THE GROWTH-FACTOR BOUND FOR THE BUNCH-KAUFMAN FACTORIZATION IS TIGHT

ALEX DRUINSKY AND SIVAN TOLEDO

ABSTRACT. We show that the growth factor bound in the Bunch-Kaufman factorization method is essentially tight. The method factors a symmetric matrix A into $A = P^T L D L^T P$ where P is a permutation matrix, L is lower triangular, and D is block diagonal with 1-by-1 and 2-by-2 diagonal blocks. The method uses one of several partial pivoting rules that ensure bounded by possibly exponential growth in the elements of the reduced matrix and the factor D (growth in L is not bounded). We show that the exponential bound is essentially tight, thereby solving a question that has been open since 1977.

1. INTRODUCTION

In 1977 Bunch and Kaufman [3] discovered a beautiful method for factoring symmetric indefinite matrices. The method is now widely used for both dense and sparse matrices. The method is used in LAPACK to solve symmetric indefinite linear systems. Variants are used in all the sparse symmetric indefinite factorization codes. Bunch and Kaufman suggested several partial pivoting rules for their method. They showed that the element growth factor in the trailing submatrix is bounded by $(1+\alpha^{-1})^{n-1}$, where $\alpha \approx 0.6404$ or $\alpha \approx 0.525$ and n is the dimension of A (the value of α varies between variants of the pivoting rule). In practice, element growth is usually much milder; the situation for LU with partial pivoting is similar. Mild element growth ensures that the factorization is backward stable [5]; dramatic element growth usually leads to instability. Bunch and Kaufman tried to find examples that show that their growth bounds are tight in the worst-case sense, but only found examples for small matrix dimensions. The question of the tightness of these bounds was posed by Higham as an important research question [6, p. 229, problem 11.10]. For LU with partial pivoting, the tightness of the growth bound has been known for decades [10].

Date: Revised March 2011.

In this paper, we show that the upper bound on growth given by Bunch and Kaufman is essentially tight.

The Bunch-Kaufman algorithm computes the factorization $PAP^T = LDL^T$ of a symmetric (and possibly indefinite) matrix A. The matrix L is unit lower-triangular, D is block-diagonal with diagonal blocks of dimension 1 or 2, and P is a permutation matrix. In each step, the algorithm selects the order of the pivot s ($s = 1$ or 2) and the s rows and columns to be eliminated. The intersection of these rows and columns induces the pivot block. The rules for selecting the pivot form the heart of the algorithm and are described in the next sections. The algorithm reorders the selected rows and columns to the first s positions in the reduced matrix by means of a permutation matrix Π, and performs an elimination step to transform the matrix

$$\Pi A \Pi^T = \begin{matrix} \\ s \\ n-s \end{matrix} \begin{matrix} \begin{matrix} s & n-s \end{matrix} \\ \begin{bmatrix} E & C^T \\ C & B \end{bmatrix} \end{matrix}$$

into s new columns of L and D and the reduced matrix $B - CE^{-1}C^T$. The process then repeats, transforming the reduced matrix into a smaller reduced matrix until A is completely reduced.

Bunch and Kaufman described several pivoting strategies named Algorithm A through Algorithm F. In general, all the pivoting rules guarantee a bound on the growth in the reduced matrix, but the $(1+\alpha^{-1})^{n-1}$ bound is exponential in the dimension n of A. Algorithms E and F only apply to pentadiagonal matrices, and we do not analyze their growth here. We also do not analyze the growth in Algorithm C, which is not widely used (and appears harder to analyze).

The rest of the paper is organized as follows. Section 2 shows that the bound for the pivoting rule called Algorithm D is exactly tight. Section 3 shows that the growth bounds for Algorithms A and B are almost tight (they are tight up to a small constant factor). The examples that we use to show the tightness of the bound are ill-conditioned, but we also show well-conditioned examples that lead to exponential growth. Section 4 proves that the worst-case growth bound can almost be achieved in floating-point arithmetic (Sections 2 and 3 assume exact arithmetic). Section 5 demonstrates experimentally that the bounds are attained in LAPACK, a widely-used implementation of the Bunch-Kaufman method. We conclude the paper in Section 6 with remarks and open problems.

