On solving unsymmetric tridiagonal systems without interchanges

Jennifer Erway,[*] Roummel F. Marcia,[†] and Joseph Tyson,[*]

Abstract— **It has been shown recently that a nonsingular tridiagonal linear system of the form $Tx = b$ can be solved in a backward-stable manner using the LBM$^{\mathbf{T}}$ decomposition, i.e., $T = LBM^T$, where L and M are unit lower triangular matrices and B is block diagonal with 1×1 and 2×2 blocks. In this paper, we demonstrate the robustness of two algorithms that compute a backward-stable LBM$^{\mathbf{T}}$ decomposition using a wide range of well-conditioned and ill-conditioned linear systems. Numerical results suggest that these algorithms are comparable to Gaussian elimination with partial pivoting (GEPP). However, unlike GEPP, these algorithms do not require row interchanges, and thus, may be used in applications where row interchanges are not possible. In addition, substantial computational savings can be achieved by carefully managing the nonzero elements of the factors L, B, and M.**

Keywords: linear algebra, tridiagonal systems, diagonal pivoting, Gaussian elimination

1 Introduction

A nonsingular tridiagonal linear system of the form

$$Tx = b, \tag{1}$$

where $T \in \mathbb{R}^{n \times n}$ and x and $b \in \mathbb{R}^n$, is often solved using matrix factorizations. If T is symmetric and positive definite, then the Cholesky decomposition or the LDL$^{\mathrm{T}}$ factorization, where L is a lower triangular matrix and D is a diagonal matrix, can be used to solve (1). If T is symmetric but indefinite, then with row and/or column permutations, the LBL$^{\mathrm{T}}$ factorization can be used, where B is block diagonal with either 1×1 or 2×2 blocks (see e.g., [2, 3, 4, 5, 6, 9]). Finally, if T is unsymmetric, then (1) can be solved using Gaussian elimination with full pivoting or with partial pivoting (GEPP). Recent work by the authors [8] shows that (1) can be solved in a backward-stable manner using the LBM$^{\mathrm{T}}$ decomposition of T, i.e., $T = LBM^T$, where B is a block diagonal matrix with either 1×1 or 2×2 blocks and L and M are unit-lower tridiagonal matrices.

In this paper, we examine two algorithms presented in [8] for forming a backward-stable LBM$^{\mathrm{T}}$ decomposition of a nonsingular tridiagonal matrix T. We demonstrate that the resulting L, B, and M factors from either algorithm can be used to solve the linear system $Tx = b$ with accuracy comparable to Gaussian eliminiation with partial pivoting (GEPP). However, unlike GEPP, neither algorithm requires row interchanges, making them particularly useful in look-ahead Lanczos methods [11] and composite-step bi-conjugate gradient methods [1] for solving unsymmetric linear systems.

2 Diagonal pivoting

Let $T \in \mathbb{R}^{n \times n}$ denote the unsymmetric nonsingular tridiagonal matrix

$$T = \begin{bmatrix} \alpha_1 & \gamma_2 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & \gamma_3 & \ddots & \vdots \\ 0 & \beta_3 & \alpha_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \gamma_n \\ 0 & \cdots & 0 & \beta_n & \alpha_n \end{bmatrix}. \tag{2}$$

We partition T in the following manner:

$$T = \begin{array}{cc} & \begin{array}{cc} d & n-d \end{array} \\ \begin{array}{c} d \\ n-d \end{array} & \begin{bmatrix} B_1 & T_{12}^T \\ T_{21} & T_{22} \end{bmatrix} \end{array}$$

The computation of the LBM$^{\mathrm{T}}$ factorization, where L and M are unit lower triangular and B is block diagonal with 1×1 and 2×2 blocks, involves choosing the dimension ($d = 1$ or 2) of the pivot B_1 at each stage:

$$T = \begin{bmatrix} I_d & 0 \\ T_{21}B_1^{-1} & I_{n-d} \end{bmatrix} \begin{bmatrix} B_1 & 0 \\ 0 & S_d \end{bmatrix} \begin{bmatrix} I_d & B_1^{-1}T_{12}^T \\ 0 & I_{n-d} \end{bmatrix}, \tag{3}$$

where $S_d = T_{22} - T_{21}B_1^{-1}T_{12}^T \in \mathbb{R}^{(n-d)\times(n-d)}$. It can be shown that an invertible B_1 exists for some d and that the Schur complement S_d is tridiagonal so that the

