GOAL-ORIENTED AND MODULAR STABILITY ANALYSIS

PAOLO BIENTINESI* AND ROBERT A. VAN DE GEIJN†

DEDICATED TO G.W. STEWART ON THE OCCASION OF HIS 70TH BIRTHDAY

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 roundoff error is introduced. The approach supports the analysis of more complex algorithms, such as the blocked LU factorization. For this operation we derive a tighter error bound than has been previously reported in the literature.

1. Introduction.

Numerical stability analysis related to dense linear algebra operations continues to be an important topic in numerical analysis. As part of the FLAME project, we approach this topic from a new perspective. We are interested in the methodology for deriving results as much as, if not more, than the results themselves. Our goal is to identify notation and a procedure for error analyses that can, in principle, be made mechanical (automatic). For the problem of deriving algorithms for linear algebra operations we have already achieved these objectives: we identified notation and exposed a systematic procedure that was reproducible by a computer algebra system [2]. 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, both in the presentation of the algorithms and in the error analyses of those algorithms.
- The derivation of error results becomes a goal-oriented: 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 is explained by applying it to a sequence of progressively more difficult algorithms for which results were already known.
- The methodology is used 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 $\check{L}$ and $\check{U}$ factors of matrix A are such that $\check{L}\check{U} = A + \Delta A$, with $|\Delta A| \leq \gamma_{\frac{n}{b}+b}(|A| + |\check{L}||\check{U}|)$. The factor $\gamma_{\frac{n}{b}+b}$ improves on the factor γ_n for 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 systematically by people with considerably less expertise.

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 [3]. The reason is that the derivation of the analysis of an algorithm mirrors the derivation of the algorithm itself, as discussed in the conclusion.

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

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

It is impossible for us to give a complete treatment of related work. We refer the reader to Higham's book [10], which lists no less than 1134 citations. Other classic references are Golub and Van Loan [8], Stewart [11], and Stewart and Sun [12].

This paper targets experts in numerical stability analysis. Many results are stated without proof when they are well-known or relatively obvious. A pedagogical version of this paper, appropriate for graduate students with a limited or no background in stability analysis, is available as a technical report [4].

The paper is structured as follows. In Section 2, we review notation for capturing and analyzing error. In Section 3, we analyze error accumulated when computing the inner (dot) product. 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. In Section 5 an (unblocked) LU factorization algorithm is analyzed. These first sections yield only well-known results. The climax comes in Section 6 in which an analysis for a blocked LU factorization that yields a tighter bound for the error is given. The paper concludes with a discussion of what the new error result means for a practical blocked LU factorization and of new opportunities that are enabled by the proposed methodology.

2. Preliminaries.

2.1. Notation.

Much of our notation related to error analysis was taken and/or inspired by the notation in [10]. The notation $\delta\!\chi$, where the symbol δ is touches a scalar variable χ, indicates a perturbation associated with the variable χ. Likewise, $\delta\!x$ and $\Delta\!X$ (Δ touches X) indicate a perturbation vector and matrix associated with vector x and matrix X, respectively. Variables indicating perturbations are called error operands.

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

We differentiate between exact and computed quantities. The function $[expression]$ returns the result of the evaluation of *expression*, where every operation is executed in floating point arithmetic[1]. Equality between the quantities *lhs* and *rhs* is denoted by $lhs = rhs$. Assignment 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 $\kappa := expression$, the notation $\check{\kappa}$ to denotes the quantity resulting from $[expression]$, which is actually stored in the variable κ.

We denote the *machine epsilon* or *unit roundoff* by $\mathbf{u}$. It 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 focus on real valued arithmetic 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

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

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

The quantity ϵ is a function of χ, ψ and op. 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)* [10] which also assumes for the basic arithmetic operations that

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

As for the standard computation model, the quantity ϵ is a function of χ, ψ and op.

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

The algorithm is said to be *backward stable* if for all $x \in \mathcal{D}$ there exists a perturbed input $\check{x} \in \mathcal{D}$, close to x, such that $\check{f}(x) = f(\check{x})$. The difference between $\check{x}$ and x, $\delta x = \check{x} - x$, is the perturbation to the original input x.

When discussing error analyses, δx, the difference between x and $\check{x}$, is the backward error and the difference $\check{f}(x) - f(x)$ is the forward error.

2.4. Absolute value of vectors and matrices. In this paper, all bounds are given in terms of the absolute values of the individual elements of the vectors and/or matrices. It is easy to convert such bounds into bounds involving norms.

DEFINITION 2.1. *Let* $\triangle \in \{<, \le, =, \ge, >\}$ *and* $x, y \in \mathbb{R}^n$. *Then* $|x| \triangle |y|$ *iff* $|\chi_i| \triangle |\psi_i|$, *with* $i = 0, \ldots, n-1$. *Similarly, given* A *and* $B \in \mathbb{R}^{m\times n}$, $|A| \triangle |B|$ *iff* $|\alpha_{ij}| \triangle |\beta_{ij}|$, *with* $i = 0, \ldots, m-1$ *and* $j = 0, \ldots, n-1$.

LEMMA 2.2. *Let* $A \in \mathbb{R}^{m\times k}$ *and* $B \in \mathbb{R}^{k\times n}$. *Then* $|AB| \le |A||B|$.

2.5. Deriving dense linear algebra algorithms. In various papers, we have shown that for a broad class of linear algebra operations one can systematically derive algorithms for computing them [9]. The primary vehicle in the derivation of algorithms is a worksheet to be filled in a prescribed order [3]. 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. We give an algorithm for $\kappa := x^T y$ and the related error results. We also introduce the error-worksheet as a framework for presenting analyses side-by-side with the algorithm.

3.1. An algorithm for computing DOT. We consider the algorithm given in Fig. 3.1(left). It uses the FLAME notation [9, 3] to express the computation

$$\kappa := \Big(\big((\chi_0\psi_0 + \chi_1\psi_1) + \cdots \big) + \chi_{n-2}\psi_{n-2} \Big) + \chi_{n-1}\psi_{n-1} \tag{3.1}$$

in the indicated order.

3.2. Preparation. Under the SCM model

$$\check{\kappa} = \sum_{i=0}^{n-1} \left(\chi_i\psi_i (1+\epsilon_*^{(i)}) \prod_{j=i}^{n-1} (1+\epsilon_+^{(j)}) \right), \tag{3.2}$$

```
Algorithm Dot: $\kappa := x^T y$
$\kappa := 0$
Partition $x \rightarrow \left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right), y \rightarrow \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right)$
    where $x_T$ and $y_T$ are empty
While $m(x_T) < m(x)$ do
    Repartition
    $\left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right) \rightarrow \left(\begin{array}{c} x_0 \\ \hline \chi_1 \\ x_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ y_2 \end{array}\right)$
        where $\chi_1$ and $\psi_1$ are scalars
    ----------------------------------------
    $\kappa := \kappa + \chi_1 \psi_1$
    ----------------------------------------
    Continue with
    $\left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right) \leftarrow \left(\begin{array}{c} x_0 \\ \chi_1 \\ \hline x_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \leftarrow \left(\begin{array}{c} y_0 \\ \psi_1 \\ \hline y_2 \end{array}\right)$
endwhile
```

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

FIG. 3.1. *Left: Algorithm for computing $\kappa := x^T y$. Right: An algorithm for solving $Lx = b$.*

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

LEMMA 3.1. ([10], 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 $|\theta_n| \leq n\mathbf{u}/(1-n\mathbf{u})$.*

The quantity θ_n is not intended to be a specific number. 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 $(1+\theta_n)$.

Two instances of the symbol θ_j typically do not represent the same number.

Since the bound on $|\theta_n|$ occurs often, we assign it a symbol:

DEFINITION 3.2. *For all $n \geq 1$ and $n\mathbf{u} < 1$, define $\gamma_n := n\mathbf{u}/(1-n\mathbf{u})$.*

Equality (3.2) simplifies to $\check{\kappa} = \chi_0\psi_0(1+\theta_n) + \chi_1\psi_1(1+\theta_n) + \cdots + \chi_{n-1}\psi_{n-1}(1+\theta_2)$, where $|\theta_j| \leq \gamma_j$, $j = 2, \ldots, n$.

