Effects of Underflow on Solving Linear Systems

James Demmel[1]

Computer Science Division
Electronic Research Laboratory
U.C. Berkeley
May 1981

ABSTRACT

In this paper we examine the effects of underflow on solving systems of linear equations using Gaussian Elimination. Our goal is to decide if reliable software for solving linear systems in the presence of underflow can be written at reasonable cost. We contrast the utilities of gradual underflow and "store zero", and show that only by using gradual underflow can we achieve reliability easily.

1. Introduction

In this paper we examine the effects of underflow on solving systems of linear equations using Gaussian elimination. In particular, we contrast the effects of gradual underflow [Coonen,1981] and "store zero" on the accuracy and stability of the algorithm.

Our ultimate goal is to decide whether reliable software for solving linear systems in the presence of underflow can be written at a reasonable cost, where by reliable we mean a piece of software that ideally

1. produces accurate results whenever they can be represented,
2. gives a warning whenever the computed results are inaccurate, and
3. avoids giving warnings when it is possible to compute accurate results.

Our motivation for this study is twofold. First, that underflow is a problem at all in linear equation solvers is not generally recognized. For example, the LINPACK Users' Guide [Dongarra et al.,1979] states in its introduction that "Any underflows which occur are harmless." This statement is almost always true, since underflow's status as an end effect means it may be safely ignored in most situations, but it is not always true. The advantage of gradual underflow over store zero is that it allows a cleaner line to be drawn between problems where underflow can be ignored and those where it cannot, and thus helps the writer of reliable software who wants to be able to guarantee when his program will work. Our first motivation, then, is to make people aware that underflow can give reasonable looking but totally inaccurate results if it is handled poorly or ignored.

Second, we wish to attenuate the controversy surrounding the recent decision on how to handle underflow by the IEEE Microprocessor Standards Committee. The decision to use gradual underflow instead of the usual store zero approach came after much argument about the advantages of one over the other in making reliable numerical software easier to write. [Coonen,1981]. In this paper we show that the advantage is significant.

1. The author gratefully acknowledges the support of the U. S. Department of Energy, Contract DE-AM03-76SF00034, Project Agreement DE-AS03-79ER10358, and the Office of Naval Research Contract N00014-76-C-0013.

Specifically, the algorithm to solve $Ax=b$ is as follows:

(1) Decompose $A=LU$ = (lower triangular)(upper triangular) using pivoting, so that the diagonal of L contains all 1's and all other entries of L are ≤ 1 in absolute value.

(2) Solve $Ly=b$ for y (forward substitution).

(3) Solve $Ux=y$ for x (back substitution).

The only gradual underflows that can possibly contribute significantly to the error are underflows in the final solution vector x. Intermediate gradual underflows contribute an error with a bound scarcely worse than the bound for the error contributed by roundoff alone. Thus, by using gradual underflow we are able to satisfy the first goal of reliable software since an alarm need not be raised unless the results themselves underflow and are not representable (in which case the data would have to be scaled to avoid underflow).

In contrast, storing zero in place of intermediate underflows during any stage of solution using store zero can contribute significantly to the error, possibly producing reasonable looking results whose error greatly exceeds the uncertainty attributable to roundoff alone (see the Examples).

We may also satisfy the second and third goals of reliable software more easily with gradual underflow than with store zero. It is easy to tell which small class of possible gradual underflows (they are the underflows in the answer itself) can contribute significant errors and require a warning; with store zero, on the other hand, we must either do tedious testing to tell which underflows require a warning, or naively raise an alarm whenever an underflow occurs. This naive approach produces "paranoid" code and many false alarms.

It is important to understand how the unavoidable uncertainty due to roundoff can affect our ability to produce accurate results. Guaranteeing results to within a certain accuracy is not possible for a price most people are willing to pay. We can only guarantee that our computed answer is the solution of a new unknown problem perturbed slightly from our original problem. This is the nature of Gaussian elimination, and confirmed by backwards error analysis, the approach used in this paper; we want to bound that perturbation, to bound how different the new problem is from the old. If the problem is ill-conditioned, then the solution of the new problem may be very different from the solution of the original problem, and there is no way, at reasonable cost, to tell. Thus, our original goal of producing accurate results should be modified to read: "satisfy the equations as closely as possible; achieve a tiny residual." This is an achievable goal.

