{ii}) \cdot \epsilon \cdot ||A||_1$ — this trades off some numerical stability for the ability to keep pivots from getting too small. The result of this pivoting approach is that the factorization is, in general, not accurate and iterative refinement will be needed.

Algorithm 2.1 describes the usual 1×1 and 2×2 Bunch-Kaufman pivoting strategy within the diagonal block corresponding to a supernode of size n_S. The pivoting strategy is supplemented with half-machine precision perturbation techniques. The Bunch–Kaufman pivoting method computes two scalars γ_1 and γ_r. The scalar γ_1 is the largest off-diagonal element, e.g. $|a_{r1}|$, in the first column of the diagonal block corresponding to the supernode of size n_s. γ_r is the largest off-diagonal element in the corresponding row r. The scalar $\alpha = (\sqrt{17} + 1)/8$ is chosen to minimize the element growth. With this choice, the element growth in the diagonal block after k steps is bounded by the factor $(2.57)^{k-1}$. The algorithm than selects either 1×1 and 2×2 pivots for the factorization. If both $|a_{11}|$ and $|\gamma_1|$ are too small, e.g. smaller than $\epsilon \cdot ||A||_1$, we apply the pivot perturbation technique as described above.

3. Symmetric reorderings to improve the results of pivoting on restricted subsets. In this section we will discuss weighted graph matchings as an additional preprocessing step. Weighted matchings $\mathcal{M}$ have been introduced by Olschowka and Neumaier [25] to obtain

ETNA
Kent State University
etna@mcs.kent.edu

$A =$ 

$$P_{\mathcal{M}}^T = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

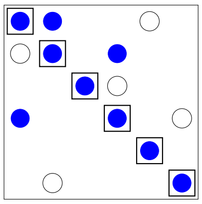
$P_{\mathcal{M}} A =$ 

FIG. 3.1. *Illustration of the row permutation. A small numerical value is indicated by a* $\circ$*-symbol and a large numerical value by an* $\bullet$*-symbol. The matched entries* $\mathcal{M}$ *are marked with squares and* $P_{\mathcal{M}} = (e_4; e_1; e_5; e_2; e_3; e_6)$.

a permutation that moves the large elements to the diagonal of a dense matrix to reduce the effort for pivoting. Duff and Koster [12] extended the approach to nonsymmetric sparse matrices to ensure a zero-free diagonal or to maximize the product of the absolute values of the diagonal entries. It is now an often used technique for solving nonsymmetric linear systems [5, 24, 27, 29].

3.1. Matching Algorithms. Let $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ be a general matrix. The nonzero elements of A define a graph with edges $\mathcal{E} = \{(i,j) : a_{ij} \neq 0\}$ of ordered pairs of row and column indices. A subset $\mathcal{M} \subset \mathcal{E}$ is called a matching or a transversal, if every row index i and every column index j appears at most once in $\mathcal{M}$. A matching $\mathcal{M}$ is called perfect if its cardinality is n. For a nonsingular matrix at least one perfect matching exists and can be found with well known algorithms. With a perfect matching $\mathcal{M}$, it is possible to define a permutation matrix $P_{\mathcal{M}} = (p_{ij})$ with:

$$p_{ij} = \begin{cases} 1 & (j,i) \in \mathcal{M} \\ 0 & \text{otherwise.} \end{cases}$$

As a consequence, the permutation matrix $P_{\mathcal{M}} A$ has nonzero elements on its diagonal. This method only takes the nonzero structure of the matrix into account. There are other approaches which maximize the diagonal values in some sense. One possibility is to look for a matrix $P_{\mathcal{M}}$, such that the product of the diagonal values of $P_{\mathcal{M}} A$ is maximal. In other words, a permutation σ has to be found, which maximizes:

$$\prod_{i=1}^{n} |a_{\sigma(i)i}|. \tag{3.1}$$

This maximization problem is solved indirectly. It can be reformulated by defining a matrix $C = (c_{ij})$ with

$$c_{ij} = \begin{cases} \log a_i - \log |a_{ij}| & a_{ij} \neq 0 \\ \infty & \text{otherwise,} \end{cases}$$

where $a_i = \max_j |a_{ij}|$, i.e. the maximum element in row i of matrix A. A permutation σ, which minimizes $\sum_{i=1}^{n} c_{\sigma(i)i}$ also maximizes the product (3.1).

The effect of nonsymmetric row permutations using a permutation associated with a matching $\mathcal{M}$ is shown in Figure 3.1. It is clearly visible that the matrix $P_{\mathcal{M}} A$ is now nonsymmetric, but has the largest nonzeros on the diagonal. For a more detailed description on matchings algorithms for large symmetric linear systems the interested reader should consult [18].

ETNA
Kent State University
etna@mcs.kent.edu

3.2. Symmetric 1×1 and 2×2 block weighted matchings. In the case of symmetric indefinite matrices, we are interested in symmetrically permuting PAP^T. The problem is that zero or small diagonal elements of A remain on the diagonal by using a symmetric permutation PAP^T. Alternatively, instead of permuting a large[1] off-diagonal element a_{ij} nonsymmetrically to the diagonal, one can try to devise a permutation $P_\mathcal{S}$ such that $P_\mathcal{S} A P_\mathcal{S}^T$ permutes this element close to the diagonal. As a result, if we form the corresponding 2×2 block to $\left[\begin{smallmatrix} a_{ii} & a_{ij} \\ a_{ij} & a_{jj} \end{smallmatrix}\right]$, we expect the off-diagonal entry a_{ij} to be large and thus the 2×2 block would form a suitable 2×2 pivot for the Supernode–Bunch–Kaufman factorization. An observation on how to build $P_\mathcal{S}$ from the information given by a weighted matching $\mathcal{M}$ was presented by Duff and Gilbert [11]. They noticed that the cycle structure of the permutation $P_\mathcal{M}$ associated with the nonsymmetric matching $\mathcal{M}$ can be exploited to derive such a permutation $P_\mathcal{S}$. For example, the permutation $P_\mathcal{M}$ from Figure 3.1 can be written in cycle representation as $P_\mathcal{C} = (e_1; e_2; e_4)(e_3; e_5)(e_6)$. This is shown in the upper graphics in Figure 3.2. The left graphic displays the cycles $(1\ 2\ 4)$, $(3\ 5)$ and (6). If we modify the original permutation $P_\mathcal{M} = (e_4; e_1; e_5; e_2; e_3; e_6)$ into this cycle permutation $P_\mathcal{C} = (e_1; e_2; e_4)(e_3; e_5)(e_6)$ and permute A symmetrically with $P_\mathcal{C} A P_\mathcal{C}^T$, it can be observed that the largest elements are permuted to diagonal blocks. These diagonal blocks are shown by filled boxes in the upper right matrix. Unfortunately, a long cycle would result into a large diagonal block and the fill-in of the factor for $P_\mathcal{C} A P_\mathcal{C}^T$ may be prohibitively large. Therefore, long cycles corresponding to P_M must be broken down into disjoint 2×2 and 1×1 cycles. These smaller cycles are used to define a symmetric permutation $P_\mathcal{S} = (c_1, \ldots, c_m)$, where m is the total number of 2×2 and 1×1 cycles.

The rule for choosing the 2×2 and 1×1 cycles from $P_\mathcal{C}$ to build $P_\mathcal{S}$ is straightforward. One has to distinguish between cycles of even and odd length. It is always possible to break down even cycles into cycles of length two. For each even cycle, there are two possibilities to break it down. We use a structural metric as described in [13] to decide which one to take. The same metric is also used for cycles of odd length, but the situation is slightly different. Cycles of length $2l+1$ can be broken down into l cycles of length two and one cycle of length one. There are $2l+1$ different possibilities to do this. The resulting 2×2 blocks will contain the matched elements of $\mathcal{M}$. However, there is no guarantee that the remaining diagonal element corresponding to the cycle of length one will be nonzero. Our implementation will randomly select one element as a 1×1 cycle from an odd cycle of length $2l + 1$. If this diagonal element a_{ii} is still zero during the numerical factorization, it will be perturbed using the strategies described in Section 2.

