Interval Computations
No 2, 1993

Direct Sparse Interval Hull Computations for Thin Non-M-Matrices

Fernando L. Alvarado[*] and Zian Wang

The computation of an enclosure for the solution of a sparse set of linear equations $Ax = b$ with a thin matrix A and an interval right hand side b is considered. Gaussian elimination works well when the matrix is an M-matrix, but fails when the matrix is a general matrix. This paper illustrates how, for sparse matrices, an ordering intended to reduce the height of the elimination tree (and thus enhance parallelism of computations) can reduce the growth in the interval solution for general non-M-matrices. For banded matrices, it is shown that the growth in intervals is bounded by n, where n is the dimension of the matrix, rather than 2^n. This result appears to hold for other types of sparse matrices. A second method based on singleton right hand side representations of b is shown to yield the hull of the solution. Best results are obtained by a combination of the singleton solver with path length shortening permutations.

Прямые вычисления интервальной оболочки для тонкой разреженной не-М-матрицы

Ф. Л. Альварадо, Ж. Ванг

Рассмотрен метод вычисления оболочки решения разреженной системы линейных уравнений вида $Ax = b$, где A — тонкая матрица, а b — интервальная правая часть. Метод исключения Гаусса хорошо работает лишь для М-матриц и не справляется с задачей в общем случае. Описан и проиллюстрирован способ, позволяющий для разреженных матриц путем упорядочивания элементов уменьшить глубину дерева исключений (увеличивая, таким образом, возможность производить вычисления параллельно). При этом уменьшается сложность интервального решения для общего случая не-М-матриц. Для ленточных матриц показано, что рост интервалов ограничен n — размерностью матрицы, а не 2^n. Данный результат применим и к другим типам разреженных матриц. Второй метод, основывающийся на представлении b в правой части в виде одноэлементного множества, позволяет решить задачу нахождения оболочки решения. Комбинация вышеуказанных методов позволяет достичь наилучших результатов.

[*]The author was sponsored by NSF contracts ECS-8907391, ECS-9216520 and DMS-9216120.

1 Introduction

The computation of an interval enclosure for the solutions x to a sparse set of interval linear equations $Ax = b$ has seen much activity lately. For the case of inverse-positive matrices (M-matrices), it is known that $2n$ solutions suffice. The case of an interval matrix whose mid-point is the identity matrix has led to results that require just one solution [1, 2]. However, these methods may fail when the matrix is a general non-M-matrix: considerable overestimation of the hull is possible. This paper considers a more narrow problem: the solution of $Ax = b$ with a thin matrix A and an interval right hand side b but where the matrix A can be a general matrix.

Two of the reasons for the overestimation that often occurs in interval mathematics are:

- The repeated use of the same interval at more than one place in a calculation sequence. Perhaps the most trivial example of this is the evaluation of x^2 when $x \in \mathbf{IR}$ and $0 \in x$. Consider, for example:

$$\begin{aligned} [-2,3]^2 &= [0,9] \\ [-2,3]\cdot[-2,3] &= [-6,9]\,. \end{aligned}$$

- The failure of the distributive law. In general, $x \cdot (y+z) \neq x \cdot y + x \cdot z$. Consider, for example:

$$\begin{aligned} [-2,3]\cdot\big([-2,1]+[3,4]\big) &= [-10,15] \\ [-2,3]\cdot[-2,1]+[-2,3]\cdot[3,4] &= [-14,16]\,. \end{aligned}$$

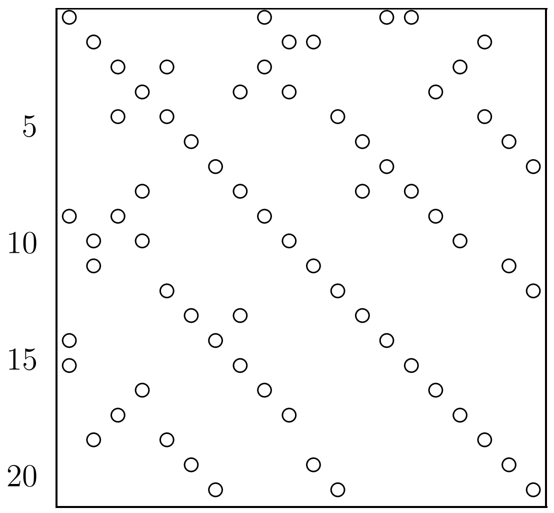
Overestimation is particularly troublesome during forward and back substitution solutions of factored linear equations. The algorithm for forward substitution using an already factored matrix $A = LDU$, where L is unit lower triangular with entries ℓ_{ij}, D is diagonal and U is unit upper triangular, can be stated as:

Forward substitution algorithm

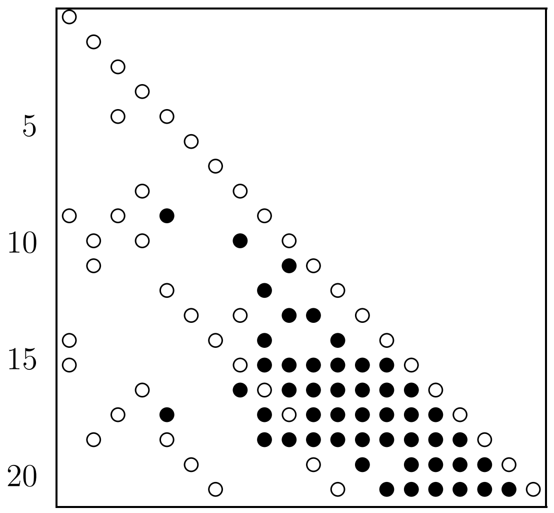
```
For $i = 1, 2, \ldots,$ n
    for all $j$ for which $\ell_{ij} \neq 0$
        $b_i \leftarrow b_i - \ell_{ij} b_j$
```

Thus, the solution to $Ly = b$ by forward substitution (denoted hereinafter as $L\backslash b$) is constructed as a sequence of modifications to the vector b itself. In this algorithm, y_1 depends only on b_1, y_2 depends on b_2 and y_1 but only *indirectly* on b_1, and so on.