In the remainder of this paper we will present examples to demonstrate the typical effects of underflow, describe the model of arithmetic used and the approach used in the error analysis, and present the conclusions drawn from the error analysis. Appendices contain program listings and an analysis of Cholesky decomposition for solving positive definite linear systems.

2. Examples

We assume the reader is acquainted with arithmetic using gradual underflow (henceforth G.U.) and store zero (henceforth S.Z.) ([Coonen,1981],[Kahan and Palmer,1979],[Coonen et al.,1979]). Let ϵ be the rounding error of the arithmetic, and λ be the underflow threshold. Suppose, for example, a binary floating point number is represented as $f \cdot 2^e$ where f lies between 1 and 2 with an n bit fraction, and e is an integer exponent in the range $e_{min} \leq e \leq e_{max}$. Then $\epsilon = 2^{-n}$ (the difference between 1 and the next largest number). The underflow threshold is $\lambda = 2^{e_{min}}$; this is the smallest positive number in S.Z. arithmetic, and the smallest normalized number for G.U. Whenever a nonzero number smaller than $2^{e_{min}}$ is generated by arithmetic, underflow is signaled and something special happens; S.Z. replaces the number by zero, and G.U. by a nearest "denormalized" number or zero. Denormalized numbers lie between 0 and $2^{e_{min}}$ in G.U. arithmetic for which the smallest nonzero number is thus $\mu = \epsilon\lambda$. We will give a more precise model of rounding and underflow errors in the next section.

In this section we present three examples of the effects of underflow on performing the decomposition $A=LU$. The first example shows how store zero can produce a reasonable looking but completely inaccurate decomposition of a well conditioned matrix, whereas gradual underflow either

produces the correct decomposition or correctly decides the matrix is singular. (There are no rounding errors or pivot growth in this example.) The second example shows that G.U. produces the correct decomposition of a matrix which S.Z. incorrectly decides is singular. Finally, we present an innocuous looking ordinary differential equation and show that the linear system arising from trying to solve it numerically leads to underflow which is handled correctly by G.U. and not by S.Z.

2.1. Example 1

Consider the family of matrices $A(x)$ where

$$A(x) = \lambda \cdot \begin{bmatrix} 2 & & & & 1 \\ & 2 & & & 1 \\ & & 2 & & 1 \\ & & & 2 & 1 \\ 1 & 1 & 1 & 1 & x \end{bmatrix}, \tag{1}$$

(blanks denote zero entries). The LU decomposition obtained by G.U. is

$$L^{GU}(x) \cdot U^{GU}(x) = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ .5 & .5 & .5 & .5 & 1 \end{bmatrix} \cdot \lambda \cdot \begin{bmatrix} 2 & & & & 1 \\ & 2 & & & 1 \\ & & 2 & & 1 \\ & & & 2 & 1 \\ & & & & x-4 \end{bmatrix} = A(x), \tag{2}$$

and by S.Z. is

$$L^{SZ}(x) \cdot U^{SZ}(x) = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ .5 & .5 & .5 & .5 & 1 \end{bmatrix} \cdot \lambda \cdot \begin{bmatrix} 2 & & & & 1 \\ & 2 & & & 1 \\ & & 2 & & 1 \\ & & & 2 & 1 \\ & & & & x \end{bmatrix} = A(x) + E, \tag{3}$$

where the error matrix E equals

$$E = \lambda \cdot \begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & -4 \end{bmatrix}. \tag{4}$$

We see using S.Z. results in a large error in the $U(x)_{55}$ entry, whereas using G.U. gives the correct decomposition. If $x=4$, we see that by using S.Z. we decide the matrix is far from singular, when in fact it is exactly singular. Note that the matrix $A(x)$ is well conditioned when x is far from 4, and if x is a small integer, no rounding errors occur in either decomposition.

2.2. Example 2

Let

$$A = \lambda \cdot \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}, \tag{5}$$

a well conditioned matrix. Using G.U. we obtain

$$L^{GU} \cdot U^{GU} = \begin{bmatrix} 1 & \\ .5 & 1 \end{bmatrix} \cdot \lambda \cdot \begin{bmatrix} 2 & 3 \\ & .5 \end{bmatrix} = A, \tag{6}$$

and by using S.Z. we obtain

$$L^{SZ} \cdot U^{SZ} = \begin{bmatrix} 1 & \\ .5 & 1 \end{bmatrix} \cdot \lambda \cdot \begin{bmatrix} 2 & 3 \\ & 0 \end{bmatrix} . \tag{7}$$

Thus, using G.U. we correctly decompose the matrix A, whereas using S.Z. we incorrectly decide the matrix is singular.