[*] Department of Mathematics, Wake Forest University, Winston-Salem, NC 27106, USA. J. Erway and J. Tyson are supported by the National Science Foundation grant DMS-0811106

[†] School of Natural Sciences, 5200 North Lake Road, University of California, Merced, Merced, CA 95343. Tel: 209-228-4874. Fax: 209-228-4060. Email: rmarcia@ucmerced.edu. R. F. Marcia is supported by the National Science Foundation grant DMS-0811062

factorization can be defined recursively. Specifically, for $d = 1$,

$$S_1 = T_{22} - \frac{\beta_2\gamma_2}{\alpha_1} e_1^{(n-1)} e_1^{(n-1)T},$$

where $e_1^{(n-1)}$ is the first column of the $(n-1) \times (n-1)$ identity matrix. For $d = 2$,

$$S_2 = T_{22} - \left(\frac{\alpha_1\beta_3\gamma_3}{\varDelta} \right) e_1^{(n-2)} e_1^{(n-2)T}.$$

where

$$\varDelta = \alpha_1\alpha_2 - \beta_2\gamma_2$$

is the determinant of the 2×2 pivot B_1 and $e_1^{(n-2)}$ is the first column of the $(n-2) \times (n-2)$ identity matrix. In both cases S_d and T_{22} differ only in the $(1,1)$ entry; thus, the LBM^T factorization can be recursively defined to obtain the matrices L, B, and M.

The algorithms in [8] can be completely described by looking at the first stage of the factorization.

2.1 Algorithm I

Let

$$L_1 = T_{21}B_1^{-1} \quad \text{and} \quad M_1^T = B_1^{-1}T_{12}^T$$

in (3). If a 1×1 pivot is used, then the $(1,1)$ elements of L_1 and M_1 are

$$(L_1)_{1,1} = \frac{\beta_2}{\alpha_1} \quad \text{and} \quad (M_1)_{1,1} = \frac{\gamma_2}{\alpha_1}. \tag{4}$$

If a 2×2 pivot is used, then the $(1,1)$ and $(1,2)$ elements of L_1 and M_1 are

$$\begin{array}{rclrcl} (L_1)_{1,1} & = & -\dfrac{\beta_2\beta_3}{\varDelta}, & (M_1)_{1,1} & = & -\dfrac{\gamma_2\gamma_3}{\varDelta}, \\ (L_1)_{1,2} & = & \dfrac{\alpha_1\beta_3}{\varDelta}, & (M_1)_{1,2} & = & \dfrac{\alpha_1\gamma_3}{\varDelta}. \end{array} \tag{5}$$

With elements of L_1 and M_1 for a 2×2 pivot scaled by the constant κ (described below) from the Bunch pivoting strategy [2], Algorithm I chooses a 1×1 pivot if the both of the entries in (4) is smaller than the largest entry in (5) in magnitude, i.e.,

$$\begin{aligned} \max \left\{ \frac{|\beta_2|}{|\alpha_1|}, \frac{|\gamma_2|}{|\alpha_1|} \right\} & \\ \leq \max \kappa \left\{ \frac{|\beta_2\beta_3|}{|\varDelta|}, \frac{|\alpha_1\beta_3|}{|\varDelta|}, \frac{|\gamma_2\gamma_3|}{|\varDelta|}, \frac{|\alpha_1\gamma_3|}{|\varDelta|} \right\}, & \end{aligned} \tag{6}$$

and a 2×2 pivot is chosen otherwise. In other words, Algorithm I chooses pivot sizes based on whichever leads to smaller entries (in magintude) in the computed factors L and M. In addition, we impose the criterion that if

$$|\alpha_1\alpha_2| \geq \kappa|\beta_2\gamma_2|, \tag{7}$$

a 1×1 pivot is chosen. This additional criterion guarantees that if T is positive definite, the LBM^T factorization reduces to the LDM^T factorization. The choice of pivot size in the first iteration is described as follows:

Algorithm I. *This algorithm determines the size of the pivot for the first stage of the LBM^T factorization applied to a tridiagonal matrix $T \in I\!R^{n \times n}$.*

```
$\kappa = (\sqrt{5} - 1)/2 \approx 0.62$
$\varDelta = \alpha_1\alpha_2 - \beta_2\gamma_2$
if $|\alpha_1\alpha_2| \geq \kappa|\beta_2\gamma_2|$ or $|\varDelta| \max\{|\beta_2|, |\gamma_2|\} \leq$
        $\kappa|\alpha_1| \max\{|\beta_2\beta_3|, |\alpha_1\beta_3|, |\gamma_2\gamma_3|, |\alpha_1, \gamma_3|\}$
    $d_I = 1$
else
    $d_I = 2$
end
```

The choice of κ, which is a root of the equation $\kappa^2 + \kappa - 1 = 0$, balances the element growth in the Schur complement for both pivot sizes by equating the maximal element growth (see [2] for details). A recursive application of Algorithm I yields a factorization $T = LBM^T$, where L and M are unit lower triangular and B is block diagonal. (In fact, one can show that L and M are such that $L_{i,j} = M_{i,j} = 0$ for $i - j > 2$.) Intuitively, Algorithm I chooses a 1×1 pivot if α_1 is sufficiently large relative to the determinant of the 2×2 pivot, i.e., a 1×1 pivot is chosen if a 2×2 pivot is relatively closer to being singular than α_1 is to zero.

2.2 Algorithm II

We also proposed an alternative pivoting strategy that is based on the strategy of Bunch and Kaufman for symmetric *tridiagonal* matrices (Section 4.2, [3]). (Note that this strategy is different from their well-known symmetric factorization using partial pivoting.) This second strategy chooses a 1×1 pivot if the $(1,1)$ diagonal entry is sufficiently large relative to the off-diagonals, i.e.,

$$|\alpha_1|\sigma_1 \geq \kappa|\beta_2\gamma_2|,$$

where

$$\sigma_1 = \max\{|\alpha_2|, |\gamma_2|, |\beta_2|, |\gamma_3|, |\beta_3|\},$$

and κ is as in Sec. 2.1. This algorithm can be completely described in the first step:

Algorithm II. *This alternative algorithm determines the size of the pivot for the first stage of the LBM^T factorization applied to a tridiagonal matrix $T \in I\!R^{n \times n}$.*

```
$\kappa = (\sqrt{5} - 1)/2 \approx 0.62$
$\sigma_1 = \max\{|\alpha_2|, |\gamma_2|, |\beta_2|, |\gamma_3|, |\beta_3|\}$
if $|\alpha_1|\sigma_1 \geq \kappa|\beta_2\gamma_2|$,
    $d_{II} = 1$
```

```
else
    $d_{II} = 2$
end
```

It can be shown that Algorithms I and II are very related in that whenever Algorithm I chooses a 1×1 pivot, so does Algorithm II. In fact, the only difference between the two algorithms is that on occasion, Algorithm I may choose a 2 × 2 pivot while Algorithm II chooses two consecutive 1 × 1 pivots.