For a sparse matrix, not all dependencies exist. Consider a sparse matrix with the following topology (nonzero pattern):



Factorization of this matrix results in the following unit lower triangular matrix structure:



Solid symbols • denote fills, elements that start as zero entries and become nonzero during the factorization. The dependencies that take place during the factorization are best illustrated by means of an elimination tree [3]. Figure 1 illustrates the structure of the elimination tree for this matrix. The last entry, y_{20}, depends indirectly on all previous entries. However, an element such as y_{14} depends only on certain other entries, y_1, y_3, y_5, y_7, y_9, and y_{12} in this case.

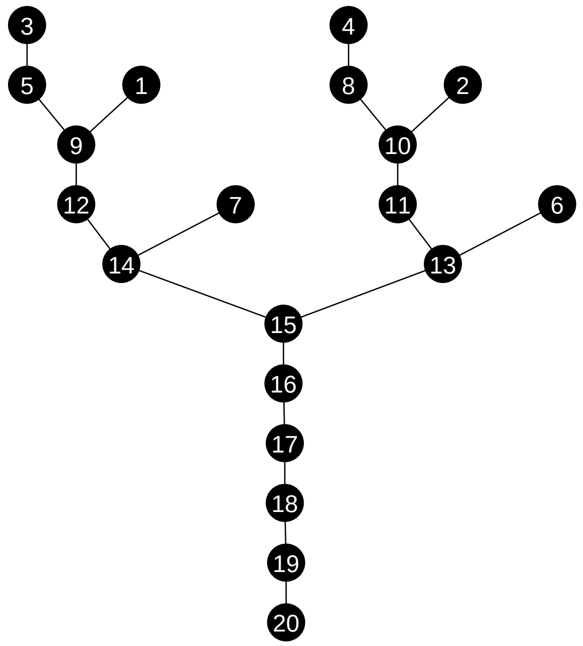


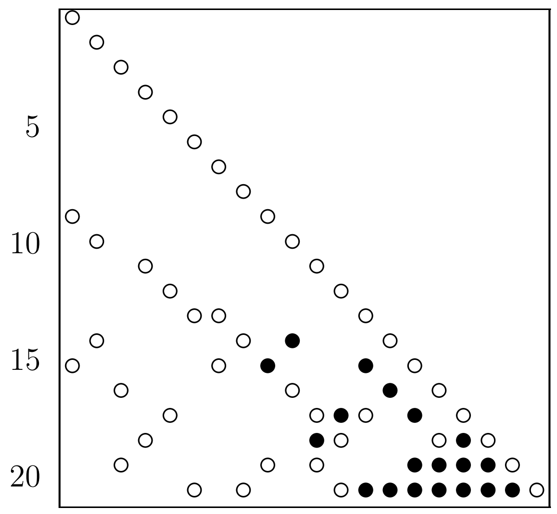
Figure 1: Elimination tree for matrix in original order.

The paper deals with two distinct lines of thinking, each with independent merit. The first is the notion that reordering of the rows and columns of a matrix can reduce the growth in the intervals by reducing the length of computational sequences (the height of the elimination tree), but this requires a counter-intuitive ordering procedure for certain kinds of matrices such as banded matrices. The second key notion in this paper is that, by processing the entries of the right hand side vector one at the time, one can obtain the tightest possible interval enclosure (the *hull*) of the solution for arbitrary thin matrices.

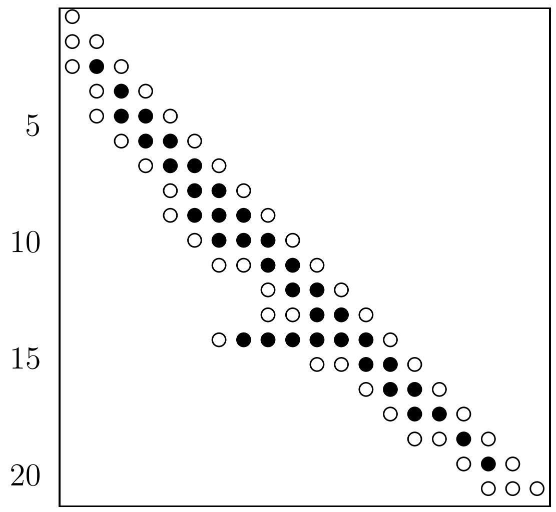
2 Reducing interval growth by reordering

In exact arithmetic, the solution of $Ax = b$ is unaffected by a renumbering of the rows and columns of A, provided b and x are renumbered accordingly. Limited precision numeric computation, however, often results in substantial differences in solutions that depend on the ordering of the matrix. As a consequence, various kinds of numerical pivoting schemes have been developed in an effort to reduce error growth. Two such examples are complete and partial pivoting.

For sparse matrices, ordering of rows and columns of a matrix also has significant impact on the total computation required, as a result of "fill in" that occurs during factorization. Thus, when the matrix A belongs to a class of sparse matrices (a class of matrices where the number of nonzero entries grows less then quadratically with matrix dimension), it is usually advantageous to renumber the rows and columns of A to maximize sparsity of L and U [4, 5]. Consider, as an example, the 20×20 sparse matrix above and its "unordered" factor L. If the matrix is reordered according to the Multiple Minimum Degree (MMD) algorithm [6] (a variant of the minimum degree algorithm), the topology of the resulting $\tilde{L}$ factor of the permuted matrix is:



This factor contains fewer fills than the unordered factor. Another popular reordering method, which reduces the bandwidth (technically, the profile) of the matrix, is the Gibbs-Poole-Stockmeyer (GPS) algorithm [7]. If both rows and columns of the matrix are renumbered according to GPS, the resulting $\tilde{L}$ factor (with fills) is:



There is another important consequence of renumbering a sparse matrix. This has to do with the precedence of the operations required to perform the forward and back substitution steps that follow the factorization. The elimination trees for the two renumbered matrices are illustrated in Figure 2.