2.3. Example 3

Consider the ordinary differential equation

$$\dot{x}(t) = \frac{1-(t/T)^M}{T-t}\, x(t) \ , \quad x(T_s) = c \ . \tag{8}$$

We try to solve this equation numerically by representing $x(t)$ as the truncated power series $x(t) = \sum_{n=1}^{N} x_n t^n$, the function $1/(T-t)$ by its truncated power series, and equating coefficients of equal powers of t on both sides of equation (8). After we scale the last row (which represents the initial condition) down to have largest entry equal to 1, we get the linear system $Ax=b$, where

$$A = \begin{bmatrix} N & -1/T & -1/T^2 & \cdots & & -1/T^N \\ & N-1 & -1/T & \cdots & & -1/T^{N-1} \\ & & N-2 & \cdots & & -1/T^{N-2} \\ & & & \cdot & & \cdot \\ & & & & \cdot & \cdot \\ & & & & & \cdot \\ & & & & 1 & -1/T \\ 1 & 1/T_s & 1/T_s^2 & \cdots & 1/T_s^{N-1} & 1/T_s^N \end{bmatrix} , \tag{9}$$

$b^T = (0, \cdots, 0, c/T_0^N)$, and $x^T = (x_N, \cdots, x_0)$.

We chose $M=15$, $N=14$, $T=512.$, $T_s=500.$, and $c=100.$ for this example. We used an implementation of the IEEE Floating Point Standard [J. Coonen et al.,1979] on a VAX11. ϵ was 2^{-23} and μ was 2^{-149}. There was a switch on the compiler to enable/disable G.U., so we were able to obtain numerical results using both G.U. and S.Z. This example was chosen to be simple but realistic; even though it can easily be solved analytically, it could be changed easily into a two dimensional problem without an explicit solution, but with the same sensitivity to underflow.

L and U have a simple structure. L will be zero below the diagonal, except for the last row, which is graded from $L_{15,1} \approx 7.14285_{10} -2$ down to $L_{15,14} \approx 5.34726_{10} -35$. U is identical to A in all but its last row. A's columns are badly scaled, although this is not obvious because no row nor column is drastically smaller in norm than any other; nonetheless, bad scaling causes A to *appear* very ill conditioned, and this ill conditioning shows up in the last row of U, making $U_{15,15}$ very small, in fact barely above the underflow threshold. S.Z. and G.U. compute all elements of L and U identically except for $U_{15,15}$. If we scale up the last column of U to avoid all underflows, we compute $U_{15,15} \approx 2.09261e{-}37$. We get the identical value using G.U., but using S.Z. we compute $U^{SZ}_{15,15} \approx 1.72763e{-}37$, a relative difference of .174. This relative difference in the last entry of U is very important, because one divides by $U_{15,15}$ in the course of solution. The relative difference in solution vectors x is

$$\frac{\| x^{GU} - x^{SZ} \|_\infty}{\| x^{GU} \|_\infty} \approx .211 \ .$$

Thus, G.U. obtains markedly better results than S.Z. This example is very interesting because there is nothing obviously wrong about the matrix. All its entries are normalized, and every row and every column contains reasonably large values, yet 11 out of 14 products $L_{15,j} * U_{j,15}$ in the sum for $U_{15,15}$ underflow just slightly below the underflow threshold. Since the true value of $U_{15,15}$ is itself not much larger than the underflow threshold, this makes for a large relative error.

3. Error Analysis

3.1. Assumptions

In this section we define our notation and our model of arithmetic. ϵ will denote the level of roundoff error and μ the smallest nonzero number when using G.U. Then $\lambda = \mu/\epsilon$ will be the underflow threshold and smallest normalized number. Using S.Z. the only number smaller than λ in magnitude is zero; using G.U. the numbers between λ and zero in magnitude are called denormalized numbers. Zero and all numbers no smaller than λ in magnitude are normalized in both G.U. and S.Z.