2.3 Backward stability result

These two pivoting strategies imply a backward stability result which demonstrates that (a) the difference between computed factorization $\widehat{L}\widehat{B}\widehat{M}^T$ and the original tridiagonal matrix T is small (i.e., it is of order machine precision), and (b) the computed solution $\widehat{x}$ is the exact solution to a nearby problem. More formally, we state the following theorem:

Theorem 1. *Assume the LBM^T factorization of the unsymmetric tridiagonal matrix $T \in \mathbb{R}^{n\times n}$ obtained using the pivoting strategy of Algorithm I or II yields the computed factorization $T \approx \widehat{L}\widehat{B}\widehat{M}^T$, and let $\widehat{x}$ be the computed solution to $Tx = b$ obtained using the factorization. Assume that all linear systems $Ey = f$ involving 2×2 pivots E are solved using the explicit inverse. Then*

$$T - \widehat{L}\widehat{B}\widehat{M}^T = \Delta T_1, \quad \textit{and} \quad (T + \Delta T_2)\widehat{x} = b,$$

where

$$\|\Delta T_i\|_{\max} \le cu\|T\|_{\max} + O(u^2), \quad i = 1, 2,$$

where c is a constant and u is the machine precision.

Proof. *See [8].*

Pictorially, the backward stability result of Theorem 1 is represented in Fig. 1.

In addition to computing a backward-stable LBM^T decomposition of T, these algorithms have minimal storage requirements. Specifically, T is tridiagonal, and thus, can stored using three vectors. Moreover, updating the Schur complement requires updating only one nonzero component of T. The matrices L and M are unit-lower triangular with $L_{i,j} = M_{i,j} = 0$ for $i - j > 2$; thus, their entries can be stored in two vectors each. Finally, B is block-diagonal with 1 × 1 or 2 × 2 blocks, requiring only three vectors for storage.

3 Numerical experiments

Numerical tests were run using Matlab implementations of Algorithm I, Algorithm II, GEPP, and the Matlab backslash command ("\"). (Specifically, Algorithm I and Algorithm II were embedded in a code that used one forward substitution to solve with L, one back substitution to solve with M^T, and a solve with the block-diagonal factor B). We compared the performance of each code on 16 types of nonsingular tridiagonal linear systems of the form $Tx = b$. The test set of system matrices contains a wide range of difficulty. Many ill-conditioned matrices were chosen as part of the test set in order to compare algorithm robustness. (Ill-conditioned matrices are often challenging for matrix-factorization algorithms.) The test set of system matrices was taken from recent literature (specifically, [7] and [10]) on estimating condition numbers of ill-conditioned matrices — a task that can become increasingly difficult the more ill-conditioned the matrix is.

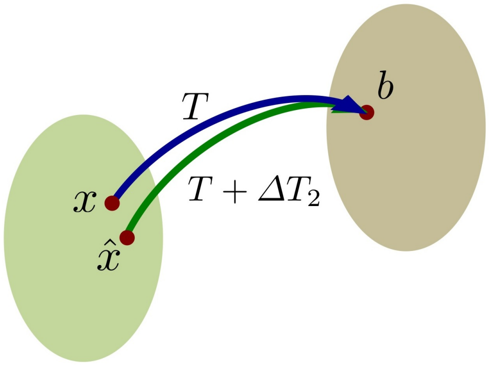

Figure 1: Theorem 1 implies not only that the product of the computed factors $\widehat{L}\widehat{B}\widehat{M}^T$ is very close to the original matrix T, but that the computed solution $\hat{x}$ is the exact solution to a nearby problem, i.e., the components of ΔT_2 are very small.

Table 1 contains a description of each tridiagonal matrix type in the test set. The first ten matrix types listed are based on test cases in [10]. Types 11-14 correspond to test cases found in [7] not found in [10], i.e., we eliminated redundant types. Finally, we include two additional tridiagonal matrices (Types 15-16) that can be generated using the Matlab command `gallery`. For our tests, T was chosen to be a 100 × 100 matrix. The elements of b were chosen from a uniform distribution on $[-1, 1]$.

One system matrix was generated from each matrix type, and together with a vector b, the same linear system of the form $Tx = b$ was solved by each algorithm. Table 2 shows the relative errors associated with each method. Columns 2-5 of the table contain the relative error

$$\frac{\|T\hat{x} - b\|_2}{\|b\|_2},$$

where $\hat{x}$ is the computed solution by each solver. The final column gives the condition number of the system matrix T. Each row in the table corresponds to one linear system from one type; that is, row i contains the relative errors associated with each solver on a linear system whose system matrix was from type i in Table 1.