A selection of $P_\mathcal{S}$ from a weighted matching $P_\mathcal{M}$ is illustrated in Figure 3.2. The permutation associated with the weighted matching, which is sorted according to the cycles, consists of $P_\mathcal{C} = (e_1; e_2; e_4)(e_3; e_5)(e_6)$. We now split the full cycle of odd length three into two cycles $(1)(24)$ — resulting in $P_\mathcal{S} = (e_1)(e_2; e_4)(e_3; e_5)(e_6)$. If $P_\mathcal{S}$ is symmetrically applied to $A \leftarrow P_\mathcal{S} A P_\mathcal{S}^T$, we see that the large elements from the nonsymmetric weighted matching $\mathcal{M}$ will be permuted close to the diagonal and these elements will have more chances to form good initial 1×1 and 2×2 pivots for the subsequent SBK factorization.

Good fill-in reducing orderings P_{Fill} are equally important for symmetric indefinite systems. The following section introduces two strategies to combine these reorderings with the symmetric graph matching permutation $P_\mathcal{S}$. This will provide good initial pivots for the SBK factorization as well as a good fill-in reduction permutation.

4. Combination of orderings P_{Fill} for fill reduction with orderings $P_\mathcal{S}$ based on weighted matchings. In order to construct the factorization efficiently, care has to be taken

[1]Large in the sense of the weighted matching $\mathcal{M}$.

ETNA
Kent State University
etna@mcs.kent.edu

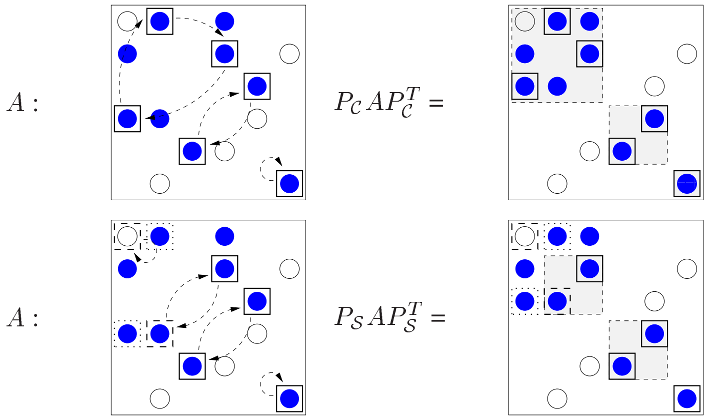


FIG. 3.2. *Illustration of a cycle permutation with* $P_{\mathcal{C}} = (e_1; e_2; e_4)(e_3; e_5)(e_6)$ *and* $P_{\mathcal{S}} = (e_1)(e_2; e_4)(e_3; e_5)(e_6)$. *The symmetric matching* $P_{\mathcal{S}}$ *has two additional elements (indicated by dashed boxes), while one element of the original matching fell out (dotted box). The two 2-cycles are permuted into* 2×2 *diagonal blocks to serve as initial* 2×2 *pivots.*

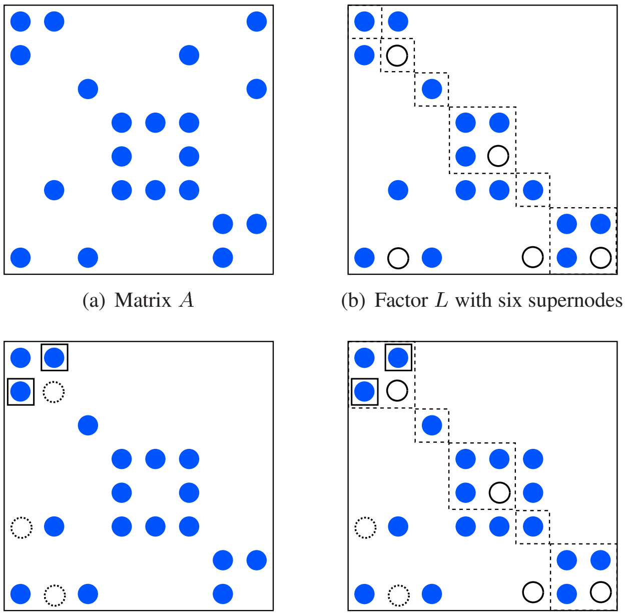
that not too much fill-in is introduced during the elimination process. We now examine two algorithms for the combination of a permutation $P_{\mathcal{S}}$ based on weighted matchings to improve the numerical quality of the coefficient matrix A with a fill-in reordering P_{Fill} based on a nested dissection from METIS [23]. The first method based on compressed subgraphs has also been used by Duff and Pralet in [13] in order to find good scalings and orderings for symmetric indefinite systems for multifrontal direct solvers.

4.1. Pivoting Variant 1: Fill-in reduction P_{Fill} based on compressed subgraphs. In order to combine the permutation $P_{\mathcal{S}}$ with a fill-in reducing permutation, we compress the graph of the reordered system $P_{\mathcal{S}}AP_{\mathcal{S}}^T$ and apply the fill-in reducing reordering to the compressed graph. In the compression step, the union of the structure of the two rows and columns corresponding to a 2×2 diagonal block is built, and used as the structure of a single, compressed row and column representing the original ones.

If $G_A = (V; E)$ is the undirected graph of A and a cycle consists of two vertices $(s, t) \in V$, then graph compression will be done on the 1×1 and 2×2 cycles, which have been found using a weighted matching $\mathcal{M}$ on the graph. The vertices (s, t) are replaced with a single supervertex $u = \{s, t\} \in V_c$ in the compressed graph $G_c = (V_c, E_c)$. An edge $e_c = (s, t) \in E_c$ between two supervertices $s = \{s_1, s_2\} \in V_c$ and $t = \{t_1, t_2\} \in V_c$ exists if at least one of the following edges exit in E : $(s_1, t_1), (s_1, t_2), (s_2, t_1)$ or (s_2, t_2). The fill-in reducing ordering is found by applying METIS on the compressed graph $G_c = (V_c, E_c)$. Expansion of that permutation to the original numbering yields P_{fill}. Hence all 2×2 cycles that correspond to a suitable 2×2 pivot block are reordered consecutively in the factor. During the expansion of the compressed graph in each 2x2 block the larger diagonal entry can be ordered first.

This strategy will be denoted as a SBK-COMP-1 factorization approach in the numerical experiments. However, due to a different row structure below a 2×2 pivot block, there is no guarantee that this factorization approach will always use the preselected 2×2 blocks.

4.2. Pivoting Variant 2: Use All Preselected 2×2 Pivots. The 1×1 and 2×2 pivoting search of the SBK method is applied within the block diagonal of a supernode. In the extreme case, the supernode exists of only one column and the SBK pivoting can degenerate to diagonal pivoting. Therefore any permutation that symmetrically permutes large-off diag-



(a) Matrix A

(b) Factor L with six supernodes

(c) Zero elements in A to increase supernode size

(d) Factor L with five supernodes

FIG. 4.1. *(a) and (b) Matrix and factor of supernodal Bunch-Kaufman pivoting with supernodes* $\{(1),(2),(3),(4,5),(6),(7,8)\}$. *(b) and (c) Matrix and factor of preselected* 2×2 *pivots with additional zero elements and supernodes* $\{(1,2),(3),(4,5),(6),(7,8)\}$.

onal entries close to the diagonal or that identifies suitable 2×2 pivots prior to the numerical factorization would further improve the accuracy. In order to enforce the use of these preselected pivot blocks during the LDL^T factorization, we merge the 2×2 column structure in L in such a way that these initial 2×2 pivot structure is maintained in the factor L.

This is illustrated in Figure 4.1. In the pivoting variant 1 we will restrict the pivoting search within diagonal-blocks of the supernodes as shown in Figure 4.1 (a) and (b). In the pivoting variant 2 we will identify 2×2 and enforce the use of these preselected pivot blocks by adding additional zero elements to the structure of A. As a result, the size of the supernode increases, e.g. we will merge column/row 1 and 2 into one supernodes of size 2, in which a 2×2 Bunch-Kaufman pivoting is performed instead of two 1×1 diagonal pivot elements.

5. Numerical Experiments.

We now present the results of the numerical experiments that demonstrate the robustness and the effectiveness of our pivoting approach.