2. Growth in Algorithm D

We begin with Algorithm D, for which our example is the simplest. In this case, our example shows that Bunch and Kaufman's upper bound on growth is tight. The pivoting rule is as follows.

(1) Find the position r and the magnitude λ of the largest off-diagonal element in column 1.
(2) If $|A_{11}| \geq \alpha\lambda$ (where α is a constant between 0 and 1), then select A_{11} as the pivot and return; otherwise, continue to Step 3.
(3) Find the magnitude σ of the largest element in column r (including the diagonal element).
(4) If $|A_{11}|\,\sigma \geq \alpha\lambda^2$, then select A_{11} as the pivot and return; otherwise, continue to Step 5.
(5) Use the 2-by-2 pivot $\begin{bmatrix} A_{11} & A_{1r} \\ A_{r1} & A_{rr} \end{bmatrix}$.

The constant $\alpha \approx 0.525$ guarantees that the worst-case growth of two 1-by-1 steps equals the worst-case growth of one 2-by-2 step.

Theorem 1. *The worst-case growth factor in the Bunch-Kaufman factorization (Algorithm D) is* $\left(1 + \frac{1}{\alpha}\right)^{n-1}$*, where* n *is the dimension of the matrix.*

Proof. Consider the matrix

$$A = \begin{bmatrix} d_1 & & & & 1 \\ & d_2 & & & 1 \\ & & \ddots & & \vdots \\ & & & d_{n-1} & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix},$$

where

$$d_k = \frac{q}{1-q} q^{-k}$$
$$q = 1 + \frac{1}{\alpha}.$$

We prove by induction that the algorithm performs only 1-by-1 elimination steps and that the reduced matrix $A^{(k)}$ after k steps is

$$\begin{bmatrix} d_{k+1} & & & & 1 \\ & d_{k+2} & & & 1 \\ & & \ddots & & \vdots \\ & & & d_{n-1} & 1 \\ 1 & 1 & \cdots & 1 & q^k \end{bmatrix}.$$

For $k = 0$, the claim is true by the construction of A. Suppose that the claim holds for $k-1$ and consider the next elimination step. Column r will be the last column with $\lambda = 1$ and $\sigma = q^{k-1}$. The algorithm now tests whether $|A_{11}^{(k-1)}|$ is large enough (Step 2 above). If the test succeeds, the algorithm uses $A_{11}^{(k-1)}$ as a pivot. If the test fails, the algorithm tests whether $\left|A_{11}^{(k-1)}\right| \sigma \geq \alpha\lambda^2$. We claim that this test, if checked, always succeeds:

$$\begin{aligned}
\left|A_{11}^{(k-1)}\right| \sigma &= |d_k|\, q^{k-1} \\
&= -\frac{q}{1-q} q^{-k} \cdot q^{k-1} \\
&= -\frac{1}{1-q} \\
&= \alpha \\
&= \alpha\lambda^2 \, .
\end{aligned}$$

We have established that either the test in Step 2 succeeds or the test in Step 4 succeeds; either way, the algorithm uses $A_{11}^{(k-1)}$ as a pivot.

We now analyze the elimination step. The only matrix element that is modified in the reduced matrix is the last diagonal element. It is updated as follows:

$$\begin{aligned}
q^{k-1} - \frac{1}{d_k} &= q^{k-1} - \frac{1-q}{q} q^k \\
&= q^{k-1} - (1-q) q^{k-1} \\
&= q^k \, .
\end{aligned}$$

The theorem follows because the last diagonal element of the reduced matrix after $n-1$ steps is $q^{n-1} = \left(1 + \frac{1}{\alpha}\right)^{n-1}$. □

In the factorization of the matrix that we used in the last example, not only elements of the reduced matrix grow exponentially, but also elements of L grow exponentially. The elements in the last row of L are d_k^{-1}.

The reader might object that the only element that grows in this example is a diagonal element; our example for Algorithms A and B shows that offdiagonal elements can also grow at the same rate.

Another concern might be that as n grows, our matrix becomes more and more ill-conditioned. It is easy to see that this is indeed the case. Columns 1 through $n-1$ of A consist of a nonzero diagonal element and a 1 in the last row. The diagonal element converges exponentially to zero, so the columns become numerically dependent.