The following relations will be useful to bound how error accumulates:

LEMMA 3.3. *If $n, b \geq 1$ then $\gamma_n \leq \gamma_{n+b}$ and $\gamma_n + \gamma_b + \gamma_n\gamma_b \leq \gamma_{n+b}$.*

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:

$$1)\ \check{\kappa} = (x + \delta x)^T y; \quad 2)\ \check{\kappa} = x^T(y + \delta y); \quad 3)\ \check{\kappa} = x^T y + \delta\kappa.$$

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). We will see that under different circumstances, different error results 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 $\Sigma = \mathrm{diag}(\sigma_0, \ldots, \sigma_{n-1})$. The following special structure of Σ will be used in the remainder of the paper:

$$\Sigma^{(n)} = \begin{cases} 0 \times 0 \text{ matrix} & \text{if } n = 0 \\ \theta_1 & \text{if } n = 1 \\ \mathrm{diag}(\theta_n, \theta_n, \theta_{n-1}, \ldots, \theta_2) & \text{otherwise.} \end{cases} \qquad (3.3)$$

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

LEMMA 3.4. *Let $k \geq 0$ and assume that $|\epsilon_1|, |\epsilon_2| \leq \mathbf{u}$, with $\epsilon_1 = 0$ if $k = 0$. Then*

$$\left(\begin{array}{c|c} I + \Sigma^{(k)} & 0 \\ \hline 0 & (1+\epsilon_1) \end{array}\right)(1+\epsilon_2) = (I + \Sigma^{(k+1)}).$$

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

THEOREM 3.5. *Let $x, y \in \mathbb{R}^n$ and let $\kappa := x^T y$ be computed by executing the algorithm in Fig. 3.1(left). Then $\check{\kappa} = [x^T y] = x^T(I + \Sigma^{(n)})y$.*

3.4. A proof in traditional format. We purposely pick notation so that the proof can be related to the alternative framework 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

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

Let $x_T, y_T \in \mathbb{R}^{k+1}$, 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

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

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

	Error side	Step
$\kappa := 0$	$\{ \ \Sigma = 0 \ \}$	1a
Partition $x \rightarrow \left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right), y \rightarrow \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right)$, **where** x_T and y_T are empty, and Σ_T is 0×0	$\Sigma \rightarrow \left(\begin{array}{cc} \Sigma_T & 0 \\ \hline 0 & \Sigma_B \end{array}\right)$	4
	$\left\{\check{\kappa} = x_T^T(I + \Sigma_T)y_T \wedge \Sigma_T = \Sigma^{(k)} \wedge m(x_T) = k \right\}$	2a
While $m(x_T) < m(x)$ **do**		3
	$\left\{\check{\kappa} = x_T^T(I + \Sigma_T)y_T \wedge \Sigma_T = \Sigma^{(k)} \wedge m(x_T) = k \right\}$	2b
Repartition $\left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right) \rightarrow \left(\begin{array}{c} x_0 \\ \hline \chi_1 \\ \hline x_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$, **where** χ_1, ψ_1, and σ_1^i are scalars	$\left(\begin{array}{cc} \Sigma_T & 0 \\ \hline 0 & \Sigma_B \end{array}\right) \rightarrow \left(\begin{array}{ccc} \Sigma_0^i & 0 & 0 \\ \hline 0 & \sigma_1^i & 0 \\ \hline 0 & 0 & \Sigma_2 \end{array}\right)$	5a
	$\left\{\check{\kappa}^i = x_0^T(I + \Sigma_0^i)y_0 \wedge \Sigma_0^i = \Sigma^{(k)} \wedge m(x_0) = k \right\}$	6
$\kappa := \kappa + \chi_1\psi_1$	$\check{\kappa}^{i+1} = \big(\check{\kappa}^i + \chi_1\psi_1(1+\epsilon_*)\big)(1+\epsilon_+)$ SCM, twice ($\epsilon_+ = 0$ if $k = 0$) $= \big(x_0^T(I + \Sigma_0^{(k)})y_0 + \chi_1\psi_1(1+\epsilon_*)\big)(1+\epsilon_+)$ Step 6: I.H. $= \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}^T \left(\begin{array}{cc} I + \Sigma_0^{(k)} & 0 \\ \hline 0 & 1+\epsilon_* \end{array}\right)(1+\epsilon_+)\begin{pmatrix} y_0 \\ \psi_1 \end{pmatrix}$ Rearrange $= \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}^T \left(I + \Sigma^{(k+1)}\right)\begin{pmatrix} y_0 \\ \psi_1 \end{pmatrix}$ Lemma 3.4	8
	$\left\{\begin{array}{l} \check{\kappa}^{i+1} = \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}^T \left(I + \left(\begin{array}{cc} \Sigma_0^{i+1} & 0 \\ \hline 0 & \sigma_1^{i+1} \end{array}\right)\right)\begin{pmatrix} y_0 \\ \psi_1 \end{pmatrix} \\ \wedge \left(\begin{array}{cc} \Sigma_0^{i+1} & 0 \\ \hline 0 & \sigma_1^{i+1} \end{array}\right) = \Sigma^{(k+1)} \wedge m\begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix} = (k+1) \end{array}\right\}$	7
Continue with $\left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right) \leftarrow \left(\begin{array}{c} x_0 \\ \hline \chi_1 \\ \hline x_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \leftarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$,	$\left(\begin{array}{cc} \Sigma_T & 0 \\ \hline 0 & \Sigma_B \end{array}\right) \leftarrow \left(\begin{array}{ccc} \Sigma_0^{i+1} & 0 & 0 \\ \hline 0 & \sigma_1^{i+1} & 0 \\ \hline 0 & 0 & \Sigma_2 \end{array}\right)$	5b
	$\left\{\check{\kappa} = x_T^T(I + \Sigma_T)y_T \wedge \Sigma_T = \Sigma^{(k)} \wedge m(x_T) = k \right\}$	2c
endwhile		
	$\left\{\check{\kappa} = x_T^T(I + \Sigma_T)y_T \wedge \Sigma_T = \Sigma^{(k)} \wedge m(x_T) = k \quad \wedge m(x_T) = m(x) \right\}$	2d
	$\left\{\check{\kappa} = x^T(I + \Sigma^{(n)})y \wedge m(x) = n \right\}$	1b

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

3.5. The error worksheet. In Fig. 3.2 we present a framework, which we call the *error worksheet*, for presenting the inductive proof of Theorem 3.5 side-by-side with the algorithm for DOT. This framework, in a slightly different form, was first introduced in [2]. The expressions enclosed by $\{ \quad \}$ (in the grey boxes) are predicates that describe the state of the variables used in the algorithms and their analysis. 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 $\check{\kappa}, x_T, y_T, \Sigma_T$ are such that the predicate

$$\{\check{\kappa} = x_T^T(I + \Sigma_T)y_T \wedge k = m(x_T) \wedge \Sigma_T = \Sigma^{(k)}\} \tag{3.4}$$

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 $\check{\kappa} = x^T(I + \Sigma^{(n)})y$, i.e., the thesis of the theorem, is satisfied too.

The inductive proof of Theorem 3.5 is captured by the error worksheet as follows.
Base case. In Step 2a, i.e., before the execution of the loop, predicate (3.4) is satisfied, as $k = m(x_T) = 0$.
Inductive step. Assume that the predicate (3.4) 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 *true* 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 σ_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 (3.4) holds for all iterations. In particular, when the loop terminates, the predicate becomes

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

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 3.6. *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.

An inventory of consequences of Theorem 3.5 follow. We will draw from this list when analyzing algorithms that utilize DOT.

COROLLARY 3.7. *Under the assumptions of Theorem 3.5 the following hold:*
R1-B: (Backward analysis) $\check{\kappa} = (x + \delta x)^T y$, *where* $|\delta x| \leq \gamma_n |x|$;
R2-B: (Backward analysis) $\check{\kappa} = x^T(y + \delta y)$, *where* $|\delta y| \leq \gamma_n |y|$;
R1-F: (Forward analysis) $\check{\kappa} = x^T y + \delta\kappa$, *where* $|\delta\kappa| \leq \gamma_n |x|^T |y|$.

Proof: For R1-F, let $\delta\kappa = x^T \Sigma^{(n)} y$, where $\Sigma^{(n)}$ is as in Theorem 3.5. Then

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

We leave the proofs of R1-B and R2-B as an exercise to the reader. □

Lemma 3.8. *Given scalars α, β and λ, consider $\sigma := (\alpha+\beta)/\lambda$. Then:*
R1-B: $\check{\sigma} = ((\alpha+\delta\alpha)+(\beta+\delta\beta))/\lambda$, *where* $|\delta\alpha| \le \gamma_2|\alpha|$ *and* $|\delta\beta| \le \gamma_2|\beta|$.
R1-F: $\lambda\check{\sigma} = \alpha+\beta+\delta\sigma$, *with* $|\delta\sigma| \le \gamma_2(|\alpha|+|\beta|)$.
R2-B: $\check{\sigma} = (\alpha+\beta)/(\lambda+\delta\lambda)$, *where* $|\delta\lambda| \le \gamma_2|\lambda|$.
R2-F: $\lambda\check{\sigma} = \alpha+\beta+\delta\sigma$, *with* $|\delta\sigma| \le \gamma_2|\check{\sigma}||\lambda|$.
R3-B: $\check{\sigma} = ((\alpha+\delta\alpha)+(\beta+\delta\beta))/(\lambda+\delta\lambda)$, *where* $|\delta\alpha| \le \gamma_1|\alpha|$, $|\delta\beta| \le \gamma_1|\beta|$, *and* $|\delta\lambda| \le \gamma_1|\lambda|$.
R3-F: $\lambda\check{\sigma} = \alpha+\beta+\delta\sigma$, *with* $|\delta\sigma| \le \gamma_1(|\alpha|+|\beta|+|\check{\sigma}||\lambda|)$.
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:

