

THE SCIENCE OF DERIVING STABILITY ANALYSES

PAOLO BIENTINESI* AND ROBERT A. VAN DE GEIJN†

FLAME Working Note #33

Abstract. We introduce a methodology for obtaining inventories of error results for families of numerical dense linear algebra algorithms. The approach for deriving the analyses is goal-oriented, systematic, and layered. The presentation places the analysis side-by-side with the algorithm so that it is obvious where error is introduced, making it attractive for use in the classroom. Yet the approach is sufficiently powerful to support the analysis of more complex algorithms, such as blocked variants that attain high performance. We provide what we believe to be the first analysis of a practical blocked LU factorization. Contrary to popular belief, the bound on the error is tighter than that of the corresponding unblocked algorithm.

1. Introduction. Numerical stability analysis related to dense linear algebra operations is one of the most important topics in numerical analysis. It is part of almost any introductory and advanced text on numerical linear algebra and earned one of its pioneers, J.H. Wilkinson, a Turing Award in 1970.

Parlett [2] observes that “*One of the major difficulties in a practical analysis is that of description. An ounce of analysis follows a pound of preparation.*” This suggests the desirability of a standardization of the description of error analyses. Higham [11] echoes this sentiment: “*Two reasons why rounding error analysis can be hard to understand are that, first, there is no standard notation and, second, error analyses are often cluttered with re-derivations of standard results.*” His statement also suggests that an inventory of results, to be used in subsequent analyses, is desirable. We believe that texts and papers alike focus on the error results and their proofs rather than on exposing a methodology that can be used by practitioners and students to obtain error results in a systematic way. Especially in a classroom setting, assignments related to stability analysis reward the recognition of tricks because no systematic methodology is explicitly passed from a typical instructor to the student.

As part of the FLAME project, we approach such problems differently. As computer scientists, we are interested in the methodology for deriving results as much as, if not more, than the results themselves. The goal for us is to identify notation and a procedure that yields the solution and that can, in principle, be made mechanical (automated). For the problem of deriving algorithms for linear algebra operations we were able to achieve these objectives: we identified notation and exposed a systematic procedure that was reproducible by a computer algebra system [3]. In this paper, we show that the same notation and procedure can be extended to equally systematically (although not yet mechanically) derive stability analyses.

This paper makes the following contributions:

- The notation we use deviates from tradition. As much as possible we abstract away from details like indices in an effort to avoid clutter, both in the presentation of the algorithms and in the error analyses of those algorithms. We are not alone in this: many texts, (e.g., [8]), partition matrices in analyses. We believe we do so more systematically and more consistent with how we present algorithms.

¹RWTH Aachen University, AICES, Pauwelsstrasse 12, 52074 Aachen, GERMANY; pauldj@aices.rwth-aachen.de.

²Department of Computer Sciences, 1 University Station C0500, The University of Texas, Austin, TX 78712; rvdg@cs.utexas.edu.

- In most texts and courses the error results are presented as theorems with proofs that incorporate one or more obscure tricks, often with different tricks for different operations. In contrast, we show that the derivation of error results is a goal-oriented exercise: given an operation and an algorithm that computes it, a possible error result is motivated and is subsequently established hand-in-hand with its proof. The methodology naturally leads to a sequence of error results that can be catalogued for future reuse. If in an analysis a subresult has not yet been catalogued, the methodology can be applied to derive that subresult.
- We motivate and demonstrate the methodology by applying it to a sequence of progressively more difficult algorithms for which results were already known. The presentation is such that it can be used as a supplement to a text being used in a class; this why we took the liberty of including exercises in this paper.
- We apply the methodology to analyze a blocked algorithm for LU factorization that is closely related to the most commonly used high-performance algorithm for LU factorization with partial pivoting. We show that the computed \tilde{L} and \tilde{U} factors of matrix A are such that $\tilde{L}\tilde{U} = A + \Delta A$, with $|\Delta A| \leq \gamma_{\frac{n}{b}+b}(|A| + |\tilde{L}||\tilde{U}|)$. The factor $\gamma_{\frac{n}{b}+b}$ improves on the known factor γ_n resulting from unblocked algorithms, yielding $\gamma_{2\sqrt{n}}$ for a suitable choice of the block size b .

We do not claim that the approach is different from what an expert does as he/she analyzes an algorithm. We do claim that we provide structure so that such analyses can be performed by people with considerably less expertise.

Experience tells us that there are many, including experts, in this particular field who think at the index level and for whom the abstractions themselves are clutter. However, we have noticed that our approach often provokes a reaction like “*As much as I have been trying to avoid [stability] analysis for the last ten years, I can only say that I like [this].*” Indeed, this pretty much summarizes our own reaction as we wrote this paper. One should not dismiss the approach without at least trying some of the more advanced exercises.

While the paper is meant to be self-contained, full understanding will come from first reading our paper on the systematic derivation of algorithms in this problem domain [4]. The reason is that the derivation of the analysis of an algorithm mirrors the derivation of the algorithm itself, as discussed in the conclusion. It is impossible for us to give a complete treatment of related work. We refer the reader to Higham’s book [11], which lists no less than 1134 citations. Other classic references are the book by Golub and Van Loan [9], Stewart [13], and by Stewart and Sun [14].

The structure of this paper is unusual. Since we claim that the presented approach makes the subject more accessible to the novice, it is written to a target audience that includes mathematically sophisticated graduate students who have not yet been exposed to numerical error analysis and includes exercises. As a result, the first sections may appear to the expert to be a review of basic notation followed by an unorthodox presentation of the subject. In Section 2, we review notation for capturing and analyzing error. In Section 3, we analyze error accumulated when computing the inner (dot) product. We also use this section to introduce what we call the “error worksheet”, first as a framework for capturing the inductive proof, of how error accumulates and next as a tool for constructively deriving the error analysis. This helps relate the new approach to the more traditional analysis that is usually presented in a

classroom setting. Next, the results for the dot product are used as part of the analysis of an algorithm for the solution of a triangular system of equations, in Section 4. This demonstrates how the error worksheet organizes the analysis of an algorithm for a somewhat more complex operation. The level of sophistication is stepped up another notch in Section 5 where an (unblocked) LU factorization algorithm is analyzed. Through these first sections, the paper is a slow crescendo that yields only well-known results. The climax comes in Section 6: an analysis for a blocked LU factorization that yields a tighter bound for the error! The paper concludes with a discussion of what the new error result means for a practical blocked LU factorization and, more importantly, of new opportunities that are enabled by the proposed methodology.

2. Preliminaries. In this section we present the minimal notation and background required for this paper.

2.1. Notation. Much of the notation related to error analysis in this paper was taken and/or inspired by the notation in [11]. We adopt the common convention that scalars, vector, and matrices will generally be denoted by lower case Greek, lower case Roman, and upper case Roman letters, respectively. Exceptions include the letters i , j , k , m , and n , which will denote scalar integers.

The letters T, B, L, R, when used as subscripts of a matrix (vector) X , indicate a 2×1 or a 1×2 partitioning of X , and denote the Top, Bottom, Left, and Right part of X , respectively. Similarly, the 4 quadrants of a 2×2 -partitioned matrix are indicated by the subscripts TL, TR, BL and BR.

The notation $\delta\chi$, where the symbol δ is connected with a scalar variable χ , indicates a perturbation associated with the variable χ . Likewise, δx and ΔX (Δ is connected with X) indicate a perturbation vector and matrix associated with vector x and matrix X , respectively. In general, variables indicating perturbations are called error operands.

The functions $m(X)$ and $n(X)$ return the row and column dimension of matrix (or vector) X , respectively. Finally, we use “ \wedge ” to represent the logical AND operator.

As is customary when discussing error analyses, we distinguish between exact and computed quantities. The function $[expression]$ returns the result of the evaluation of $expression$, where every operation is executed in floating point arithmetic¹. Equality between the quantities lhs and rhs is denoted by $lhs = rhs$. Assignment of rhs to lhs is denoted by $lhs := rhs$ (lhs becomes rhs). In the context of a program, the statements $lhs := rhs$ and $lhs := [rhs]$ are equivalent. Given an assignment $\kappa := expression$, we use the notation $\tilde{\kappa}$ to denote the quantity resulting from $[expression]$, which is actually stored in the variable κ .

We will denote the *machine epsilon* or *unit roundoff* by \mathbf{u} . It is typically of the order of 10^{-8} and 10^{-16} for single and double precision arithmetic, respectively². The unit roundoff is defined as the maximum positive floating point number which can be added to the number stored as 1 without changing the number stored as 1: $[1 + \mathbf{u}] = 1$.

2.2. Floating point computation. We introduce definitions and results regarding floating point arithmetic. In this paper, we focus on real valued arithmetic

¹Assuming that the expressions are evaluated from left to right, $[x + y + z/w]$ is equivalent to $[[[x] + [y]] + [[z] / [w]]]$.

² \mathbf{u} is machine dependent; it is a function of the parameters characterizing the machine arithmetic: $\mathbf{u} = \frac{1}{2}\beta^{1-t}$, where β is the base and t is the precision of the floating point number system for the machine.

only. Extensions to complex arithmetic are straightforward.

The *Standard Computational Model (SCM)* assumes that, for any two floating point numbers χ and ψ , the basic arithmetic operations satisfy the equality

$$[\chi \text{ op } \psi] = (\chi \text{ op } \psi)(1 + \epsilon), \quad |\epsilon| \leq \mathbf{u}, \text{ and } \text{op} \in \{+, -, *, /\}.$$

The quantity ϵ is a function of χ, ψ and op . Sometimes we add a subscript ($\epsilon_+, \epsilon_*, \dots$) to indicate what operation generated the $(1 + \epsilon)$ error factor. We always assume that all the input variables to an operation are floating point numbers.

For certain problems it is convenient to use the *Alternative Computational Model (ACM)* [11] which also assumes for the basic arithmetic operations that

$$[\chi \text{ op } \psi] = \frac{\chi \text{ op } \psi}{1 + \epsilon}, \quad |\epsilon| \leq \mathbf{u}, \text{ and } \text{op} \in \{+, -, *, /\}.$$

As for the standard computation model, the quantity ϵ is a function of χ, ψ and op . Note that the ϵ 's produced using the standard and alternative models are generally not equal.

2.3. Stability of a numerical algorithm. In the presence of round-off error, an algorithm involving numerical computations cannot be expected to yield the exact result. Thus, the notion of “correctness” applies only to the execution of algorithms in exact arithmetic. Here we briefly introduce the notion of “stability” of algorithms.

Let $f : \mathcal{D} \rightarrow \mathcal{R}$ be a mapping from the domain \mathcal{D} to the range \mathcal{R} and let $\tilde{f} : \mathcal{D} \rightarrow \mathcal{R}$ represent the mapping that captures the execution in floating point arithmetic of a given algorithm which computes f .

The algorithm is said to be *backward stable* if for all $x \in \mathcal{D}$ there exists a perturbed input $\tilde{x} \in \mathcal{D}$, close to x , such that $\tilde{f}(x) = f(\tilde{x})$. In other words, the computed result equals the result obtained when the exact function is applied to slightly perturbed data. The difference between \tilde{x} and x , $\delta x = \tilde{x} - x$, is the perturbation to the original input x .

The reasoning behind backward stability is as follows. The input to a function typically has some errors associated with it. Uncertainty may be due to measurement errors when obtaining the input and/or may be the result of converting real numbers to floating point numbers when storing the input on a computer. If it can be shown that an implementation is backward stable, then it has been proved that the result could have been obtained through exact computations performed on slightly corrupted input. Thus, one can think of the error introduced by the implementation as being comparable to the error introduced when obtaining the input data in the first place.

When discussing error analyses, δx , the difference between x and \tilde{x} , is the backward error and the difference $\tilde{f}(x) - f(x)$ is the forward error. Throughout the remainder of this paper we will be concerned with bounding the backward and/or forward errors introduced by the algorithms executed with floating point arithmetic.

2.4. Absolute value of vectors and matrices. In the above discussion of error, the vague notions of “near” and “slightly perturbed” are used. Making these notions exact usually requires the introduction of measures of size for vectors and matrices, i.e., norms. Instead, for the operations analyzed in this paper, all bounds are given in terms of the absolute values of the individual elements of the vectors and/or matrices. While it is easy to convert such bounds to bounds involving norms, the converse is not true.

DEFINITION 2.1. Given $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$,

$$|x| = \begin{pmatrix} |\chi_0| \\ \vdots \\ |\chi_{n-1}| \end{pmatrix} \quad \text{and} \quad |A| = \begin{pmatrix} |\alpha_{0,0}| & \dots & |\alpha_{0,n-1}| \\ \vdots & \ddots & \vdots \\ |\alpha_{m-1,0}| & \dots & |\alpha_{m-1,n-1}| \end{pmatrix}.$$

DEFINITION 2.2. Let $\Delta \in \{<, \leq, =, \geq, >\}$ and $x, y \in \mathbb{R}^n$. Then $|x| \Delta |y|$ iff $|\chi_i| \Delta |\psi_i|$, with $i = 0, \dots, n-1$. Similarly, given A and $B \in \mathbb{R}^{m \times n}$, $|A| \Delta |B|$ iff $|\alpha_{ij}| \Delta |\beta_{ij}|$, with $i = 0, \dots, m-1$ and $j = 0, \dots, n-1$.

The next Lemma is exploited in later sections:

LEMMA 2.3. Let $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{k \times n}$. Then $|AB| \leq |A||B|$.

EXERCISE 2.4. Prove Lemma 2.3.

The fact that the bounds that we establish can be easily converted into bounds involving norms is a consequence of the following theorem, where $\|\cdot\|_F$ indicates the Frobenius matrix norm.

THEOREM 2.5. Let $A, B \in \mathbb{R}^{m \times n}$. If $|A| \leq |B|$ then $\|A\|_1 \leq \|B\|_1$, $\|A\|_\infty \leq \|B\|_\infty$, and $\|A\|_F \leq \|B\|_F$.

EXERCISE 2.6. Prove Theorem 2.5.

2.5. Deriving dense linear algebra algorithms. In various papers, we have shown that for a broad class of linear algebra operations, given an operation one can systematically derive algorithms for computing it [10]. The primary vehicle in the derivation of algorithms is a worksheet to be filled in a prescribed order [4]. We do not discuss the derivation worksheet in this paper in order to keep the focus of on the derivation of error analyses. However, we encourage the reader to compare the error worksheet, introduced next, to the worksheet for deriving algorithms, as the order in which the error worksheet is filled mirrors that of the derivation worksheet.

3. Stability of the Dot Product Operation and Introduction to the Error Worksheet. The triangular solve algorithm discussed in the next section requires the computation of the dot (inner) product (DOT) of vectors $x, y \in \mathbb{R}^n$: $\kappa := x^T y$. In this section, we give an algorithm for this operation and the related error results. We also introduce the error-worksheet as a framework for presenting the error analysis side-by-side with the algorithm.

3.1. An algorithm for computing DOT. We will consider the algorithm given in Fig. 3.1. It uses the FLAME notation [10, 4] to express the computation

$$\kappa := \left((\chi_0 \psi_0 + \chi_1 \psi_1) + \dots \right) + \chi_{n-2} \psi_{n-2} + \chi_{n-1} \psi_{n-1} \quad (3.1)$$

in the indicated order.