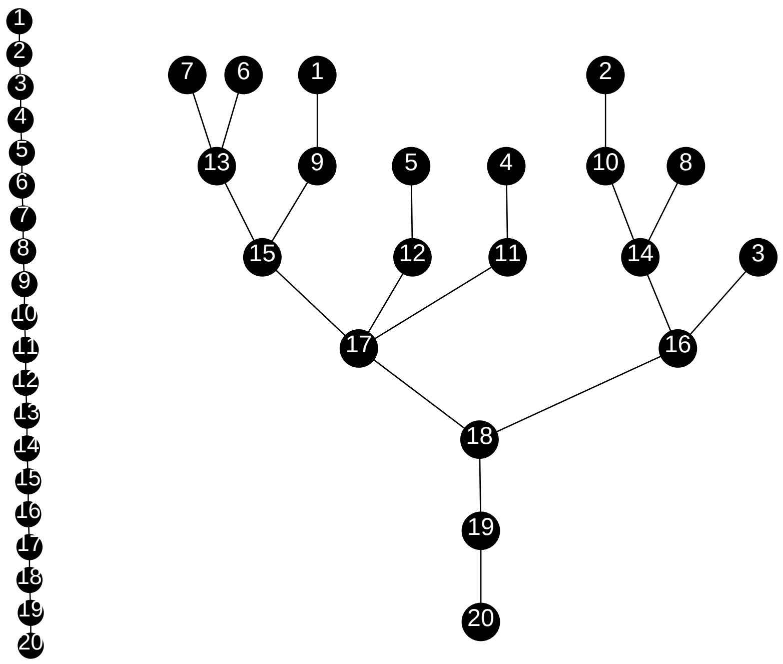


Figure 2: Elimination tree for matrix ordered according to GPS algorithm (left) and multiple minimum degree algorithm (right).

Assume we want to solve $y = \tilde{L}\backslash b$. These trees illustrate direct data dependencies among the entries of b while computing y. A similar interpretation exists for the back substitution step using the same tree. The observation from these figures is that the computation of the forward substitution using the GPS algorithm results in a significantly longer sequence of data dependencies, while that from the MMD method results in a shorter sequence of dependencies.

When the matrix A is a (possibly dense) matrix, it is a well known result that renumbering of matrix rows and columns to help reduce error accumulation is essential for large scale computation. The renumbering process is called pivoting. Two different types of pivoting are generally considered: partial pivoting and complete pivoting. Partial pivoting guarantees a bound on the growth of the solution error of 2^n, while complete pivoting guarantees a better bound. This bound is conservative. For diagonal dominant and other well-behaved matrices, one can sometimes dispense with numerical pivoting altogether. Or rather, one is free to perform any (generally diagonal) renumbering required for the preservation of sparsity during factorization. In the case of unsymmetric, indefinite or non-M-matrices, one must often use numerical pivoting in conjunction with sparsity preserving matrix renumbering. Since the two objectives are often in conflict, a good compromise is often "threshold pivoting", the idea of performing pivoting for sparsity preservation reasons, unless the pivot so chosen is worse than the numerically chosen pivot by more than some specified factor [4].

Definition 1. *A thin matrix, vector or scalar is defined as a matrix, a vector or a scalar whose entries belong to the set of real numbers* $\mathbf{R}$.

Definition 2. *An interval matrix, vector or scalar is defined as a matrix, vector or scalar whose entries belong to the set of interval real numbers* $\mathbf{IR}$. *A real interval number is representable as an ordered pair of two real numbers,* $[\underline{\alpha}, \bar{\alpha}]$, *where* $\underline{\alpha} \in \mathbf{R}$ *and* $\bar{\alpha} \in \mathbf{R}$, $\underline{\alpha} \leq \bar{\alpha}$.

A thin matrix, vector or scalar can be viewed as a special case of an interval.

Definition 3. *The radius* $\operatorname{rad}(\alpha)$ *of an interval number is defined as a real number* $r \in \mathbf{R}$, *where* $r = (\bar{\alpha} - \underline{\alpha})/2$.

Definition 4. *An M-matrix* A *is a matrix where* $A^{-1} > 0$, *that is, every entry of its inverse is positive.*

Definition 5. *A matrix A is said to be banded with bandwidth β if $a_{ij} = 0$ if $|i - j| > \beta$.*

Definition 6. *A vector b^i of dimension n is defined as a singleton interval vector if b^i_i is an interval scalar and all other entries of b^i are zero. The vector e^i is a unit singleton interval vector if it is an interval vector and in addition $e^i_i = [-1, 1]$. The vector $\bar{e}^i$ is a unit singleton vector (thin) if $\bar{e}^i_i = 1$.*