Let be one of the operations $(+,-,*,/)$ and let fl(a b) denote the floating point result of the indicated computation. Traditionally, error analyses have used the formula

$$fl(a \quad b) = (a \quad b)(1 + e) \qquad \text{unless } a \quad b \text{ underflows or overflows.} \tag{10}$$

To take underflow into account, we write ([Kahan and Palmer, 1979])

$$fl(a \quad b) = (a \quad b)(1 + e) + \eta \qquad \text{unless } a \quad b \text{ overflows.} \tag{11}$$

In the case of G.U. we have the following constraints on e and η:

(1) $|e| \leq \epsilon$ and $|\eta| \leq \lambda\epsilon$,

(2) $\eta \times e = 0$,

(3) $\eta = 0$ if is either addition or subtraction.

In the case of S.Z. we have the following somewhat different constraints on e and η:

(1) $|e| \leq \epsilon$ and $|\eta| \leq \lambda$, and

(2) $\eta \times e = 0$.

Note that η need not be 0 when performing an addition or subtraction using S.Z.

We define the function $UN(x)$ to be

$$UN(x) = \begin{cases} 1 & \text{if } x \text{ underflows } (x<\lambda) \\ 0 & \text{if it does not} \end{cases}$$

We assume no overflow occurs.

We ignore terms which are $O(\epsilon^2)$ or $O(\epsilon\mu)$. This allows us to make the following substitutions when convenient:

Replace $1/(1+e_1)$ by $1+e_2$,

Replace $(1+e_1)^j$ by $1+je_2$,

Replace $\prod_{j=1}^{m}(1+e_j)/\prod_{i=1}^{n}(1+e'_i)$ by $1+(n+m)e$

Replace $\eta_1*(1+e_1)^j$ by η_2 .

By replacing every appearance of an original datum a by $1. \cdot a$, and using the above formula for error in multiplication, we can take into account effects of rounding and underflow errors in the original data.

$||A||$ and $||b||$ denote the norms of the matrix A and vector b. $||A||_\infty$ denotes the infinity norm of A, and similarly for $||b||_\infty$. $k(A) = ||A||_\infty ||A^{-1}||_\infty$ denotes the condition number of the matrix A.

$|A|$ ($|b|$) denotes the matrix (vector) whose entries are the absolute values of the corresponding entries in A (b). Inequalities like $|A| > |B|$ ($|a| > |b|$) are meant componentwise.

3.2. Approach

As stated in the introduction, we use backward error analysis. Thus, when trying to solve

$$Ax = b \tag{12}$$

for x we obtain an approximation $\hat{x} = x + \delta x$ which satisfies the perturbed problem

$$(A + \delta A)\,\hat{x} = b + \delta b \ . \tag{13}$$

The task of backwards error analysis is to obtain bounds on δA and δb. These bounds can be used in turn to bound the residual r:

$$r = A\hat{x} - b = -\delta A\hat{x} + \delta b \tag{14}$$

and the error δx.

Wilkinson's approach [Wilkinson,1963] is to obtain a bound ω on the errors

$$\| \delta A \|_\infty \leq \omega \| A \|_\infty \quad \text{and} \quad \| \delta b \|_\infty \leq \omega \| b \|_\infty \tag{15}$$

and then show

$$\| r \|_\infty \leq \omega \left[\| A \|_\infty \| \hat{x} \|_\infty + \| b \|_\infty \right] \tag{16}$$

and so

$$\frac{\| \delta x \|_\infty}{\| x \|_\infty + \| \hat{x} \|_\infty} \leq \omega\, k(A) \ . \tag{17}$$

Skeel's approach [Skeel,1979], modified slightly here, is to obtain bounds on the relative error ω in each entry of A and b:

$$|\delta A| \leq \omega |A| \quad \text{and} \quad |\delta b| \leq \omega |b| \tag{18}$$

by first obtaining an inequality (see [Skeel,1979])

$$\| r \|_\infty \leq \omega' \| \, |A| \, |\hat{x}| + |b| \, \|_\infty \tag{19}$$

from which he shows

$$\omega = \frac{\max_i (|A| \, |\hat{x}| + |b|)_i}{\min_i (|A| \, |\hat{x}| + |b|)_i} \cdot \omega' \tag{20}$$