However, we can achieve exactly the same growth if we embed the $n/2$-by-$n/2$ example matrix A in a well-conditioned matrix of twice the dimension. Let $\epsilon > 0$ be a small number, and consider the matrix

$$\begin{bmatrix} A & (1-\epsilon)I \\ (1-\epsilon)I & 0 \end{bmatrix}.$$

Until the algorithm factors $n/2$ rows and columns, the row index r in Step 1 of the algorithm is always in the first block, because $1 - \epsilon < 1$; rows from the $(1-\epsilon)I$ block are never selected as pivots. Therefore, the algorithm always produces a factorization of A, so the growth in the last diagonal element in the first block is $\left(1 + \frac{1}{\alpha}\right)^{n/2-1}$. The entire n-by-n matrix is well-conditioned. This growth bound is only a square root of the upper bound, but it is still dramatic. This example can be adapted to odd dimensions.

Finally, it is interesting to ask whether the exponential growth can occur in floating-point arithmetic. Section 4 shows that this is indeed the case (the analysis that we show below is for Algorithm A, because it is implemented in LAPACK and is the subject of our numerical experiments).

3. Growth in Algorithms A and B

The pivoting rule in Algorithms A and B is a little different, in that it allows the algorithm to eliminate column r without eliminating column 1.

(1) Find the position r and the magnitude λ of the largest off-diagonal element in column 1.
(2) If $|A_{11}| \geq \alpha\lambda$ ($0 < \alpha < 1$ is defined below), then select A_{11} as the pivot and return; otherwise, continue to Step 3.
(3) Find the magnitude σ of the largest *off-diagonal* element in column r.
(4) If $|A_{11}|\,\sigma \geq \alpha\lambda^2$, then select A_{11} as the pivot and return.
(5) If $|A_{rr}| \geq \alpha\sigma$, then select A_{rr} as the pivot and return.
(6) Use the 2-by-2 pivot $\begin{bmatrix} A_{11} & A_{1r} \\ A_{r1} & A_{rr} \end{bmatrix}$.

In these pivoting rules, the constant $\alpha = (1 + \sqrt{17})/8 \approx 0.6404$ guarantees equal worst-case growth in two 1-by-1 or in one 2-by-2 step. Algorithms A and B differ only in the ordering of independent arithmetic operations, so they produce exactly the same results (Algorithm A is right looking and Algorithm B is left looking). We now show that these pivoting rules can cause almost as much growth as the upper bound allows.

Theorem 2. *The growth factor in the Bunch-Kaufman factorization (Algorithms A and B) can be as much as* $\left(1+\frac{1}{\alpha}\right)^{n-2}$*, where* n *is the dimension of the matrix.*

Proof. We now use the matrix

$$A = \begin{bmatrix} d_1 & & & & 1 & 1 \\ & d_2 & & & 1 & 1 \\ & & \ddots & & \vdots & \vdots \\ & & & d_{n-2} & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 & 1 \end{bmatrix},$$

where

$$d_k = \frac{q}{1-q} q^{-k}$$
$$q = 1 + \frac{1}{\alpha}.$$

The rest of the proof is exactly the same as Theorem 1, except that the entire trailing 2-by-2 block grows. After k steps, the trailing submatrix is

$$\begin{bmatrix} d_{k+1} & & & & 1 & 1 \\ & d_{k+2} & & & 1 & 1 \\ & & \ddots & & \vdots & \vdots \\ & & & d_{n-2} & 1 & 1 \\ 1 & 1 & \cdots & 1 & q^k & q^k \\ 1 & 1 & \cdots & 1 & q^k & q^k \end{bmatrix}.$$

We omit the details. The matrix A is exactly singular, but the example can be easily modified so that A is non singular. For example, we can set $A_{nn} = 0$ rather than $A_{nn} = 1$. □

The role of the extra row is to ensure that σ grows exponentially during the algorithm, which in turn ensures that the test in Step 4 of the pivoting rule always succeeds. In Algorithm D, σ is the magnitude of the largest element in column r, *including the diagonal element*, which grows in our example. In Algorithms A and B, the definition of σ excludes the diagonal, so we had to generate an exponentially growing element outside the diagonal.