3.2. Preparation. Under the computational model given in Section 2.2, the computed result of (3.1), $\tilde{\kappa}$, satisfies

$$\begin{aligned} \tilde{\kappa} &= \left(\left((\chi_0 \psi_0 (1 + \epsilon_*^{(0)}) + \chi_1 \psi_1 (1 + \epsilon_*^{(1)})) (1 + \epsilon_+^{(1)}) + \dots \right) (1 + \epsilon_+^{(n-2)}) \right. \\ &\quad \left. + \chi_{n-1} \psi_{n-1} (1 + \epsilon_*^{(n-1)}) \right) (1 + \epsilon_+^{(n-1)}) \\ &= \sum_{i=0}^{n-1} \left(\chi_i \psi_i (1 + \epsilon_*^{(i)}) \prod_{j=i}^{n-1} (1 + \epsilon_+^{(j)}) \right), \end{aligned} \quad (3.2)$$

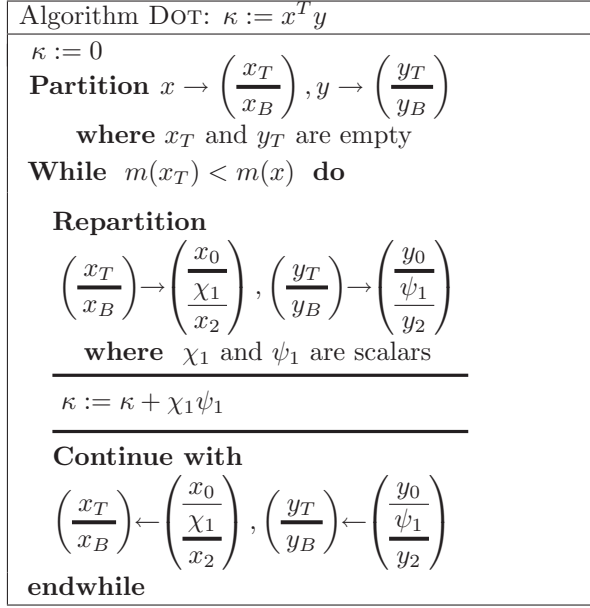


FIG. 3.1. Algorithm for computing $\kappa := x^T y$.

where $\epsilon_+^{(0)} = 0$ and $|\epsilon_*^{(0)}|, |\epsilon_*^{(j)}|, |\epsilon_+^{(j)}| \leq \mathbf{u}$ for $j = 1, \dots, n-1$. Clearly, a notation to keep expressions from becoming unreadable is desirable.

LEMMA 3.1. *Let $\epsilon_i \in \mathbb{R}$, $0 \leq i \leq n-1$, $n\mathbf{u} < 1$, and $|\epsilon_i| \leq \mathbf{u}$. Then $\exists \theta_n \in \mathbb{R}$ such that $\prod_{i=0}^{n-1} (1 + \epsilon_i)^{\pm 1} = 1 + \theta_n$, with $|\theta_n| \leq n\mathbf{u}/(1 - n\mathbf{u})$.*

Proof: By Mathematical Induction.

Base case. $n = 1$. Trivial.

Inductive Step. The Inductive Hypothesis (I.H.) tells us that for all $\epsilon_i \in \mathbb{R}$, $0 \leq i \leq n-1$, $n\mathbf{u} < 1$, and $|\epsilon_i| \leq \mathbf{u}$, there exists a $\theta_n \in \mathbb{R}$ such that $\prod_{i=0}^{n-1} (1 + \epsilon_i)^{\pm 1} = 1 + \theta_n$, with $|\theta_n| \leq n\mathbf{u}/(1 - n\mathbf{u})$.

We will show that if $\epsilon_i \in \mathbb{R}$, $0 \leq i \leq n$, $(n+1)\mathbf{u} < 1$, and $|\epsilon_i| \leq \mathbf{u}$, then there exists a $\theta_{n+1} \in \mathbb{R}$ such that $\prod_{i=0}^n (1 + \epsilon_i)^{\pm 1} = 1 + \theta_{n+1}$, with $|\theta_{n+1}| \leq (n+1)\mathbf{u}/(1 - (n+1)\mathbf{u})$.

Case 1: $\prod_{i=0}^n (1 + \epsilon_i)^{\pm 1} = \prod_{i=0}^{n-1} (1 + \epsilon_i)^{\pm 1} (1 + \epsilon_n)$. See Exercise 3.2.

Case 2: $\prod_{i=0}^n (1 + \epsilon_i)^{\pm 1} = (\prod_{i=0}^{n-1} (1 + \epsilon_i)^{\pm 1}) / (1 + \epsilon_n)$. By the I.H. there exists a θ_n such that $(1 + \theta_n) = \prod_{i=0}^{n-1} (1 + \epsilon_i)^{\pm 1}$ and $|\theta_n| \leq n\mathbf{u}/(1 - n\mathbf{u})$. Then

$$\frac{\prod_{i=0}^{n-1} (1 + \epsilon_i)^{\pm 1}}{1 + \epsilon_n} = \frac{1 + \theta_n}{1 + \epsilon_n} = 1 + \underbrace{\frac{\theta_n - \epsilon_n}{1 + \epsilon_n}}_{\theta_{n+1}},$$

which tells us how to pick θ_{n+1} . Now

$$\begin{aligned} |\theta_{n+1}| &= \left| \frac{\theta_n - \epsilon_n}{1 + \epsilon_n} \right| \leq \frac{|\theta_n| + \mathbf{u}}{1 - \mathbf{u}} \leq \frac{\frac{n\mathbf{u}}{1 - n\mathbf{u}} + \mathbf{u}}{1 - \mathbf{u}} = \frac{n\mathbf{u} + (1 - n\mathbf{u})\mathbf{u}}{(1 - n\mathbf{u})(1 - \mathbf{u})} \\ &= \frac{(n+1)\mathbf{u} - n\mathbf{u}^2}{1 - (n+1)\mathbf{u} + n\mathbf{u}^2} \leq \frac{(n+1)\mathbf{u}}{1 - (n+1)\mathbf{u}}. \end{aligned}$$

By the Principle of Mathematical Induction, the result holds. \square

EXERCISE 3.2. Complete the proof of Lemma 3.1.

The quantity θ_n will be used throughout the paper. It is not intended to be a specific number. Instead, it is an order of magnitude identified by the subscript n , which indicates the number of error factors of the form $(1 + \epsilon_i)$ and/or $(1 + \epsilon_i)^{-1}$ that are grouped together to form $(1 + \theta_n)$. Since the bound on $|\theta_n|$ occurs often, we assign it a symbol as follows:

DEFINITION 3.3. For all $n \geq 1$ and $n\mathbf{u} < 1$, define $\gamma_n := n\mathbf{u}/(1 - n\mathbf{u})$.

With this notation, (3.2) simplifies to

$$\tilde{\kappa} = \chi_0\psi_0(1 + \theta_n) + \chi_1\psi_1(1 + \theta_n) + \cdots + \chi_{n-1}\psi_{n-1}(1 + \theta_2) \quad (3.3)$$

where $|\theta_j| \leq \gamma_j$, $j = 2, \dots, n$.

Two instances of the symbol θ_n , appearing even in the same expression, typically do not represent the same number. For example, in (3.3) a $(1 + \theta_n)$ multiplies each of the terms $\chi_0\psi_0$ and $\chi_1\psi_1$, but these two instances of θ_n , as a rule, do not denote the same quantity.

As part of the analyses the following bounds will be useful to bound error that accumulates:

LEMMA 3.4. If $n, b \geq 1$ then $\gamma_n \leq \gamma_{n+b}$ and $\gamma_n + \gamma_b + \gamma_n\gamma_b \leq \gamma_{n+b}$.

EXERCISE 3.5. Prove Lemma 3.4.

3.3. Target result. It is of interest to accumulate the roundoff error encountered during computation as a perturbation of input and/or output parameters:

- $\tilde{\kappa} = (x + \delta x)^T y$;
- $\tilde{\kappa} = x^T (y + \delta y)$;
- $\tilde{\kappa} = x^T y + \delta \kappa$.

The first two are backward error results (error is accumulated onto input parameters) while the last one is a forward error result (error is accumulated onto the answer). Under different circumstances, a different error result may be needed by analyses of operations that require a dot product.

Let us focus on the second result. Ideally one would show that each of the entries of y is slightly perturbed relative to that entry:

$$\delta y = \begin{pmatrix} \sigma_0\psi_0 \\ \vdots \\ \sigma_{n-1}\psi_{n-1} \end{pmatrix} = \begin{pmatrix} \sigma_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_{n-1} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{n-1} \end{pmatrix} = \Sigma y,$$

where each σ_i is “small” and $\Sigma = \text{diag}(\sigma_0, \dots, \sigma_{n-1})$. The following special structure of Σ will be used in the remainder of the paper:

$$\Sigma^{(n)} = \begin{cases} 0 \times 0 \text{ matrix} & \text{if } n = 0 \\ \theta_1 & \text{if } n = 1 \\ \text{diag}(\theta_n, \theta_n, \theta_{n-1}, \dots, \theta_2) & \text{otherwise.} \end{cases} \quad (3.4)$$

Recall that θ_j is an order of magnitude variable with $|\theta_j| \leq \gamma_j$.

EXERCISE 3.6. Let $k \geq 0$ and assume that $|\epsilon_1|, |\epsilon_2| \leq \mathbf{u}$, with $\epsilon_1 = 0$ if $k = 0$.

Show that $\left(\begin{array}{c|c} I + \Sigma^{(k)} & 0 \\ \hline 0 & (1 + \epsilon_1) \end{array} \right) (1 + \epsilon_2) = (I + \Sigma^{(k+1)})$. Hint: reason the case where $k = 0$ separately from the case where $k > 0$.

We state a theorem that captures how error is accumulated by the algorithm.

THEOREM 3.7. Let $x, y \in \mathbb{R}^n$ and let $\kappa := x^T y$ be computed by executing the algorithm in Fig. 3.1. Then $\tilde{\kappa} = [x^T y] = x^T (I + \Sigma^{(n)})y$.

3.4. A proof in traditional format. In the below proof, we will pick symbols to denote vectors so that the proof can be easily related to the alternative framework to be presented in Section 3.5.

Proof: By Mathematical Induction on n , the length of vectors x and y .

Base case. $m(x) = m(y) = 0$. Trivial.

Inductive Step. I.H.: Assume that if $x_T, y_T \in \mathbb{R}^k$, $k > 0$, then

$$[x_T^T y_T] = x_T^T (I + \Sigma_T) y_T, \text{ where } \Sigma_T = \Sigma^{(k)}.$$

We will show that when $x_T, y_T \in \mathbb{R}^{k+1}$, the equality $[x_T^T y_T] = x_T^T (I + \Sigma_T) y_T$ holds true again. Assume that $x_T, y_T \in \mathbb{R}^{k+1}$, and partition $x_T \rightarrow \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}$ and $y_T \rightarrow \begin{pmatrix} y_0 \\ \psi_1 \end{pmatrix}$.

Then

$$\begin{aligned} \left[\begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}^T \begin{pmatrix} y_0 \\ \psi_1 \end{pmatrix} \right] &= [[x_0^T y_0] + [\chi_1 \psi_1]] && \text{(definition)} \\ &= [x_0^T (I + \Sigma_0) y_0 + [\chi_1 \psi_1]] && \text{(I.H. with } x_T = x_0, \\ & && y_T = y_0, \text{ and } \Sigma_0 = \Sigma^{(k)}) \\ &= (x_0^T (I + \Sigma_0) y_0 + \chi_1 \psi_1 (1 + \epsilon_*)) (1 + \epsilon_+) && \text{(SCM, twice)} \\ &= \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}^T \left(\begin{array}{c|c} (I + \Sigma_0) & 0 \\ \hline 0 & (1 + \epsilon_*) \end{array} \right) (1 + \epsilon_+) \begin{pmatrix} y_0 \\ \psi_1 \end{pmatrix} && \text{(rearrangement)} \\ &= x_T^T (I + \Sigma_T) y_T && \text{(renaming),} \end{aligned}$$

where $|\epsilon_*|, |\epsilon_+| \leq \mathbf{u}$, $\epsilon_+ = 0$ if $k = 0$, and $(I + \Sigma_T) = \begin{pmatrix} (I + \Sigma_0) & 0 \\ \hline 0 & (1 + \epsilon_*) \end{pmatrix} (1 + \epsilon_+)$ so that $\Sigma_T = \Sigma^{(k+1)}$.

By the Principle of Mathematical Induction, the result holds. \square

3.5. The error worksheet. We focus the reader's attention on Fig. 3.2 in which we present a framework, which we will call the *error worksheet*, for presenting the inductive proof of Theorem 3.7 side-by-side with the algorithm for DOT. This framework, in a slightly different form, was first introduced in [3]. The expressions enclosed by $\{ \}$ (in the grey boxes) are predicates describing the state of the variables used in the algorithms and in their analysis. In the worksheet, we use superscripts to indicate the iteration number, thus, the symbols v^i and v^{i+1} do not denote two different variables, but two different states of variable v .

The proof presented in Fig. 3.2 goes hand in hand with the algorithm, as it shows that before and after each iteration of the loop that computes $\kappa := x^T y$, the variables $\tilde{\kappa}, x_T, y_T, \Sigma_T$ are such that the predicate

$$\{\tilde{\kappa} = x_T^T (I + \Sigma_T) y_T \wedge k = m(x_T) \wedge \Sigma_T = \Sigma^{(k)}\} \quad (3.5)$$

holds true. This relation is satisfied at each iteration of the loop, so it is also satisfied when the loop completes. Upon completion, the loop guard is $m(x_T) = m(x) = n$, which implies that $\tilde{\kappa} = x^T (I + \Sigma^{(n)}) y$, i.e., the thesis of the theorem, is satisfied too.

In details, the inductive proof of Theorem 3.7 is captured by the error worksheet as follows:

Base case. In Step 2a, i.e. before the execution of the loop, predicate (3.5) is satisfied, as $k = m(x_T) = 0$.