5.1. Sparse Symmetric Indefinite Matrices.

The test problems used here are either of augmented symmetric indefinite type

$$A = \begin{pmatrix} A_{11} & B \\ B^T & 0 \end{pmatrix}$$

or of general symmetric indefinite type. The matrices A_{11} and B in the augmented system A are large and sparse, and no assumption is made for the upper left block A_{11}. Most of the test problems are taken from extensive surveys [16, 17] of direct solvers for 61 indefinite

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 5.1
General symmetric indefinite test matrices. n denotes the total number of unknowns, and nnz the nonzeros in the matrix.

no.	matrix	n	nnz	Application
1	bcsstk35	30,237	740,200	Stiffness matrix — automobile seat
2	bcsstk37	25,503	5,832,490	Stiffness matrix — rack ball
3	bcsstk39	46,772	1,068,033	Stiffness shuttle rocket booster
4	bmw3_2	227,362	5,757,996	Linear static analysis — car body
5	copter2	55,476	407,714	Helicopter rotor blade
6	crystk02	13,965	491,274	Stiffness matrix, crystal free vibration
7	crystk03	24,696	887,937	Stiffness matrix, crystal free vibration
8	dawson5	51,537	531,157	Aeroplane actuator system
9	DIXMAANL	60,000	179,999	Dixon-Maany optimization example
10	HELM2D03	392,257	1,567,096	Helmholtz problem
11	HELM3D01	32,226	230,335	Helmholtz problem
12	LINVERSE	11,999	53,988	Matrix inverse approximation
13	NCVXBQP1	50,000	199,984	Nonconvex QP Hessian
14	qa8fk	66,127	863,353	FE matrix from 3D acoustics
15	SPMSRTLS	29,995	129,971	Sparse matrix square root
16	vibrobox	12,328	177,578	Vibroacoustic problem

systems[2]. This set is rather challenging and several of the matrices could not be solved by some of the sparse direct solvers under the constraints imposed in the reports [16, 17]. The matrices comprise a variety of application areas and Table 5.1 and 5.2 give a rough classification of the matrices including the number of unknowns, the number of zero diagonal elements, the total number of nonzeros in the matrix, and the application area. In addition to these matrices, Table 5.2 also shows two augmented symmetric indefinite matrices `lnt09` and `mass06` from [21] and two large augmented sparse indefinite systems `cont5_2` and `cont5_3` from the interior point optimization package IPOPT [31]. The matrices have been especially included in order to demonstrate the robustness benefit due to symmetric weighted matchings as an additional preprocessing strategy for symmetric indefinite sparse factorization methods. Furthermore, the matrices `AUG2D`, `AUG2DC`, `AUG3D`, `DTOC`, and `K1_SAN` are structurally singular matrices. This is recognized by the matching algorithms, too, because a perfect matching is supposed to exist in the algorithms described. In order to deal with these systems, the matching was extend in such a way that the resulting matching describes a complete maximal matching.

5.2. Test Environment and Codes. All numerical tests were run on a double processor Pentium III 1.3 GHz system with 2 GB of memory, but only one processor was used for the testing. The system runs a recent Linux distribution. In order to provide realistic measurements, the timings were determined using the `gettimeofday` standard C library function, which provides a high accuracy wall clock time. To compensate for the variations, we provide the best result out of three runs for each measurement.

All algorithms were implemented in Fortran 77 and C and were integrated into the PARDISO 3.1 solver package[3], a suite of publicly available parallel sparse linear solvers. The

[2] The matrices can be downloaded at `ftp://ftp.numerical.rl.ac.uk/pub/matrices/symmetric` and from the University of Florida Sparse Matrix Collection [8].

[3] `http://www.computational.unibas.ch/cs/scicomp/software/pardiso`

ETNA
Kent State University
etna@mcs.kent.edu

code was compiled by *g77* and *gcc* with the -O3 optimization option and linked with the Automatically Tuned Linear Algebra Software ATLAS library[4] for the basic linear algebra subprograms optimized for Intel architectures. The weighted matching code MPS, which is part of the PARDISO package, was provided by S. Röllin from ETH Zurich, and it is based on [18]. In general, the performance and the results of the MPS code are comparable to the MC64 code from the HSL library [19].

In all numerical experiments in Section 5.5, we choose a random right hand side b. In Section 5.6 we used the original right-hand side that was generated by the interior point optimization package. For all our tests, scaling was found to make an insignificant difference and hence we do not report on the effects on scalings here.

We used two steps of iterative refinement in cases where perturbations have been performed during the numerical factorization and a factorization is considered to be successful if the backward error

$$\|\boldsymbol{b} - A\boldsymbol{x}\|_\infty \; < \; \epsilon \; (\; \|A\|_\infty \; \|\boldsymbol{x}\|_\infty + \|\boldsymbol{b}\|_\infty \;)$$

is smaller than $\epsilon = 10^{-4}$. This corresponds to the same accuracy as used by Gould and Scott in [17]. In addition to [17], we also consider a factorization as not successful, if the backward error significantly grows during one iterative refinement process. It should already be mentioned that the backward error for our methods is for almost all highly indefinite matrices significantly smaller than $\epsilon = 10^{-14}$ and almost on the order of the machine precision.

5.3. Performance Profiles. In order to evaluate the quality of the different pivoting methods for symmetric indefinite linear systems we used performance profiles as a tool for benchmarking and for comparing the algorithms. Theses profiles were firstly proposed in [9] for benchmarking optimization software and recently used in [16, 17] to evaluate various sparse direct linear solvers.

The profiles are generated by running the set of pivoting methods μ on the set of sparse matrices σ and recording information of interest, e.g. time for numerical factorization, memory consumption, and backward error accuracy. Let us assume that a pivoting method $m \in \mu$ reports a result $t_{ms} \geq 0$ for the sparse indefinite matrix $s \in \sigma$ and that a smaller result t_{ms} indicates a better solution strategy. We can further define $\tilde{t}_s = \min\{ \; t_{ms}, m \in \mu \; \}$, which represents the best result for a given sparse matrix m. Then for $\alpha \geq 0$ and each $m \in \mu$ and $s \in \sigma$ we define

$$k(t_{ms}, \tilde{t}_s, \alpha) = \begin{cases} 1 & \text{if } t_{ms} \leq \alpha \, \tilde{t}_s \\ 0 & \text{otherwise.} \end{cases}$$

The performance profile $p_m(\alpha)$ of the pivoting method m is then defined by

$$p_m(\alpha) = \frac{\sum_{s \in \sigma} k(t_{ms}, \tilde{t}_s, \alpha)}{|\sigma|}$$

Thus, in these profiles, the values of $p_m(\alpha)$ indicate the fraction of all examples, which can be solved within α times. The best strategy, e.g. $p_m(1)$, gives the fraction of which the pivoting method m is the most effective method and $\lim_{\alpha \to \infty}$ indicates the fraction for which the algorithm succeeded.

[4] https://sourceforge.net/projects/math-atlas

TABLE 5.2
Augmented symmetric indefinite test matrices. n denotes the total number of unknowns, m the number of zero diagonals, and nnz the nonzeros in the matrix.

no.	matrix	n	m	nn,	Application
1	A0NSDSIL	80,016	35,008	200,021	Linear Complementarity
2	A2NNSNSL	80,016	35,008	196,115	Linear Complementarity
3	A5ESINDL	60,008	25,004	145,004	Linear Complementarity
4	AUG2D	29,008	29,008	38,416	2D PDE Expanded system
5	AUG2DC	30,200	30,200	40,000	2D PDE Expanded system
6	AUG3D	24,300	24,300	34,992	3D PDE Expanded system
7	AUG3DCQP	35,543	8,000	77,829	3D PDE Expanded system
8	BLOCKQP1	60,012	20,001	340,022	QP with block structure
9	BLOWEYA	30,004	20,003	90,006	Cahn-Hilliard problem
10	BOYD1	93,279	18	652,246	KKT — convex QP
11	BOYD2	466,316	186,531	890,093	KKT — convex QP
12	BRAINPC2	27,607	13,800	96,601	Biological model
13	BRATU3D	27,792	24,334	88,627	3D Bratu problem
14	CONT-201	80,595	40,198	239,596	KKT — convex QP
15	CONT-300	180,895	90,298	562,496	KKT — convex QP
16	DARCY003	389,874	155,746	116,768	Darys's KKT
17	DTOC	24,993	24,993	34,986	Discrete-time control
18	D_PRETOK	182,730	53,570	885,416	Mine model
19	K1_SAN	67,759	20,805	303,364	Mine model
20	mario001	38,434	15,304	114,643	Stokes equation
21	mario002	389,874	155,746	1'167,685	Stokes equation
22	NCVXQP1	12,111	5,000	40,537	KKT — nonconvex QP
23	NCVXQP3	75,000	25,000	324,982	KKT — nonconvex QP
24	NCVXQP5	62,500	12,500	237,483	KKT — nonconvex QP
25	NCVXQP7	87,500	37,500	312,481	KKT — nonconvex QP
26	NCVXQP9	16,554	7,500	31,547	KKT — nonconvex QP
27	olesnik0	88,263	27,233	402,623	Mine model
28	SIT100	10,262	3,120	34,094	Mine model
29	stokes128	49,666	16,384	295,938	Stokes equation
30	stokes64	12,546	4,096	74,242	Stokes equation
31	stokes64s	12,546	4,096	74,242	Stokes equation
32	tuma1	22,967	9,607	66,592	Mine model
33	tuma2	12,992	5,477	37,458	Mine model
34	turan_M	189,924	56,110	912,345	Model of uraniummine
35	lnt09	17,990	14,483	49,401	optimal control matrix
36	mass06	33,794	512	145,253	mass matrix
37	cont5_2	2,012,000	1,004,000	6,020,000	interior point optimization
38	cont5_3	2,012,000	1,004,000	6,020,000	interior point optimization