Table 2 suggests that all four algorithms are comparable on a wide variety of linear systems. When the system matrix is well-conditioned (e.g., Types 1, 4, 6, 8, 10, 13, 14, 16), the algorithms have relative errors that are close to machine precision and that are comparable to one another. On the other hand, more significant differences occur when the system matrix is very ill-conditioned. For Type 5, the Algorithms I and II perform particularly poorly, and the solutions are offered by GEPP and the MATLAB backslash command are also very poor, with an error near 10^{16}. In fact, due to the ill-conditioning of the system matrix, all methods failed to solve the linear system. Finally, for Types 2, 3, 7, 9, 11, 12, and 15, the system matrices are ill-conditioned, however all the methods obtain relative errors within an order of magnitude of each other.

4 Conclusions and future work

Algorithms I and II, proposed in [8], were shown to compute a backward-stable LBM^T decomposition of any nonsingular tridiagonal matrix. Numerical results on a wide range of linear systems suggest that the performance of Algorithms I and II are comparable to GEPP and the MATLAB backslash command.

Future work will focus on embedding Algorithm I and II into methods such as the look-ahead Lanczos methods and composite-step bi-conjugate gradient methods for solving unsymmetric linear systems.

References

[1] R. E. BANK AND T. F. CHAN, *A composite step bi-conjugate gradient algorithm for nonsymmetric linear systems*, Numer. Algorithms, 7, pp. 1–16, 1994.

[2] J. R. BUNCH, *Partial pivoting strategies for symmetric matrices*, SIAM J. Numer. Anal., 11, pp. 521–528, 1974.

[3] J. R. BUNCH AND L. KAUFMAN, *Some stable methods for calculating inertia and solving symmetric linear systems*, Math. Comp., 31, pp. 163–179, 1977.

[4] J. R. BUNCH AND R. F. MARCIA, *A pivoting strategy for symmetric tridiagonal matrices*, Numer. Linear Algebra Appl., 12, pp. 911–922, 2005.

[5] ——, *A simplified pivoting strategy for symmetric tridiagonal matrices*, Numer. Linear Algebra Appl., 13, pp. 865–867, 2006.

[6] J. R. BUNCH AND B. N. PARLETT, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., 8, pp. 639–655, 1971.

[7] I. S. DHILLON, *Reliable computation of the condition number of a tridiagonal matrix in $O(n)$ time*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 776-796.

[8] J. B. ERWAY AND R. F. MARCIA, *A backward stability analysis of diagonal pivoting methods for solving unsymmetric tridiagonal systems without interchanges.* Accepted for publication in Numerical Linear Algebra with Applications.

[9] H. R. FANG AND D. P. O'LEARY, *Stable factorizations of symmetric tridiagonal and triadic matrices*, SIAM Journal on Matrix Analysis and Applications, 28, pp. 576–595, 2006.

[10] G. I. HARGREAVES, *Computing the Condition Number of Tridiagonal and Diagonal-Plus-Semiseparable Matrices in Linear Time*, Numerical Analysis Report 447, Manchester Centre for Computational Mathematics, Manchester, 2004.

[11] B. N. PARLETT, D. R. TAYLOR, AND Z. A. LIU, *A look-ahead Lánczos algorithm for unsymmetric matrices*, Math. Comput., 44, pp. 105–124, 1985.