This paper concerns itself with thin non-M-matrices and interval vectors. Ordering of the rows and columns of a matrix is often important. We motivate the importance of matrix ordering by means of a well known example [8]. Consider the following banded matrix:

$$\begin{bmatrix}
1&1&1&&&&&&&&&&&&&&&&& \\
1&2&2&1&&&&&&&&&&&&&&&& \\
1&2&3&2&1&&&&&&&&&&&&&&& \\
&1&2&3&2&1&&&&&&&&&&&&&& \\
&&1&2&3&2&1&&&&&&&&&&&&& \\
&&&1&2&3&2&1&&&&&&&&&&&& \\
&&&&1&2&3&2&1&&&&&&&&&&& \\
&&&&&1&2&3&2&1&&&&&&&&&& \\
&&&&&&1&2&3&2&1&&&&&&&&& \\
&&&&&&&1&2&3&2&1&&&&&&&& \\
&&&&&&&&1&2&3&2&1&&&&&&& \\
&&&&&&&&&1&2&3&2&1&&&&&& \\
&&&&&&&&&&1&2&3&2&1&&&&& \\
&&&&&&&&&&&1&2&3&2&1&&&& \\
&&&&&&&&&&&&1&2&3&2&1&&& \\
&&&&&&&&&&&&&1&2&3&2&1&& \\
&&&&&&&&&&&&&&1&2&3&2&1& \\
&&&&&&&&&&&&&&&1&2&3&2&1 \\
&&&&&&&&&&&&&&&&1&2&3&2 \\
&&&&&&&&&&&&&&&&&1&2&3
\end{bmatrix}$$

Its L factor is:

$$\begin{bmatrix}
1 &&&&&&&&&&&&&&&&&&& \\
1 & 1 &&&&&&&&&&&&&&&&&& \\
1 & 1 & 1 &&&&&&&&&&&&&&&&& \\
& 1 & 1 & 1 &&&&&&&&&&&&&&&& \\
&& 1 & 1 & 1 &&&&&&&&&&&&&&& \\
&&& 1 & 1 & 1 &&&&&&&&&&&&&& \\
&&&& 1 & 1 & 1 &&&&&&&&&&&&& \\
&&&&& 1 & 1 & 1 &&&&&&&&&&&& \\
&&&&&& 1 & 1 & 1 &&&&&&&&&&& \\
&&&&&&& 1 & 1 & 1 &&&&&&&&&& \\
&&&&&&&& 1 & 1 & 1 &&&&&&&&& \\
&&&&&&&&& 1 & 1 & 1 &&&&&&&& \\
&&&&&&&&&& 1 & 1 & 1 &&&&&&& \\
&&&&&&&&&&& 1 & 1 & 1 &&&&&& \\
&&&&&&&&&&&& 1 & 1 & 1 &&&&& \\
&&&&&&&&&&&&& 1 & 1 & 1 &&&& \\
&&&&&&&&&&&&&& 1 & 1 & 1 &&& \\
&&&&&&&&&&&&&&& 1 & 1 & 1 && \\
&&&&&&&&&&&&&&&& 1 & 1 & 1 & \\
&&&&&&&&&&&&&&&&& 1 & 1 & 1
\end{bmatrix}$$

Forward substitution using forward elimination results in extremely large interval growth. Consider the following two right hand side interval vectors:

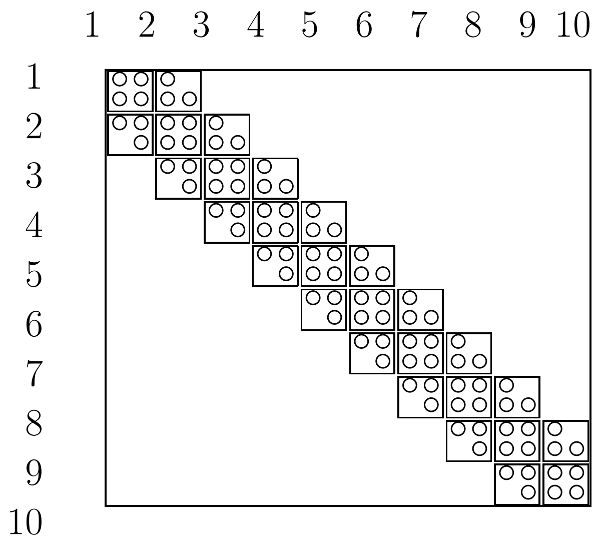
$$e^1 = \begin{bmatrix} [-1,1] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \\ [0,0] \end{bmatrix} \qquad b = \begin{bmatrix} [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \\ [-1,1] \end{bmatrix}$$

The respective y interval vectors from the forward substitution are:

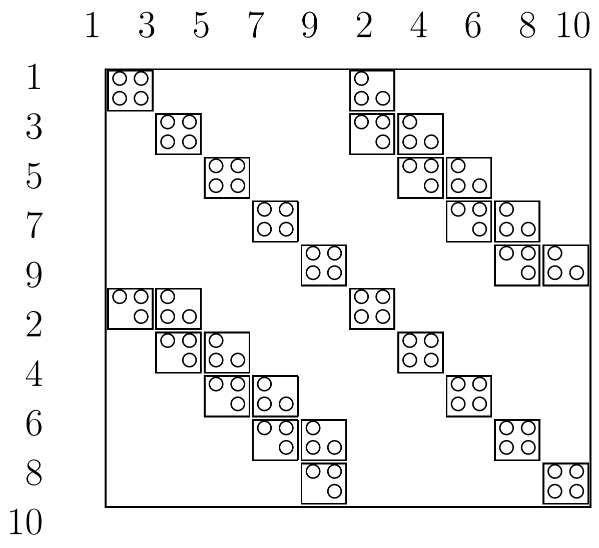
$$L\backslash e^1 = \begin{bmatrix} [-1,1] \\ [-1,1] \\ [-2,2] \\ [-3,3] \\ [-5,5] \\ [-8,8] \\ [-13,13] \\ [-21,21] \\ [-34,34] \\ [-55,55] \\ [-89,89] \\ [-144,144] \\ [-233,233] \\ [-377,377] \\ [-610,610] \\ [-987,987] \\ [-1597,1597] \\ [-2584,2584] \\ [-4181,4181] \\ [-6765,6765] \end{bmatrix} \quad \text{and} \quad L\backslash b = \begin{bmatrix} [-1,1] \\ [-2,2] \\ [-4,4] \\ [-7,7] \\ [-12,12] \\ [-20,20] \\ [-33,33] \\ [-54,54] \\ [-88,88] \\ [-143,143] \\ [-232,232] \\ [-376,376] \\ [-609,609] \\ [-986,986] \\ [-1596,1596] \\ [-2583,2583] \\ [-4180,4180] \\ [-6764,6764] \\ [-10945,10945] \\ [-17710,17710] \end{bmatrix}$$

For the singleton interval vector, the interval radius grows as the Fibonacci numbers. Thus, ordinary forward and back substitution result in considerable overestimation of the solution. The elimination tree for this matrix consists of a single unbranched structure with 20 levels. Thus, the computation of y_{20} depends directly or indirectly on all previous 19 entries of y.