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

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 3.9. *Let $n \ge 1$, $\alpha, \lambda, \mu, \nu \in \mathbb{R}$, and $x, y \in \mathbb{R}^n$. Assume that $\lambda \ne 0$ and consider the assignments $\mu := \alpha - (x^Ty)$ and $\nu := \left(\alpha - (x^Ty)\right)/\lambda$; the parentheses identify the evaluation order. Then*
R1-B: $\check{\mu} = \alpha + \delta\alpha - (x+\delta x)^Ty$, *where* $|\delta\alpha| \le \gamma_1|\alpha|$ *and* $|\delta x| \le \gamma_{n+1}|x|$.
R1-F: $\check{\mu} = \alpha - x^Ty + \delta\mu$, *where* $|\delta\mu| \le \gamma_{n+1}|x|^T|y| + \gamma_1|\alpha| \le \gamma_{n+1}\left(|x|^T|y|+|\alpha|\right)$.
R2-F: $\check{\mu} = \alpha - x^Ty + \delta\mu$, *where* $|\delta\mu| \le \gamma_n|x|^T|y| + \gamma_1|\check{\mu}| \le \gamma_n\left(|x|^T|y|+|\check{\mu}|\right)$.
Also,
R3-B: $\lambda\check{\nu} = \alpha + \delta\alpha - (x+\delta x)^Ty$, *where* $|\delta\alpha| \le \gamma_2|\alpha|$ *and* $|\delta x| \le \gamma_{n+2}|x|$.
R3-F: $\lambda\check{\nu} = \alpha - x^Ty + \delta\nu$, *where* $|\delta\nu| \le \gamma_2|\alpha| + \gamma_{n+2}|x|^T|y| \le \gamma_{n+2}\left(|\alpha|+|x|^T|y|\right)$.
R4-B: $(\lambda+\delta\lambda)\check{\nu} = \alpha - (x+\delta x)^Ty$, *where* $|\delta\lambda| \le \gamma_2|\lambda|$ *and* $|\delta x| \le \gamma_n|x|$.
R4-F: $\lambda\check{\nu} = \alpha - x^Ty + \delta\nu$, *where*
$|\delta\nu| \le \gamma_n|x|^T|y| + \gamma_2|\lambda||\check{\nu}| \le \max(\gamma_2,\gamma_n)\left(|x|^T|y|+|\lambda||\check{\nu}|\right)$.
R5-B: $(\lambda+\delta\lambda)\check{\nu} = \alpha + \delta\alpha - (x+\delta x)^Ty$, *where*
$|\delta\alpha| \le \gamma_1|\alpha|$, $|\delta x| \le \gamma_{n+1}|x|$, *and* $|\delta\lambda| \le \gamma_1|\lambda|$.
R5-F: $\lambda\check{\nu} = \alpha - x^Ty + \delta\nu$, *where*
$|\delta\nu| \le \gamma_1|\alpha| + \gamma_{n+1}|x|^T|y| + \gamma_1|\lambda||\check{\nu}| \le \gamma_{n+1}\left(|\alpha|+|x|^T|y|+|\lambda||\check{\nu}|\right)$.

Proof: R1 follows Theorem 3.5 and SCM: $\check{\mu} = \left(\alpha - x^T(I+\Sigma^{(n)})y\right)(1+\epsilon_1)$, algebraic manipulation, and bounding. R2-F follows Theorem 3.5 and ACM: $\check{\mu} = \left(\alpha - x^T(I+\Sigma^{(n)})y\right)/(1+\epsilon_2)$, algebraic manipulation, and bounding. Results R3, R4 and R5 are obtained similarly, invoking Theorem 3.5 and Lemma 3.8. $\square$

4. Analysis of Triangular Linear Systems and Introduction to Goal-Oriented Error Analysis.

We discuss a goal-oriented methodology for analyzing the error generated by an algorithm hand-in-hand with the analysis of an algorithm for the solution of a triangular linear system.

Error side	Step
$\{\ \Delta L = 0\ \}$	1a
Partition $L \rightarrow \left(\begin{array}{cc} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right), x \rightarrow \left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right), b \rightarrow \left(\begin{array}{c} b_T \\ \hline b_B \end{array}\right), \quad \Delta L \rightarrow \left(\begin{array}{cc} \Delta L_{TL} & 0 \\ \hline \Delta L_{BL} & \Delta L_{BR} \end{array}\right)$ **where** L_{TL}, x_T, b_T, and ΔL_{TL} are empty	4
$\left\{(L_{TL} + \Delta L_{TL})\check{x}_T = b_T \wedge \lvert \Delta L_{TL} \rvert \le \max(\gamma_2, \gamma_{k-1}) \lvert L_{TL} \rvert \wedge m(x_T) = k\right\}$	2a
While $m(x_T) < m(x)$ **do**	3
$\left\{(L_{TL} + \Delta L_{TL})\check{x}_T = b_T \wedge \lvert \Delta L_{TL} \rvert \le \max(\gamma_2, \gamma_{k-1}) \lvert L_{TL} \rvert \wedge m(x_T) = k\right\}$	2b
Repartition $\left(\begin{array}{cc} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{ccc} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ L_{20} & l_{21} & L_{22} \end{array}\right), \quad \left(\begin{array}{cc} \Delta L_{TL} & 0 \\ \hline \Delta L_{BL} & \Delta L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{ccc} \Delta L_{00} & 0 & 0 \\ \hline \delta l_{10}^T & \delta\lambda_{11} & 0 \\ \Delta L_{20} & \delta l_{21} & \Delta L_{22} \end{array}\right)$ $\left(\begin{array}{c} \check{x}_T \\ \hline \check{x}_B \end{array}\right) \rightarrow \left(\begin{array}{c} \check{x}_0 \\ \hline \check{\chi}_1 \\ \check{x}_2 \end{array}\right), \left(\begin{array}{c} b_T \\ \hline b_B \end{array}\right) \rightarrow \left(\begin{array}{c} b_0 \\ \hline \beta_1 \\ b_2 \end{array}\right)$ **where** λ_{11}, $\delta\lambda_{11}$, χ_1, and β_1 are scalars	5a
$\left\{(L_{00} + \Delta L_{00})\check{x}_0 = b_0 \wedge \lvert \Delta L_{00} \rvert \le \max(\gamma_2, \gamma_{k-1}) \lvert L_{00} \rvert \wedge m(x_0) = k\right\}$	6
$\chi_1 := \dfrac{\beta_1 - l_{10}^T x_0}{\lambda_{11}}$; $\left.\begin{array}{l} b_0 = (L_{00} + \Delta L_{00})\check{x}_0 \\ \quad \wedge \lvert \Delta L_{00} \rvert \le \max(\gamma_2, \gamma_{k-1}) \lvert L_{00} \rvert \end{array}\right\}$ Step 6: I.H. ; $\left.\begin{array}{l} \beta_1 = \left(l_{10}^T + \delta l_{10}^T\right) \check{x}_0 + (\lambda_{11} + \delta\lambda_{11})\,\check{\chi}_1 \\ \quad \wedge \lvert \left(\delta l_{10}^T \,\vert\, \delta\lambda_{11} \right) \rvert \le \max(\gamma_2, \gamma_k) \lvert \left(l_{10}^T \,\vert\, \lambda_{11} \right) \rvert \end{array}\right\}$ Th. 3.9 R4-B	8
$\left\{\begin{array}{l} \left(\left(\begin{array}{cc} L_{00} & 0 \\ \hline l_{10}^T & \lambda \end{array}\right) + \left(\begin{array}{cc} \Delta L_{00} & 0 \\ \hline \delta l_{10}^T & \delta\lambda \end{array}\right)\right) \left(\begin{array}{c} \check{x}_0 \\ \hline \check{\chi}_1 \end{array}\right) = \left(\begin{array}{c} b_0 \\ \hline \beta_1 \end{array}\right) \\ \wedge \left\lvert \left(\begin{array}{cc} \Delta L_{00} & 0 \\ \hline \delta l_{10}^T & \delta\lambda \end{array}\right) \right\rvert \le \max(\gamma_2, \gamma_k) \left\lvert \left(\begin{array}{cc} L_{00} & 0 \\ \hline l_{10}^T & \lambda \end{array}\right) \right\rvert \wedge m \left(\begin{array}{c} x_0 \\ \hline \chi_1 \end{array}\right) = k + 1 \end{array}\right\}$	7
Continue with L, x and b as in Fig. 3.1, and $\left(\begin{array}{cc} \Delta L_{TL} & 0 \\ \hline \Delta L_{BL} & \Delta L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{ccc} \Delta L_{00} & 0 & 0 \\ \delta l_{10}^T & \delta\lambda_{11} & 0 \\ \hline \Delta L_{20} & \delta l_{21} & \Delta L_{22} \end{array}\right)$	5b
$\left\{(L_{TL} + \Delta L_{TL})\check{x}_T = b_T \wedge \lvert \Delta L_{TL} \rvert \le \max(\gamma_2, \gamma_{k-1}) \lvert L_{TL} \rvert \wedge m(x_T) = k\right\}$	2c
endwhile	
$\left\{\begin{array}{r} (L_{TL} + \Delta L_{TL})\check{x}_T = b_T \wedge \lvert \Delta L_{TL} \rvert \le \max(\gamma_2, \gamma_{k-1}) \lvert L_{TL} \rvert \\ \wedge\, m(x_T) = m(x) \wedge m(x_T) = k \end{array}\right\}$	2d
$\left\{(L + \Delta L)\check{x} = b \wedge \lvert \Delta L \rvert \le \max(\gamma_2, \gamma_{n-1}) \lvert L \rvert \wedge m(x) = n\right\}$	1b