Matrix type	Description
1	Randomly generated matrix from a uniform distribution on $[-1, 1]$.
2	`gallery('randsvd',n,1e15,2,1,1)` – Randomly generated matrix with condition number `1e15` and one small singular value.
3	`gallery('randsvd',n,1e15,3,1,1)` – Randomly generated matrix with condition number `1e15` and geometrically distributed singular values.
4	Toeplitz matrix T with $T_{ii} = 10^8$ for $i = 1 \dots n$, and the elements T_{ij} for $i \neq j$ are randomly generated from a uniform distribution on $[-1, 1]$.
5	Toeplitz matrix T with $T_{ii} = 10^{-8}$ for $i = 1 \dots n$, and the elements T_{ij} for $i \neq j$ are randomly generated from a uniform distribution on $[-1, 1]$.
6	`gallery('lesp',n)` – Matrix with sensitive eigenvalues that are smoothly distributed in the approximate interval $[-2n - 3.5, -4.5]$.
7	`gallery('dorr',n,1e-4)` – Ill-conditioned, diagonally dominant matrix.
8	Randomly generated matrix from a uniform distribution on $[-1, 1]$; the 50th subdiagonal element is then multiplied by 10^{-50}.
9	Matrix whose elements are all generated from a uniform normal distribution on $[-1, 1]$; the lower diagonal are then multiplied by 10^{-50}.
10	Main diagonal elements generated randomly from a uniform distribution on $[-1, 1]$; off-diagonal elements each chosen with 50% probability as either zero or generated randomly from a uniform distribution on $[-1, 1]$.
11	`gallery('randsvd',n,1e15,1,1,1)` – Randomly generated matrix with condition number `1e15` and one large singular value.
12	`gallery('randsvd',n,1e15,4,1,1)` – Randomly generated matrix with condition number `1e15` and arithmetically distributed singular values.
13	Toeplitz matrix T with $T_{ii} = 64$ for $i = 1 \dots n$, and the elements T_{ij} for $i \neq j$ are randomly generated from a uniform distribution on $[-1, 1]$.
14	Toeplitz matrix T with $T_{ii} = 0$ for $i = 1 \dots n$, and the elements of T_{ij} for $i \neq j$ are randomly generated from a uniform distribution on $[-1, 1]$.
15	`gallery('clement',n,0)` – Main diagonal elements are zero; eigenvalues include plus and minus the numbers $n - 1, n - 3, n - 5, \dots 1$.
16	`inv(gallery('kms',n,0.5))` – Inverse of a Kac-Murdock-Szego Toeplitz ($A_{ij} = (0.5)^{\|i-j\|}$).

Table 1: Tridiagonal matrix types used in the numerical experiments

Matrix Type	Algorithm I	Algorithm II	GEPP	MATLAB Backslash	Condition Number
1	3.358363e-15	2.759199e-15	1.815859e-15	1.837180e-15	6.329660e+02
2	1.517113e-03	1.517113e-03	1.517113e-03	1.517113e-03	1.068959e+15
3	1.422747e-05	2.261760e-05	8.733909e-06	6.084902e-06	1.002161e+15
4	9.931598e-17	9.931598e-17	8.106222e-17	8.106222e-17	1.000000e+00
5	3.932643e+23	3.932643e+23	1.302276e+16	1.302276e+16	1.570243e+41
6	1.513163e-16	1.513163e-16	1.357171e-16	1.357171e-16	6.748520e+01
7	4.472943e+01	4.472943e+01	6.609908e+01	6.609908e+01	2.784168e+16
8	1.796253e-15	1.807151e-15	7.290777e-16	1.089269e-15	1.093088e+03
9	2.298240e-10	2.298240e-10	1.307689e-10	1.307689e-10	1.101082e+08
10	3.857912e-14	3.857912e-14	3.017280e-14	3.017259e-14	1.489926e+04
11	5.856221e-05	5.856221e-05	2.155033e-05	2.155033e-05	1.259243e+15
12	3.526433e-03	3.124242e-03	1.539025e-03	1.655561e-03	8.930231e+14
13	8.951786e-17	8.951786e-17	6.956100e-17	6.956100e-17	1.006886e+00
14	1.706070e-14	1.706081e-14	5.410519e-15	5.411247e-15	5.378205e+05
15	2.765054e-02	2.765054e-02	1.212209e-02	1.186110e-02	3.144575e+15
16	1.633022e-16	1.633022e-16	2.816457e-16	2.766310e-16	8.981329e+00

Table 2: Relative errors for four methods for solving the tridiagonal system $Tx = b$.