	Error side	Step
$\kappa := 0$	$\{ \Sigma = 0 \}$	1a
Partition $x \rightarrow \left(\frac{x_T}{x_B} \right), y \rightarrow \left(\frac{y_T}{y_B} \right),$ where x_T and y_T are empty, and Σ_T is 0×0	$\Sigma \rightarrow \left(\frac{\Sigma_T \mid 0}{0 \mid \Sigma_B} \right)$	4
	$\{ \tilde{\kappa} = x_T^T (I + \Sigma_T) y_T \wedge \Sigma_T = \Sigma^{(k)} \wedge m(x_T) = k \}$	2a
While $m(x_T) < m(x)$ do		3
	$\{ \tilde{\kappa} = x_T^T (I + \Sigma_T) y_T \wedge \Sigma_T = \Sigma^{(k)} \wedge m(x_T) = k \}$	2b
Repartition $\left(\frac{x_T}{x_B} \right) \rightarrow \left(\frac{x_0}{\chi_1} \right), \left(\frac{y_T}{y_B} \right) \rightarrow \left(\frac{y_0}{\psi_1} \right),$ where $\chi_1, \psi_1,$ and σ_1^i are scalars	$\left(\frac{\Sigma_T \mid 0}{0 \mid \Sigma_B} \right) \rightarrow \left(\frac{\Sigma_0^i \mid 0 \mid 0}{0 \mid \sigma_1^i \mid 0} \right)$	5a
	$\{ \tilde{\kappa}^i = x_0^T (I + \Sigma_0^i) y_0 \wedge \Sigma_0^i = \Sigma^{(k)} \wedge m(x_0) = k \}$	6
$\kappa := \kappa + \chi_1 \psi_1$	$\tilde{\kappa}^{i+1} = (\tilde{\kappa}^i + \chi_1 \psi_1 (1 + \epsilon_*)) (1 + \epsilon_+)$ SCM, twice $(\epsilon_+ = 0 \text{ if } k = 0)$ $= (x_0^T (I + \Sigma_0^{(k)}) y_0 + \chi_1 \psi_1 (1 + \epsilon_*)) (1 + \epsilon_+)$ Step 6: I.H. $= \left(\frac{x_0}{\chi_1} \right)^T \left(\frac{I + \Sigma_0^{(k)} \mid 0}{0 \mid 1 + \epsilon_*} \right) (1 + \epsilon_+) \left(\frac{y_0}{\psi_1} \right)$ Rearrange $= \left(\frac{x_0}{\chi_1} \right)^T (I + \Sigma^{(k+1)}) \left(\frac{y_0}{\psi_1} \right)$ Exercise 3.6	8
	$\left\{ \begin{array}{l} \tilde{\kappa}^{i+1} = \left(\frac{x_0}{\chi_1} \right)^T \left(I + \left(\frac{\Sigma_0^{i+1} \mid 0}{0 \mid \sigma_1^{i+1}} \right) \right) \left(\frac{y_0}{\psi_1} \right) \\ \wedge \left(\frac{\Sigma_0^{i+1} \mid 0}{0 \mid \sigma_1^{i+1}} \right) = \Sigma^{(k+1)} \wedge m \left(\frac{x_0}{\chi_1} \right) = (k+1) \end{array} \right\}$	7
Continue with $\left(\frac{x_T}{x_B} \right) \leftarrow \left(\frac{x_0}{\chi_1} \right), \left(\frac{y_T}{y_B} \right) \leftarrow \left(\frac{y_0}{\psi_1} \right),$ $\left(\frac{\Sigma_T \mid 0}{0 \mid \Sigma_B} \right) \leftarrow \left(\frac{\Sigma_0^{i+1} \mid 0 \mid 0}{0 \mid \sigma_1^{i+1} \mid 0} \right)$		5b
	$\{ \tilde{\kappa} = x_T^T (I + \Sigma_T) y_T \wedge \Sigma_T = \Sigma^{(k)} \wedge m(x_T) = k \}$	2c
endwhile		
	$\{ \tilde{\kappa} = x_T^T (I + \Sigma_T) y_T \wedge \Sigma_T = \Sigma^{(k)} \wedge m(x_T) = k \wedge m(x_T) = m(x) \}$	2d
	$\{ \tilde{\kappa} = x^T (I + \Sigma^{(n)}) y \wedge m(x) = n \}$	1b

FIG. 3.2. Error worksheet completed to establish the backward error result for the given algorithm that computes the DOT operation.

Inductive step. Assume that the predicate (3.5) holds *true* at Step 2b, i.e., at the top of the loop. Then Steps 6, 7, and 8 in Fig. 3.2 prove that the predicate is satisfied again at Step 2c, i.e., the bottom of the loop. Specifically,

- Step 6 holds by virtue of the equalities $x_0 = x_T, y_0 = y_T,$ and $\Sigma_0^i = \Sigma_T.$
- The update in Step 8-left introduces the error indicated in Step 8-right (SCM, twice), yielding the results for Σ_0^{i+1} and $\sigma_1^{i+1},$ leaving the variables in the state indicated in Step 7.
- Finally, the redefinition of Σ_T in Step 5b transforms the predicate in Step 7 into that of Step 2c, completing the inductive step.

By the Principle of Mathematical Induction, the predicate (3.5) holds for all iterations. In particular, when the loop terminates, the predicate becomes

$$\tilde{\kappa} = x^T (I + \Sigma^{(n)}) y \wedge n = m(x_T).$$

This completes the discussion of the proof as captured by Fig. 3.2.

In the derivation of algorithms, the concept of *loop-invariant* plays a central role. Let \mathcal{L} be a loop and \mathcal{P} a predicate. If \mathcal{P} is *true* before the execution of \mathcal{L} , at the beginning and at the end of each iteration of \mathcal{L} , and after the completion of \mathcal{L} , then predicate \mathcal{P} is a *loop-invariant* with respect to \mathcal{L} . Similarly, we give the definition of *error-invariant*.

DEFINITION 3.8. *We call the predicate involving the operands and error operands in Steps 2a-d the error-invariant for the analysis. This predicate is true before and after each iteration of the loop.*

For any algorithm, the loop-invariant and the error-invariant are related in that the former describes the status of the computation at the beginning and the end of each iteration, while the latter captures an error result for the computation indicated by the loop-invariant.

The reader will likely think that the error worksheet is an overkill when proving the error result for the dot product. We agree. We use the dot product merely as a vehicle to introduce the reader to the methodology. As the operations being analyzed become more complex, the benefits of the structure that the error worksheet provides will become more obvious.

3.6. Results. A number of useful consequences of Theorem 3.7 follow. These will be used later as an inventory (library) of error results from which to draw when analyzing operations and algorithms that utilize DOT.

COROLLARY 3.9. *Under the assumptions of Theorem 3.7 the following relations hold:*

- R1-B: (Backward analysis) $\tilde{\kappa} = (x + \delta x)^T y$, where $|\delta x| \leq \gamma_n |x|$, and $\tilde{\kappa} = x^T (y + \delta y)$, where $|\delta y| \leq \gamma_n |y|$;*
- R1-F: (Forward analysis) $\tilde{\kappa} = x^T y + \delta \kappa$, where $|\delta \kappa| \leq \gamma_n |x|^T |y|$.*

Proof: We leave the proof of R1-B as an exercise. For R1-F, let $\delta \kappa = x^T \Sigma^{(n)} y$, where $\Sigma^{(n)}$ is as in Theorem 3.7. Then

$$\begin{aligned} |\delta \kappa| &= |x^T \Sigma^{(n)} y| \leq |\chi_0| |\theta_n| |\psi_0| + |\chi_1| |\theta_n| |\psi_1| + \cdots + |\chi_{n-1}| |\theta_2| |\psi_{n-1}| \\ &\leq \gamma_n |\chi_0| |\psi_0| + \gamma_n |\chi_1| |\psi_1| + \cdots + \gamma_2 |\chi_{n-1}| |\psi_{n-1}| \leq \gamma_n |x|^T |y|. \end{aligned}$$

□

EXERCISE 3.10. *Prove R1-B.*

LEMMA 3.11. *Let α, β and λ be scalars and consider the assignment $\sigma := (\alpha + \beta)/\lambda$. The following relations hold:*

- R1-B: $\tilde{\sigma} = ((\alpha + \delta \alpha) + (\beta + \delta \beta))/\lambda$, where $|\delta \alpha| \leq \gamma_2 |\alpha|$ and $|\delta \beta| \leq \gamma_2 |\beta|$.*
- R1-F: $\lambda \tilde{\sigma} = \alpha + \beta + \delta \sigma$, with $|\delta \sigma| \leq \gamma_2 (|\alpha| + |\beta|)$.*
- R2-B: $\tilde{\sigma} = (\alpha + \beta)/(\lambda + \delta \lambda)$, where $|\delta \lambda| \leq \gamma_2 |\lambda|$.*
- R2-F: $\lambda \tilde{\sigma} = \alpha + \beta + \delta \sigma$, with $|\delta \sigma| \leq \gamma_2 |\tilde{\sigma}| |\lambda|$.*
- R3-B: $\tilde{\sigma} = ((\alpha + \delta \alpha) + (\beta + \delta \beta))/(\lambda + \delta \lambda)$, where $|\delta \alpha| \leq \gamma_1 |\alpha|$, $|\delta \beta| \leq \gamma_1 |\beta|$, and $|\delta \lambda| \leq \gamma_1 |\lambda|$.*
- R3-F: $\lambda \tilde{\sigma} = \alpha + \beta + \delta \sigma$, with $|\delta \sigma| \leq \gamma_1 (|\alpha| + |\beta| + |\tilde{\sigma}| |\lambda|)$.*

Also, if $\lambda = 1$, results R1-B, R2-B and R2-F hold with γ_1 in place of γ_2 , and R3-B holds with either $\delta \lambda = 0$ or both $\delta \alpha = 0$ and $\delta \beta = 0$.

Proof: R1-B follows directly from the definition of SCM. R2-B is obtained by applying the definition of ACM (twice) and Lemma 3.1:

$$\begin{aligned}\check{\sigma} &= \left[\frac{\alpha + \beta}{\lambda} \right] = \left[\frac{[\alpha + \beta]}{\lambda} \right] = \left[\frac{\alpha + \beta}{\lambda(1 + \epsilon_+)} \right] = \frac{\alpha + \beta}{\lambda(1 + \epsilon_+)(1 + \epsilon_j)} \\ &= \frac{\alpha + \beta}{\lambda(1 + \theta_2)} = \frac{\alpha + \beta}{\lambda + \delta\lambda}, \quad \text{where } \delta\lambda = \lambda\theta_2.\end{aligned}$$

R3-B is derived similarly, and R2-F is an immediate consequence of R2-B. \square

Relations R1-B, R2-B and R3-B state that the computed result $\check{\sigma}$ equals the exact result of the assignment $(\alpha + \beta)/\lambda$ with slightly perturbed inputs. These relations indicate how the error generated in performing such an assignment can be “thrown back” in different ways at the α , β , and/or λ input variables; in other words, they establish that the computation of $[(\alpha + \beta)/\lambda]$ is backward stable.

The following theorem prepares us for the analyses of algorithms that use DOT as a suboperation.

THEOREM 3.12. *Let $n \geq 1$, $\alpha, \lambda, \mu, \nu \in \mathbb{R}$, and $x, y \in \mathbb{R}^n$. Assume that $\lambda \neq 0$ and consider the assignments $\check{\mu} := \alpha - (x^T y)$ and $\check{\nu} := (\alpha - (x^T y)) / \lambda$; the parentheses identify the evaluation order. Then*

R1-B: $\check{\mu} = \alpha + \delta\alpha - (x + \delta x)^T y$, where $|\delta\alpha| \leq \gamma_1 |\alpha|$ and $|\delta x| \leq \gamma_{n+1} |x|$.

R1-F: $\check{\mu} = \alpha - x^T y + \check{\delta}\mu$, where $|\check{\delta}\mu| \leq \gamma_{n+1} |x|^T |y| + \gamma_1 |\alpha| \leq \gamma_{n+1} (|x|^T |y| + |\alpha|)$.

R2-F: $\check{\mu} = \alpha - x^T y + \check{\delta}\mu$, where $|\check{\delta}\mu| \leq \gamma_n |x|^T |y| + \gamma_1 |\check{\mu}| \leq \gamma_n (|x|^T |y| + |\check{\mu}|)$.

Also,

R3-B: $\lambda\check{\nu} = \alpha + \delta\alpha - (x + \delta x)^T y$, where $|\delta\alpha| \leq \gamma_2 |\alpha|$ and $|\delta x| \leq \gamma_{n+2} |x|$.

R3-F: $\lambda\check{\nu} = \alpha - x^T y + \delta\nu$, where $|\delta\nu| \leq \gamma_2 |\alpha| + \gamma_{n+2} |x|^T |y| \leq \gamma_{n+2} (|\alpha| + |x|^T |y|)$.

R4-B: $(\lambda + \delta\lambda)\check{\nu} = \alpha - (x + \delta x)^T y$, where $|\delta\lambda| \leq \gamma_2 |\lambda|$ and $|\delta x| \leq \gamma_n |x|$.

R4-F: $\lambda\check{\nu} = \alpha - x^T y + \delta\nu$, where

$$|\delta\nu| \leq \gamma_n |x|^T |y| + \gamma_2 |\lambda| |\check{\nu}| \leq \max(\gamma_2, \gamma_n) (|x|^T |y| + |\lambda| |\check{\nu}|).$$

R5-B: $(\lambda + \delta\lambda)\check{\nu} = \alpha + \delta\alpha - (x + \delta x)^T y$, where

$$|\delta\alpha| \leq \gamma_1 |\alpha|, \quad |\delta x| \leq \gamma_{n+1} |x|, \quad \text{and } |\delta\lambda| \leq \gamma_1 |\lambda|.$$

R5-F: $\lambda\check{\nu} = \alpha - x^T y + \delta\nu$, where

$$|\delta\nu| \leq \gamma_1 |\alpha| + \gamma_{n+1} |x|^T |y| + \gamma_1 |\lambda| |\check{\nu}| \leq \gamma_{n+1} (|\alpha| + |x|^T |y| + |\lambda| |\check{\nu}|).$$

Proof: R1 follows Theorem 3.7 and SCM: $\check{\mu} = (\alpha - x^T \Sigma^{(n)} y)(1 + \epsilon_1)$, algebraic manipulation, and bounding. R2-F follows Theorem 3.7 and ACM: $\check{\mu} = (\alpha - x^T \Sigma^{(n)} y)/(1 + \epsilon_2)$, algebraic manipulation, and bounding.

Results R3, R4 and R5 are obtained similarly, invoking Theorem 3.7 and Lemma 3.11. \square

EXERCISE 3.13. *Work out the details of the proof of Theorem 3.12.*

4. Analysis of Triangular Linear Systems and Introduction to Goal-Oriented Error Analysis. In this section, we discuss a goal-oriented methodology for analyzing the error accumulated by an algorithm when executed under the model of floating point arithmetic given in Section 2.2. We introduce the methodology hand-in-hand with the analysis of an algorithm for the solution of a triangular linear system.

4.1. Solving a lower triangular linear system. The solution of a lower triangular system of equations (LTRSV) can be stated as $Lx = b$, where $L \in \mathbb{R}^{n \times n}$ is a nonsingular lower triangular matrix, L and b are input operands and x is the output

<p>Algorithm LTRSV: Compute x such that $Lx = b$</p> <p>Partition $L \rightarrow \left(\begin{array}{c c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array} \right), x \rightarrow \begin{pmatrix} x_T \\ x_B \end{pmatrix}, b \rightarrow \begin{pmatrix} b_T \\ b_B \end{pmatrix}$</p> <p>where $L_{TL}, x_T,$ and b_T are empty</p> <p>While $m(x_T) < m(x)$ do</p> <p>Repartition</p> <p>$\left(\begin{array}{c c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c c c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ L_{20} & l_{21} & L_{22} \end{array} \right), \begin{pmatrix} x_T \\ x_B \end{pmatrix} \rightarrow \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} b_T \\ b_B \end{pmatrix} \rightarrow \begin{pmatrix} b_0 \\ \beta_1 \\ b_2 \end{pmatrix}$</p> <p>where $\lambda_{11}, \chi_1,$ and β_1 are scalars</p> <hr/> <p>$\chi_1 := (\beta_1 - l_{10}^T x_0) / \lambda_{11}$</p> <hr/> <p>Continue with</p> <p>$\left(\begin{array}{c c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c c c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ L_{20} & l_{21} & L_{22} \end{array} \right), \begin{pmatrix} x_T \\ x_B \end{pmatrix} \leftarrow \begin{pmatrix} x_0 \\ \chi_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} b_T \\ b_B \end{pmatrix} \leftarrow \begin{pmatrix} b_0 \\ \beta_1 \\ b_2 \end{pmatrix}$</p> <p>endwhile</p>
--

FIG. 4.1. An algorithm for solving $Lx = b$.

operand³. In Fig. 4.1 we show one specific algorithm for computing x . This algorithm has the property that during its execution the variables satisfy the predicate (loop-invariant) $\{L_{TL}x_T = b_T\}$ at the beginning and the end of each loop iteration.

4.2. Analysis. The analysis of the algorithm in Fig. 4.1 is carried out in the worksheet in Fig. 4.2; in the following we explain how the worksheet is constructively filled out to generate the analysis. At this stage, please pretend that all the gray-shaded boxes in Fig. 4.2 are empty. Similarly, pretend that only the left part of Step 8 (computational updates) is already complete, being copied over from Fig. 4.2. The empty boxes will be filled out in the order indicated in the “Steps” column.