FIG. 4.1. *Error worksheet for deriving the backward stability error result for the algorithm in Fig. 3.1(right).*

4.1. Solving a lower triangular linear system (LTRSV). Consider $Lx = b$, 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 operand[2]. In Fig. 3.1(right) we show one specific algorithm for computing x. During the execution of this algorithm 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. 3.1(right) is in Fig. 4.1: in the following we explain how the worksheet is constructively filled out to generate the analysis. Pretend that all the gray-shaded boxes in Fig. 4.1 are initially empty and

[2]In 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.

that only Step 8-left (the computational update) is already complete, having been copied over from Fig. 4.1. The empty boxes will be filled out in the order indicated in the "Steps" column.

Step 1: Error-precondition and error-postcondition. We show that the computed solution, $\check{x}$, satisfies the backward error result $(L + \Delta L)\check{x} = b$ 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 $\chi_1 = (\beta_1 - l_{10}^T x_0)/\lambda$. This assignment involves a dot product with vectors of length $n - 1$. Thus, results R4 from Theorem 3.9 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 $(L+\Delta L)\check{x} = b$. In this stage error bounds are not considered, as the only concern is to push the error generated by the computational updates onto 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 verified. 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 that describes the target error result, the *error-postcondition*. In Fig. 4.1 these predicates are placed in Step 1a and 1b, respectively. The error-precondition is $\{\Delta L = 0\}$ (no error has yet accumulated), and the error-postcondition is

$$\{(L + \Delta L)\check{x} = b \wedge |\Delta L| \leq \max(\gamma_2, \gamma_{n-1})|L| \wedge m(x) = n\}.$$

Step 2: Error-invariant. Recall that the *error-invariant* for the algorithm is the predicate that describes the state of the error operands in Steps 2a–d. The loop-invariant is the predicate that describes the computation performed when the algorithm reaches Steps 2a–d. The error-invariant should equal the subexpression of the error-postcondition that corresponds to the state described by the loop-invariant. The algorithm in Fig. 3.1(right) has the property that before and after each iteration the contents of x_T are such that $\{L_{TL}x_T = b_T\}$, the loop-invariant. Thus, we expect the accumulated error to satisfy $(L_{TL}+\Delta L_{TL})\check{x}_T = b_T$. Adding also bounds on ΔL_{TL}, the error-invariant, which is entered in Steps 2a–d, becomes

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

Step 3: Loop-guard. The loop guard is part of the algorithm to be analyzed.

Step 4: Initialization. In this step the error operands are partitioned conformally to the operands in the algorithm with respect to the error-postcondition. In the example, ΔL is partitioned so that the dimensions of the variables in the expression $(L + \Delta L)\check{x} = b$, with the operands partitioned as in Fig. 3.1(right), are conformal.

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

Step 6: State of the variables before the update in Step 8. Step 5a consists of the following relabelling statements: $L_{TL} \to L_{00}$, $x_T \to x_0$, $b_T \to b_0$, and $\Delta L_{TL} \rightarrow \Delta L_{00}$. Thus, given the state of the variables in Step 2b, the contents of the submatrices and subvectors exposed in Step 5a are described by

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

at Step 6. This predicate expresses the state of the error operands before the execution of the computational statements listed in Step 8-left.

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 $L_{TL} \leftarrow \left(\begin{array}{c|c} L_{00} & 0 \\ \hline l_{10}^T & \lambda_{11} \end{array}\right)$, $x_T \leftarrow \left(\begin{array}{c} x_0 \\ \hline \chi_1 \end{array}\right)$, $b_T \leftarrow \left(\begin{array}{c} b_0 \\ \hline \beta_1 \end{array}\right)$, and $\Delta L_{TL} \leftarrow \left(\begin{array}{c|c} \Delta L_{00} & 0 \\ \hline \delta l_{10}^T & \delta\lambda_{11} \end{array}\right)$.Thus, given the state in which the variables *must be* in Step 2c, the contents of the submatrices and subvectors before Step 5b *must*, at Step 7, become

$$\left(\left(\begin{array}{c|c} L_{00} & 0 \\ \hline l_{10}^T & \lambda_{11} \end{array}\right) + \left(\begin{array}{c|c} \Delta L_{00} & 0 \\ \hline \delta l_{10}^T & \delta\lambda_{11} \end{array}\right)\right)\left(\begin{array}{c} \check{x}_0 \\ \hline \check{\chi}_1 \end{array}\right) = \left(\begin{array}{c} b_0 \\ \hline \beta_1 \end{array}\right) \wedge m\left(\begin{array}{c} x_0 \\ \hline \chi_1 \end{array}\right) = k+1$$
$$\wedge \left|\left(\begin{array}{c|c} \Delta L_{00} & 0 \\ \hline \delta l_{10}^T & \delta\lambda_{11} \end{array}\right)\right| \leq \max(\gamma_2, \gamma_k)\left|\left(\begin{array}{c|c} L_{00} & 0 \\ \hline l_{10}^T & \lambda_{11} \end{array}\right)\right|. \tag{4.2}$$

Step 8: Inductive step. When the computation reaches this step, the predicate (4.1), described in Step 6, is satisfied. The question is whether the computational update $\chi_1 := (\beta_1 - l_{10}^T x_0)/\lambda_{11}$, 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 completes, the error variables must satisfy

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

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 3.9 Result R4-B, which is chosen because it matches the requirement of how error is propogated. The analysis is also given in Step 8-right of Fig. 4.1.

4.3. Results for TRSV.

Fig. 4.1, provides the proof for R1-B of the next theorem.

THEOREM 4.1. *Let $L \in \mathbb{R}^{n\times n}$ be a nonsingular lower triangular matrix and $x, b \in \mathbb{R}^n$. If the solution x of the linear system $Lx = b$ is computed via the algorithm in Fig. 4.1(left), then $\check{x}$ satisfies*
R1-B: $(L + \Delta L)\check{x} = b$, where $|\Delta L| \leq \max(\gamma_2, \gamma_{n-1})|L|$.
R1-F: $L\check{x} = b + \delta b$ where $|\delta b| \leq \max(\gamma_2, \gamma_{n-1})|L||\check{x}|$.

4.4. Other algorithmic variants.

Theorems for other algorithmic variants can be derived in a similar manner [2].

5. Analysis of an unblocked algorithm for LU factorization.

We now 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.

```
Algorithm $LU = A$
Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right), U \rightarrow \left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right)$
    where $A_{TL}, L_{TL}, U_{TL},$ are $0 \times 0$
While $m(A_{TL}) \leq m(A)$ do
    Repartition
    $\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & 1 & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right),$
    $\left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} U_{00} & u_{01} & U_{02} \\ \hline 0 & \upsilon_{11} & u_{12}^T \\ \hline 0 & 0 & U_{22} \end{array}\right)$
        where $\alpha_{11}, 1, \upsilon_{11}$ are scalars
    ----------------------------------------
    $\upsilon_{11} := \alpha_{11} - l_{10}^T u_{01}$
    $u_{12}^T := a_{12}^T - l_{10}^T U_{02}$
    $l_{21} := (a_{21} - L_{20} u_{01}) / \upsilon_{11}$
    ----------------------------------------
    Continue with
    $\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & 1 & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right), \cdots$
endwhile
```

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

5.1. An algorithm for computing the LU factorization. We analyze the variant that is often called the Crout variant [7], which allows the reader to compare and contrast our exposition with the one in [10]. The algorithm is given in Fig. 5.1. This variant computes L and U so that before and after each iteration L_{TL}, U_{TL}, L_{BL} and U_{TR} satisfy[3]

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

5.2. Preparation. 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: $\check{v} = y - Ax + \delta v$, *where* $|\delta v| \leq \gamma_{n+1}|A||x| + \gamma_1|y| \leq \gamma_{n+1}(|A||x| + |y|)$.

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

Also,

R3-F: $\lambda\check{w} = y - Ax + \delta v$, *where*
$|\delta v| \leq \gamma_2|y| + \gamma_{n+2}|A||x| \leq \gamma_{n+2}(|y| + |A||x|)$.

R4-F: $\lambda\check{w} = y - Ax + \delta v$, *where*
$|\delta v| \leq \gamma_n|A||x| + \gamma_2|\lambda||\check{w}| \leq \max(\gamma_2, \gamma_n)(|A||x| + |\lambda||\check{w}|)$.