(where the min in the denominator is over the nonzero values of $(|A| \, |\hat{x}|)_i$ only). From this Skeel obtains

$$\frac{\| \delta x \|_\infty}{\| x \|_\infty} \leq \omega \cdot \frac{\| \, |A^{-1}| \, |A| \, |x| + |A^{-1}| \, |b| \, \|_\infty}{(1 - \omega \| \, |A^{-1}| \, |A| \, \|_\infty) \| x \|_\infty} \ . \tag{21}$$

We proceed by breaking the computation into three steps:

1) Decomposition : Try to decompose A into LU and obtain L and U where $A = LU - E$; bound E in terms of A.
2) Forward Substitution : Try to solve $Ly = b$ and obtain y where $(L + \delta L)y = b + \Delta b$; bound δL in terms of L and Δb in terms of b.
3) Backward Substitution : Try to solve $Ux = y$ and obtain $\hat{x}$ where $(U + \delta U)\hat{x} = y + \delta y$; bound δU in terms of U and δy in terms of y.

Thus, combining the above three steps we obtain

$$(A + E + L\,\delta U + \delta L U + \delta L\,\delta U)\,\hat{x} = b + \Delta b + (L + \delta L)\delta y \tag{22}$$

or

$$(A + \delta A)\,\hat{x} = b + \delta b \ . \tag{23}$$

The residual $r = A\hat{x} - b$ is

$$r = -\delta A\hat{x} + \delta b = -(E + L\delta U + \delta L(U + \delta U))\hat{x} + \Delta b + (L + \delta L)\delta y \qquad (24)$$
$$= -\delta A\hat{x} + \delta b \ .$$

Equation (24) shows that

$$\delta A = E + L\delta U + \delta L(U + \delta U) \qquad (25)$$

and

$$\delta b = \Delta b + (L + \delta L)\delta y \ . \qquad (26)$$

In the style of Wilkinson we obtain the bound ω in equation (15) by bounding

$$\frac{||E||_\infty}{||A||_\infty} \quad \text{and} \quad \frac{||L\delta U + \delta L(U + \delta U)||_\infty}{||A||_\infty} \ .$$

We also obtain bounds on

$$\frac{||\Delta b||_\infty}{||b||_\infty} \quad \text{and} \quad \frac{||(L + \delta L)\delta y||_\infty}{||b||_\infty}$$

by showing them to be no larger than $||\delta A||_\infty / ||A||_\infty$ if certain scaling conditions on A and b are met. These inequalities are derived in the next section. We conclude with Theorem 1, which summarizes the results by giving explicit and simple requirements on A and b for the bound on the error contributed by underflow (using either G.U. or S.Z.) to be scarcely worse than the bound on the error from roundoff alone.

In the style of Ṣkeel, we obtain the bound ω' in equation (19) by bounding

$$\frac{||E\hat{x}||_\infty}{||\ |A|\ |\hat{x}|\ ||_\infty} \quad \text{and} \quad \frac{||(L\delta U + \delta L(U + \delta U)\hat{x}||_\infty}{||\ |A|\ |\hat{x}|\ ||_\infty} \ .$$

We also bound

$$\frac{||\Delta b||_\infty}{||b||_\infty} \quad \text{and} \quad \frac{||(L + \delta L)\delta y||_\infty}{||b||_\infty}$$

by showing them to be no larger than $||\delta A\hat{x}||_\infty / ||\ |A|\ |\hat{x}|\ ||_\infty$ if certain scaling conditions on A and b are met. These conditions are nearly identical to the conditions derived in the Wilkinson style analysis above. These inequalities are derived in the next section. We conclude with Theorem 2 which summarizes the results as in the Wilkinson style analysis.

3.3. Error Analysis Details

We first state two propositions which analyze the error on computing inner products.

Proposition 1.

Let g' be a bound on the partial sums and individual terms of the inner product $\sum_{i=1}^{n} a_i b_i$:

$$g' = \max_{1 \le i \le n} (float(\sum_{j=1}^{i} a_j b_j),\ a_i b_i) \ . \qquad (27)$$

We can bound the error in computing $\sum_{i=1}^{n} a_i b_i$ as follows:

In the absence of underflow we have

$$|\ fl(\sum_{i=1}^{n} a_i b_i) - \sum_{i=1}^{n} a_i b_i\ | \le (2n-1)\epsilon g \qquad (28)$$

where $g = g'/(1-\epsilon)$.