Consider, now, a permutation of the original matrix A. The permutation vector is generated as follows. First, group all entries of the original banded matrix of bandwidth 2 as 2×2 blocks. (In general, the block dimension is $\beta \times \beta$.) The structure of the blocked matrix becomes:



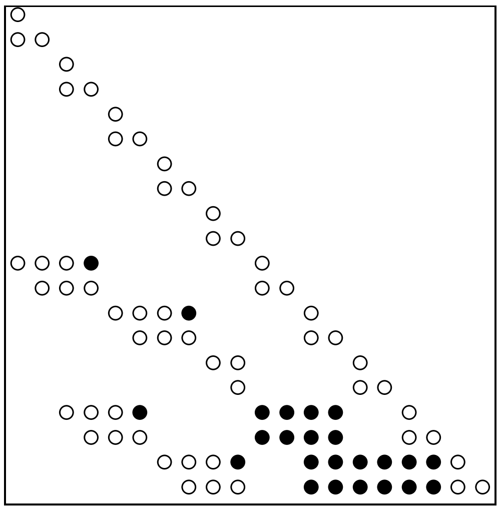
We now apply an odd-even permutation recursively to the blocks of this matrix. We illustrate the process by first considering the non-recursive odd-even permutation of this matrix:



That is, number odd blocks ahead of even blocks and add fills. The remaining even blocks form a new tridiagonal matrix of half the original size. Order once again according to the odd-even permutation, add fills, and so on. The permutation vector generated according to this recursive procedure is:

$$(1\ 3\ 5\ 7\ 9) \quad (2\ 6\ 10) \quad (4\ 8)$$

This permutation is applied to the blocked matrix. The structure of the resulting matrix factor $\tilde{L}$ after application of this permutation to the blocked matrix, followed by a subsequent "unblocking" and addition of fills is:



Performing a forward substitution for this matrix results in a significant reduction in the growth of the intervals for both cases under consideration:

$$\tilde{L}\backslash e^1 = \begin{bmatrix} [-1.0, 1.0] \\ [-1.0, 1.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [-2.0, 2.0] \\ [-2.0, 2.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [0.0, 0.0] \\ [-3.0, 3.0] \\ [-3.8, 3.8] \\ [-5.6, 5.6] \\ [-7.8, 7.8] \end{bmatrix} \quad \text{and} \quad \tilde{L}\backslash b = \begin{bmatrix} [-1.0000, 1.0000] \\ [-2.0000, 2.0000] \\ [-1.0000, 1.0000] \\ [-1.6667, 1.6667] \\ [-1.0000, 1.0000] \\ [-1.6667, 1.6667] \\ [-1.0000, 1.0000] \\ [-1.6667, 1.6667] \\ [-1.0000, 1.0000] \\ [-1.6667, 1.6667] \\ [-5.0000, 5.0000] \\ [-6.5000, 6.5000] \\ [-3.6667, 3.6667] \\ [-4.4667, 4.4667] \\ [-2.6667, 2.6667] \\ [-4.0000, 4.0000] \\ [-15.1016, 15.1016] \\ [-20.4419, 20.4419] \\ [-37.4552, 37.4552] \\ [-51.8895, 51.8895] \end{bmatrix}$$

The key observation that leads to this result is that the height of the elimination tree (and thus the length of the data dependencies) has been reduced from 20 levels to 8. The elimination tree for the reordered (blocked) matrix is illustrated in Figure 3. Entries from even levels can be paired with entries from odd levels, and a shorter tree of height 4 based on "blocked entries" can be built.

The overall observation from this section can be stated succinctly: reordering of the matrix rows and columns with a view to reducing the height of the elimination tree can significantly reduce the overestimation of solutions when the matrix in question is not an M-matrix. Orderings that reduce the length of the elimination tree have been considered in connection with a variety of other applications: increase of parallelism [9, 10, 11], improvements in sparse vector methods [12], and improvements to partitioned inverse methods [13, 14].

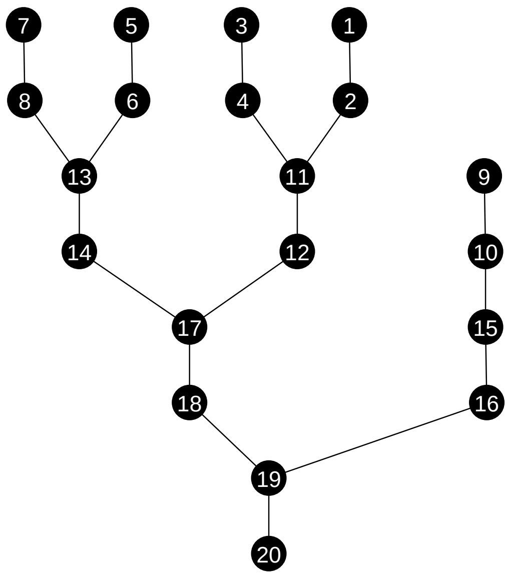


Figure 3: The elimination tree for the matrix after recursive blocked binary permutation. All entries appear in pairs.