[3]In Eqns. 5.1 and following, the dash indicates that no relation is specified for the contents of this portion of the matrix.

R5-F: $\lambda \check{w} = y - Ax + \delta v$, *where*
$|\delta v| \leq \gamma_1 |y| + \gamma_{n+1}|A||x| + \gamma_1 |\lambda||\check{w}| \leq \gamma_{n+1}(|y| + |A||x| + |\lambda||\check{w}|)$.

Proof: All these results follow from Theorem 3.9. □

5.3. Analysis. In practice the approach starts with a formulation of the analysis that does not take bounds into account. Our methodology then makes it easy to experiment with loose analyses to gain insights that lead to a tighter formulation. For space considerations, we start the analysis already with tight bounds. We will show that $\check{L}\check{U} = A + \Delta A$ with $|\Delta A| \leq \gamma_n |\check{L}||\check{U}|$. This error-postcondition is entered in Step 1b of Fig. 5.2.

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 discussed 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

$$m(A_{TL}) = k \wedge \left(\begin{array}{c|c} \check{L}_{TL}\check{U}_{TL} = A_{TL} + \Delta A_{TL} & \check{L}_{TL}\check{U}_{TR} = A_{TR} + \Delta A_{TR} \\ \hline \check{L}_{BL}\check{U}_{TR} = A_{BL} + \Delta A_{BL} & - \end{array}\right)$$
$$\wedge \left(\begin{array}{c|c} |\Delta A_{TL}| \leq \gamma_k |\check{L}_{TL}||\check{U}_{TL}| & |\Delta A_{TR}| \leq \gamma_k |\check{L}_{TL}||\check{U}_{TR}| \\ \hline |A_{BL}| \leq \gamma_k |\check{L}_{BL}||\check{U}_{TL}| & - \end{array}\right),$$

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

$$\begin{array}{l} m(A_{00}) = k \wedge \\ \left(\begin{array}{c|c|c} \check{L}_{00}\check{U}_{00} = A_{00} + \Delta A_{00} & \check{L}_{00}\check{u}_{01} = a_{01} + \delta a_{01} & \check{L}_{00}\check{U}_{02} = A_{02} + \Delta A_{02} \\ \hline \check{l}_{10}^T\check{U}_{00} = a_{10}^T + \delta a_{10}^T & - & - \\ \hline \check{L}_{20}\check{U}_{00} = A_{20} + \Delta A_{02} & - & - \end{array}\right) \wedge \\ \left|\left(\begin{array}{c|c|c} \Delta A_{00} & \delta a_{01} & \Delta A_{02} \\ \hline \delta a_{10}^T & - & - \\ \hline \Delta A_{20} & - & - \end{array}\right)\right| \leq \gamma_k \left(\begin{array}{c|c|c} |\check{L}_{00}||\check{U}_{00}| & |\check{L}_{00}||\check{u}_{01}| & |\check{L}_{00}||\check{U}_{02}| \\ \hline |\check{l}_{10}^T||\check{U}_{00}| & - & - \\ \hline |\check{L}_{20}||\check{U}_{00}| & - & - \end{array}\right). \end{array} \tag{5.2}$$

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

[4]In the error worksheet of Fig. 4.1, 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

$$
\begin{aligned}
& m\left(\begin{array}{c|c} A_{00} & a_{01} \\ \hline a_{10}^T & \alpha_{11} \end{array}\right) = k+1 \ \wedge \qquad (5.3) \\
& \left(\begin{array}{c|c|c} \check{L}_{00}\check{U}_{00} = A_{00} + \Delta A_{00} & \check{L}_{00}\check{u}_{01} = a_{01} + \delta a_{01} & \check{L}_{00}\check{U}_{02} = A_{02} + \Delta A_{02} \\ \hline \check{l}_{10}^T\check{U}_{00} = a_{10}^T + \delta a_{10}^T & \check{l}_{10}^T\check{u}_{01} + \check{v}_{11} = \alpha_{11} + \delta\alpha_{11} & \check{l}_{10}^T\check{U}_{02} + \check{u}_{12}^T = a_{12}^T + \delta a_{12}^T \\ \hline \check{L}_{20}\check{U}_{00} = A_{20} + \Delta A_{20} & \check{L}_{20}\check{u}_{01} + \check{l}_{21}\check{v}_{11} = a_{21} + \delta a_{21} & \text{—} \end{array}\right) \wedge \\
& \left|\left(\begin{array}{c|c|c} \Delta A_{00} & \delta a_{01} & \Delta A_{02} \\ \hline \delta a_{10}^T & \delta\alpha_{11} & \delta a_{12}^T \\ \hline \Delta A_{20} & \delta a_{21} & \text{—} \end{array}\right)\right| \le \gamma_{k+1} \left(\begin{array}{c|c|c} |\check{L}_{00}||\check{U}_{00}| & |\check{L}_{00}||\check{u}_{01}| & |\check{L}_{00}||\check{U}_{02}| \\ \hline |\check{l}_{10}^T||\check{U}_{00}| & |\check{l}_{10}^T||\check{u}_{01}| + |\check{v}_{11}| & |\check{l}_{10}^T||\check{U}_{02}| + |\check{u}_{12}^T| \\ \hline |\check{L}_{20}||\check{U}_{00}| & |\check{L}_{20}||\check{u}_{01}| + |\check{l}_{21}||\check{v}_{11}| & \text{—} \end{array}\right),
\end{aligned}
$$

which is the predicate that must be *true* after the execution of the 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 $\gamma_k \le \gamma_{k+1}$. 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 $\upsilon_{11} := \alpha_{11} - l_{10}^T u_{01}$ is executed in a state where $m(A_{00}) = k$. Theorem 3.9 R2-F states that there exists $\delta\upsilon_{11}$ such that

$$l_{10}^T u_{01} + \check{\upsilon}_{11} = \alpha_{11} + \delta\upsilon_{11}, \quad \text{where } |\delta\upsilon_{11}| \le \gamma_{k+1}(|l_{10}^T||u_{01}| + |\check{\upsilon}_{11}|),$$

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

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

$$l_{10}^T U_{02} + \check{u}_{12}^T = a_{12}^T + \delta u_{12}^T, \quad \text{where } |\delta u_{12}^T| \le \gamma_{k+1}\left(|l_{10}^T|\,|U_{02}| + |\check{u}_{12}^T|\right);$$

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

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

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

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

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 and Fig. 5.2 prove the following theorem.

THEOREM 5.2. *Let $A \in \mathbb{R}^{n\times n}$. Then $\check{L}$ and $\check{U}$, computed by the algorithm in Fig. 5.1, satisfy $\check{L}\check{U} = A + \Delta A$, where $|\Delta A| \le \gamma_n |\check{L}||\check{U}|$.*

The backward stability result in this theorem agrees with the one in [10]. The attentive reader will have noticed that the none of the bounds used to determine $\delta\alpha_{11}, \delta a_{12}^T, \delta a_{21}$ was tight. As a result, it can be shown that the bound for ΔA can be improved, replacing γ_n by $\max(\gamma_2, \gamma_{n-1})$.

5.5. Other algorithmic variants. The approach can be similarly applied to the other four variants for computing the LU factorization. In [2] it is shown how to use the error worksheet to derive an error result for the so-called bordered variant.