5.4. Factorization Pivoting Algorithms. Apart from the supernode Bunch and Kaufman approach, we provide comparisons with several other symmetric factorization pivoting methods as well as with symmetric matchings as an additional preprocessing method. In particular, we will consider the following options:

ETNA
Kent State University
etna@mcs.kent.edu

CHOLESKY	METIS ordering and LL^T factorization for symmetric positive definite systems. The diagonal elements of all test matrices are changed in such a way that the resulting symmetric system is positive definite.
SBK	METIS ordering and LDL^T factorization with Supernode–Bunch–Kaufman pivoting using 1×1 and 2×2 pivots combined with a pivoting perturbation, if the actual absolute pivot is less than $\epsilon \cdot \|\|A\|\|_1$ with $\epsilon = 10^{-8}$.
SBK-COMP-1	METIS orderings based on compressed subgraphs that are obtained by a symmetric matching. LDL^T factorization with Supernode–Bunch–Kaufman pivoting using 1×1 and 2×2 pivots combined with a pivoting perturbation, if the actual absolute pivot is less than $\epsilon \cdot \|\|A\|\|_1$ with $\epsilon = 10^{-8}$.
SBK-COMP-2	as SBK-COMP-1, but with preselected enlarged supernodes to respect the 2x2 pivot blocks

As already mentioned, we always used multilevel nested dissection[5][23] to symmetrically reorder the rows and columns of all matrices prior to factoring them. This is either performed on the original adjacency graph for the methods CHOLESKY and SBK or on the compressed 1×1 and 2×2 subgraph for the method SBK-COMP-1 and SBK-COMP-2. We have also included the CHOLESKY factorization in the numerical experiments in order to show the performance of the indefinite symmetric factorization methods in relation to the CHOLESKY factorization. In this case we change the diagonal elements of the original matrix A such that the new system $\tilde{A}$ has the same sparsity structure as A, but is now symmetric positive definite. This allows us to use the Cholesky method and we report on timing results for a Level-3 BLAS Cholesky factorization method. This Cholesky performance profile represents in most of the cases an upper bound and it is useful to assess the quality of the algorithms for indefinite systems.

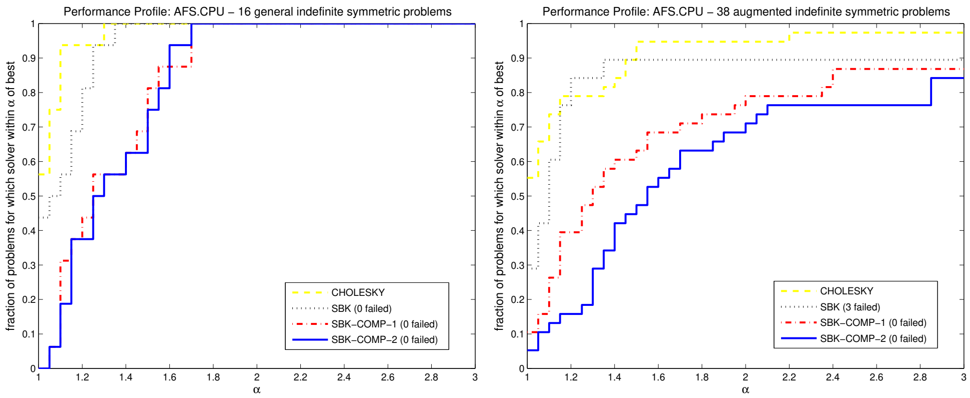


FIG. 5.1. *Performance profile CPU time for the complete solution (analysis, factor, solve) for both sets of symmetric indefinite matrices.*

---

[5]We used the METIS Version 4.0.1 Interface METIS_NodeND with default option for the general indefinite systems and the option array $[1, 3, 1, 2, 0, 1, 200, 1]$ for augmented indefinite systems.

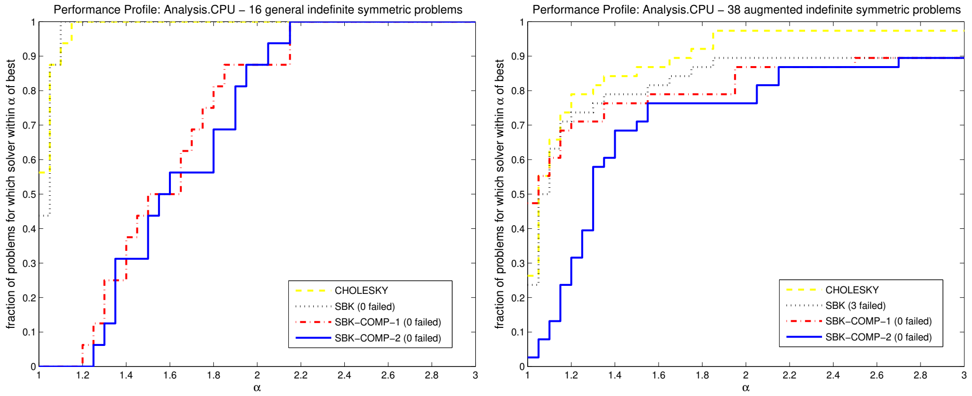


FIG. 5.2. *Performance profile CPU time for the analysis for both sets of symmetric indefinite matrices.*

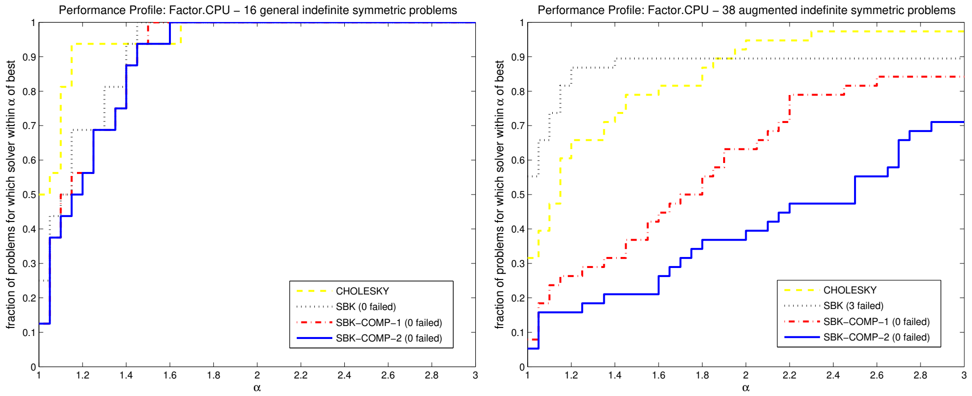


FIG. 5.3. *Performance profile CPU time for the numerical factorization for both sets of symmetric indefinite matrices.*

5.5. Numerical results for augmented and general symmetric indefinite systems. We present in Figure 5.1 the performance profiles of the CPU time for the complete solution process including analysis, factorization, forward/backward solution (abbreviated by 'solve') and potentially two steps of iterative refinement in the presence of pivot perturbations. In Figure 5.1 and all other Figures 5.2 to 5.4 the left graphic always shows the performance profiles for the general indefinite systems whereas the right graphic shows profile information for the augmented symmetric indefinite systems.