3 Obtaining bounds on interval growth

The experiments from the last section can be studied more rigorously. The following theorems establish some of the properties of unordered and reordered forward substitution steps. *Important*: In all these theorems, it is assumed that $|\ell_{i,j}| \leq 1 \ \forall \ i \neq j$. An extension to the case where this assumption is relaxed is possible by blocking entries, but this is not pursued in this paper. Before proceeding with the theorems, it is useful to establish a preliminary result:

Lemma 1. *The roots of polynomials of the form* $\lambda^n - \lambda^{n-1} - \lambda^{n-2} - \cdots - \lambda - 1$ *are bounded by 2.*

Proof: This result can be proven by looking at the Gershgorin disks of the companion matrix. The form of the companion matrix is:

$$\begin{bmatrix} 1 & 1 & 1 & & 1 & 1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & \ldots & 1 & 0 & 0 \\ 0 & 0 & 0 & & 0 & 1 & 0 \end{bmatrix}$$

Gershgorin disks for this matrix establish the result immediately [15]. □

Theorem 2. *For a banded thin matrix A of bandwidth β and a singleton unit interval right hand side vector e^i, the growth in the radius of the interval in a forward elimination in otherwise exact arithmetic is bounded by 2^n.*

Proof: The worst case is when $i = 1$, so we only prove this case. Let $e_1 = [-\epsilon, \epsilon]$. Without loss of generality, the computation of elements of y proceeds as follows:

$$\begin{aligned} y_1 &= e_1 \\ y_2 &= -\ell_{21} y_1 \\ y_3 &= -\ell_{31} y_1 - \ell_{32} y_2 \\ &\vdots \\ y_k &= -\sum_{j=k-\beta}^{k-1} \ell_{kj} y_j \\ &\vdots \end{aligned}$$

Because the elements of L are bounded by one, the worst case is when $\ell_{ij} = 1$. Let ϵ_k denote a bound for $\text{rad}(y_k)$. The growth in the radii of the intervals y_k (taking into consideration that all operations are interval operations) is bounded by:

$$\begin{aligned} \epsilon_1 &= \epsilon \\ \epsilon_2 &= \epsilon_1 \\ \epsilon_3 &= \epsilon_1 + \epsilon_2 \\ &\vdots \\ \epsilon_k &= \sum_{j=k-\beta}^{k-1} \epsilon_j \\ &\vdots \end{aligned}$$

The generic recursion defines a difference equation. A closed-form solution to this difference equation can be obtained by finding the eigenvalues of the companion matrix for the difference equation. Of interest is the largest eigenvalue λ, since the solution is dominated by λ^k. This recursion has the same form as the polynomial in Lemma 1. Therefore, its companion matrix has the same form, and its largest eigenvalue is bounded by 2. This means that the dominant term in the solution is bounded by 2^n. For the special case of $\beta = 2$, the largest eigenvalue is only $(1+\sqrt{5})/2 \approx 1.618$, which is one of the known closed form solutions for the Fibonacci series. □

Theorem 2 is consistent with the knowledge that when partial pivoting is performed, the error growth is bounded by 2^n.

The case of a full interval vector b can also be considered as a generalization. Consider the following more general theorem:

Theorem 3. *For a banded thin matrix A of bandwidth β and a dense interval right hand side vector b, the growth of the interval in a forward elimination process is bounded by* $\max\big[\mathrm{rad}(b_i)\big]\ 2^n$.

Proof: The proof proceeds by first recognizing that if all values of b_i are replaced by ϵ, where ϵ is defined as the largest radius for any b_i, maximum growth in the radius will take place. Thus, the same interval radius ϵ can be used for all entries of b. The remainder of the proof is similar to the proof of the previous theorem, except that the recursion equation changes to:

$$\epsilon_k \;=\; \epsilon + \sum_{j=k-\beta}^{k-1} \epsilon_j .$$

□

Another generalization that may be considered is the removal of the requirement of the subsequent exact computation of all multiplications and additions. If it is assumed that every floating point multiplication may itself introduce a relative error also equal to ϵ (a gross overestimation of the error, which in most cases will be in reality about 2^{-t}, where t is the machine precision), then simple expressions for this case can be derived. The form of the recursion equation under these assumptions becomes:

$$\epsilon_k \;=\; \beta\,\epsilon + \sum_{j=k-\beta}^{k-1} \epsilon_j .$$

Since the results are qualitatively the same as the previous cases, these generalizations are not considered in detail here. Instead, we now seek to place bounds on *permuted* repeat solutions. To this effect, we first establish a result about general banded matrices:

Lemma 4. *For a banded matrix of bandwidth β, there exists an ordering that results in an elimination tree of height* $\left[\log_2(n/\beta)+1\right]\beta$.

Proof: If the matrix is "blocked" in blocks of dimension $\beta \times \beta$, the result is a tridiagonal blocked matrix of dimension bounded by n/β. It can be readily shown that a recursive odd-even permutation will reduce the height of the elimination tree in terms of these blocks to $\log_2(n/\beta)+1$. Since each block itself has height β, the total height for the tree is $\left[\log_2(n/\beta)+1\right]\beta$. □

Corollary 5. *For a tridiagonal matrix, the height of the elimination tree is bounded by* $\log_2 n + 1$.

Corollary 5 is actually a well known result [16, 17].

Corollary 6. *For a matrix of bandwidth 2, the height of the elimination tree is bounded by* $2\log_2(n/2)+1$.

During the factorization that follows the odd-even permutation, fill-in occurs in the renumbered matrix. This fill-in increases the density (the number of nonzeros) of subsequent rows in the matrix. The following lemma establishes a bound on the number of nonzero entries in the rows of the reordered matrix $\tilde{L}$.