Step 1: The error-precondition and error-postcondition. The first question is what error result is desired. We will show that the computed solution, \tilde{x} , satisfies the backward error result $(L + \Delta L)\tilde{x} = b$ with $|\Delta L| \leq \max(\gamma_2, \gamma_{n-1})|L|$, i.e., $|\Delta L| \leq \gamma_{n-1}|L|$ when $n > 2$.

The reasoning for the factor γ_{n-1} is as follows. We would expect the maximal error to be incurred during the final iteration when computing $\chi_1 = (\beta_1 - l_{10}^T x_0) / \lambda_{11}$. This assignment involves a dot product with vectors of length $n - 1$. Thus, results R4 from Theorem 3.12 suggest that it is reasonable to expect the indicated factor.

In practice, the analysis proceeds in two stages. In the first stage one proves, constructively, the existence of a matrix ΔL , the error-operand, such that $(L + \Delta L)\tilde{x} = b$. In this stage error bounds are not considered, as the only concern is to assign the error generated by the computational updates to the error-operands. This process involves error results regarding the suboperations that appear in the updates. These error results then allow one to make an educated guess of the bounds that can be established for the error operands. In the second stage the bounds on ΔL are determined. For space considerations, in this paper we incorporate the proof of the bounds on the error operands into the proof of existence.

We call the predicate that describes the state of the error operands, in this case the matrix ΔL , before the algorithm is executed, the *error-precondition*, and the predicate

³In this section we make no assumption on the diagonal entries of matrix L . In the next section we will use the symbol L to indicate a unit lower triangular matrix.

	Error side	Step
	$\{\Delta\mathcal{L} = 0\}$	1a
Partition $L \rightarrow \left(\begin{array}{c c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array} \right), x \rightarrow \left(\begin{array}{c} x_T \\ \hline x_B \end{array} \right), b \rightarrow \left(\begin{array}{c} b_T \\ \hline b_B \end{array} \right), \Delta\mathcal{L} \rightarrow \left(\begin{array}{c c} \Delta\mathcal{L}_{TL} & 0 \\ \hline \Delta\mathcal{L}_{BL} & \Delta\mathcal{L}_{BR} \end{array} \right)$ where $L_{TL}, x_T, b_T,$ and $\Delta\mathcal{L}_{TL}$ are empty		4
	$\{(L_{TL} + \Delta\mathcal{L}_{TL})\tilde{x}_T = b_T \wedge \Delta\mathcal{L}_{TL} \leq \max(\gamma_2, \gamma_{k-1}) L_{TL} \wedge m(x_T) = k\}$	2a
While $m(x_T) < m(x)$ do		3
	$\{(L_{TL} + \Delta\mathcal{L}_{TL})\tilde{x}_T = b_T \wedge \Delta\mathcal{L}_{TL} \leq \max(\gamma_2, \gamma_{k-1}) L_{TL} \wedge m(x_T) = k\}$	2b
Repartition $\left(\begin{array}{c c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c c c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array} \right), \left(\begin{array}{c c} \Delta\mathcal{L}_{TL} & 0 \\ \hline \Delta\mathcal{L}_{BL} & \Delta\mathcal{L}_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c c c} \Delta\mathcal{L}_{00} & 0 & 0 \\ \hline \delta_{10}^T & \delta_{11} & 0 \\ \hline \Delta\mathcal{L}_{20} & \delta_{21} & \Delta\mathcal{L}_{22} \end{array} \right)$ $\left(\begin{array}{c} \tilde{x}_T \\ \hline \tilde{x}_B \end{array} \right) \rightarrow \left(\begin{array}{c} \tilde{x}_0 \\ \hline \tilde{\chi}_1 \\ \hline \tilde{x}_2 \end{array} \right), \left(\begin{array}{c} b_T \\ \hline b_B \end{array} \right) \rightarrow \left(\begin{array}{c} b_0 \\ \hline \beta_1 \\ \hline b_2 \end{array} \right)$ where $\lambda_{11}, \delta_{11}, \chi_1,$ and β_1 are scalars		5a
	$\{(L_{00} + \Delta\mathcal{L}_{00})\tilde{x}_0 = b_0 \wedge \Delta\mathcal{L}_{00} \leq \max(\gamma_2, \gamma_{k-1}) L_{00} \wedge m(x_0) = k\}$	6
$\chi_1 := \frac{\beta_1 - l_{10}^T x_0}{\lambda_{11}}$	$\left. \begin{array}{l} b_0 = (L_{00} + \Delta\mathcal{L}_{00})\tilde{x}_0 \\ \wedge \Delta\mathcal{L}_{00} \leq \max(\gamma_2, \gamma_{k-1}) L_{00} \end{array} \right\}$ Step 6: I.H. $\left. \begin{array}{l} \beta_1 = (l_{10}^T + \delta_{10}^T)\tilde{x}_0 + (\lambda_{11} + \delta_{11})\tilde{\chi}_1 \\ \wedge (\delta_{10}^T \delta_{11}) \leq \max(\gamma_2, \gamma_k) (l_{10}^T \lambda_{11}) \end{array} \right\}$ Th. 3.12 R4-B	8
	$\left\{ \left(\left(\begin{array}{c c} L_{00} & 0 \\ \hline l_{10}^T & \lambda \end{array} \right) + \left(\begin{array}{c c} \Delta\mathcal{L}_{00} & 0 \\ \hline \delta_{10}^T & \delta \end{array} \right) \right) \left(\begin{array}{c} \tilde{x}_0 \\ \hline \tilde{\chi}_1 \end{array} \right) = \left(\begin{array}{c} b_0 \\ \hline \beta_1 \end{array} \right) \right. \\ \left. \wedge \left \left(\begin{array}{c c} \Delta\mathcal{L}_{00} & 0 \\ \hline \delta_{10}^T & \delta \end{array} \right) \right \leq \max(\gamma_2, \gamma_k) \left \left(\begin{array}{c c} L_{00} & 0 \\ \hline l_{10}^T & \lambda \end{array} \right) \right \wedge m \left(\begin{array}{c} x_0 \\ \hline \chi_1 \end{array} \right) = k + 1 \right\}$	7
Continue with L, x and b as in Fig. 4.1, and	$\left(\begin{array}{c c} \Delta\mathcal{L}_{TL} & 0 \\ \hline \Delta\mathcal{L}_{BL} & \Delta\mathcal{L}_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c c c} \Delta\mathcal{L}_{00} & 0 & 0 \\ \hline \delta_{10}^T & \delta_{11} & 0 \\ \hline \Delta\mathcal{L}_{20} & \delta_{21} & \Delta\mathcal{L}_{22} \end{array} \right)$	5b
	$\{(L_{TL} + \Delta\mathcal{L}_{TL})\tilde{x}_T = b_T \wedge \Delta\mathcal{L}_{TL} \leq \max(\gamma_2, \gamma_{k-1}) L_{TL} \wedge m(x_T) = k\}$	2c
endwhile		
	$\left\{ \begin{array}{l} (L_{TL} + \Delta\mathcal{L}_{TL})\tilde{x}_T = b_T \wedge \Delta\mathcal{L}_{TL} \leq \max(\gamma_2, \gamma_{k-1}) L_{TL} \\ \wedge m(x_T) = m(x) \wedge m(x_T) = k \end{array} \right\}$	2d
	$\{(L + \Delta\mathcal{L})\tilde{x} = b \wedge \Delta\mathcal{L} \leq \max(\gamma_2, \gamma_{n-1}) L \wedge m(x) = n\}$	1b

FIG. 4.2. Error worksheet for deriving the backward stability error result for the algorithm in Fig. 4.1.

that describes the target error result, the *error-postcondition*. In Fig. 4.2 these predicates are placed in Step 1a and 1b, respectively. The error-precondition is $\{\Delta\mathcal{L} = 0\}$ (no error has yet accumulated), and the error-postcondition is $\{(L + \Delta\mathcal{L})\tilde{x} = b \wedge |\Delta\mathcal{L}| \leq \max(\gamma_2, \gamma_{n-1})|L| \wedge m(x) = n\}$.

Step 2: The error-invariant. Now we pick an *error-invariant* for the algorithm, i.e., a predicate that describes the state of the error operands in Steps 2a–d. Since the loop-invariant is the predicate that describes the computation performed when the algorithm reaches these stages, it makes sense to impose the error-invariant to equal the portion of the error-postcondition that corresponds to the state described by the loop-invariant. The algorithm in Fig. 4.1 has the property that before and after each iteration the contents of x_T are such that $\{L_{TL}x_T = b_T\}$. This predicate is the loop-invariant. Thus, we expect the accumulated error to satisfy $(L_{TL} + \Delta\mathcal{L}_{TL})\tilde{x}_T = b_T$. Considering also bounds on $\Delta\mathcal{L}_{TL}$, the error-invariant be-

comes

$$\{(L_{TL} + \Delta L_{TL})\tilde{x}_T = b_T \wedge |\Delta L_{TL}| \leq \max(\gamma_2, \gamma_{k-1})|L_{TL}| \wedge m(x_T) = k\},$$

and it appears in Steps 2a–d.

Step 3: The loop-guard. There is nothing to be done for this step, as the loop guard comes as part of the algorithm to analyze. We retain this step number so that the error worksheet closely resembles the worksheet we use for deriving algorithms [4, 3, 15].

Step 4: Initialization. In this step the error operand, ΔL , is partitioned conformally to the operands in the algorithm with respect to the error-postcondition. In other words, ΔL is partitioned so that the dimensions of the variables in the expression $(L + \Delta L)\tilde{x} = b$, with the operands partitioned as in Fig. 4.1, are conformal.

Step 5: Moving through the error operand. In Steps 5a and 5b, the error operand is repartitioned conformally to the repartitioning of the operands in the algorithm.

Step 6: State of the variables before the update in Step 8. Step 5a consists of the following relabelling statements:

$$L_{TL} \rightarrow L_{00}, \quad x_T \rightarrow x_0, \quad b_T \rightarrow b_0, \quad \text{and} \quad \Delta L_{TL} \rightarrow \Delta L_{00}.$$

Thus, given the state of the variables in Step 2b, the contents of the submatrices and subvectors exposed in Step 5a are described by

$$(L_{00} + \Delta L_{00})\tilde{x}_0 = b_0 \wedge |\Delta L_{00}| \leq \max(\gamma_2, \gamma_{k-1})|L_{00}| \wedge m(x_0) = k \quad (4.1)$$

at Step 6. This predicate expresses the state of the error operands before the execution of the computational statements listed in Step 8, on the left. Such a replacement of symbols is called *textual substitution*.

Step 7: State of the variables after the update in Step 8. At the bottom of the loop, the variables must again be in a state where the loop-invariant holds, as indicated by Step 2c. In Step 5b, the different quadrants of L and ΔL , and the subvectors of x and b , are redefined so that

$$L_{TL} \leftarrow \left(\frac{L_{00}}{l_{10}^T} \middle| \frac{0}{\lambda_{11}} \right), \quad x_T \leftarrow \left(\frac{x_0}{\chi_1} \right), \quad b_T \leftarrow \left(\frac{b_0}{\beta_1} \right), \quad \text{and} \quad \Delta L_{TL} \leftarrow \left(\frac{\Delta L_{00}}{\delta_{10}^T} \middle| \frac{0}{\delta_{11}} \right).$$

Thus, given the state in which the variables *must be* in Step 2c, the contents of the submatrices and subvectors before Step 5b *must be*

$$\begin{aligned} & \left(\left(\frac{L_{00}}{l_{10}^T} \middle| \frac{0}{\lambda_{11}} \right) + \left(\frac{\Delta L_{00}}{\delta_{10}^T} \middle| \frac{0}{\delta_{11}} \right) \right) \begin{pmatrix} \tilde{x}_0 \\ \tilde{\chi}_1 \end{pmatrix} = \begin{pmatrix} b_0 \\ \beta_1 \end{pmatrix} \wedge m \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix} = k + 1 \\ & \wedge \left| \left(\frac{\Delta L_{00}}{\delta_{10}^T} \middle| \frac{0}{\delta_{11}} \right) \right| \leq \max(\gamma_2, \gamma_k) \left| \left(\frac{L_{00}}{l_{10}^T} \middle| \frac{0}{\lambda_{11}} \right) \right| \end{aligned} \quad (4.2)$$

at Step 6. This predicate is attained via textual substitution and algebraic manipulation.

Step 8: The inductive step. When the computation reaches this step the predicate (4.1), described in Step 6, is satisfied. The question now is whether the computational update

$$\chi_1 := (\beta_1 - l_{10}^T x_0) / \lambda_{11}, \quad \text{with } m(x_0) = k,$$

generates errors for which one can find assignments to the error variables ΔL_{00} , δl_{10}^T , and $\delta \lambda_{11}$ so that the predicate (4.2), described in Step 7, is also satisfied.

Manipulating (4.2), we find that after the computational update is complete, the error variables must satisfy

$$\begin{aligned} b_0 &= (L_{00} + \Delta L_{00}) \tilde{x}_0 && \wedge |\Delta L_{00}| \leq \max(\gamma_2, \gamma_k) |L_{00}| \\ \beta_1 &= (l_{10}^T + \delta l_{10}^T) \tilde{x}_0 + (\lambda_{11} + \delta \lambda_{11}) \tilde{\chi}_1 && \wedge (|\delta l_{10}^T| |\delta \lambda_{11}|) \leq \max(\gamma_2, \gamma_k) |(l_{10}^T | \lambda_{11})|. \end{aligned}$$

Since L_{00} , x_0 , and b_0 are not modified by the assignment in Step 8-left, the top constraint is satisfied by virtue of (4.1) and $\gamma_{k-1} \leq \gamma_k$. In the language of an inductive proof, this constraint is *true* by the I.H. For the second constraint, we consult our inventory of error results and find Theorem 3.12 Result R4-B. The analysis is also given in Step 8-right of Fig. 4.2.

4.3. Results for TRSV. The above discussion in Sec. 4.2, together with Fig. 4.2, is the proof for R1-B of this next theorem.

THEOREM 4.1. *Let $L \in \mathbb{R}^{n \times n}$ be a nonsingular lower triangular matrix and $x, b \in \mathbb{R}^n$. Assume that the solution x of the linear system $Lx = b$ is computed via the algorithm in Fig. 4.2(left side). When executed in floating point arithmetic, the algorithm yields \tilde{x} that satisfies*

R1-B: $(L + \Delta L)\tilde{x} = b$, where $|\Delta L| \leq \max(\gamma_2, \gamma_{n-1})|L|$.

R1-F: $L\tilde{x} = b + \delta$ where $|\delta| \leq \max(\gamma_2, \gamma_{n-1})|L||\tilde{x}|$.

EXERCISE 4.2. *Prove R1-F of Theorem 4.1.*

EXERCISE 4.3. *In Theorem 4.1, assume that the diagonal entries of L are all 1s. How does the factor γ improve as a result of this assumption? Indicate what changes must be made to Section 4.2 and Fig. 4.2, and what result from Theorem 3.12 should be used.*

4.4. Other algorithmic variants. There is another algorithmic variant for computing the triangular solve for which the error accumulates slightly differently if b is overwritten by the solution x without the use of extra workspace. Theorems for that algorithm can be derived in a similar manner [3].

5. Analysis of an unblocked algorithm for LU factorization. Having introduced the methodology for deriving analyses as well as a few error results for simpler operations, we can move on to a more difficult operation, the LU factorization of a square matrix: given a non-singular square matrix A , we seek matrices L and U such that $LU = A$, where L is a lower triangular with unit diagonal and U is upper triangular.