As in the previous section, this example matrix becomes very ill-conditioned as n grows, because the diagonal elements shrink (this happens even if we modify the trailing 2-by-2 block to ensure that the last two rows are not identical). As in the previous section, we can

embed this matrix in a larger one to create a well-conditioned matrix that leads to a dramatic growth.

4. Growth in Floating-Point Arithmetic

We now show that the worst-case bound for growth in Algorithm A can be attained in floating point (Theorems 1 and 2 assumed exact arithmetic, for simplicity). More specifically, we show that the growth factor can be as large as $(1 + 1/\alpha)^{n-2}(1 - O(u))$ where u is the unit roundoff for the arithmetic.

We assume that the arithmetic satisfies the following conditions:

$$\begin{aligned}
&\text{(FPOP1)} && \mathrm{fl}\,(x \operatorname{op} y) = (x \operatorname{op} y)\,(1+\delta_1)\\
&\text{(FPOP2)} && \mathrm{fl}\,(x \operatorname{op} y)\,(1+\delta_2) = x \operatorname{op} y\\
&\text{(FPSQRT)} && \mathrm{fl}\left(\sqrt{x}\right) = \sqrt{x}\,(1+\delta_3)\\
&\text{(FPUNITY)} && \mathrm{fl}(1\cdot x) = \mathrm{fl}(x\cdot 1) = x\\
&\text{(FPSMALLU)} && (11n-22)\,u < 1\ ,
\end{aligned}$$

where op $= +, -, \cdot, \div$ and $|\delta_i| \le u$ for $i = 1, 2, 3$. We also assume that the constants $1 + 4u$, 1, 8, and 17 have exact floating-point representations and that $\mathrm{fl}(-x) = -\mathrm{fl}(x)$. Double-precision IEEE-754 floating-point arithmetic (with round to 0 or round to nearest) satisfies these conditions up to a huge n. Our main result also holds for most weaker arithmetics that only satisfy FPOP1 and FPSQRT. In such cases, the constant hidden in the $O(u)$ term might be a little larger. Also, the proof becomes more cumbersome, so we omit the details.

For conciseness, we extend the $\mathrm{fl}(\cdot)$ notation to expressions: $\mathrm{fl}(x + y - z)$ is defined to be $\mathrm{fl}(\mathrm{fl}(x + y) - z)$, for example.

We rely on the following lemmas from [6], which use the notation $\gamma_n = nu/(1 - nu)$.

Lemma 3. [6, Lemma 3.1] *If $|\delta_i| \le u$ for $i = 1, 2, \ldots, n$ and $nu < 1$ then*

$$\frac{(1+\delta_1)\,(1+\delta_2)\cdots(1+\delta_m)}{(1+\delta_{m+1})\,(1+\delta_{m+2})\cdots(1+\delta_n)} = 1+\theta,$$

where $|\theta| \le \gamma_n$.

Lemma 4. [6, part of Lemma 3.3] *If $|\theta_m| \le \gamma_m$ and $|\theta_n| \le \gamma_n$ then $(1 + \theta_m)(1 + \theta_n) = (1 + \theta_{m+n})$ where $|\theta_{m+n}| \le \gamma_{m+n}$.*

We now prove the lower bound on the worst-case growth in floating-point arithmetic.

Theorem 5. *When the Bunch-Kaufman Algorithm A is carried out in floating-point arithmetic satisfying the conditions above, its worst-case growth factor is at least* $(1 + 1/\alpha)^{n-2}(1 - \gamma_{11n-22})$.

Proof. The matrix that demonstrates the lower bound has the same structure as that in the proof of Theorem 2,

$$A = \begin{bmatrix} d_1 & & & & 1 & 1 \\ & d_2 & & & 1 & 1 \\ & & \ddots & & \vdots & \vdots \\ & & & d_{n-2} & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 & 1 \end{bmatrix},$$

but with diagonal elements that are slightly larger than those that appeared in the proof of Theorem 2, to ensure that rounding errors never cause the algorithm to perform a 2-by-2 pivoting step.