It is immediately apparent that augmented symmetric indefinite systems are much harder to solve than general indefinite systems. This has also been observed in [16, 17]. However, it is clearly visible that the overall reliability of the methods SBK, SBK-COMP-1, and SBK-COMP-2 is generally high and the absolute CPU time is on the same order of magnitude as that of the CHOLESKY method. This is already a good indication of the robustness and performance of these approaches. The results also show that weighted graph matchings have a strong influence on the overall performance of the augmented symmetric indefinite systems, where the method without symmetric matchings SBK is superior if it works to the SBK-COMP-1and SBK-COMP-2 strategies. For the general indefinite systems there is only a minor difference due to the reason that, for almost all systems, the permutation produced by the symmetric matchings is identical or very close to the identity. A close examination of the

ETNA
Kent State University
etna@mcs.kent.edu

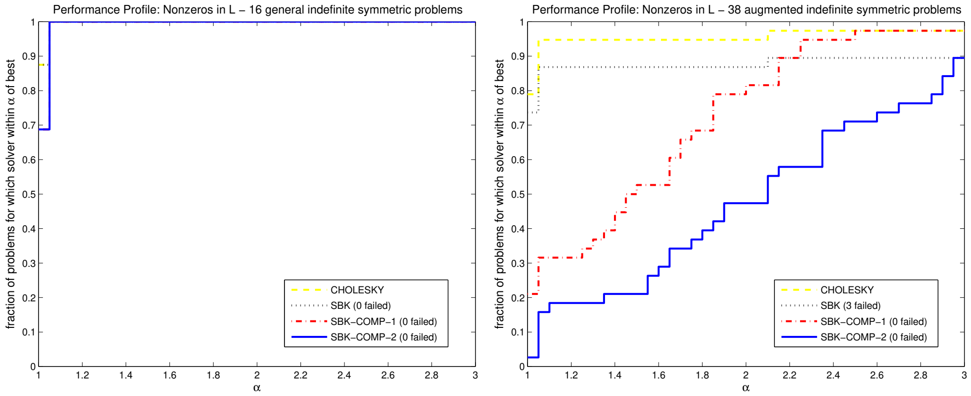


FIG. 5.4. *Performance profile for the numbers of nonzeros in the factor L for both sets of symmetric indefinite matrices.*

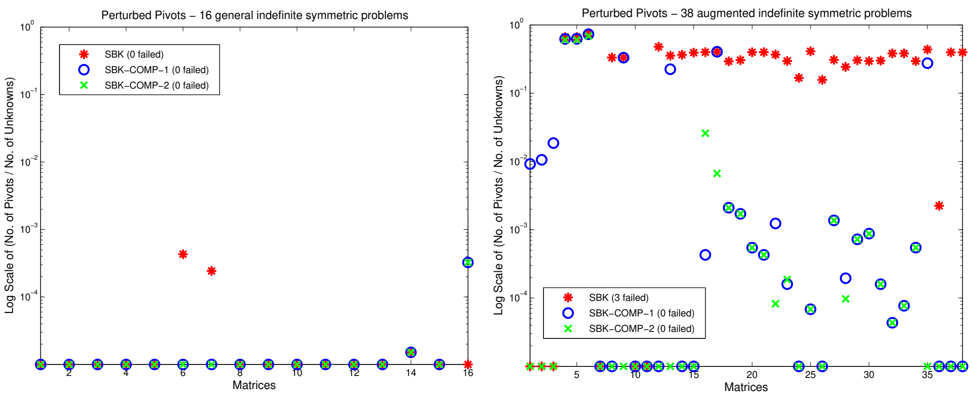


FIG. 5.5. *Percentage of perturbed pivots for both sets of symmetric indefinite matrices.*

separate analysis, factorization, and forward/backward solution (abbreviated by 'solve') time will reveal this in Figures 5.2 to 5.8.

Figure 5.2 compares the profiles for the analysis. The analysis time for the methods CHOLESKYand SBK are very similar. Matching algorithms are not applied, hence these methods have the fastest analysis time for both sets of indefinite systems. The graph-weighted matching methods SBK-COMP-1and SBK-COMP-2 produced a nontrivial permutation for six general indefinite matrices `copter2`, `dawson5`, `DIXMAANL`, `LINVERSE`, `NCVXBQP1`, `SPMSRTLS` and for all augmented symmetric indefinite systems.

In Figures 5.3 and 5.4 we compare the factorization time and numbers of nonzeros in the factors for all methods. Firstly, it can be noticed that the SBK factorization method is faster than CHOLESKY for the augmented symmetric indefinite linear systems. The reason is that we use slightly different LAPACK factorization routines[6]. Secondly, when applying symmetric matchings to the augmented indefinite systems, the factorization using SBK-COMP-1 is in general about a factor of two slower due to the fact that the resulting factors require more nonzeros compared to SBK.

Now we turn our attention to the numbers of perturbed pivots during factorization. Figure

[6]DSYTRF has been used for symmetric indefinite matrices and DGETRF for symmetric positive definite.

ETNA
Kent State University
etna@mcs.kent.edu

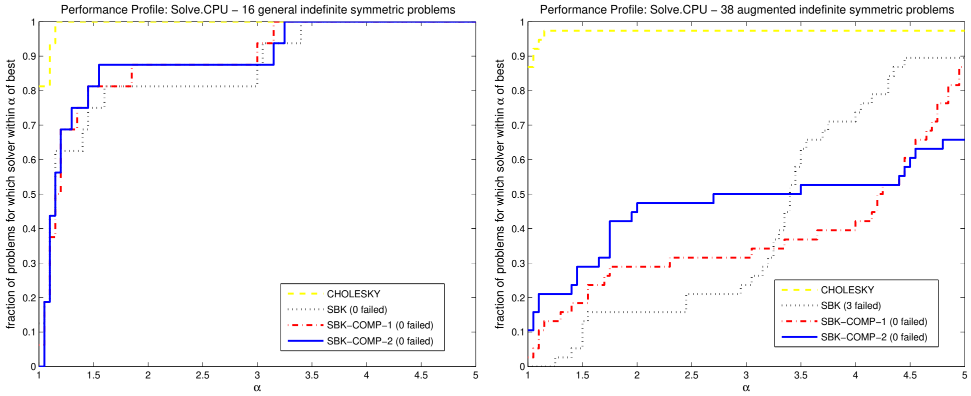


FIG. 5.6. *Performance profile CPU time for the solve including iterative refinement for both sets of symmetric indefinite matrices.*

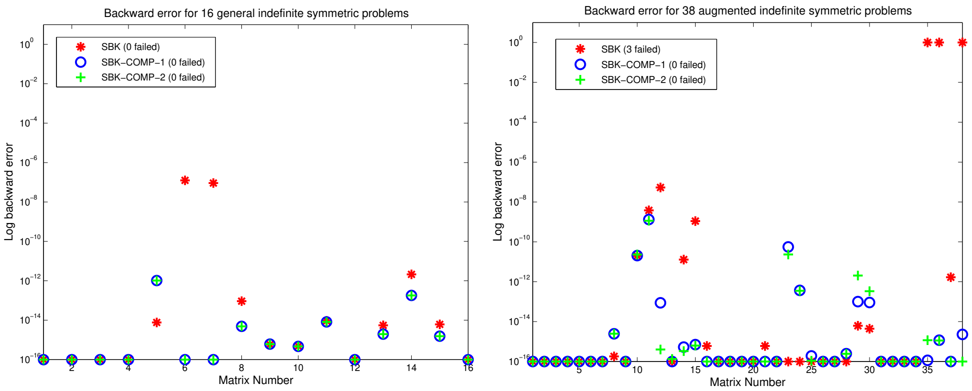


FIG. 5.7. *Size of the backward error for both sets of symmetric indefinite matrices.*

5.5 demonstrates that only three of the sixteen general indefinite systems are affected by pivot perturbation. This is different for the augmented indefinite systems, where for many systems pivot perturbations occur during factorization. The Figure 5.5 shows the percentage of perturbed pivots and it can be seen that the additional preprocessing can reduce the occurrence of pivot perturbation by several orders of magnitude due to the effective initial construction of 1×1 and 2×2 cycles during the analysis. This increases the factorization time by about a factor of two, but it also adds reliability that might be needed in various applications.

We now evaluate the solution time for all methods in Figure 5.6. As already mentioned in Section 5.2, we perform two steps of an iterative refinement method if any pivot perturbation has been encountered during factorization for the methods SBK, SBK-COMP-1, and SBK-COMP-2. It is clear that the solve step is affected by this strategy and the figures demonstrate that the influence is higher for the augmented linear systems.