Error side	Step
$\{\ \Delta A = 0\ \}$	1a
Partition $A \rightarrow \left(\begin{array}{cc} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), L \rightarrow \left(\begin{array}{cc} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right), U \rightarrow \left(\begin{array}{cc} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right), \quad \Delta A \rightarrow \left(\begin{array}{cc} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & \Delta A_{BR} \end{array}\right)$ **where** A_{TL}, L_{TL}, U_{TL} and ΔA_{TL} are empty	4
$\left\{ \begin{array}{l} m(A_{TL}) = k \wedge \left(\begin{array}{cc} \check{L}_{TL}\check{U}_{TL} = A_{TL} + \Delta A_{TL} & \check{L}_{TL}\check{U}_{TR} = A_{TR} + \Delta A_{TR} \\ \hline \check{L}_{BL}\check{U}_{TR} = A_{BL} + \Delta A_{BL} & — \end{array}\right) \\ \wedge \left\lvert \left(\begin{array}{cc} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & — \end{array}\right) \right\rvert \leq \gamma_k \left(\begin{array}{cc} \lvert\check{L}_{TL}\rvert\lvert\check{U}_{TL}\rvert & \lvert\check{L}_{TL}\rvert\lvert\check{U}_{TR}\rvert \\ \hline \lvert\check{L}_{BL}\rvert\lvert\check{U}_{TL}\rvert & — \end{array}\right) \end{array} \right\}$	2a
While $m(A_{TL}) \leq m(A)$ **do**	3
Repartition A, L, U partitioned as in Fig. 5.1, and $\left(\begin{array}{cc} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & \Delta A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{ccc} \Delta A_{00} & \delta a_{01} & \Delta A_{02} \\ \hline \delta a_{10}^T & \delta\alpha_{11} & \delta a_{12}^T \\ \hline \Delta A_{20} & \delta a_{21} & \Delta A_{22} \end{array}\right)$ **where** $\delta\alpha_{11}$ is a scalars	5a
$\left\{ \begin{array}{l} m(A_{00}) = k \\ \wedge \left(\begin{array}{ccc} \check{L}_{00}\check{U}_{00} = A_{00} + \Delta A_{00} & \check{L}_{00}\check{u}_{01} = a_{01} + \delta a_{01} & \check{L}_{00}\check{U}_{02} = A_{02} + \Delta A_{02} \\ \hline \check{l}_{10}^T\check{U}_{00} = a_{10}^T + \delta a_{10}^T & — & — \\ \hline \check{L}_{20}\check{U}_{00} = A_{20} + \Delta A_{02} & — & — \end{array}\right) \\ \wedge \left\lvert \left(\begin{array}{ccc} \Delta A_{00} & \delta a_{01} & \Delta A_{02} \\ \hline \delta a_{10}^T & — & — \\ \hline \Delta A_{20} & — & — \end{array}\right) \right\rvert \leq \gamma_k \left(\begin{array}{ccc} \lvert\check{L}_{00}\rvert\lvert\check{U}_{00}\rvert & \lvert\check{L}_{00}\rvert\lvert\check{u}_{01}\rvert & \lvert\check{L}_{00}\rvert\lvert\check{U}_{02}\rvert \\ \hline \lvert\check{l}_{10}^T\rvert\lvert\check{U}_{00}\rvert & — & — \\ \hline \lvert\check{L}_{20}\rvert\lvert\check{U}_{00}\rvert & — & — \end{array}\right) \end{array} \right\}$	6
$\upsilon_{11} := \alpha_{11} - l_{10}^T u_{01}$ $u_{12}^T := a_{12}^T - l_{10}^T U_{02}$ $l_{21} := (a_{21} - L_{20}u_{01})/\upsilon_{11}$ $\check{\upsilon}_{11} + \delta\alpha_{11} = \alpha_{11} - l_{10}^T u_{01}$ $\wedge \lvert\delta\alpha_{11}\rvert \leq \gamma_{k+1}(\lvert l_{10}^T\rvert\lvert u_{01}\rvert + \lvert\check{\upsilon}_{11}\rvert)$ Th. 3.9 R2-F $\check{u}_{12}^T + \delta a_{12}^T = a_{12}^T - l_{10}^T U_{02}$ $\wedge \lvert\delta a_{12}^T\rvert \leq \gamma_{k+1}(\lvert l_{10}^T\rvert\lvert U_{02}\rvert + \lvert\check{u}_{12}^T\rvert)$ Th. 5.1 R2-F $\check{l}_{21}\upsilon_{11} + \delta a_{21} = a_{21} - L_{20}u_{01}$ $\wedge \lvert\delta a_{21}\rvert \leq \gamma_{k+1}(\lvert\check{L}_{20}\rvert\lvert\check{u}_{01}\rvert + \lvert\check{l}_{21}\rvert\lvert\upsilon_{11}\rvert)$ Th. 5.1 R4-F	8
$\left\{ \begin{array}{l} m\left(\begin{array}{cc} A_{00} & a_{01} \\ \hline a_{10}^T & \alpha_{11} \end{array}\right) = k+1 \\ \wedge \left(\begin{array}{ccc} \check{L}_{00}\check{U}_{00} = A_{00} + \Delta A_{00} & \check{L}_{00}\check{u}_{01} = a_{01} + \delta a_{01} & \check{L}_{00}\check{U}_{02} = A_{02} + \Delta A_{02} \\ \hline \check{l}_{10}^T\check{U}_{00} = a_{10}^T + \delta a_{10}^T & \check{l}_{10}^T\check{u}_{01} + \check{\upsilon}_{11} = \alpha_{11} + \delta\alpha_{11} & \check{l}_{10}^T\check{U}_{02} + \check{u}_{12}^T = a_{12}^T + \delta a_{12}^T \\ \hline \check{L}_{20}\check{U}_{00} = A_{20} + \Delta A_{20} & \check{L}_{20}\check{u}_{01} + \check{l}_{21}\check{\upsilon}_{11} = a_{21} + \delta a_{21} & — \end{array}\right) \\ \wedge \left\lvert \left(\begin{array}{ccc} \Delta A_{00} & \delta a_{01} & \Delta A_{02} \\ \hline \delta a_{10}^T & \delta\alpha_{11} & \delta a_{12}^T \\ \hline \Delta A_{20} & \delta a_{21} & — \end{array}\right) \right\rvert \leq \gamma_{k+1} \left(\begin{array}{ccc} \lvert\check{L}_{00}\rvert\lvert\check{U}_{00}\rvert & \lvert\check{L}_{00}\rvert\lvert\check{u}_{01}\rvert & \lvert\check{L}_{00}\rvert\lvert\check{U}_{02}\rvert \\ \hline \lvert\check{l}_{10}^T\rvert\lvert\check{U}_{00}\rvert & \lvert\check{l}_{10}^T\rvert\lvert\check{u}_{01}\rvert + \lvert\check{\upsilon}_{11}\rvert & \lvert\check{l}_{10}^T\rvert\lvert\check{U}_{02}\rvert + \lvert\check{u}_{12}^T\rvert \\ \hline \lvert\check{L}_{20}\rvert\lvert\check{U}_{00}\rvert & \lvert\check{L}_{20}\rvert\lvert\check{u}_{01}\rvert + \lvert\check{l}_{21}\rvert\lvert\check{\upsilon}_{11}\rvert & — \end{array}\right) \end{array} \right\}$	7
Continue with A, L and U as in Fig. 5.1, and $\left(\begin{array}{cc} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & \Delta A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{ccc} \Delta A_{00} & \delta a_{01} & \Delta A_{02} \\ \hline \delta a_{10}^T & \delta\alpha_{11} & \delta a_{12}^T \\ \hline \Delta A_{20} & \delta a_{21} & \Delta A_{22} \end{array}\right)$	5b
endwhile	
$\left\{ m(A) = n \wedge \check{L}\check{U} = (A + \Delta A) \wedge \lvert\Delta A\rvert \leq \gamma_n \lvert L\rvert\lvert U\rvert \right\}$	1b

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

5.6. Solution of a linear system. The following theorem summarizes how the computed solution, $\check{x}$, is the exact solution of a perturbed linear system $(A+\Delta A)\check{x} = y$:

THEOREM 5.3. *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. 3.1(right). Let $\check{x}$ 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. 3.1(right). Then $\check{x}$ satisfies $(A + \Delta A)\check{x} = b$ with $|\Delta A| \leq (3\gamma_n + \gamma_n^2)|\check{L}||U|$.*

Proof: From previous sections we have learned that

$$\begin{array}{rcll} \check{L}\check{U} &=& A + E_1 \quad \text{with } |E_1| \leq \gamma_n|\check{L}||\check{U}| & \text{(Th. 5.2)},\\ (\check{L} + E_2)\check{z} &=& b \quad \text{with } |E_2| \leq \gamma_n|\check{L}| & \text{(Th. 4.1)},\\ (\check{U} + E_3)\check{x} &=& \check{z} \quad \text{with } |E_3| \leq \gamma_n|\check{U}| & \text{(Th. 4.1)}. \end{array}$$

Therefore

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

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

□

5.7. Partial pivoting. The analysis of LU factorization without partial pivoting is related to 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 that the computations occur in a similar order on similar data and that therefore the numerical properties are similar. A traditional analysis of this algorithm, yielding a less tight bound then the one we derive, can be found in [5, 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

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

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

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

is satisfied at the beginning and at the end of each iteration. Superscripts denote the iteration number. For simplicity, in our analysis, we assume that n is a multiple of the block size b.

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 5.2. Here we present theorems related to triangular solve with multiple right-hand sides and matrix-matrix multiplication.