We begin by analyzing the effects of rounding errors on the constant $\alpha = (1 + \sqrt{17})/8$. We denote the computed value by $\hat{\alpha} = \mathrm{fl}((1 + \sqrt{17})/8)$. We have

$$\begin{aligned} 1 + \frac{1}{\hat{\alpha}} &= 1 + \frac{1}{\left(1 + \sqrt{17}\,(1+\eta_1)\right)(1+\eta_2)/8\,(1+\eta_3)} && |\eta_i| \le u \\ &&& i = 1,2,3 \\ &\ge 1 + \frac{1}{\left(1 + \sqrt{17}\,(1+u)\right)(1+u)/8\,(1+u)} \\ &\ge 1 + \frac{1}{\alpha}\frac{1}{(1+u)^3} \\ &\ge \left(1 + \frac{1}{\alpha}\right)\frac{1}{(1+u)^3}. \end{aligned}$$

We now define the diagonal values of A using recursive formulas that also involve the values in the trailing 2-by-2 block of the reduced matrix (from here on, the trailing block). The formulas also use the constant $\phi = 1 + 4u$. We note that ϕ satisfies $(1+u)^3 \le \phi \le (1+u)^4$. The lower bound on ϕ will ensure that rounding errors cannot force the algorithm to perform a 2-by-2 pivoting step. The upper bound will ensure large element growth.

$$
\begin{aligned}
&\sigma_0 = 1 \\
&d_k = \mathrm{fl}\left(-\frac{\alpha}{\sigma_{k-1}}\phi\right) \\
&\sigma_k = \mathrm{fl}\left(\sigma_{k-1} - \frac{1}{d_k}\right) \qquad (\text{for } k = 1,\, 2,\, \ldots,\, n-2)\,.
\end{aligned}
$$

We first show by induction that every pivoting step is a 1-by-1 step and that the reduced matrix has the form

$$
\begin{bmatrix}
d_{k+1} & & & & 1 & 1 \\
& d_{k+2} & & & 1 & 1 \\
& & \ddots & & \vdots & \vdots \\
& & & d_{n-2} & 1 & 1 \\
1 & 1 & \cdots & 1 & \sigma_k & \sigma_k \\
1 & 1 & \cdots & 1 & \sigma_k & \sigma_k
\end{bmatrix}.
$$

This is trivially true before the first step. Assume that the claim holds in steps 1, 2, ..., k.

The algorithm now tests whether $|d_{k+1}| \geq \mathrm{fl}(\hat{\alpha}\lambda)$ (Step 2 in the pivoting rule), where λ is the largest-magnitude element in column $k+1$, which is 1. If the test succeeds, the pivoting step is 1-by-1. If not, the algorithm tests the condition $\mathrm{fl}(|d_{k+1}|\sigma_k) \geq \mathrm{fl}(\hat{\alpha}\lambda^2) = \hat{\alpha}$ (Step 4 in the pivoting rule),

$$
\mathrm{fl}(|d_{k+1}|\sigma_k) = |d_{k+1}|\,\sigma_k \frac{1}{(1+\delta_1)} \geq \hat{\alpha}\,. \qquad |\delta_1| \leq u
$$

Substituting the expression for d_{k+1} we obtain

$$
\frac{\hat{\alpha}}{\sigma_k}\phi\frac{1}{(1+\delta_2)\,(1+\delta_3)}\sigma_k\frac{1}{(1+\delta_1)} \geq \hat{\alpha}\,, \qquad \begin{matrix} |\delta_i| \leq u \\ i = 1,\, 2,\, 3 \end{matrix}
$$

which is equivalent to

$$
\hat{\alpha}\phi \geq \hat{\alpha}\prod_{i=1}^{3}(1+\delta_i)\,,
$$

which is satisfied because $\phi \geq (1+u)^3$. Therefore, step $k+1$ must be a 1-by-1 step. This implies that the reduced matrix has the structure we claimed, because σ_k was defined to be the result of the appropriate rank-1 modification.