5.1. An algorithm for computing the LU factorization. We analyze the variant that is often called the Crout variant [7]. This choice allows the reader, if so inclined, to compare and contrast our exposition with the one in the book by Nick Higham [11]. The Crout variant computes L and U so that before and after each iteration L_{TL} , U_{TL} , L_{BL} and U_{TR} satisfy

$$\left(\begin{array}{c|c} L_{TL}U_{TL} = A_{TL} & L_{TL}U_{TR} = A_{TR} \\ \hline L_{BL}U_{TL} = A_{BL} & \end{array} \right). \quad (5.1)$$

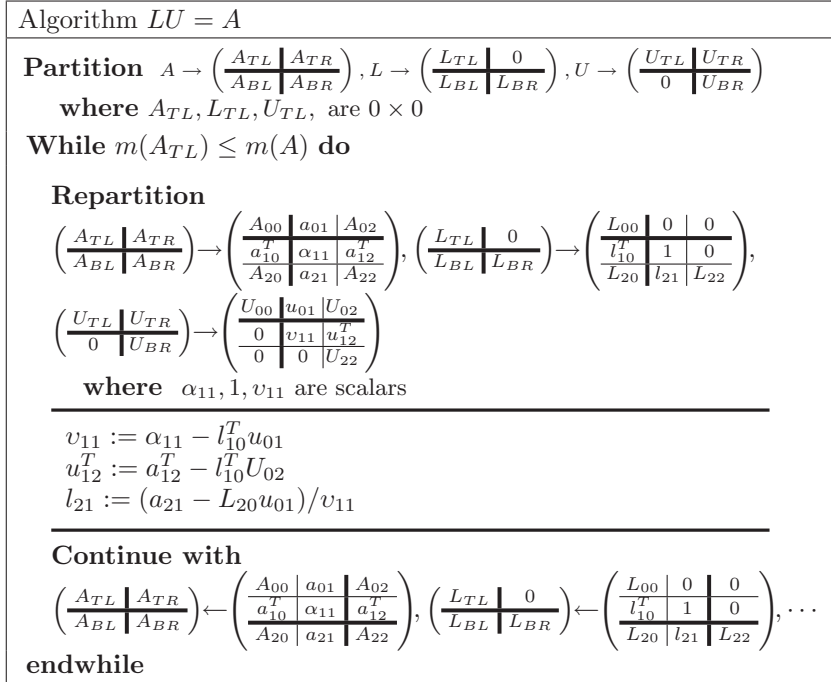


FIG. 5.1. The unblocked Crout variant for computing the LU factorization of a matrix.

The algorithm is given in Fig. 5.1.

5.2. Preparation. The Crout variant casts the computation in terms of inner products ($v_{11} := \alpha_{11} - l_{10}^T u_{01}$) and matrix-vector multiplications ($u_{12}^T := a_{12}^T - l_{10}^T U_{02}$ and $l_{21} := (a_{21} - L_{20} u_{01}) / v_{11}$), therefore its stability analysis will build on top of the analyses for these operations. In Section 3 we have created a collection of error results for inner products; the next theorem provides error analyses for matrix-vector multiplications.

THEOREM 5.1. Error results for matrix-vector multiplication. *Let $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $y, v, w \in \mathbb{R}^m$, $\lambda \in \mathbb{R}$ and consider the assignments $v := y - Ax$ and $w := (y - Ax) / \lambda$. Assume that v and w are computed one element at a time by subtracting from y the result of an inner product, and then dividing it by λ for the second operation. These equalities hold:*

R1-F: $\tilde{v} = y - Ax + \delta v$, where $|\delta v| \leq \gamma_{n+1} |A| |x| + \gamma_1 |y| \leq \gamma_{n+1} (|A| |x| + |y|)$.

R2-F: $\tilde{v} = y - Ax + \delta v$, where $|\delta v| \leq \gamma_n |A| |x| + \gamma_1 |\tilde{v}| \leq \gamma_n (|A| |x| + |\tilde{v}|)$.

Also,

R3-F: $\lambda \tilde{w} = y - Ax + \delta w$, where

$$|\delta w| \leq \gamma_2 |y| + \gamma_{n+2} |A| |x| \leq \gamma_{n+2} (|y| + |A| |x|).$$

R4-F: $\lambda \tilde{w} = y - Ax + \delta w$, where

$$|\delta w| \leq \gamma_n |A| |x| + \gamma_2 |\lambda| |\tilde{w}| \leq \max(\gamma_2, \gamma_n) (|A| |x| + |\lambda| |\tilde{w}|).$$

R5-F: $\lambda \tilde{w} = y - Ax + \delta w$, where

$$|\delta w| \leq \gamma_1 |y| + \gamma_{n+1} |A| |x| + \gamma_1 |\lambda| |\tilde{w}| \leq \gamma_{n+1} (|y| + |A| |x| + |\lambda| |\tilde{w}|).$$

Proof: All these results follow from Theorem 3.12. \square

EXERCISE 5.2. Prove Theorem 5.1.

EXERCISE 5.3. Which of the results in Theorem 5.1 admit a backward formulation? For those, state and prove the backward results.

5.3. Analysis. We will show that the computed factors \check{L} and \check{U} satisfy the equality $\check{L}\check{U} = A + \Delta A$ with $|\Delta A| \leq \gamma_n |\check{L}||\check{U}|$. This is the error-postcondition, which appears in Step 1b of Fig. 5.2. The implications of this result on the stability of the algorithm are discussed in Section 6.4.

The *a priori* justification for the bound is not as straightforward as it was for the solution of the triangular system. The factor γ_n comes from the fact that each element of L and U is exposed to the effects of at most n individual operations. In addition, experience tells us that bounds for the backward error of various algorithms for LU factorization have the form $c_1\gamma_n|A| + c_2\gamma_n|L||U|$, where $c_1 \geq 0$ and $c_2 \geq 1$. We stress that in practice one would start with a formulation of the analysis without taking bounds into account; our methodology makes it easy to experiment with loose analyses to gain insight that leads to a tighter formulation. For space considerations, we start the analysis already with tight bounds.

In Fig. 5.2, we show both the algorithm, on the left, and the error side of the error worksheet, on the right. The latter is filled out in the order indicated by the Step column, as we demonstrated in Sec. 4. We have already established the expressions for the error-precondition and error-postcondition (Step 1), in which the matrix ΔA acts as the error operand. In Step 2 one has to select a feasible error-invariant, i.e., an error result for the computations performed up to this stage. To this end, we restrict the error-postcondition to the computation described by the loop-invariant (5.1), yielding the error-invariant

$$m(A_{TL}) = k \wedge \left(\frac{\check{L}_{TL}\check{U}_{TL} = A_{TL} + \Delta A_{TL} \mid \check{L}_{TL}\check{U}_{TR} = A_{TR} + \Delta A_{TR}}{\check{L}_{BL}\check{U}_{TR} = A_{BL} + \Delta A_{BL}} \right) \wedge \left(\frac{|\Delta A_{TL}| \leq \gamma_k |\check{L}_{TL}||\check{U}_{TL}| \mid |\Delta A_{TR}| \leq \gamma_k |\check{L}_{TL}||\check{U}_{TR}|}{|A_{BL}| \leq \gamma_k |\check{L}_{BL}||\check{U}_{TR}|} \right),$$

which appears in Step 2.⁴ Given the error-invariant, Steps 4–7 only require symbolic manipulation. In particular, the predicates indicated in Steps 6 and 7 follow immediately from the error-invariant by substituting the submatrices from Steps 5a and 5b, respectively. Step 6 becomes

$$m(A_{00}) = k \wedge \left(\frac{\check{L}_{00}\check{U}_{00} = A_{00} + \Delta A_{00} \mid \check{L}_{00}\check{u}_{01} = a_{01} + \delta a_{01} \mid \check{L}_{00}\check{U}_{02} = A_{02} + \Delta A_{02}}{\frac{\check{L}_{10}^T\check{U}_{00} = a_{10}^T + \delta a_{10}^T}{\check{L}_{20}\check{U}_{00} = A_{20} + \Delta A_{02}}} \right) \wedge \left(\frac{\left(\frac{\Delta A_{00} \mid \delta a_{01} \mid \Delta A_{02}}{\delta a_{10}^T} \right)}{\left(\frac{|\check{L}_{00}||\check{U}_{00}| \mid |\check{L}_{00}||\check{u}_{01}| \mid |\check{L}_{00}||\check{U}_{02}|}{|\check{L}_{10}^T||\check{U}_{00}|} \right)} \leq \gamma_k \left(\frac{|\check{L}_{00}||\check{U}_{00}| \mid |\check{L}_{00}||\check{u}_{01}| \mid |\check{L}_{00}||\check{U}_{02}|}{|\check{L}_{20}||\check{U}_{00}|} \right). \quad (5.2)$$

Step 2 tells us that this predicate is *true* before the execution of the computational statements in the left box of Step 8. In other words, Step 6 represents our Inductive

⁴In the error worksheet of Fig. 4.2, the error-invariant appears in four different places. In order to save space, here we list it only before the loop.

Hypothesis (I.H.). Algebraic manipulation of Step 7 yields

$$\begin{aligned}
m \left(\frac{A_{00}}{a_{10}^T} \middle| \frac{a_{01}}{\alpha_{11}} \right) &= k + 1 \wedge \\
&\left(\frac{\check{L}_{00}\check{U}_{00} = A_{00} + \Delta A_{00} \middle| \check{L}_{00}\check{u}_{01} = a_{01} + \delta a_{01} \middle| \check{L}_{00}\check{U}_{02} = A_{02} + \Delta A_{02}}{\check{l}_{10}^T\check{U}_{00} = a_{10}^T + \delta a_{10}^T \middle| \check{l}_{10}^T\check{u}_{01} + \check{v}_{11} = \alpha_{11} + \delta \alpha_{11} \middle| \check{l}_{10}^T\check{U}_{02} + \check{u}_{12}^T = a_{12}^T + \delta a_{12}^T} \right) \wedge \\
&\left(\frac{\check{L}_{20}\check{U}_{00} = A_{20} + \Delta A_{20} \middle| \check{L}_{20}\check{u}_{01} + \check{l}_{21}\check{v}_{11} = a_{21} + \delta a_{21}}{\left(\frac{\Delta A_{00} \middle| \delta a_{01} \middle| \Delta A_{02}}{\check{\delta} a_{10}^T \middle| \delta \alpha_{11} \middle| \check{\delta} a_{12}^T} \right)} \leq \gamma_{k+1} \left(\frac{\check{L}_{00}|\check{U}_{00}| \middle| \check{L}_{00}|\check{u}_{01}| \middle| \check{L}_{00}|\check{U}_{02}|}{\check{l}_{10}^T|\check{U}_{00}| \middle| \check{l}_{10}^T|\check{u}_{01}| + |\check{v}_{11}| \middle| \check{l}_{10}^T|\check{U}_{02}| + |\check{u}_{12}^T|} \right), \\
&\left(\frac{\Delta A_{20} \middle| \delta a_{21}}{\check{L}_{20}|\check{U}_{00}| \middle| \check{L}_{20}|\check{u}_{01}| + |\check{l}_{21}|\check{v}_{11}|} \right)
\end{aligned}$$

which is the predicate that must be *true* after the execution of the computational statements.

The goal is to show that the computation in Step 8, when executed in a state that satisfies predicate (5.2), generates errors that satisfy the relations in (5.3). For simplicity, in the latter predicate we have highlighted the submatrices that are affected by the computation. The relations in the other submatrices are satisfied “by the I.H.”, in the terminology of an inductive proof, since $\gamma_k \leq \gamma_{k+1}$. Thus it remains to show that there exist $\delta \alpha_{11}$, δa_{12}^T , and δa_{21} that satisfy the constraints in the grey-shaded boxes. To this end, we examine the error introduced by the computational updates to determine how error is contributed to each of these variables:

Determining $\delta \alpha_{11}$: The assignment $v_{11} := \alpha_{11} - l_{10}^T u_{01}$ is executed in a state where $m(A_{00}) = k$. Theorem 3.12 R2-F states that there exists δv_{11} such that

$$\begin{aligned}
l_{10}^T u_{01} + \check{v}_{11} &= \alpha_{11} + \delta v_{11}, \quad \text{where } |\delta v_{11}| \leq \gamma_k (|l_{10}^T| |u_{01}| + |\check{v}_{11}|) \\
&\leq \gamma_{k+1} (|l_{10}^T| |u_{01}| + |\check{v}_{11}|),
\end{aligned}$$

therefore we choose $\delta \alpha_{11} := \delta v_{11}$.

Determining δa_{12}^T : The assignment $u_{12}^T := a_{12}^T - l_{10}^T U_{02}$ executed in a state where $m(A_{00}) = k$ and Theorem 5.1 R2-F imply that there exists δv_{12} such that

$$\begin{aligned}
l_{10}^T U_{02} + \check{u}_{12}^T &= a_{12}^T + \delta v_{12}^T, \quad \text{where } |\delta v_{12}^T| \leq \gamma_k (|l_{10}^T| |U_{02}| + |\check{u}_{12}^T|) \\
&\leq \gamma_{k+1} (|l_{10}^T| |U_{02}| + |\check{u}_{12}^T|);
\end{aligned}$$

thus, $\delta a_{12}^T := \delta v_{12}^T$ is the desired update.

Determining δa_{21} : The assignment $l_{21} := (a_{21} - L_{20} u_{01}) / v_{11}$ executed in a state where $m(A_{00}) = k$, and Theorem 5.1 R4-F imply that there exists δv_{21} such that

$$L_{20} u_{01} + v_{11} \check{l}_{21} = a_{21} + \delta v_{21}, \quad \text{where } |\delta v_{21}| \leq \gamma_{k+1} (|L_{20}| |u_{01}| + |\check{l}_{21}| |v_{11}|);$$

therefore $\delta a_{21} := \delta v_{21}$ is the desired update.

This completes the proof of the Inductive Step. The proof is also summarized in Step 8 of Fig. 5.2.