In the case of G.U. we have

$$|fl(\sum_{i=1}^{n} a_i b_i) - \sum_{i=1}^{n} a_i b_i| \leq (n-1)\epsilon g + n\epsilon\max(\lambda, g) \tag{29}$$

$$\leq (2n-1)\epsilon g \quad \text{if} \quad g \geq \lambda$$

where $g = g'/(1-\epsilon)$.

In the case of S.Z. we have

$$|fl(\sum_{i=1}^{n} a_i b_i) - \sum_{i=1}^{n} a_i b_i| \leq (2n-1)\epsilon\max(\frac{\lambda}{\epsilon}, g) \tag{30}$$

$$\leq (2n-1)\epsilon g \quad \text{if} \quad g > \frac{\lambda}{\epsilon}$$

where $g = (g' + \lambda)/(1-\epsilon)$.

Note that the g used in equation (29) may differ from the g used in equation (30) because g depends on the kind of arithmetic used (G.U. or S.Z.). Also, g depends on the order of the terms $a_i b_i$.

Proof: Let $S_m = fl(\sum_{i=1}^{m} a_i b_i)$. Then

$$S_1 = a_1 b_1 + \epsilon_1 a_1 b_1 + \eta_1 = a_1 b_1 + E_1$$

$$\begin{aligned} S_m &= (S_{m-1} + a_m b_m(1+\epsilon_m) + \eta_m)(1 + e_{m-1}) + z_{m-1} \\ &= S_{m-1} + a_m b_m + \epsilon_m a_m b_m + e_{m-1}[(S_m - z_{m-1})/(1 + \epsilon_{m-1})] + z_{m-1} + \eta_m \\ &= S_{m-1} + a_m b_m + E_m \ . \end{aligned}$$

In the absence of underflow

$$|E_m| \leq 2\epsilon g \quad \text{and} \quad |E_1| \leq \epsilon g \ .$$

In the case of G.U., $z_{m-1} \equiv 0$ and $\epsilon_m \eta_m = 0$ so

$$|E_m| \leq \epsilon g + \epsilon\max(g, \lambda) \quad \text{and} \quad |E_1| \leq \epsilon\max(g, \lambda) \ .$$

In the case of S.Z. we have only $\epsilon_m \eta_m = 0$, so

$$|E_m| \leq 2\epsilon\max(g, \frac{\lambda}{\epsilon}) \quad \text{and} \quad |E_1| \leq \epsilon\max(g, \frac{\lambda}{\epsilon}) \ .$$

Now $|S_n - \sum_{i=1}^{n} a_i b_i| \leq \sum_{i=1}^{n} |E_i|$, and so the result follows by adding the bounds for $|E_i|$ above.

Q.E.D.

Proposition 2.

To analyze solving a triangular system, we need another expression for the error in computing an inner product:

$$float(\sum_{i=1}^{n} a_i b_i) = \sum_{i=1}^{n} a_i b_i(1+E_i) + \eta \ . \tag{31}$$

In the absence of underflow we have

$$\begin{aligned} &|E_1| \leq n\epsilon \\ &|E_j| \leq (n+2-j)\epsilon \quad \text{if} \quad j>1 \ . \end{aligned} \tag{32}$$

In the case of G.U. we have the same bounds on the $|E_i|$, and

$$|\eta| \leq n\lambda\epsilon \quad . \tag{33}$$

In the case of S.Z. we have the same bounds on the $|E_i|$, and

$$|\eta| \leq n\lambda \quad . \tag{34}$$

Proof. Letting

$$\begin{aligned} S_m &= fl(\sum_{i=1}^{m} a_i b_i) \\ &= fl(S_{m-1} + a_m b_m) \\ &= (S_{m-1} + a_m b_m(1+e_m) + \eta_m)(1+w_{m-1}) + z_{m-1} \end{aligned}$$

it is easy to prove by induction on m that

$$\begin{aligned} S_m = {} & a_1 b_1(1+e_1)\prod_{i=1}^{m-1}(1+w_i) + \eta_1\prod_{i=1}^{m-1}(1+w_i) + \\ & \sum_{k=2}^{m} a_k b_k(1+e_k)\prod_{i=k-1}^{m-1}(1+w_i) + \sum_{k=2}^{m}\eta_k\prod_{i=k-1}^{m-1}(1+w_i) + \\ & \sum_{k=1}^{m-1} z_k \prod_{i=k+1}^{m-1}(1+w_i). \end{aligned}$$