Figure 5.7 and Figure 5.8 show the accuracy for all factorization pivoting methods and all general and augmented indefinite systems. Figure 5.7 compares the backward error ($\|\boldsymbol{b} - A\boldsymbol{x}\|_\infty \,/\, (\,\|A\|_\infty \, \|\boldsymbol{x}\|_\infty + \|\boldsymbol{b}\|_\infty\,)$). A backward error larger than 10^{-4} indicates a failure of the method. It can be seen that the backward error is for most of the general indefinite systems of order 10^{-16}. For the augmented indefinite systems the SBK-COMP-1 and SBK-COMP-2 methods also often produce backward error close to machine precision. The SBK backward

ETNA
Kent State University
etna@mcs.kent.edu

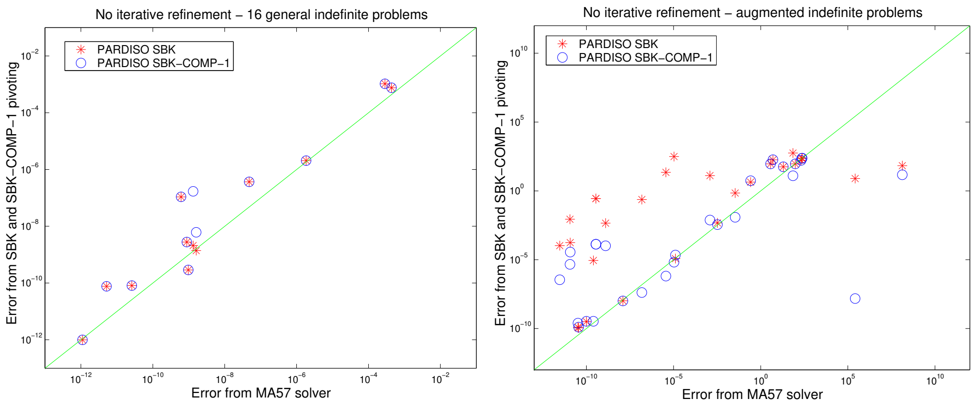


FIG. 5.8. *Comparison of relative error without iterative refinement with* SBK-COMP-1*,* SBK *and* MA57 *for both sets of symmetric indefinite matrices.*

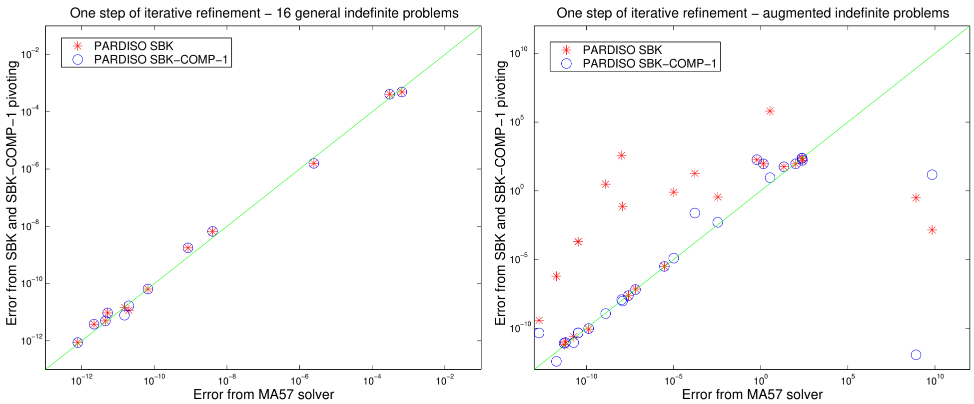


FIG. 5.9. *Comparison of relative error with one step of iterative refinement with* SBK-COMP-1*,* SBK *and* MA57 *for both sets of symmetric indefinite matrices.*

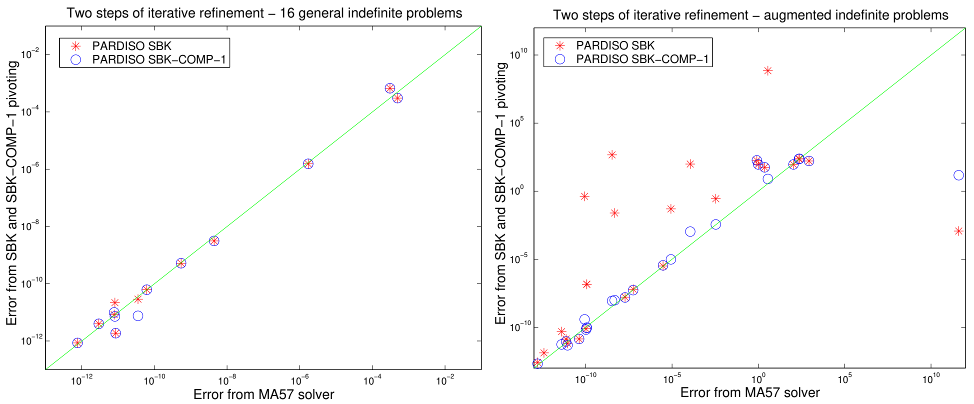


FIG. 5.10. *Comparison of relative error with two steps of iterative refinement with* SBK-COMP-1*,* SBK *and* MA57 *for both sets of symmetric indefinite matrices.*

errors are a bit larger and for four systems, the backward errors are on the order of 10^{-10}.

Figure 5.8 assesses the accuracy of the SBK and SBK-COMP-1 methods for all sparse symmetric indefinite matrices that are not structural singular. The Figure plots the error $\|x - \mathbf{1}\|_\infty$ from both methods versus the relative error from a pivoting method that uses threshold Duff–Reid pivoting for the complete matrix. In particular we selected MA57 from [19] as a direct solver for the x-axis in the Figure. A dot on the diagonal means that the two errors were the same, a dot above means that MA57 is more accurate, and a dot below the diagonal means that the SBK and SBK-COMP-1 methods are more accurate. As already mentioned, most of the augmented systems are much harder to solve than the general indefinite systems but it can be concluded from these figures that the error of MA57 and the SBK-COMP-1 method in PARDISO are of the same order. The error of SBK can be higher for hard to solve problems, but the general conclusion is that even in the presence of small pivots, the iterative refinement can effectively recover the loss of accuracy during factorization.

5.6. Numerical results for interior point optimization matrices. In Tables 5.3 and 5.5 and Figure 5.11 we report the performance of PARDISO using the SBK method for interior point optimization matrices from IPOPT [31]. The whole set consists of 57 matrices of increasing size. It can be found at [4] and we selected a representative subset of 16 matrices. The core step of the IPOPT optimization algorithm is a damped Newton iteration, where in each step a linear system of the form

$$A = \begin{pmatrix} W + \Sigma + \delta_w I & B \\ B^T & -\delta_c I \end{pmatrix}$$

is solved. In our examples, the upper left block $A_{11} = W + \Sigma + \delta_w I$ is a diagonal matrix, B is of full rank, and the scalar δ_c is set to 10^{-8}. W and Σ are related to the Hessian of the Lagrangian function, B denotes the Jacobian of the constraints and the scalar δ_w is modified in such a way that the inertia of the system is exactly $(n, m, 0)$. Here n is the dimension of the matrix A_{11} and m denotes the rank of B. A detailed description of the interior point optimization method is given in [31]. The default solver in IPOPT is the MA27 from [20], but we refer to the newest version of MA57 [7,8].

Table 5.3 shows the name of the IPOPT matrices, and the numbers of positive and negative eigenvalues. It can be seen that all the matrices are very indefinite and in general hard to solve. The table also shows the time in seconds for the analysis, factorization, and forward/backward solution (abbreviated by 'solve'). It can be seen that for larger examples the factorization speedup of PARDISO using SBK method compared to MA57 is on the order of 2, whereas solve and analysis time are similar due to the fact that both solvers use METIS in the reordering and no pivot perturbation occurred for these matrices. As a result, both PARDISO and MA57 performed only one forward and backward solve. It is also visible that the SBK-COMP-1 is the slowest method but it can have an accuracy advantage over SBK which we will discuss later.

Table 5.4 shows the speedup of PARDISO using SBK on a SGI Altix 3700/BX2 with 8 Intel Itanium2 processors with 1.6 GHz and we report on the factorization speedup for the larger IPOPT matrices.

Table 5.5 shows the numbers of nonzeros in the factors and the total memory requirement for both methods. The total memory requirement consists of the amount used for the factors

[7]We used MA57 Version 3.0.0 (March 2005). The options for MA57 for these interior point matrices have been recommended by Iain Duff, who is the author of MA57.