COROLLARY 6.1. **Error results for matrix-matrix multiplication.** *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 $Z \rightarrow \left(z_0 | \cdots | z_{n-1}\right), B \rightarrow \left(b_0 | \cdots | b_{n-1}\right)$, and $C \rightarrow \left(c_0 | \cdots | c_{n-1}\right)$, so that $Z = \left(c_0 - Ab_0 | \cdots | c_{n-1} - Ab_{n-1}\right) = C - AB$. Then*

R1-F: $AB + \check{Z} = C + \Delta Z$, where $|\Delta Z| \le \gamma_k|A||B| + \gamma_1|\check{Z}|$.
R2-F: $AB + \check{Z} = C + \Delta C$, where $|\Delta C| \le \gamma_{k+1}|A||B| + \gamma_1|\check{C}|$.

$$\vdots$$

$$\left\{\begin{array}{l} m(A_{TL}) = k \wedge \left(\begin{array}{c|c} \check{L}_{TL}\check{U}_{TL} = A_{TL} + \Delta A_{TL} & \check{L}_{TL}\check{U}_{TR} = A_{TR} + \Delta A_{TR} \\ \hline \check{L}_{BL}\check{U}_{TL} = A_{BL} + \Delta A_{BL} & \check{A}^i_{BR} = A_{BR} - \check{L}_{BL}\check{U}_{TR} + \Delta A^i_{BR} \end{array}\right) \\ \wedge \left|\left(\begin{array}{c|c} \Delta A_{TL} & \Delta A_{TR} \\ \hline \Delta A_{BL} & \Delta A^i_{BR} \end{array}\right)\right| \le \gamma_{\frac{n}{b}+b} \left(\begin{array}{c|c} |A_{TL}| + |\check{L}_{TL}||\check{U}_{TL}| & |A_{TR}| + |\check{L}_{TL}||\check{U}_{TR}| \\ \hline |A_{TL}| + |\check{L}_{BL}||\check{U}_{TL}| & |A_{BR}| + |\check{L}_{BL}||\check{U}_{TR}| \end{array}\right) \end{array}\right\} \quad \text{2a}$$

$$\vdots$$

$$\left\{\begin{array}{l} m(A_{00}) = k \wedge \left(\begin{array}{c|c|c} \star & \star & \star \\ \hline \star & \check{A}^i_{11} = A_{11} - \check{L}_{10}\check{U}_{01} + \Delta A^i_{11} & \check{A}^i_{12} = A_{12} - \check{L}_{10}\check{U}_{02} + \Delta A^i_{12} \\ \hline \star & \check{A}^i_{21} = A_{21} - \check{L}_{20}\check{U}_{01} + \Delta A^i_{21} & \check{A}^i_{22} = A_{22} - \check{L}_{20}\check{U}_{02} + \Delta A^i_{22} \end{array}\right) \\ \wedge \left|\left(\begin{array}{c|c|c} \star & \star & \star \\ \hline \star & \Delta A^i_{11} & \Delta A^i_{12} \\ \hline \star & \Delta A^i_{21} & \Delta A^i_{22} \end{array}\right)\right| \le \gamma_{\frac{k}{b}+b} \left(\begin{array}{c|c|c} \star & \star & \star \\ \hline \star & |A_{11}| + |\check{L}_{10}|\,|\check{U}_{01}| & |A_{12}| + |\check{L}_{10}|\,|\check{U}_{02}| \\ \hline \star & |A_{21}| + |\check{L}_{20}|\,|\check{U}_{01}| & |A_{22}| + |\check{L}_{20}|\,|\check{U}_{02}| \end{array}\right) \end{array}\right\} \quad 6$$

$[L_{11}, U_{11}] := \mathrm{LU}(A_{11})$	$\check{L}_{11}\check{U}_{11} = A_{11} + \Delta A_{11}$	$\wedge\ \lvert\Delta\check{A}_{11}\rvert \le \gamma_b\lvert\check{L}_{11}\rvert\lvert\check{U}_{11}\rvert$	Theorem 5.2	8
$U_{12} := L_{11}^{-1}A_{12}$	$\check{L}_{11}\check{U}_{12} = A_{12} + \Delta A_{12}$	$\wedge\ \lvert\Delta\check{A}_{12}\rvert \le \gamma_b\lvert\check{L}_{11}\rvert\lvert\check{U}_{12}\rvert$	Corollary 6.2	
$L_{21} := A_{21}U_{11}^{-1}$	$\check{L}_{21}\check{U}_{11} = A_{21} + \Delta A_{21}$	$\wedge\ \lvert\Delta\check{A}_{21}\rvert \le \gamma_b\lvert\check{L}_{21}\rvert\lvert\check{U}_{11}\rvert$	Corollary 6.2	
$A_{22} := A_{22} - L_{21}U_{12}$	$\check{A}^{i+1}_{22} + \check{L}_{21}\check{U}_{12} = A^i_{22} + \Delta\check{A}^{i+1}_{22}$	$\wedge\ \lvert\Delta\check{A}^{i+1}_{22}\rvert \le \gamma_{b+1}(\lvert\check{A}^i_{22}\rvert + \lvert\check{L}_{21}\rvert\lvert\check{U}_{12}\rvert)$	Corollary 6.1	

$$\left\{\begin{array}{l} m\left(\begin{array}{c|c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array}\right) = k + b \wedge \left(\begin{array}{c|c|c} \star & \star & \star \\ \hline \star & \check{L}_{11}\check{U}_{11} = A_{11} - \check{L}_{10}\check{U}_{01} + \Delta A^{i+1}_{11} & \check{L}_{10}\check{U}_{02} + \check{L}_{11}\check{U}_{12} = A_{12} + \Delta A^{i+1}_{12} \\ \hline \star & \check{L}_{20}\check{U}_{01} + \check{L}_{21}\check{U}_{11} = A_{21} + \Delta A^{i+1}_{21} & \check{A}^{i+1}_{22} = A_{22} - \check{L}_{20}\check{U}_{02} - \check{L}_{21}\check{U}_{12} + \Delta A^{i+1}_{22} \end{array}\right) \\ \wedge \left|\left(\begin{array}{c|c|c} \star & \star & \star \\ \hline \star & \Delta A^{i+1}_{11} & \Delta A^{i+1}_{12} \\ \hline \star & \Delta A^{i+1}_{21} & \Delta A^{i+1}_{22} \end{array}\right)\right| \le \gamma_{\frac{k}{b}+b+1} \left(\begin{array}{c|c|c} \star & \star & \star \\ \hline \star & |A_{11}| + |\check{L}_{10}|\,|\check{U}_{01}| + |\check{L}_{11}|\,|\check{U}_{11}| & |A_{12}| + |\check{L}_{10}|\,|\check{U}_{02}| + |\check{L}_{11}|\,|\check{U}_{12}| \\ \hline \star & |A_{21}| + |\check{L}_{20}|\,|\check{U}_{01}| + |\check{L}_{21}|\,|\check{U}_{11}| & |A_{22}| + |\check{L}_{20}|\,|\check{U}_{02}| + |\check{L}_{21}|\,|\check{U}_{12}| \end{array}\right) \end{array}\right\} \quad 7$$

$$\vdots$$

FIG. 6.2. *$LU = A$. Details for Steps 6–8 in the proof of Theorem 6.3.*

The following result follows immediately from Theorem 4.1:

COROLLARY 6.2. **Error results for triangular solve with multiple right-hand sides.** *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. 4.1(left). Then $\check{X}$ satisfies $L\check{X} = B + \Delta B$ where $|\Delta B| \leq \gamma_n |L||\check{X}|$.*

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 6.3. *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*

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

The worksheet containing the proof of Theorem 6.3 is shown in Figs. 6.1 (right) and 6.2. In the remainder of the section we prove that the error bounds indicated in the worksheet hold. Consider the error-invariant

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

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 $\check{A}_{BR} = A_{BR} + \Delta A_{BR}$ with $|\Delta A_{BR}| \leq \gamma_b |A_{BR}|$, which is true as no computation has been performed yet. Thus $\check{A}_{BR}$ equals A_{BR}, and $\Delta A_{BR} = 0$.

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

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

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

$$m\left(\begin{array}{c|c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array}\right) = k + b \wedge \tag{6.3}$$

$$\left(\begin{array}{c|c|c} \star & \star & \star \\ \hline \star & \check{L}_{10}\check{U}_{01} + \check{L}_{11}\check{U}_{11} = A_{11} + \Delta A_{11}^{i+1} & \check{L}_{10}\check{U}_{02} + \check{L}_{11}\check{U}_{12} = A_{12} + \Delta A_{12}^{i+1} \\ \hline \star & \check{L}_{20}\check{U}_{01} + \check{L}_{21}\check{U}_{11} = A_{21} + \Delta A_{21}^{i+1} & A_{22}^{i+1} = A_{22} - \check{L}_{20}\check{U}_{02} - \check{L}_{21}\check{U}_{12} + \Delta A_{22}^{i+1} \end{array}\right) \wedge$$

$$\left|\left(\begin{array}{c|c|c} \star & \star & \star \\ \hline \star & \Delta A_{11}^{i+1} & \Delta A_{12}^{i+1} \\ \hline \star & \Delta A_{21}^{i+1} & \Delta A_{22}^{i+1} \end{array}\right)\right| \le \gamma_{\frac{k}{b}+b+1} \left(\begin{array}{c|c|c} \star & \star & \star \\ \hline \star & |A_{11}| + |\check{L}_{10}||\check{U}_{01}| + |\check{L}_{11}||\check{U}_{11}| & |A_{12}| + |\check{L}_{10}||\check{U}_{02}| + |\check{L}_{11}||\check{U}_{12}| \\ \hline \star & |A_{21}| + |\check{L}_{20}||\check{U}_{01}| + |\check{L}_{21}||\check{U}_{11}| & |A_{22}| + |\check{L}_{20}||\check{U}_{02}| + |\check{L}_{21}||\check{U}_{12}| \end{array}\right).$$

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 $\gamma_{\frac{k}{b}+b} \le \gamma_{\frac{k}{b}+b+1}$.