Lemma 7. *The number of nonzero entries in each row after the factorization of a tridiagonal matrix of dimension n after a recursive binary odd-even permutation is bounded by* $2\left(\log_2 n + 1\right)$.

Proof: A simple odd-even permutation (before any fills are added) increases the lower triangular entry count in the last $n/2$ rows from 1 to 2. The factorization process adds one more fill. Thus, every step of the factorization for everyone of the $\log_2 n$ terms increases the lower tridiagonal count by 2. For a matrix of dimension n, this produces a bound on the number of lower triangular entries in its densest row equal to $2(\log_2 n + 1)$. □

A corollary of this lemma for the arbitrary bandwidth case is that the number of entries grows as $2\left[\log_2(n/\beta)+1\right]\beta$.

Theorem 8. *For a banded matrix of bandwidth β and a singleton interval right hand side vector e^i, the interval radius growth under the assumption of no error growth during the multiplications or additions and diagonal dominance by $\tilde{L}$ is bounded by $2n$.*

Proof: The proof follows directly from the assumption of diagonal dominance, a recursion on the number of terms in the summations according to Lemma 7, and the number of recursions needed to compute any of the terms of interest. In our case, the number of recursions m is $\log_2 n$, which dominates the computational order. The earlier bounds on recursion still hold: the growth in intervals is bounded by 2^m, where m is the number of recursion steps (the height of the tree minus one), rather than the dimension of the matrix. Thus, the expected growth in intervals is $2^{\log_2 n}$, or simply n. $\Box$

Corollary 9. *The growth of the radii of the inverse of A, when computed in this manner, remains of computational order n.*

The next section presents results applicable to dense right hand side vectors.

4 Singleton solvers

We start now on a new line of thought. The reason behind the overestimation during forward elimination is only partly due to the length of the path. A more fundamental reason is the direct and indirect repeated use of each entry of b at various places during the computation without recognizing that we are dealing with the effects from the same number. One way to eliminate this problem is to find the (exact) explicit inverse of the thin matrix A first, then simply perform the product $A^{-1}b$. However, even if the exact inverse of A could be found reliably, the method would not work well for sparse matrices because the inverse of A is full. This section considers an alternative based on the factors of A instead.

Consider first the case of a singleton interval vector b^i. This vector can be expressed as the product of a scalar interval α_i times the thin unit vector $\bar{e}^i$. Consider now the forward substitution involving b^i. The forward substitution can be obtained either from:

$$y^i = L \backslash b^i$$

or alternatively as a two-step process from:

$$\begin{aligned} \bar{y}^i &= L\backslash\bar{e}^i \\ y^i &= \alpha_i\,\bar{y}^i\ . \end{aligned}$$

The intermediate result $\bar{y}^i$ is a thin vector. If the right hand side vector is not a singleton interval, it is possible to express it as a sum of singleton unit vectors $\bar{e}^i$ and scalar intervals α_i:

$$b \;=\; \alpha_1\bar{e}^1 + \alpha_2\bar{e}^2 + \cdots + \alpha_n\bar{e}^n.$$

We then solve the problem first for each singleton unit vector $\bar{e}^i$ (thin), then scale all solutions by their intervals α_i, and finally aggregate the result using a single vector interval sum. That is, we solve:

$$\bar{y}^i \;=\; L\backslash\bar{e}^i$$

for the thin vector $\bar{y}^i$, and then construct the result of the forward substitution from:

$$\bar{y} \;=\; \alpha_1\bar{y}^1 + \alpha_2\bar{y}^2 + \cdots + \alpha_n\bar{y}^n\ .$$

Definition 7. *The process of applying the singleton method for forward substitution to an interval vector b is designated as $L^G b$.*

Application of this method to the solution of our test problem in its original banded order results in the following:

$$L^G e^1 = \begin{bmatrix} [-1,1] \\ [-1,1] \\ [0,0] \\ [-1,1] \\ [-1,1] \\ [0,0] \\ [-1,1] \\ [-1,1] \\ [0,0] \\ [-1,1] \\ [-1,1] \\ [0,0] \\ [-1,1] \\ [-1,1] \\ [0,0] \\ [-1,1] \\ [-1,1] \\ [0,0] \\ [-1,1] \\ [-1,1] \end{bmatrix} \quad \text{and} \quad L^G b = \begin{bmatrix} [-1,1] \\ [-2,2] \\ [-2,2] \\ [-3,3] \\ [-4,4] \\ [-4,4] \\ [-5,5] \\ [-6,6] \\ [-6,6] \\ [-7,7] \\ [-8,8] \\ [-8,8] \\ [-9,9] \\ [-10,10] \\ [-10,10] \\ [-11,11] \\ [-12,12] \\ [-12,12] \\ [-13,13] \\ [-14,14] \end{bmatrix}$$

The back substitution step can be performed likewise.

Definition 8. *The process of applying the singleton method for forward substitution, performing a diagonal scaling, and applying the singleton back substitution method is designated as $A^G b$. Exact arithmetic is assumed for all thin matrix operations.*

The result of back substitution is:

$$A^G e^1 = \begin{bmatrix} [-14,14] \\ [-7,7] \\ [-6,6] \\ [-12,12] \\ [-6,6] \\ [-5,5] \\ [-10,10] \\ [-5,5] \\ [-4,4] \\ [-8,8] \\ [-4,4] \\ [-3,3] \\ [-6,6] \\ [-3,3] \\ [-2,2] \\ [-4,4] \\ [-2,2] \\ [-1,1] \\ [-2,2] \\ [-1,1] \end{bmatrix} \quad \text{and} \quad A^G b = \begin{bmatrix} [-105,105] \\ [-98,98] \\ [-90,90] \\ [-102,102] \\ [-94,94] \\ [-85,85] \\ [-95,95] \\ [-86,86] \\ [-76,76] \\ [-84,84] \\ [-74,74] \\ [-63,63] \\ [-69,69] \\ [-58,58] \\ [-46,46] \\ [-50,50] \\ [-38,38] \\ [-25,25] \\ [-27,27] \\ [-14,14] \end{bmatrix}$$

These vectors are the hulls of the respective solutions. This result follows directly from the observation that $A^G \bar{e}^i$ is simply the ith column of the inverse of A.