5.4. Results. The above discussion, summarized in Fig. 5.2, proves the following theorem:

THEOREM 5.4. *Let $A \in \mathbb{R}^{n \times n}$, $n > 2$, and let \check{L} and \check{U} be computed via the Crout variant of the LU factorization, as given in Fig. 5.1. Then the computed factors \check{L} and \check{U} satisfy $\check{L}\check{U} = A + \Delta A$, where $|\Delta A| \leq \gamma_n |\check{L}||\check{U}|$.*

The backward stability result in this theorem agrees with the one in [11]. The attentive reader will have noticed that the none of the bounds used to determine

	Error side	Step
	$\{ \Delta A = 0 \}$	1a
Partition $A \rightarrow \left(\begin{array}{c c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right), L \rightarrow \left(\begin{array}{c c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array} \right), U \rightarrow \left(\begin{array}{c c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array} \right), \quad \Delta A \rightarrow \left(\begin{array}{c c} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & \Delta A_{BR} \end{array} \right)$ where A_{TL}, L_{TL}, U_{TL} and ΔA_{TL} are empty	4	
$\left\{ \begin{array}{l} m(A_{TL}) = k \wedge \left(\frac{\tilde{L}_{TL} \tilde{U}_{TL} = A_{TL} + \Delta A_{TL} \mid \tilde{L}_{TL} \tilde{U}_{TR} = A_{TR} + \Delta A_{TR}}{L_{BL} U_{TR} = A_{BL} + \Delta A_{BL}} \right) \\ \wedge \left(\frac{\Delta A_{TL} \mid \Delta A_{TR}}{\Delta A_{BL}} \right) \leq \gamma_k \left(\frac{ \tilde{L}_{TL} \tilde{U}_{TL} \mid \tilde{L}_{TL} \tilde{U}_{TR} }{ L_{BL} U_{TR} } \right) \end{array} \right\}$		2a
While $m(A_{TL}) \leq m(A)$ do		3
Repartition A, L, U partitioned as in Fig. 5.1, and $\left(\frac{\Delta A_{TL} \mid \Delta A_{TR}}{\Delta A_{BL} \mid \Delta A_{BR}} \right) \rightarrow \left(\frac{\Delta A_{00} \mid \delta \alpha_{01} \mid \Delta A_{02}}{\delta \alpha_{10}^T \mid \delta \alpha_{11} \mid \delta \alpha_{12}^T} \right)$ where $\delta \alpha_{11}$ is a scalars		5a
$\left\{ \begin{array}{l} m(A_{00}) = k \\ \wedge \left(\frac{\tilde{L}_{00} \tilde{U}_{00} = A_{00} + \Delta A_{00} \mid \tilde{L}_{00} \tilde{u}_{01} = a_{01} + \delta \alpha_{01} \mid \tilde{L}_{00} \tilde{U}_{02} = A_{02} + \Delta A_{02}}{l_{10}^T \tilde{U}_{00} = a_{10}^T + \delta \alpha_{10}^T \mid \tilde{L}_{20} \tilde{U}_{00} = A_{20} + \Delta A_{02}} \right) \\ \wedge \left(\frac{\Delta A_{00} \mid \delta \alpha_{01} \mid \Delta A_{02}}{\delta \alpha_{10}^T \mid \Delta A_{20}} \right) \leq \gamma_k \left(\frac{ \tilde{L}_{00} \tilde{U}_{00} \mid \tilde{L}_{00} \tilde{u}_{01} \mid \tilde{L}_{00} \tilde{U}_{02} }{ l_{10}^T \tilde{U}_{00} \mid L_{20} \tilde{U}_{00} } \right) \end{array} \right\}$		6
$\begin{array}{l} v_{11} := \alpha_{11} - l_{10}^T u_{01} \\ u_{12}^T := a_{12}^T - l_{10}^T U_{02} \\ l_{21} := (a_{21} - L_{20} u_{01}) / v_{11} \end{array}$	$\begin{array}{l} \tilde{v}_{11} + \delta \alpha_{11} = \alpha_{11} - l_{10}^T u_{01} \\ \wedge \delta \alpha_{11} \leq \gamma_{k+1} (l_{10}^T u_{01} + \tilde{v}_{11}) \quad \text{Th. 3.12 R2-F} \\ \tilde{u}_{12}^T + \delta \alpha_{12}^T = a_{12}^T - l_{10}^T U_{02} \\ \wedge \delta \alpha_{12}^T \leq \gamma_{k+1} (l_{10}^T U_{02} + \tilde{u}_{12}^T) \quad \text{Th. 5.1 R2-F} \\ \tilde{l}_{21} v_{11} + \delta \alpha_{21} = a_{21} - L_{20} u_{01} \\ \wedge \delta \alpha_{21} \leq \gamma_{k+1} (\tilde{L}_{20} \tilde{u}_{01} + \tilde{l}_{21} v_{11}) \quad \text{Th. 5.1 R4-F} \end{array}$	8
$\left\{ \begin{array}{l} m \left(\frac{A_{00} \mid a_{01}}{a_{10}^T \mid \alpha_{11}} \right) = k + 1 \\ \wedge \left(\frac{\tilde{L}_{00} \tilde{U}_{00} = A_{00} + \Delta A_{00} \mid \tilde{L}_{00} \tilde{u}_{01} = a_{01} + \delta \alpha_{01} \mid \tilde{L}_{00} \tilde{U}_{02} = A_{02} + \Delta A_{02}}{l_{10}^T \tilde{U}_{00} = a_{10}^T + \delta \alpha_{10}^T \mid l_{10}^T \tilde{u}_{01} + \tilde{v}_{11} = \alpha_{11} + \delta \alpha_{11} \mid l_{10}^T \tilde{U}_{02} + \tilde{u}_{12}^T = a_{12}^T + \delta \alpha_{12}^T} \right) \\ \wedge \left(\frac{\Delta A_{00} \mid \delta \alpha_{01} \mid \Delta A_{02}}{\delta \alpha_{10}^T \mid \delta \alpha_{11} \mid \delta \alpha_{12}^T} \right) \leq \gamma_{k+1} \left(\frac{ \tilde{L}_{00} \tilde{U}_{00} \mid \tilde{L}_{00} \tilde{u}_{01} \mid \tilde{L}_{00} \tilde{U}_{02} }{ l_{10}^T \tilde{U}_{00} \mid l_{10}^T \tilde{u}_{01} + \tilde{v}_{11} \mid l_{10}^T \tilde{U}_{02} + \tilde{u}_{12}^T } \right) \end{array} \right\}$		7
Continue with A, L and U as in Fig. 5.1, and $\left(\frac{\Delta A_{TL} \mid \Delta A_{TR}}{\Delta A_{BL} \mid \Delta A_{BR}} \right) \leftarrow \left(\frac{\Delta A_{00} \mid \delta \alpha_{01} \mid \Delta A_{02}}{\delta \alpha_{10}^T \mid \delta \alpha_{11} \mid \delta \alpha_{12}^T} \right)$		5b
endwhile		
	$\{ m(A) = n \wedge \tilde{L} \tilde{U} = (A + \Delta A) \wedge \Delta A \leq \gamma_n L U \}$	1b

FIG. 5.2. $LU = A$. Error worksheet for proving the backward stability of the Crout variant for the LU factorization.

$\delta\alpha_{11}, \delta\alpha_{12}^T, \delta\alpha_{21}$ was tight. As a result, with a little more work it can be shown that the bound for ΔA can be improved:

EXERCISE 5.5. *Prove Theorem 5.4 replacing γ_n by $\max(\gamma_2, \gamma_{n-1})$. Hint: use Exercise 4.3.*

5.5. Other algorithmic variants. The described approach can be similarly applied to the other four variants for computing the LU factorization. In [3] it is shown how to use the error worksheet to derive an error result for the so-called bordered variant.

5.6. Solution of a linear system. The result of an LU factorization is most often used to solve a linear system of equations: If $Ax = b$, then $L(Ux) = b$ so that, after factoring A , the triangular solves $Lz = b$ followed by $Ux = z$ can be performed, yielding the solution vector x . The following theorem summarizes how the computed solution, \tilde{x} , is the exact solution of a perturbed linear system $(A + \Delta A)\tilde{x} = y$:

THEOREM 5.6. *Let $A \in \mathbb{R}^{n \times n}$ be a nonsingular matrix with $n > 2$. Let \check{L} and \check{U} be the LU factorization computed by the algorithm in Fig. 5.1. Let \check{z} be the solution to $\check{L}z = y$ computed with the algorithm in Fig. 4.1. Let \tilde{x} be the solution to $\check{U}x = \check{z}$ computed with an algorithm for solving an upper triangular system similar to the algorithm in Fig. 4.1. Then \tilde{x} satisfies $(A + \Delta A)\tilde{x} = b$ with $|\Delta A| \leq (3\gamma_n + \gamma_n^2)|\check{L}||\check{U}|$.*

Proof: From previous sections we have learned that

$$\begin{aligned} \check{L}\check{U} &= A + E_1 \quad \wedge \quad |E_1| \leq \gamma_n |\check{L}||\check{U}| && \text{(Th. 5.4)} \\ (\check{L} + E_2)\check{z} &= b \quad \wedge \quad |E_2| \leq \gamma_n |\check{L}| && \text{(Th. 4.1)} \\ (\check{U} + E_3)\tilde{x} &= \check{z} \quad \wedge \quad |E_3| \leq \gamma_n |\check{U}| && \text{(Th. 4.1)}. \end{aligned}$$

Now,

$$\begin{aligned} b &= (\check{L} + E_2)(\check{U} + E_3)\tilde{x} = (\check{L}\check{U} + E_2\check{U} + E_3\check{L} + E_2E_3)\tilde{x} \\ &= (A + \underbrace{E_1 + E_2\check{U} + E_3\check{L} + E_2E_3}_{\Delta A})\tilde{x} = (A + \Delta A)\tilde{x}, \end{aligned}$$

where

$$\begin{aligned} |\Delta A| &= |E_1 + E_2\check{U} + E_3\check{L} + E_2E_3| \leq |E_1| + |E_2||\check{U}| + |E_3||\check{L}| + |E_2||E_3| \\ &\leq \gamma_n |\check{L}||\check{U}| + \gamma_n |\check{L}||\check{U}| + \gamma_n |\check{L}||\check{U}| + \gamma_n^2 |\check{L}||\check{U}| = (3\gamma_n + \gamma_n^2)|\check{L}||\check{U}|. \end{aligned}$$

□

5.7. Partial pivoting. We remind the reader now of how the analysis of LU factorization without partial pivoting is related that of LU factorization with partial pivoting. Invariably, it is argued that LU with partial pivoting is equivalent to the LU factorization without partial pivoting on a pre-permuted matrix $PA = LU$, where P is a permutation matrix. The permutation doesn't involve any floating point operations and therefore does not generate error. It is then argued that, as a result, the error that is accumulated is equivalent with or without partial pivoting.

6. Analysis of a Blocked Algorithm for LU Factorization. When targeting high-performance, dense matrix computations like the LU factorization are formulated as blocked algorithms so that the bulk of computation is cast in terms of highly efficient matrix-matrix multiplications [1, 6]. The numerical properties of the resulting algorithms are rarely analyzed. Rather, an informal argument is given

	Error side	Step
	$\{\Delta A = 0\}$	1a
Partition $X \rightarrow \left(\begin{array}{c c} X_{TL} & X_{TR} \\ \hline X_{BL} & X_{BR} \end{array} \right)$ with $X \in \{A, L, U\}$, where X_{TL} and ΔA_{TL} are empty	$\Delta A \rightarrow \left(\begin{array}{c c} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & \Delta A_{BR} \end{array} \right)$	4
$\left\{ \begin{array}{l} m(A_{TL}) = k \wedge \\ \left(\begin{array}{c c} \check{L}_{TL}\check{U}_{TL} = A_{TL} + \Delta A_{TL} & \check{L}_{TL}\check{U}_{TR} = A_{TR} + \Delta A_{TR} \\ \hline \check{L}_{BL}\check{U}_{TL} = A_{BL} + \Delta A_{BL} & A_{BR} = A_{BR} - \check{L}_{BL}\check{U}_{TR} + \Delta A_{BR} \end{array} \right) \wedge \\ \left \left(\begin{array}{c c} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & \Delta A_{BR} \end{array} \right) \right \leq \gamma_{\frac{k}{b}+b} \left(\begin{array}{c c c} A_{TL} + \check{L}_{TL} \check{U}_{TL} & A_{TR} + \check{L}_{TL} \check{U}_{TR} & \\ \hline A_{BL} + \check{L}_{BL} \check{U}_{TL} & A_{BR} + \check{L}_{BL} \check{U}_{TR} & \end{array} \right) \end{array} \right\}$		2a
While $m(A_{TL}) \leq m(A)$ do		3
Repartition $\left(\begin{array}{c c} X_{TL} & X_{TR} \\ \hline X_{BL} & X_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c c c} X_{00} & X_{01} & X_{02} \\ \hline X_{10} & X_{11} & X_{12} \\ \hline X_{20} & X_{21} & X_{22} \end{array} \right)$, $\left(\begin{array}{c c} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & \Delta A_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c c c} \Delta A_{00} & \Delta A_{01} & \Delta A_{02} \\ \hline \Delta A_{10} & \Delta A_{11} & \Delta A_{12} \\ \hline \Delta A_{20} & \Delta A_{21} & \Delta A_{22} \end{array} \right)$ where $X \in \{A, L, U\}$, and X_{11} and ΔA_{11} are $b \times b$		5a
	$\{\text{See Fig. 6.2}\}$	6
$[L_{11}, U_{11}] := \text{LU}(A_{11})$ $U_{12} := L_{11}^{-1}A_{12}$ $L_{21} := A_{21}U_{11}^{-1}$ $A_{22} := A_{22} - L_{21}U_{12}$	See Fig. 6.2	8
	$\{\text{See Fig. 6.2}\}$	7
Continue with $\left(\begin{array}{c c} X_{TL} & X_{TR} \\ \hline X_{BL} & X_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c c c} X_{00} & X_{01} & X_{02} \\ \hline X_{10} & X_{11} & X_{12} \\ \hline X_{20} & X_{21} & X_{22} \end{array} \right)$, $\left(\begin{array}{c c} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & \Delta A_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c c c} \Delta A_{00} & \Delta A_{01} & \Delta A_{02} \\ \hline \Delta A_{10} & \Delta A_{11} & \Delta A_{12} \\ \hline \Delta A_{20} & \Delta A_{21} & \Delta A_{22} \end{array} \right)$ with $X \in \{A, L, U\}$		5b
endwhile		
	$\{m(A) = n \wedge \check{L}\check{U} = (A + \Delta A) \wedge \Delta A \leq \gamma_{\frac{n}{b}+b}(A + \check{L} \check{U})\}$	1b

FIG. 6.1. $LU = A$. Error worksheet for proving the backward stability of the blocked LU factorization computed via the right-looking variant.

that the computations occur in a similar order on similar data and that therefore the numerical properties are similar. For LU factorization, a paper by Schreiber, Demmel, and Higham is often referenced as the paper that analyzes a blocked LU factorization [5]. Unfortunately, the blocked algorithm that is analyzed in that paper does not resemble the algorithm that is used in practice by libraries like LAPACK and FLAME [1, 10].

In this section we apply the methodology to an algorithm that is very close to the one used in practice, the so-called right-looking variant. How the analysis extends to the practical algorithm is briefly discussed at the end of the section.