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 $[L_{11}, U_{11}] := \mathrm{LU}(\check{A}_{11})$, with $\check{A}_{11} \in \mathbb{R}^{b\times b}$. Theorem 5.2 states that there exists a matrix $\Delta A_{11}^{(\mathrm{LU})}$ such that

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

From Step 6 (predicate (6.2)), we know that $\check{A}_{11} = A_{11} - \check{L}_{10}\check{U}_{01} + \Delta A_{11}^{i}$, with $\Delta A_{11}^{i} \le \gamma_{\frac{k}{b}+b}(|A_{11}| + |\check{L}_{10}||\check{U}_{01}|)$, and substituting $\check{A}_{11}$ into (6.4), it results:

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

Therefore, rearranging and setting $\Delta A_{11}^{i+1} = \Delta A_{11}^{i} + \Delta A_{11}^{(\mathrm{LU})}$, 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 $|\Delta A_{11}^{i+1}| \le \gamma_{\frac{k}{b}+b+1}(|A_{11}| + |\check{L}_{10}||\check{U}_{01}| + |\check{L}_{11}||\check{U}_{11}|)$:

$$\begin{array}{rll} |\Delta A_{11}^{i+1}| = \left|\Delta A_{11}^{i} + \Delta A_{11}^{(\mathrm{LU})}\right| & \le |\Delta A_{11}^{i}| + \gamma_b |\check{L}_{11}||\check{U}_{11}| & \text{Def., Tr.In., Thrm.5.2} \\ & \le \gamma_{\frac{k}{b}+b}(|A_{11}| + |\check{L}_{10}||\check{U}_{01}|) + \gamma_b |\check{L}_{11}||\check{U}_{11}| & \text{I.H.} \\ & \le \gamma_{\frac{k}{b}+b+1}(|A_{11}| + |\check{L}_{10}||\check{U}_{01}| + |\check{L}_{11}||\check{U}_{11}|). & \Box \end{array}$$

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 6.2 states that there exists an error matrix $\Delta A_{12}^{(\mathrm{TRSM})}$ such that

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

From Step 6 (predicate (6.2)), $\check{A}_{12} = A_{12} - \check{L}_{10}\check{U}_{02} + \Delta A_{12}^{i}$, with $|\Delta A_{12}^{i}| \le \gamma_{\frac{k}{b}+b}(|A_{12}| + |\check{L}_{10}||\check{U}_{02}|)$. Substituting $\check{A}_{12}$ into (6.5) we obtain $\check{L}_{11}\check{U}_{12} =$

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

$$\begin{aligned}
|\Delta A^{i+1}_{12}| &= \left|\Delta A^{i}_{12} + \Delta A^{(\mathrm{TRSM})}_{12}\right| \leq |\Delta A^{i}_{12}| + \gamma_b|\check{L}_{11}||\check{U}_{12}| && \text{Def., Tr.In., Cor.6.2}\\
&\leq \gamma_{\frac{k}{b}+b}(|A_{12}| + |\check{L}_{10}||\check{U}_{02}|) + \gamma_b|\check{L}_{11}||\check{U}_{12}| && \text{I.H.}\\
&\leq \gamma_{\frac{k}{b}+b+1}(|A_{12}| + |\check{L}_{10}||\check{U}_{02}| + |\check{L}_{11}||\check{U}_{12}|). && \square
\end{aligned}$$

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

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

Predicate (6.2) from Step 6 tells us that $\check{A}^{i}_{22} = A_{22} - \check{L}_{20}\check{U}_{02} + \Delta A^{i}_{22}$, with $|\Delta A^{i}_{22}| \leq \gamma_{\frac{k}{b}+b}(|A_{22}| + |\check{L}_{20}||\check{U}_{02}|)$, and substituting $\check{A}^{i}_{22}$ into (6.6), yields $\check{A}^{i+1}_{22} = A_{22} - \check{L}_{20}\check{U}_{02} + \Delta A^{i}_{22} - \check{L}_{21}\check{U}_{12} + \Delta A^{(\mathrm{GEMM})}_{22}$. Rearranging, and setting $\Delta A^{i+1}_{22} = \Delta A^{i}_{22} + \Delta A^{(\mathrm{GEMM})}_{22}$, we obtain $\check{A}^{i+1}_{22} = A_{22} - \check{L}_{20}\check{U}_{02} - \check{L}_{21}\check{U}_{12} + \Delta A^{i+1}_{22}$, which is the expression we were seeking as shown in (6.3). Next, we prove that $|\Delta A^{i+1}_{22}| \leq \gamma_{\frac{k}{b}+b+1}(|A_{22}| + |\check{L}_{20}||\check{U}_{02}| + |\check{L}_{21}||\check{U}_{12}|)$:

$$\begin{aligned}
|\Delta A^{i+1}_{22}| &= \left|\Delta A^{i}_{22} + \Delta A^{(\mathrm{GEMM})}_{22}\right| \leq |\Delta A^{i}_{22}| + \left|\Delta A^{(\mathrm{GEMM})}_{22}\right| && \text{Def., Tr.In.}\\
&\leq |\Delta A^{i}_{22}| + \gamma_{b+1}|\check{L}_{21}||\check{U}_{12}| + \gamma_1|\check{A}^{i}_{22}| && \text{Cor. 6.1}\\
&\leq |\Delta A^{i}_{22}| + \gamma_{b+1}|\check{L}_{21}||\check{U}_{12}| + \gamma_1(|A_{22}| + |\check{L}_{20}||\check{U}_{02}| + |\Delta A^{i}_{22}|) && \text{I.H., Tr.In.}\\
&= (1+\gamma_1)|\Delta A^{i}_{22}| + \gamma_{b+1}|\check{L}_{21}||\check{U}_{12}| + \gamma_1(|A_{22}| + |\check{L}_{20}||\check{U}_{02}|)\\
&\leq (1+\gamma_1)\gamma_{\frac{k}{b}+b}(|A_{22}| + |\check{L}_{20}||\check{U}_{02}|) +\\
&\quad \gamma_{b+1}|\check{L}_{21}||\check{U}_{12}| + \gamma_1(|A_{22}| + |\check{L}_{20}||\check{U}_{02}|) && \text{I.H.}\\
&\leq (\gamma_1 + \gamma_{\frac{k}{b}+b} + \gamma_1\gamma_{\frac{k}{b}+b})(|A_{22}| + |\check{L}_{20}||\check{U}_{02}|) + \gamma_{b+1}|\check{L}_{21}||\check{U}_{12}|\\
&\leq \gamma_{\frac{k}{b}+b+1}(|A_{22}| + |\check{L}_{20}||\check{U}_{02}|) + \gamma_{b+1}|\check{L}_{21}||\check{U}_{12}| && \text{Lem. 3.3}\\
&\leq \gamma_{\frac{k}{b}+b+1}(|A_{22}| + |\check{L}_{20}||\check{U}_{02}| + |\check{L}_{21}||\check{U}_{12}|). && \square
\end{aligned}$$

This concludes the proof of the inductive step of Theorem 6.3. 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 $\left(\frac{A_{11}}{A_{21}}\right)$, 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. It extends the FLAME methodology for deriving algorithms so that numerical stability results are established in a goal-oriented and modular fashion. In addition, it has yielded a new bound for the backward stability of blocked LU factorization.

While the paper was written so that later results build on earlier ones, the methodology is best applied by starting from the target algorithm to be analyzed and working backwards. For example, we could have started with the blocked LU factorization. As part of the analysis, it would have become clear that 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. The results would have then slowly built back up to the analysis of the blocked algorithm.

Just like it has been shown that systematic derivation of algorithms can be made mechanical [2], we believe the proposed approach can also be made mechanical by a system that understands the rules of linear algebra. Automation and the application of the proposed techniques to more complex operations are the topic of future research.

8. Acknowledgments. We are indebted to G.W. (Pete) Stewart for his encouragements as we pursued this research. We also thank Enrique Quintana-Ortí and Maggie Myers for feedback on an early draft.

The first author gratefully acknowledges the Deutsche Forschungsgemeinschaft (German Research Association) for financial support through grant GSC 111. This research 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. 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.

[3] 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.

[4] P. Bientinesi and R. A. van de Geijn, *The science of deriving stability analyses. FLAME Working Note #33*, Technical Report AICES-2008-2, Aachen Institute for Computational Engineering Sciences, RWTH Aachen, November 2008.

[5] J. W. Demmel and N. J. Higham, *Stability of block algorithms with fast level-3 blas*, ACM Transactions on Mathematical Software, 18 (1992), pp. 274–291.

[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. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 3nd ed., 1996.

[9] 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.

[10] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second ed., 2002.

[11] G. W. Stewart, *Matrix Algorithms. Volume I: Basic Decompositions*, SIAM, Philadelphia, 1998.

[12] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory (Computer Science and Scientific Computing) (Computer Science and Scientific Computing)*, Academic Press, June 1990.