[8]Detailed comparisons of sparse direct linear solver for symmetric indefinite systems can be found in [16, 17].

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 5.3
Comparison of two sparse direct linear solvers PARDISO *using* SBK *and* MA57 *using default options for a subset of the IPOPT interior point optimization matrices. The table shows time in seconds for analysis, factorization and one forward/backward substitution (abbreviated by 'solve').* n_+ *and* n_- *show the number of positive and negative eigenvalues. We also show the factorization time in brackets using* PARDISO *with* SBK-COMP-1*. All tests were run on a Pentium III 1.3 GHz.*

Matrix			PARDISO			MA57		
name	n_+	n_-	anal.	factor	solve	anal.	factor	solve
c-20	1'621	1'300	0.06	0.01/(0.01)	.002	0.01	0.02	.001
c-22	2'130	1'662	0.09	0.02/(0.03)	.003	0.01	0.03	.002
c-27	2'621	1'942	0.09	0.02/(0.02)	.002	0.01	0.02	.002
c-30	2'823	2'498	0.10	0.01/(0.02)	.004	0.02	0.05	.003
c-33	3'526	2'791	0.10	0.02/(0.03)	.006	0.03	0.05	.004
c-40	5'477	4'464	0.14	0.02/(0.03)	.006	0.02	0.07	.005
c-42	5'930	4'541	0.27	0.02/(0.08)	.012	0.04	0.11	.009
c-55	19'121	13'659	1.57	6.54/(18.1)	0.11	1.88	14.4	0.11
c-58	22'461	15'134	1.80	5.99/(15.4)	0.10	2.10	11.2	0.09
c-62	25'158	16'573	2.11	16.2/(43.1)	0.20	2.72	37.7	0.20
c-68	36'546	28'264	2.70	14.7/(50.7)	0.21	4.07	57.3	0.18
c-70	39'302	29'622	3.27	4.90/(16.1)	0.14	3.62	13.8	0.13
c-71	44'814	31'824	4.03	47.6/(164.)	0.41	5.04	110.	0.40
c-72	47'950	36'114	3.60	2.96/(10.2)	0.15	4.14	7.75	0.13
c-73	86'417	83'005	5.52	1.14/(3.41)	0.18	4.06	3.16	0.11
c-big	201'877	143'364	19.9	243./(615.)	1.34	18.14	487.	1.22

TABLE 5.4
Time in seconds for the numerical factorization with PARDISO *using* SBK *for 1, 4, and 8 Intel Itanium2 processors with 1.6 GHz.*

Matrix	PARDISO-SBK		
name	1 proc.	4 procs.	8 procs.
c-68	2.48	0.70	0.38
c-70	1.16	0.50	0.32
c-71	7.98	4.01	0.99
c-72	0.80	0.45	0.36
c-73	0.47	0.25	0.61
c-big	33.9	8.48	4.40

and for additional data structures that the solver consumes during the complete solution process.

Finally, the relative error $||x - \mathbf{1}||_\infty$ of both PARDISO methods versus MA57 is demonstrated in Figure 5.11. As in Figure 5.8, a dot on the diagonal means that the two errors were the same, a dot above means that MA57 is more accurate, and a dot below the diagonal means that the SBK and SBK-COMP-1 methods are more accurate. The relative error for all methods is shown without iterative refinement, and additionally with one and two steps of iterative refinements for all direct solvers. We do not list the backward error here since we noticed that both solvers always produced errors close to machine precision. It is clearly visible that for all IPOPT matrices the SBK-COMP-1 method produced similar errors as MA57 even in the case where no iterative refinement is performed. The relative error without iterative re-

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 5.5

Comparison of the direct linear solver PARDISO *using* SBK *and* SBK-COMP-1 *with* MA57 *using default options for a subset of the IPOPT interior optimization matrices. The table shows the numbers of nonzero in the factor L and the total memory requirement in MByte.*

Matrix name	SBK nnz in L	SBK memory	SBK-COMP-1 nnz in L	SBK-COMP-1 memory	MA57 nnz in L	MA57 memory
c-20	31'236	1	39'082	1	40'121	1
c-22	43'044	1	61'080	2	56'214	1
c-27	40'615	1	52'159	2	53'032	1
c-30	43'717	2	46'798	2	69'237	2
c-33	60'958	2	80'801	2	93'151	2
c-40	68'215	4	81'550	4	123'154	4
c-42	125'161	5	147'490	5	213'369	4
c-55	3'567'897	36	5'925'470	58	3'842'167	48
c-58	2'799'201	32	4'537'988	49	3'043'645	49
c-62	6'760'217	64	10'506'202	104	7'378'230	88
c-68	5'584'384	57	10'636'496	104	5'986'413	79
c-70	3'302'552	42	6'011'189	65	4'107'653	65
c-71	13'768'053	126	22'163'118	211	13'967'446	178
c-72	2'977'867	45	4'928'324	50	3'479'461	50
c-73	1'513'369	68	1'867'347	74	2'239'597	69
c-big	39'032'735	365	63'858'706	593	39'893'061	548

finement of the SBK method is higher in comparison to SBK-COMP-1, but after one step of iterative refinement all methods produced similar errors.

The computation of the inertia is a very important feature in the context of interior point optimization and it has been demonstrated in [30] that the method is very effective for large-scale nonlinear programming problems arising e.g. in optimal control of PDEs.

Finally, we evaluate the runtime of each step of the SBK-COMP-1 in Figure 5.12 for all symmetric indefinite systems (general, augmented, IPOPT). Figure 5.12 shows the fraction of runtime for the computation of the symmetric matching, the reordering and the solution including iterative refinement with respect to the time for the numerical factorization. For large enough examples, the LDL^T factorization dominates all other steps. The computation of the matching is for all matrices smaller than the reordering time except for the matrix BOYD2 from the test set in Table 5.2. We noticed that our matching code MPS required a disproportionate amount of time for matrix BOYD2, which seems to be due to a large number of dense rows in this matrix.

6. Conclusions. This paper demonstrates the effectiveness of new pivoting methods for sparse symmetric indefinite systems. As opposed to many existing pivoting methods, SBK uses dynamically 1×1 and 2×2 pivoting within a diagonal block associated to a supernode. The coefficient matrix is perturbed whenever numerically acceptable pivots can not be found within the diagonal block and only one or two passes of iterative refinement may be required to correct the effect of the perturbations required. We demonstrated the effectiveness and the numerical accuracy of the algorithm and also showed that a high performance implementation is feasible for this algorithm.

In addition, we discussed two algorithms to identify large entries in the coefficient matrix A that, if permuted close to the diagonal, permit the factorization process to identify more acceptable pivots and proceed with fewer pivot perturbations. The methods are based on

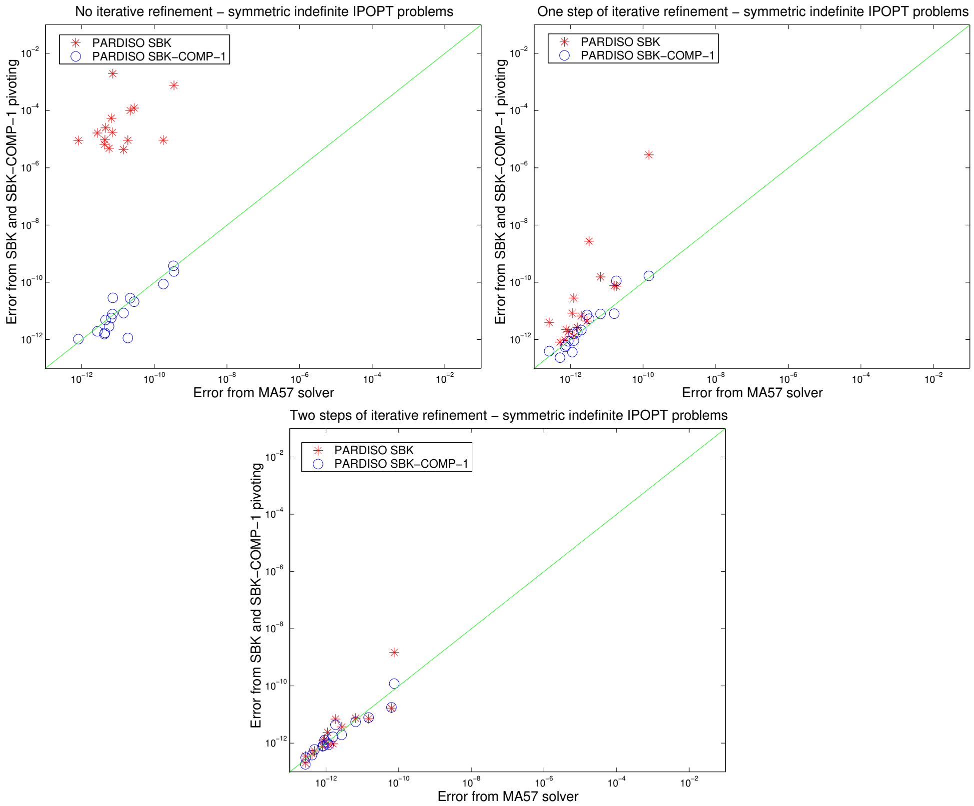


FIG. 5.11. *Comparison of relative error with* SBK-COMP-1*,* SBK *and* MA57 *for general symmetric indefinite* IPOPT *matrices without iterative refinement (left), one step iterative refinement (right), two steps iterative refinement (middle).*