6.1. A blocked algorithm for computing the LU factorization. A right-looking algorithm for computing the LU factorization of a square matrix A is given on the left side of Fig. 6.1. While in practice A is overwritten, for the analysis we assume that new matrices L and U are computed. The computation performed by this algorithm is such that the predicate

$$\left(\begin{array}{c|c} L_{TL}U_{TL} = A_{TL} & L_{TL}U_{TR} = A_{TR} \\ \hline L_{BL}U_{TL} = A_{BL} & A_{BR}^{i+1} = A_{BR} - L_{BL}U_{TR} \end{array} \right)$$

$m(ATL) = k \wedge \left(\frac{L_{TL} \tilde{U}_{TL} = A_{TL} + \Delta_{TL}}{L_{BL} \tilde{U}_{TL} = A_{BL} + \Delta_{BL}} \right) \wedge \left(\frac{\Delta_{TL}}{\Delta_{BL}} \right) \leq \gamma_b^{k+b}$	\dots	$m(A00) = k \wedge \left(\frac{L_{TL} \tilde{U}_{TR} = A_{TR} + \Delta_{TR}}{A_{BR} = A_{BR} - L_{BL} \tilde{U}_{TR} + \Delta_{BR}} \right) \wedge \left(\frac{A_{TR}}{A_{BR}} \right) \leq \gamma_b^{k+b}$	2a
$m(A00) = k \wedge \left(\frac{A_{11}^i = A_{11} - \tilde{L}_{10} \tilde{U}_{01} + \Delta_{11}^i}{A_{21}^i = A_{21} - \tilde{L}_{20} \tilde{U}_{01} + \Delta_{21}^i} \right) \wedge \left(\frac{\Delta_{11}^i}{\Delta_{21}^i} \right) \leq \gamma_b^{k+b}$	\dots	$\left(\frac{A_{11} + \tilde{L}_{10} \tilde{U}_{01}}{A_{21} + \tilde{L}_{20} \tilde{U}_{01}} \right) \leq \gamma_b^{k+b}$	6
$[L_{11}, \tilde{U}_{11}] := LU(A_{11})$ $U_{12} := L_{11}^{-1} A_{12}$ $L_{21} := A_{21} U_{11}^{-1}$ $A_{22} := A_{22} - L_{21} U_{12}$	$\tilde{L}_{11} \tilde{U}_{11} = A_{11} + \Delta_{11}$ $\tilde{L}_{11} \tilde{U}_{12} = A_{12} + \Delta_{12}$ $\tilde{L}_{21} \tilde{U}_{11} = A_{21} + \Delta_{21}$ $\tilde{A}_{22}^{i+1} + \tilde{L}_{21} \tilde{U}_{12} = A_{22}^i + \Delta_{22}^i \wedge \tilde{A}_{22}^{i+1} \leq \gamma_{b+1} (B_{22}^i + \tilde{L}_{21} \tilde{U}_{12})$	$\wedge \Delta_{11}^i \leq \gamma_b \tilde{L}_{11} \tilde{U}_{11} $ $\wedge \Delta_{12}^i \leq \gamma_b \tilde{L}_{11} \tilde{U}_{12} $ $\wedge \Delta_{21}^i \leq \gamma_b \tilde{L}_{21} \tilde{U}_{11} $ Theorem 5.4 Corollary 6.3 Corollary 6.3 Corollary 6.1	8
$m \left(\begin{array}{c c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array} \right) = k + b \wedge \left(\frac{\tilde{L}_{11} \tilde{U}_{11} = A_{11} - \tilde{L}_{10} \tilde{U}_{01} + \Delta_{11}^{i+1}}{\tilde{L}_{20} \tilde{U}_{01} + \tilde{L}_{21} \tilde{U}_{11} = A_{21} + \Delta_{21}^{i+1}} \right) \wedge \left(\frac{\Delta_{11}^{i+1}}{\Delta_{21}^{i+1}} \right) \leq \gamma_b^{k+b+1}$	\dots	$\left(\frac{\tilde{L}_{10} \tilde{U}_{02} + \tilde{L}_{11} \tilde{U}_{12} = A_{12} + \Delta_{12}^{i+1}}{\tilde{A}_{22}^{i+1} = A_{22} - \tilde{L}_{20} \tilde{U}_{02} - \tilde{L}_{21} \tilde{U}_{12} + \Delta_{22}^{i+1}} \right) \wedge \left(\frac{A_{11} + \tilde{L}_{10} \tilde{U}_{01}}{A_{21} + \tilde{L}_{20} \tilde{U}_{01}} \right) \leq \gamma_b^{k+b+1}$	7

FIG. 6.2. $LU = A$. Details for Steps 6–8 in the proof of Theorem 6.5.

is satisfied at the beginning and at the end of each iteration. Superscripts denote the iteration number.

6.2. Preparation. The computation in Fig. 6.1 (left) is cast in terms of an unblocked LU factorization, $LU(A_{11})$, two triangular solves with multiple right-hand sides, $L_{11}U_{12} = A_{12}$ and $L_{21}U_{11} = A_{21}$, and one matrix-matrix multiplication, $A_{22} - L_{21}U_{12}$. An error result for the unblocked LU factorization, if the Crout variant is used, is given in Theorem 5.4. Here we present theorems related to triangular solve with multiple right hand sides and matrix-matrix multiplication.

Error results for matrix-matrix multiplication.

COROLLARY 6.1. *Let $C \in \mathbb{R}^{m \times n}$, $A \in \mathbb{R}^{m \times k}$, and $B \in \mathbb{R}^{k \times n}$. Assume that $Z := C - AB$ is computed one column at a time by the matrix-vector multiplication discussed in Theorem 5.1. Partition Z, C and B by columns as $Z \rightarrow (z_0 | \cdots | z_{n-1})$, $B \rightarrow (b_0 | \cdots | b_{n-1})$, and $C \rightarrow (c_0 | \cdots | c_{n-1})$, so that $Z = (c_0 - Ab_0 | \cdots | c_{n-1} - Ab_{n-1}) = C - AB$. Then*

R1-F: $AB + \check{Z} = C + \Delta Z$, where $|\Delta Z| \leq \gamma_k |A| |B| + \gamma_1 |\check{Z}|$.

R2-F: $AB + \check{Z} = C + \Delta C$, where $|\Delta C| \leq \gamma_{k+1} |A| |B| + \gamma_1 |\check{C}|$.

EXERCISE 6.2. *Prove Corollary 6.1. Hint: use Theorem 5.1.*

Error results for triangular solve with multiple right-hand sides. Let $L \in \mathbb{R}^{m \times m}$, be a lower triangular matrix (no assumptions on the diagonal entries), and $X, B \in \mathbb{R}^{m \times n}$, where L and B are input operands, and matrix X is to be computed. Partition $X \rightarrow (x_0 | \cdots | x_{n-1})$ and $B \rightarrow (b_0 | \cdots | b_{n-1})$. Then

$$LX = L(x_0 | \cdots | x_{n-1}) = (Lx_0 | \cdots | Lx_{n-1}) = (b_0 | \cdots | b_{n-1})$$

so that $Lx_j = b_j$. This explains the name of the operation: each column of X is computed by solving a lower triangular system with the corresponding column of B as right-hand side.

The following result follows immediately from Theorem 4.1:

COROLLARY 6.3. *Let $L \in \mathbb{R}^{m \times m}$ be nonsingular lower triangular matrix and $X, B \in \mathbb{R}^{m \times n}$. Assume that the solution X to $LX = B$ is computed one column at a time via the algorithm in Fig. 4.2(left side). Then \check{X} satisfies $L\check{X} = B + \Delta B$ where $|\Delta B| \leq \gamma_n |L| |\check{X}|$.*

EXERCISE 6.4. *Discuss why in Cor. 6.3 there is no result that corresponds to Theorem 4.1 R1.*

There are a number of different algorithms for computing this operation, each of which has a similar error result.

6.3. Analysis. As for the unblocked algorithm discussed in the previous section, the right-looking LU factorization algorithm, in the presence of round-off error, computes factors \check{L} and \check{U} . This section provides the proof to the following theorem.

THEOREM 6.5. *Given $A \in \mathbb{R}^{n \times n}$, assume that the blocked right-looking algorithm in Fig. 6.1 completes. Then the computed factors \check{L} and \check{U} are such that*

$$\check{L}\check{U} = A + \Delta A, \quad \text{where } |\Delta A| \leq \gamma_{\frac{n}{b}+b}(|A| + |\check{L}||\check{U}|).$$

The worksheet containing the proof of Theorem 6.5 is shown in in Figg. 6.1 (right) and 6.2. In the remainder of the section we will prove that the error bounds indicated

in the worksheet hold. Consider the error-invariant

$$\begin{aligned}
m(A_{TL}) &= k \wedge \\
&\left(\frac{\check{L}_{TL}\check{U}_{TL} = A_{TL} + \Delta A_{TL} \mid \check{L}_{TL}\check{U}_{TR} = A_{TR} + \Delta A_{TR}}{\check{L}_{BL}\check{U}_{TL} = A_{BL} + \Delta A_{BL} \mid \check{A}_{BR} = A_{BR} - \check{L}_{BL}\check{U}_{TR} + \Delta A_{BR}} \right) \wedge \\
&\left| \left(\frac{\Delta A_{TL} \mid \Delta A_{TR}}{\Delta A_{BL} \mid \Delta A_{BR}} \right) \right| \leq \gamma_{\frac{k}{b}+b} \left(\frac{|A_{TL}| + |\check{L}_{TL}|\|\check{U}_{TL}\| \mid |A_{TR}| + |\check{L}_{TL}|\|\check{U}_{TR}\|}{|A_{TL}| + |\check{L}_{BL}|\|\check{U}_{TL}\| \mid |A_{BR}| + |\check{L}_{BL}|\|\check{U}_{TR}\|} \right)
\end{aligned} \tag{6.1}$$

which results from restricting the target error result to the state of the variables at the top of each iteration. The base case of the proof requires this predicate to be satisfied right after the initialization of Step 4. The initialization sets A_{TL} to be an empty matrix; therefore predicate (6.1) reduces to

$$\check{A}_{BR} = A_{BR} + \Delta A_{BR} \wedge |\Delta A_{BR}| \leq \gamma_b |A_{BR}|,$$

which is true as no computation has been performed yet, thus \check{A}_{BR} equals A_{BR} , and $\Delta A_{BR} = 0$.

We know that the predicate in Step 6 of the worksheet represents the inductive hypothesis. This predicate follows immediately by substituting the submatrices from Steps 5a into the error-invariant. Algebraic manipulation yields

$$\begin{aligned}
m(A_{00}) &= k \wedge \\
&\left(\frac{\check{L}_{00}\check{U}_{00} = A_{00} + \Delta A_{00} \mid \check{L}_{00}\check{U}_{01} = A_{01} + \Delta A_{01} \mid \check{L}_{00}\check{U}_{02} = A_{02} + \Delta A_{02}}{\check{L}_{10}\check{U}_{00} = A_{10} + \Delta A_{10} \mid \check{A}_{11} = A_{11} - \check{L}_{10}\check{U}_{01} + \Delta A_{11}^i \mid \check{A}_{12} = A_{12} - \check{L}_{10}\check{U}_{02} + \Delta A_{12}^i} \right) \wedge \\
&\left(\frac{\check{L}_{20}\check{U}_{00} = A_{20} + \Delta A_{02} \mid \check{A}_{21} = A_{21} - \check{L}_{20}\check{U}_{01} + \Delta A_{21}^i \mid \check{A}_{22} = A_{22} - \check{L}_{20}\check{U}_{02} + \Delta A_{22}^i}{\left(\frac{\Delta A_{00} \mid \Delta A_{01} \mid \Delta A_{02}}{\Delta A_{10} \mid \Delta A_{11}^i \mid \Delta A_{12}^i} \right)} \right) \leq \gamma_{\frac{k}{b}+b} \left(\frac{\begin{array}{c} |A_{00}| + |\check{L}_{00}|\|\check{U}_{00}\| \mid |A_{01}| + |\check{L}_{00}|\|\check{U}_{01}\| \mid |A_{02}| + |\check{L}_{00}|\|\check{U}_{02}\| \\ |A_{10}| + |\check{L}_{10}|\|\check{U}_{00}\| \mid |A_{11}| + |\check{L}_{10}|\|\check{U}_{01}\| \mid |A_{12}| + |\check{L}_{10}|\|\check{U}_{02}\| \\ |A_{20}| + |\check{L}_{20}|\|\check{U}_{00}\| \mid |A_{21}| + |\check{L}_{20}|\|\check{U}_{01}\| \mid |A_{22}| + |\check{L}_{20}|\|\check{U}_{02}\| \end{array}}{\left(\frac{\Delta A_{00} \mid \Delta A_{01} \mid \Delta A_{02}}{\Delta A_{10} \mid \Delta A_{11}^i \mid \Delta A_{12}^i} \right)} \right).
\end{aligned} \tag{6.2}$$

Similarly, the predicate in Step 7 represents the relations that have to be satisfied at the end of each iteration. The predicate is obtained by substituting the submatrices from Steps 5b into the error-invariant. Algebraic manipulation yields

$$\begin{aligned}
m \left(\frac{A_{00} \mid A_{01}}{A_{10} \mid A_{11}} \right) &= k + b \wedge \\
&\left(\frac{\begin{array}{c} \star \mid \star \\ \star \check{L}_{10}\check{U}_{01} + \check{L}_{11}\check{U}_{11} = A_{11} + \Delta A_{11}^{i+1} \mid \check{L}_{10}\check{U}_{02} + \check{L}_{11}\check{U}_{12} = A_{12} + \Delta A_{12}^{i+1} \\ \star \check{L}_{20}\check{U}_{01} + \check{L}_{21}\check{U}_{11} = A_{21} + \Delta A_{21}^{i+1} \mid \check{A}_{22}^{i+1} = A_{22} - \check{L}_{20}\check{U}_{02} - \check{L}_{21}\check{U}_{12} + \Delta A_{22}^{i+1} \end{array}}{\left(\frac{\star \mid \star \mid \star}{\star \Delta A_{11}^{i+1} \mid \Delta A_{12}^{i+1}} \right)} \right) \wedge \\
&\left| \left(\frac{\begin{array}{c} \star \mid \star \mid \star \\ \star \Delta A_{11}^{i+1} \mid \Delta A_{12}^{i+1} \\ \star \Delta A_{21}^{i+1} \mid \Delta A_{22}^{i+1} \end{array}}{\left(\frac{\star \mid \star \mid \star}{\star |A_{11}| + |\check{L}_{10}|\|\check{U}_{01}\| + |\check{L}_{11}|\|\check{U}_{11}\| \mid |A_{12}| + |\check{L}_{10}|\|\check{U}_{02}\| + |\check{L}_{11}|\|\check{U}_{12}\|} \right)} \right) \right| \leq \gamma_{\frac{k}{b}+b+1} \left(\frac{\begin{array}{c} \star \mid \star \mid \star \\ \star |A_{11}| + |\check{L}_{10}|\|\check{U}_{01}\| + |\check{L}_{11}|\|\check{U}_{11}\| \mid |A_{12}| + |\check{L}_{10}|\|\check{U}_{02}\| + |\check{L}_{11}|\|\check{U}_{12}\| \\ \star |A_{21}| + |\check{L}_{20}|\|\check{U}_{01}\| + |\check{L}_{21}|\|\check{U}_{11}\| \mid |A_{22}| + |\check{L}_{20}|\|\check{U}_{02}\| + |\check{L}_{21}|\|\check{U}_{12}\| \end{array}}{\left(\frac{\star \mid \star \mid \star}{\star \Delta A_{11}^{i+1} \mid \Delta A_{12}^{i+1}} \right)} \right).
\end{aligned} \tag{6.3}$$

where the \star indicates that the expression is identical to that in the corresponding result in (6.2) above.

The goal is to prove that the updates in Step 8 (left), when executed in a state that satisfies predicate (6.2), generate errors that satisfy the predicate (6.3). The constraints in (6.3) that are not highlighted are already satisfied in Step 6, and since the computation only affects submatrices L_{11} , L_{21} , U_{11} , U_{12} and A_{22} , they are also satisfied in Step 7, as $\gamma_{\frac{k}{b}+b} \leq \gamma_{\frac{k}{b}+b+1}$.