We now show that σ_k grows as quickly as the theorem claims. For $k = 1, 2, \ldots, n-2$ we have

$$\begin{aligned}
\sigma_k &= \left(\sigma_{k-1} - \frac{1}{d_k}(1+\epsilon_1)\right)(1+\epsilon_2) && |\epsilon_i| \le u \\
&&& i = 1,2,3,4 \\
&= \left(\sigma_{k-1} + \frac{\sigma_{k-1}}{\hat{\alpha}}\frac{1}{\phi}(1+\epsilon_1)(1+\epsilon_3)(1+\epsilon_4)\right)(1+\epsilon_2) \\
&= \sigma_{k-1}\left(1 + \frac{1}{\hat{\alpha}\phi}(1+\epsilon_1)(1+\epsilon_3)(1+\epsilon_4)\right)(1+\epsilon_2) \\
&\ge \sigma_{k-1}\left(1 + \frac{1}{\hat{\alpha}}\frac{(1-u)^3}{(1+u)^4}\right)(1-u) \\
&\ge \sigma_{k-1}\left(1 + \frac{1}{\hat{\alpha}}\right)\frac{(1-u)^4}{(1+u)^4}\,.
\end{aligned}$$

We have already shown that

$$1 + \frac{1}{\hat{\alpha}} \ge \left(1 + \frac{1}{\alpha}\right)\frac{1}{(1+u)^3}\,.$$

Therefore,

$$\sigma_k \ge \sigma_{k-1}\left(1 + \frac{1}{\alpha}\right)\frac{1}{(1+u)^3}\frac{(1-u)^4}{(1+u)^4}\,,$$

which according to Lemma 3 gives

$$\sigma_k \ge \sigma_{k-1}\left(1 + \frac{1}{\alpha}\right)(1-\gamma_{11})\ ,$$

so by Lemma 4,

$$\sigma_{n-2} \ge \left(1 + \frac{1}{\alpha}\right)^{n-2}(1-\gamma_{11n-22})\ .$$

This completes the proof. □

LAPACK's symmetric indefinite factorization routine xSYTRF uses a slight variant of Algorithm A, in which the test $|A_{11}|\sigma \ge \alpha\lambda^2$ is replaced by $|A_{11}| \ge \alpha\lambda(\lambda/\sigma)$; also, in case of a 1-by-1 pivot, the reduced matrix is updated according to the formula

$$A_{ij}^{(k+1)} = A_{i+1,j+1}^{(k)} - (1/A_{11}^{(k)})A_{i+1,1}^{(k)}A_{1,j+1}^{(k)}$$

where $A_{11}^{(k)}$ is explicitly inverted. The proof of Theorem 5 can be adjusted for this variant; the bound actually becomes sharper, $(1 + 1/\alpha)^{n-2}(1-\gamma_{7n-14})$. The details are omitted.

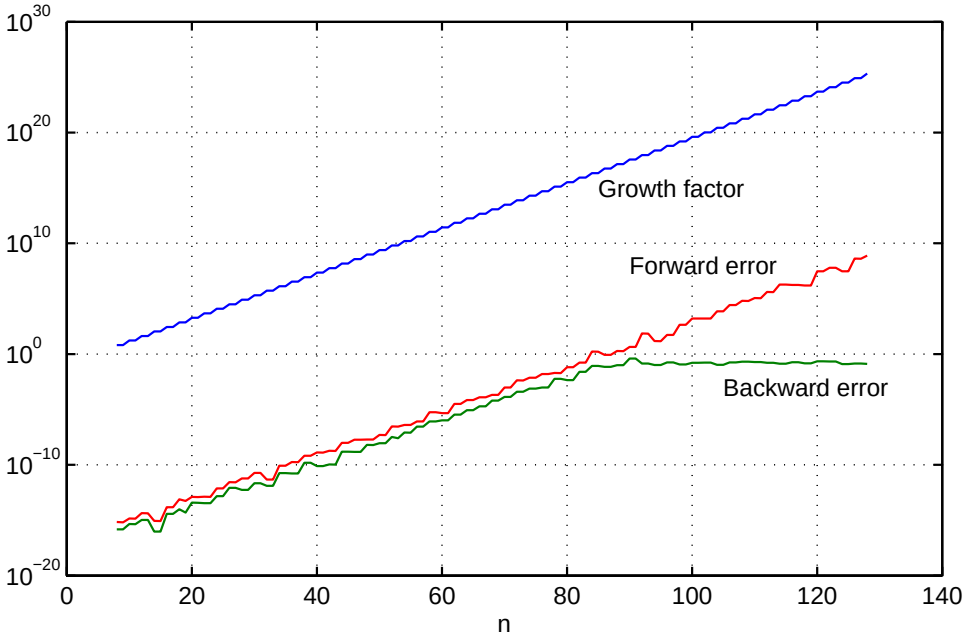

FIGURE 5.1. Actual growth in the factorization of matrices constructed as shown in Section 4. The factorization was computed using LAPACK. The graph also shows the relative backward and forward error produced by using these factorizations to solve linear systems of equations.