$z_{m-1} = 0$ always for G.U. In the case of S.Z., if $\eta_m \neq 0$, so that $fl(a_m b_m)$ has underflowed to 0, $S_m = S_{m-1}$, so $z_{m-1} = 0$. Thus, for either G.U. or S.Z., $z_{m-1}\eta_m = 0$, so the contribution from underflow is at most n times the bound for η_m and z_{m-1}. Approximating $(1+e_k)\prod_{i=}^{n-1}(1+w_i)$ by $1+(n-r+1)\epsilon$ we get the result. Note that the $\sum_{i=1}^{n} a_i b_i(1+E_i)$ term is the same term as would have been given by round-off alone. Q.E.D.

We now consider the decomposition of A.

Lemma 1: Error Analysis for Decomposition.

We assume

$|L_{ij}| \leq 1$ and $L_{ii} = 1$ because of scaling and pivoting.

The matrix has been permuted so row (or column) exchanges are not needed (to simplify notation).

Inner products are accumulated with roundoff error ϵ_s and smallest number μ_s, and other operations are done with roundoff error $\epsilon \geq \epsilon_s$ and smallest number $\mu \geq \mu_s$. We also assume $\lambda \geq \lambda_s$, i.e. that the underflow threshold for inner products is less than or equal to the underflow threshold for other operations.

Let $\hat{g}_{ij}$ be the largest absolute intermediate value encountered while computing L_{ij} (for $i>j$) and U_{ij} (for $i \leq j$). Note that $\hat{g}_{ij}$ depends on the type of arithmetic used, as indicated by superscripts G.U., S.Z., and no superscript in the absence of underflow. (The program used is in Appendix 1.)

Then in the absence of underflow

$$\text{if } i>j \quad |E_{ij}| \leq (2j-1)\epsilon_s\hat{g}_{ij} + \epsilon|U_{jj}L_{ij}| \tag{35a}$$

$$\text{if } i \leq j \quad |E_{ij}| \leq ((2i-1)\epsilon_s + \epsilon)\hat{g}_{ij} \quad . \tag{35b}$$

If G.U. is used

$$\text{if } i>j \quad |E_{ij}| \leq (j-1)\epsilon_s\hat{g}_{ij}^{GU} + j\epsilon_s\max(\lambda_s, \hat{g}_{ij}^{GU}) + \epsilon|U_{jj}|\max(|L_{ij}|, \lambda) \tag{36a}$$

$$\text{if } i \le j \quad |E_{ij}| \le (i-1)\epsilon_s \hat{g}_{ij}^{GU} + i\epsilon_s \max(\lambda_s, \hat{g}_{ij}^{GU}) + \epsilon \max(\lambda, \hat{g}_{ij}^{GU}) \ . \tag{36b}$$

If S.Z. is used

$$\text{if } i > j \quad |E_{ij}| \le (2j-1)\epsilon_s \max(\frac{\lambda_s}{\epsilon_s}, \hat{g}_{ij}^{SZ}) + \epsilon |U_{jj}| \max(|L_{ij}|, \frac{\lambda}{\epsilon}) \ . \tag{37a}$$

$$\text{if } i \le j \quad |E_{ij}| \le (2i-1)\epsilon_s \max(\frac{\lambda_s}{\epsilon_s}, \hat{g}_{ij}^{SZ}) + \epsilon \max(\frac{\lambda}{\epsilon}, \hat{g}_{ij}^{SZ}) \ . \tag{37b}$$

If G.U. is used and $\hat{g}_{ij}^{GU} \ge \lambda_s$ the bounds become more similar to the bounds without underflow:

$$\text{if } i > j \quad |E_{ij}| \le (2j-1)\epsilon_s \hat{g}_{ij}^{GU} + \epsilon |U_{jj}| \max(|L_{ij}|, \lambda) \tag{38a}$$

$$\text{if } i \le j \quad |E_{ij}| \le ((2i-1)\epsilon_s + \epsilon)\hat{g}_{ij}^{GU} \ . \tag{38b}$$