It remains to show that there exist ΔA_{11}^{i+1} , ΔA_{12}^{i+1} , ΔA_{21}^{i+1} , and ΔA_{22}^{i+1} that satisfy the constraints in the grey-shaded boxes. We examine the error introduced by the computational updates to establish how error is contributed to each of these variables: **Determining ΔA_{11}^{i+1} :** The update is $[L_{11}, U_{11}] := \text{LU}(\check{A}_{11})$, with $\check{A}_{11} \in \mathbb{R}^{b \times b}$. Theorem 5.4 states that there exists a matrix $\Delta A_{11}^{(\text{LU})}$ such that

$$\check{L}_{11}\check{U}_{11} = \check{A}_{11} + \Delta A_{11}^{(\text{LU})}, \text{ with } |\Delta A_{11}^{(\text{LU})}| \leq \gamma_b |\check{L}_{11}| |\check{U}_{11}|. \quad (6.4)$$

From Step 6 (predicate (6.2)), we know that

$$\check{A}_{11} = A_{11} - \check{L}_{10}\check{U}_{01} + \Delta A_{11}^i, \text{ with } |\Delta A_{11}^i| \leq \gamma_{\frac{k}{b}+b} (|A_{11}| + |\check{L}_{10}| |\check{U}_{01}|),$$

and substituting \check{A}_{11} into (6.4), it results:

$$\check{L}_{11}\check{U}_{11} = A_{11} - \check{L}_{10}\check{U}_{01} + \Delta A_{11}^i + \Delta A_{11}^{(\text{LU})}.$$

Therefore, rearranging and setting $\Delta A_{11}^{i+1} = \Delta A_{11}^i + \Delta A_{11}^{(\text{LU})}$, we obtain

$$\check{L}_{10}\check{U}_{01} + \check{L}_{11}\check{U}_{11} = A_{11} + \Delta A_{11}^{i+1},$$

which is the expression we were looking for, as shown in (6.3). Next, we prove that $|\Delta A_{11}^{i+1}| \leq \gamma_{\frac{k}{b}+b+1} (|A_{11}| + |\check{L}_{10}| |\check{U}_{01}| + |\check{L}_{11}| |\check{U}_{11}|)$:

$$\begin{aligned} |\Delta A_{11}^{i+1}| &= \left| \Delta A_{11}^i + \Delta A_{11}^{(\text{LU})} \right| && \text{Def.} \\ &\leq \left| \Delta A_{11}^i \right| + \left| \Delta A_{11}^{(\text{LU})} \right| && \text{Triangle Ineq.} \\ &\leq |\Delta A_{11}^i| + \gamma_b |\check{L}_{11}| |\check{U}_{11}| && \text{Theorem 5.4} \\ &\leq \gamma_{\frac{k}{b}+b} (|A_{11}| + |\check{L}_{10}| |\check{U}_{01}|) + \gamma_b |\check{L}_{11}| |\check{U}_{11}| && \text{I.H.} \\ &\leq \gamma_{\frac{k}{b}+b} (|A_{11}| + |\check{L}_{10}| |\check{U}_{01}| + |\check{L}_{11}| |\check{U}_{11}|) \\ &\leq \gamma_{\frac{k}{b}+b+1} (|A_{11}| + |\check{L}_{10}| |\check{U}_{01}| + |\check{L}_{11}| |\check{U}_{11}|). && \square \end{aligned}$$

Determining ΔA_{12}^{i+1} and ΔA_{21}^{i+1} : We first analyze the update $U_{12} := \check{L}_{11}^{-1} \check{A}_{12}$, with $\check{L}_{11} \in \mathbb{R}^{b \times b}$. The same analysis is directly applicable to $L_{12} := A_{21} U_{11}^{-1}$ too, by transposing the operands.

Corollary 6.3 states that there exists an error matrix $\Delta A_{12}^{(\text{TRSM})}$ such that

$$\check{L}_{11}\check{U}_{12} = \check{A}_{12} + \Delta A_{12}^{(\text{TRSM})}, \text{ with } |\Delta A_{12}^{(\text{TRSM})}| \leq \gamma_b |\check{L}_{11}| |\check{U}_{12}|. \quad (6.5)$$

From Step 6 (predicate (6.2)), we know that

$$\check{A}_{12} = A_{12} - \check{L}_{10}\check{U}_{02} + \Delta A_{12}^i, \text{ with } |\Delta A_{12}^i| \leq \gamma_{\frac{k}{b}+b} (|A_{12}| + |\check{L}_{10}| |\check{U}_{02}|),$$

and substituting \check{A}_{12} into (6.5) we obtain

$$\check{L}_{11}\check{U}_{12} = A_{12} - \check{L}_{10}\check{U}_{02} + \Delta A_{12}^i + \Delta A_{12}^{(\text{TRSM})}.$$

Rearranging, and setting $\Delta A_{12}^{i+1} = \Delta A_{12}^i + \Delta A_{12}^{(\text{TRSM})}$, it results

$$\check{L}_{10}\check{U}_{02} + \check{L}_{11}\check{U}_{12} = A_{12} + \Delta A_{12}^{i+1},$$

which is the expression we were looking for, as shown in (6.3). Next, we prove that $|\Delta A_{12}^{i+1}| \leq \gamma_{\frac{k}{b}+b+1}(|A_{12}| + |\check{L}_{10}||\check{U}_{02}| + |\check{L}_{11}||\check{U}_{12}|)$:

$$\begin{aligned}
|\Delta A_{12}^{i+1}| &= \left| \Delta A_{12}^i + \Delta A_{12}^{(\text{TRSM})} \right| && \text{Def.} \\
&\leq |\Delta A_{12}^i| + \left| \Delta A_{12}^{(\text{TRSM})} \right| && \text{Triangle Ineq.} \\
&\leq |\Delta A_{12}^i| + \gamma_b |\check{L}_{11}| |\check{U}_{12}| && \text{Corollary 6.3} \\
&\leq \gamma_{\frac{k}{b}+b} (|A_{12}| + |\check{L}_{10}||\check{U}_{02}|) + \gamma_b |\check{L}_{11}| |\check{U}_{12}| && \text{I.H.} \\
&\leq \gamma_{\frac{k}{b}+b} (|A_{12}| + |\check{L}_{10}||\check{U}_{02}| + |\check{L}_{11}||\check{U}_{12}|) \\
&\leq \gamma_{\frac{k}{b}+b+1} (|A_{12}| + |\check{L}_{10}||\check{U}_{02}| + |\check{L}_{11}||\check{U}_{12}|). && \square
\end{aligned}$$

Determining ΔA_{22}^{i+1} : The update $A_{22}^{i+1} := \check{A}_{22}^i - \check{L}_{21} \check{U}_{12}$, where $m(U_{12}) = n(L_{21}) = b$. Corollary 6.1 states that there exists an error matrix $\Delta A_{22}^{(\text{GEMM})}$ such that

$$\check{A}_{22}^{i+1} = \check{A}_{22}^i - \check{L}_{21} \check{U}_{12} + \Delta A_{22}^{(\text{GEMM})}, \quad \text{with } |\Delta A_{22}^{(\text{GEMM})}| \leq \gamma_{b+1} |\check{L}_{21}| |\check{U}_{12}| + \gamma_1 |\check{A}_{22}^i|. \quad (6.6)$$

Predicate (6.2) from Step 6 tells us that

$$\check{A}_{22}^i = A_{22} - \check{L}_{20} \check{U}_{02} + \Delta A_{22}^i, \quad \text{with } |\Delta A_{22}^i| \leq \gamma_{\frac{k}{b}+b} (|A_{22}| + |\check{L}_{20}||\check{U}_{02}|),$$

and substituting \check{A}_{22}^i into (6.6), it results

$$\check{A}_{22}^{i+1} = A_{22} - \check{L}_{20} \check{U}_{02} + \Delta A_{22}^i - \check{L}_{21} \check{U}_{12} + \Delta A_{22}^{(\text{GEMM})}.$$

Rearranging, and setting $\Delta A_{22}^{i+1} = \Delta A_{22}^i + \Delta A_{22}^{(\text{GEMM})}$, we obtain

$$\check{A}_{22}^{i+1} = A_{22} - \check{L}_{20} \check{U}_{02} - \check{L}_{21} \check{U}_{12} + \Delta A_{22}^{i+1},$$

which is the expression we were seeking as shown in (6.3). Next, we prove that $|\Delta A_{22}^{i+1}| \leq \gamma_{\frac{k}{b}+b+1} (|A_{22}| + |\check{L}_{20}||\check{U}_{02}| + |\check{L}_{21}||\check{U}_{12}|)$:

$$\begin{aligned}
|\Delta A_{22}^{i+1}| &= \left| \Delta A_{22}^i + \Delta A_{22}^{(\text{GEMM})} \right| && \text{Def.} \\
&\leq |\Delta A_{22}^i| + \left| \Delta A_{22}^{(\text{GEMM})} \right| && \text{Tr. Ineq.} \\
&\leq |\Delta A_{22}^i| + \gamma_{b+1} |\check{L}_{21}| |\check{U}_{12}| + \gamma_1 |\check{A}_{22}^i| && \text{Cor. 6.1} \\
&= |\Delta A_{22}^i| + \gamma_{b+1} |\check{L}_{21}| |\check{U}_{12}| + \gamma_1 |A_{22} - \check{L}_{20} \check{U}_{02} + \Delta A_{22}^i| && \text{I.H.} \\
&\leq |\Delta A_{22}^i| + \gamma_{b+1} |\check{L}_{21}| |\check{U}_{12}| + \gamma_1 (|A_{22}| + |\check{L}_{20}||\check{U}_{02}| + |\Delta A_{22}^i|) && \text{Tr. Ineq.} \\
&= (1 + \gamma_1) |\Delta A_{22}^i| + \gamma_{b+1} |\check{L}_{21}| |\check{U}_{12}| + \gamma_1 (|A_{22}| + |\check{L}_{20}||\check{U}_{02}|) \\
&\leq (1 + \gamma_1) \gamma_{\frac{k}{b}+b} (|A_{22}| + |\check{L}_{20}||\check{U}_{02}|) + \\
&\quad \gamma_{b+1} |\check{L}_{21}| |\check{U}_{12}| + \gamma_1 (|A_{22}| + |\check{L}_{20}||\check{U}_{02}|) && \text{I.H.} \\
&\leq (\gamma_1 + \gamma_{\frac{k}{b}+b} + \gamma_1 \gamma_{\frac{k}{b}+b}) (|A_{22}| + |\check{L}_{20}||\check{U}_{02}|) + \gamma_{b+1} |\check{L}_{21}| |\check{U}_{12}| \\
&\leq \gamma_{\frac{k}{b}+b+1} (|A_{22}| + |\check{L}_{20}||\check{U}_{02}|) + \gamma_{b+1} |\check{L}_{21}| |\check{U}_{12}| && \text{Lem. 3.4} \\
&\leq \gamma_{\frac{k}{b}+b+1} (|A_{22}| + |\check{L}_{20}||\check{U}_{02}| + |\check{L}_{21}||\check{U}_{12}|). && \square
\end{aligned}$$

This concludes the proof of the inductive step of Theorem 6.5. The proof is also summarized in Fig.6.2.

6.4. A comment about practical implementations. In practice, the factorization of A_{11} and subsequent updating of A_{21} is accomplished by computing an LU factorization with partial pivoting of the panel of columns $\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$, after which the row exchanges are applied to the remainder of the matrix. As we argued for the unblocked algorithm, pivoting does not introduce error and therefore does not change the analysis. Once pivoting is taken out of the equation, the factorization of the panel of columns can be thought of as a simultaneous factorization of A_{11} and subsequent update of A_{21} . Thus, the analyses of these separate operations are equivalent to the analysis of the factorization of the panel and the error result established for the blocked algorithm holds.

7. Conclusion. In this paper, we described a systematic approach to deriving numerical stability results for linear algebra algorithms. The approach leverages notation for representing algorithms that was developed as part of the FLAME project. It extends the FLAME methodology for deriving algorithms so that numerical stability results are established in a goal-oriented and modular fashion. The presented methodology alleviates the difficulties of description and preparation mentioned in the quote by Parlett. It also overcomes the problem of re-derivation of standard results mentioned in the quote by Higham, facilitating the establishment of an inventory of results. In addition, it has yielded a previously unpublished result for the backward stability of blocked LU factorization.

While the paper was written so that later results build on earlier results, the best way to unleash the methodology is to start with the final algorithm to be analyzed, and work backwards. For example, we could have started with the blocked LU factorization. As part of the analysis, it would have become clear that algorithms and stability results were needed for unblocked LU factorization, triangular solve with multiple right-hand sides, and matrix-matrix multiplication. In turn, each of these operations would have exposed other suboperations and so forth. Eventually, the analysis would have reached the fundamental operations to which the SCM and ACM can be directly applied. From that stage, the results would have then slowly built back up to the analysis of the blocked algorithm. However, such a structure would have made the paper difficult to read and of little pedagogical value.

Just like it has been shown that systematic derivation of algorithms can be made mechanical [3], we believe the proposed system could also be made mechanical by a system that understands the rules of linear algebra. This will be the topic of future research, as will be the application of the proposed techniques to more complex matrix operations.

8. Acknowledgments. The idea of systematically deriving stability analyses occurred to us shortly after the idea of deriving algorithms was first proposed [10, 4]. At that time, we considered it high hanging fruit and ignored the possibility. Then, in response to a paper with Enrique Quintana-Ortí on the derivation of algorithms for the triangular Sylvester equation [12] that yielded a number of new algorithms for that operation, a referee pushed us to add a stability analysis for the algorithms. This referee, who we correctly guessed was G.W. (Pete) Stewart, gave us three choices: (1) show empirical results; (2) present an analysis that shows the algorithms to have equivalent error accumulation; or (3) extend the derivation process so that it also yields the stability analysis. We invited Pete to work on (3) with us, to which he responded that some Holy Grails are best left unpursued. And thus we opted for (1) in the revision of that paper. It took a few more years before one of the authors

decided to pursue the suggestion as part of his dissertation [3]. We are grateful to Pete for his insight and encouragement.

In addition, we thank Enrique Quintana-Ortí and Maggie Myers for feedback on an early draft.

This work was supported in part by NSF grants ACI-0305163, CCF-0342369, CCF-0540926, and CCF-0702714. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

- [1] E. ANDERSON, Z. BAI, J. DEMMEL, J. E. DONGARRA, J. DUCROZ, A. GREENBAUM, S. HAMMARLING, A. E. MCKENNEY, S. OSTROUCHOV, AND D. SORENSEN, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992.
- [2] P. BERESFORD, *Matrix eigenvalue problems*, The American Mathematical Monthly, 72 (1965).
- [3] P. BIENTINESI, *Mechanical Derivation and Systematic Analysis of Correct Linear Algebra Algorithms*, PhD thesis, Department of Computer Sciences, The University of Texas, 2006. Technical Report TR-06-46. September 2006.
- [4] P. BIENTINESI, J. A. GUNNELS, M. E. MYERS, E. S. QUINTANA-ORTÍ, AND R. A. VAN DE GEIJN, *The science of deriving dense linear algebra algorithms*, ACM Transactions on Mathematical Software, 31 (2005), pp. 1–26.
- [5] J. W. DEMMEL, N. J. HIGHAM, AND R. S. SCHREIBER, *Stability of block lu factorization*, Numer. Lin. Algebra Applic, 2 (1995), pp. 173–190.
- [6] J. J. DONGARRA, J. DU CROZ, S. HAMMARLING, AND I. DUFF, *A set of level 3 basic linear algebra subprograms*, ACM Trans. Math. Soft., 16 (1990), pp. 1–17.
- [7] J. J. DONGARRA, I. S. DUFF, D. C. SORENSEN, AND H. A. VAN DER VORST, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, Philadelphia, PA, 1991.
- [8] G. H. GOLUB AND C. F. V. LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 2nd ed., 1989.
- [9] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 3rd ed., 1996.
- [10] J. A. GUNNELS, F. G. GUSTAVSON, G. M. HENRY, AND R. A. VAN DE GEIJN, *FLAME: Formal Linear Algebra Methods Environment*, ACM Trans. Math. Soft., 27 (2001), pp. 422–455.
- [11] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second ed., 2002.
- [12] E. S. QUINTANA-ORTÍ AND R. A. VAN DE GEIJN, *Formal derivation of algorithms: The triangular Sylvester equation*, ACM Trans. Math. Soft., 29 (2003), pp. 218–243.
- [13] G. W. STEWART, *Matrix Algorithms. Volume I: Basic Decompositions*, SIAM, Philadelphia, 1998.
- [14] G. W. STEWART AND J.-G. SUN, *Matrix Perturbation Theory (Computer Science and Scientific Computing) (Computer Science and Scientific Computing)*, Academic Press, June 1990.
- [15] R. A. VAN DE GEIJN AND E. S. QUINTANA-ORTÍ, *The Science of Programming Matrix Computations*, www.lulu.com, 2008.