5 Singleton solvers with path length reduction

The reordering method of Section 2 produces interval hull solutions for general thin matrices and singleton right hand sides and it is capable of operating in finite-precision arithmetic. The singleton method of Section 4 is capable of handling dense right hand side vectors but requires exact arithmetic. For general non-M-matrices the precise computation of the solution to the thin problem $A\tilde{x}^i = \tilde{e}^i$ is not always possible in sufficient precision. For non-M-matrices, the error in this solution may itself grow as 2^n as a result of numerical errors in L and U.

Consider now the case of a thin but general matrix A and an interval right hand side. It is our contention that the use of the singleton solver method *in conjunction with* the recursive permutation method holds the greatest promise for extreme robustness. Furthermore, parallelism of computations can be exploited in this case.

6 Conclusions

The case made in this paper has significant implications upon much of what is cherished about numerical error control during the direct solution of linear equations by Gauss-type methods. This paper says that for certain classes of problems, pivoting intended to reduce the length of the path of the computational sequences can be most effective, even at the expense of fill-in or additional computation. These types of orderings are, incidentally, also quite effective for parallel processing. Thus, a technique that is useful for parallel computation proves also useful in reducing interval overestimation. This has been the central theme of this paper.

Another key observation of this paper is the singleton solver method. This method is capable of determining the hull of the solution for thin matrices and interval right hand sides provided the underlying "thin" solution is error-free. The advantage of this method is that sparsity is preserved, and yet the hull is obtained. The combined use of the singleton solver method along with the use of tree-shortening ordering is seen as the most robust method for linear interval solvers.

The paper has illustrated some experiments graphically. The experiments and graphic results illustrated in this paper are reproducible using an interval version of the Sparse Matrix Manipulation System [18]. This software package has been made available via netlib.

Acknowledgements

The authors thank William Sethares, Hasan Dãg, and Greg Coxson for their comments during the preparation of this paper.

References

[1] Hansen, E. R. *Bounding the solution of interval linear equations.* SIAM Journal of Numerical Analysis **29** (5) (October 1992), pp. 1493–1503.

[2] Rohn, J. *Cheap and tight bounds: the recent result by E. Hansen.* Poster session presentation at the Conference on Numerical Analysis with Automatic Result Verification, February 1993.

[3] Liu, Joseph W. H. *A tree model for sparse symmetric indefinite matrix factorization.* SIAM Journal of Matrix Analysis and Applications **9** (1) (January 1988), pp. 27–39.

[4] Duff, I. S., Erisman, A., and Reid, J. K. *Direct methods for sparse matrices.* Oxford Science Publishers, Clarendon Press, 1986.

[5] Tinney, W. F. and Walker, J. W. *Direct solutions of sparse network equations by optimally ordered triangular factorization.* Proceedings of the IEEE **55** (11) (November 1967), pp. 1801–1809.

[6] Liu, Joseph W. H. *Modification of the minimum degree algorithm by multiple elimination.* ACM Transactions on Mathematical Software **11** (1985), pp. 141–153.

[7] Gibbs, N. E., Poole, W. G., and Stockmeyer, P. K. *An algorithm for reducing the bandwidth and profile of a sparse matrix.* SIAM Journal of Numerical Analysis **13** (1976), pp. 236–250.

[8] Neumeier, A. *Interval methods for systems of equations.* Cambridge University Press, 1990.

[9] Duff, Iain S. *Parallel implementation of multifrontal schemes.* Parallel Computation **3** (1986), pp. 193–204.

[10] Gilbert, J. R. and Schreiber, R. *Highly parallel sparse Cholesky factorization.* SIAM Journal on Scientific and Statistical Computing **13** (5) (September 1992), pp. 1151–1172.

[11] Liu, Joseph W. H. *Reordering sparse matrices for parallel elimination.* Parallel Computing **11** (1) (1989), pp. 73–91.

[12] Gómez, A. and Franquelo, L. G. *An efficient ordering algorithm to improve sparse vector methods.* IEEE Transactions on Power Systems **3** (4) (November 1988), pp. 1538–1544.

[13] Alvarado, F. L. and Schreiber, R. *Optimal parallel solution of sparse triangular systems.* SIAM Journal on Scientific and Statistical Computing (March 1993), pp. 446–460.

[14] Pothen, A. and Alvarado, F. L. *A fast reordering algorithm for parallel sparse triangular solution.* SIAM Journal on Scientific and Statistical Computing (March 1992), pp. 645–653.

[15] Stoer, J. and Burlirsch, R. *Introduction to numerical analysis.* Springer-Verlag, 1980.

[16] Alvarado, F. L. *Parallel solution of transient problems by trapezoidal integration.* IEEE Transactions on Power Apparatus and Systems, PAS–**98** (3) (May/June 1979), pp. 1080–1090.

[17] Stone, H. S. *An efficient parallel algorithm for the solution of a tridiagonal system of equations.* Journal of the Association for Computing Machinery **20** (1) (January 1973), pp. 183–208.

[18] Alvarado, F. L. *Manipulation and visualization of sparse matrices.* ORSA (Operations Research Society of America) Journal on Computing **2** (2) (Spring 1990), pp. 186–207.

Dept. of ECE
University of Wisconsin
1425 Johnson Dr.
Madison WI 53706–1607
USA