maximum weighted matchings and improve the quality of the factor in a complementary way to the alternative idea of using more complete pivoting techniques during the factorization. The numerical experiments show that the additional effort of symmetric weighted matchings for producing good initial 1×1 and 2×2 pivots during the analysis is not always required. However, these two methods add an additional level of reliability without severely decreasing the performance of the solver. For a large number of real-world examples, SBK-COMP-1 and SBK-COMP-2 reorderings are capable of providing a feasible pivoting order for the factorization, while the cost of this preprocessing step is often negligible. However, the memory requirements and the factorization times increase and MA57 looks like the more efficient approach in these circumstances. Further existing possibilities to merge matching and fill-in reduction information should be studied in the future to reduce the overhead introduced. The matching information drastically decreases the number of perturbations, hence other correction techniques, e.g. Sherman-Morrison-Woodbury, may now be applied efficiently and without harming the parallelization capacity.

In addition, the methods open possibilities to achieve scalability for symmetric indefinite factorization on parallel architectures, since similar data structures and communication patterns, as in the highly scalable sparse Cholesky case, can be exploited as shown in Table 5.4. The threshold pivoting approach in MA57 clearly has a precision advantage. On the other hand, it has been shown in [30] that iterative refinement in conjunction with SBK can often

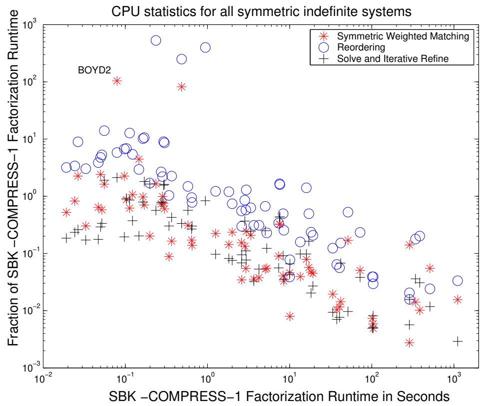


FIG. 5.12. *The runtime of the* SBK-COMP-1 *method versus other steps (computation of the matching, reordering and solution with iterative refinement) for all symmetric indefinite systems.*

recover the loss of accuracy during the factorization with iterative refinements.

Acknowledgments. The authors thank Jennifer Scott for discussions during the surveys and providing the matrices from [17], and Andreas Wächter for the IPOPT interior point optimization matrices. We would also like to thank Iain Duff for providing us MA57.

REFERENCES

[1] J. AASEN, *On the reduction of a symmetric matrix to triagonal form*, BIT, 11 (1971), pp. 233–242.
[2] C. ASHCRAFT AND R. GRIMES, *The influence of relaxed supernode partitions on the multifrontal method*, ACM Trans. Math. Software, 15 (1989), pp. 291–309.
[3] C. ASHCRAFT, R. GRIMES, AND J. LEWIS, *Accurate symmetric indefinite linear equation solvers*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 513–561.
[4] *Basel Sparse Matrix Collection*. `http://computational.unibas.ch/cs/scicomp/matrices`.
[5] M. BENZI, J. HAWS, AND M. TUMA, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.
[6] J. BUNCH AND L. KAUFMANN, *Some stable methods for calculating inertia and solving symmetric linear systems*, Math. Comp., 31 (1977), pp. 162–179.
[7] J. BUNCH AND B. PARLETT, *Direct methods for solving indefinite systems of linear equations*, SIAM J. Numerical Analysis, 8 (1971), pp. 639–655.
[8] T. DAVIS, *University of Florida Sparse Matrix Collection*, University of Florida, Gainesville. `http://www.cise.ufl.edu/davis/sparse/`.
[9] E. D. DOLAN AND J. MORÉ, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[10] I. DUFF, A. ERISMAN, AND J.K.REID, *Direct Methods for Sparse Matrices*, Oxford Science Publications, 1986.
[11] I. S. DUFF AND J. R. GILBERT, *Maximum-weighted matching and block pivoting for symmetric indefinite systems*, in Abstract book of Householder Symposium XV, June 17-21 2002, pp. 73–75.
[12] I. S. DUFF AND J. KOSTER, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 889–901.
[13] I. S. DUFF AND S. PRALET, *Strategies for scaling and pivoting for sparse symmetric indefinite problems*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 312–340.
[14] ———, *Towards a stable static pivoting strategy for the sequential and parallel solution of sparse symmetric indefinite systems*, Technical Report RAL-TR-2005-007, Rutherford Appleton Laboratory, 2005.
[15] I. S. DUFF AND J. K. REID, *A comparison of sparsity orderings for obtaining a pivotal sequence in Gaussian elimination*, J. Institute of Mathematics and its Applications, 14 (1974), pp. 281–291.
[16] N. GOULD, Y. HU, AND J. SCOTT, *A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations*, Tech. Report RAL-TR-2005-005, Rutherford Appleton Laboratory, 2005.

[17] N. GOULD AND J. SCOTT, *A numerical evaluation of HSL packages for the direct solution of large sparse, symmetric linear systems of equationss*, ACM Trans. Math. Software, 30 (2004), pp. 300–325.

[18] A. GUPTA AND L. YING, *On algorithms for finding maximum matchings in bipartite graphs*, Tech. Report RC 21576 (97320), IBM T. J. Watson Research Center, Yorktown Heights, NY, October 25, 1999.

[19] *Harwell Subroutine Library, AEA Technology, Harwell, Oxfordshire, England catalogue of subroutines*, 2004.

[20] *Harwell Subroutine Library Archive, AEA Technology, Harwell, Oxfordshire, England catalogue of subroutines*, 2002.

[21] J. HAWS AND C. MEYER, *Preconditioning KKT systems*, Tech. Report M&CT-TECH-01-021, Mathematics and Computing Technology, The Boeing Company, 2001.

[22] N. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[23] G. KARYPIS AND V. KUMAR, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[24] X. S. LI AND J. W. DEMMEL, *SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems*, ACM Trans. Math. Software, 29 (2003), pp. 110–140.

[25] M. OLSCHOWKA AND A. NEUMAIER, *A new pivoting strategy for gaussian elimination*, Linear Algebra Appl., 240 (1996), pp. 131–151.

[26] S. RÖLLIN AND O. SCHENK, *Maximum-weighted matching strategies and the application to symmetric indefinite systems*, Lecture Notes in Computer Science, 3732, Springer, 2006, pp. 808–817.

[27] O. SCHENK AND K. GÄRTNER, *Solving unsymmetric sparse systems of linear equations with PARDISO*, Journal of Future Generation Computer Systems, 20 (2004), pp. 475–487.

[28] O. SCHENK, K. GÄRTNER, AND W. FICHTNER, *Efficient sparse LU factorization with left-right looking strategy on shared memory multiprocessors*, BIT, 40 (2000), pp. 158–176.

[29] O. SCHENK, S. RÖLLIN, AND A. GUPTA, *The effects of unsymmetric matrix permutations and scalings in semiconductor device and circuit simulation*, IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, 23 (2004), pp. 400–411.

[30] O. SCHENK, A. WÄCHTER, AND M. HAGEMANN, *Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization*, Linear Algebra Issues Arising in Interior Point Methods, special issue of Comput. Optim. Appl., in press.

[31] A. WÄCHTER AND L. T. BIEGLER, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Math. Program., 106 (2006), pp. 25–57.