5. Numerical Experiments

LAPACK's [1] symmetric indefinite solver xSYSV uses Algorithm A to factor the matrix[1]. We ran the solver on a well-conditioned version of the matrix described in Section 4 and verified that the growth is indeed exponential. We also measured the forward and backward errors and verified that they grow exponentially as well. The solution vector was the vector of all 1s and the right-hand side was obtained by explicitly multiplying this vector by A. Figure 5.1 shows the results.

The results demonstrate that the exponential growth indeed occurs even in floating-point arithmetic, and even in LAPACK's production code.

[1]As of version 3.3.0 (the current version), LAPACK's symmetric indefinite factorization uses Algorithm A; it does not use rook pivoting.

6. Remarks and Open Problems

The pivoting rule of Algorithms A and B is used in LAPACK. It is important in that it is almost always used in practice to factor dense symmetric indefinite matrices. Algorithm D and variants of it are used in banded factorizations [7, 8] and in sparse factorizations, because it perturbs the row/column preordering less than Algorithms A and B. Some sparse codes use weaker variants of Algorithm D in an attempt to pivot less, possibly at the expense of even more growth [9]. Other sparse codes search for pivots within diagonal blocks (called *fronts* in the sparse-matrix literature); this is a good strategy given the data structures that are used, but it cannot be classified as either full or partial pivoting (see, e.g. [4]).

In 1998, Ashcraft, Grimes, and Lewis [2] noticed that Bunch-Kaufman solvers sometimes produce inaccurate results because L is not bounded (even when the trailing submatrix is). They proposed more sophisticated and more expensive pivoting rules that guarantee that L is bounded. Our analysis does not cover these pivoting rules. Showing examples of high growth for these methods, as well as several others referenced in [2], remains an important question [6].

Algorithm C in Bunch and Kaufman's paper starts each pivoting step by finding the largest diagonal element and permuting it to the $(1, 1)$ position. Then it continues as Algorithm A. This strategy breaks our examples. We do not know whether the growth bound can be attained in this variant. If not, it might be a safer choice in practice.

It would also be interesting to know whether the Bunch-Kaufman bounds are tight or almost tight on well-conditioned matrices; we have showed that the Bunch-Kaufman method can produce exponential growth on well-conditioned matrices, but not growth that attains the upper bound.

Acknowledgements. We thank the two anonymous referees and Nick Higham, the editor, for helpful comments and suggestions. We also thank Linda Kaufman for encouraging us to work on this problem. This research was supported in part by an IBM Faculty Partnership Award and by grant 1045/09 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities).

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] Cleve Ashcraft, Roger G. Grimes, and John G. Lewis. Accurate symmetric indefinite linear equation solvers. *SIAM Journal on Matrix Analysis and Applications*, 20(2):513–561, 1998.

[3] James R. Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation*, 31(137):163–179, 1977.

[4] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Trans. Math. Softw.*, 9(3):302–325, 1983.

[5] Nicholas J. Higham. Stability of the diagonal pivoting method with partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 18:52–65, 1997.

[6] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002.

[7] Mark T. Jones and Merrell L. Patrick. Bunch–Kaufman factorization for real symmetric indefinite banded matrices. *SIAM Journal on Matrix Analysis and Applications*, 14:553–559, April 1993.

[8] Linda Kaufman. The retraction algorithm for factoring banded symmetric matrices. *Numerical Linear Algebra with Applications*, 14:237–254, April 2007.

[9] Joseph W. H. Liu. A partial pivoting strategy for sparse symmetric matrix decomposition. *ACM Trans. Math. Softw.*, 13(2):173–182, 1987.

[10] J. H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. ACM*, 8(3):281–330, 1961.