If S.Z. is used and $\hat{g}_{ij}^{SZ} \ge \lambda_s/\epsilon_s$, then the bounds also become more similar to the bounds without underflow:

$$\text{if } i > j \quad |E_{ij}| \le (2j-1)\epsilon_s \hat{g}_{ij}^{SZ} + \epsilon |U_{jj}| \max(|L_{ij}|, \frac{\lambda}{\epsilon}) \tag{39a}$$

$$\text{if } i \le j \quad |E_{ij}| \le ((2i-1)\epsilon_s + \epsilon)\hat{g}_{ij}^{SZ} \ . \tag{39b}$$

Proof: The proof is an application of Proposition 1 to the program in Appendix 1. There are two cases, $i > j$, and $i \le j$.

Case $i > j$: We compute L_{ij} by the following formula (see Appendix 1):

$$L_{ij} = float((A_{ij} - \sum_{k=1}^{j-1} L_{ik} U_{kj})/U_{jj}) \ . \tag{40}$$

Let $\hat{g}_{ij}$ be the largest intermediate value computed:

$$\hat{g}_{ij} = \max_{1 \le j' \le j} (float(A_{ij} - \sum_{k=1}^{j'-1} L_{ik} U_{kj}), float(L_{ij'} U_{j'j})) \ , \tag{41}$$

where $\hat{g}_{ij}$ depends on the arithmetic. We write equation (40) as

$$L_{ij} = (A_{ij} - \sum_{k=1}^{j-1} L_{ik} U_{kj} + E'_{ij})/U_{jj}(1+e) + \eta' \tag{42}$$

and apply Proposition 1 to bound E'_{ij} as follows:

$$|E'_{ij}| \le (2j-1)\epsilon_s \hat{g}_{ij} \tag{43}$$

in the absence of underflow,

$$\begin{aligned} |E'_{ij}| &\le (j-1)\epsilon_s \hat{g}_{ij}^{GU} + j\epsilon_s \max(\lambda_s, \hat{g}_{ij}^{GU}) \\ &\le (2j-1)\epsilon_s \hat{g}_{ij}^{GU} \quad \text{if} \quad \hat{g}_{ij}^{GU} \ge \lambda_s \end{aligned} \tag{44}$$

in the case of G.U., and

$$\begin{aligned} |E'_{ij}| &\le (2j-1)\epsilon_s \max(\frac{\lambda_s}{\epsilon_s}, \hat{g}_{ij}^{SZ}) \\ &\le (2j-1)\epsilon_s \hat{g}_{ij}^{SZ} \quad \text{if} \quad \hat{g}_{ij}^{SZ} \ge \frac{\lambda_s}{\epsilon_s} \end{aligned} \tag{45}$$

in the case of S.Z.

So

$$A_{ij} = \sum_{k=1}^{j} L_{ik} U_{kj} - E'_{ij} + eL_{ij} U_{jj} - \eta' U_{jj}(1+e) \tag{46}$$

$$= \sum_{k=1}^{i} L_{ik} U_{kj} + E_{ij}$$

where

$$|E_{ij}| \leq |E'_{ij}| + \max(|L_{ij} U_{jj} e|, |\eta U_{jj}|) \qquad (47)$$
$$\leq |E'_{ij}| + \epsilon |U_{jj}| \max(|L_{ij}|, |\eta|/\epsilon) \ .$$

$|\eta| = |\eta'(1+e)|$ is essentially bounded by $\lambda\epsilon$ for G.U. and λ for S.Z.

Thus, in the absence of underflow,

$$|E_{ij}| \leq (2j-1)\epsilon_s \hat{g}_{ij} + \epsilon |U_{jj} L_{ij}| \ . \qquad (48)$$

When using G.U.

$$|E_{ij}| \leq (j-1)\epsilon_s \hat{g}_{ij}^{GU} + \epsilon_s j \max(\lambda_s, \hat{g}_{ij}^{GU}) + \epsilon |U_{jj}| \max(|L_{ij}|, \lambda) \qquad (49)$$
$$\leq (2j-1)\epsilon_s \hat{g}_{ij}^{GU} + \epsilon |U_{jj}| \max(|L_{ij}|, \lambda) \quad \text{if} \quad \hat{g}_{ij}^{GU} \geq \lambda_s \ .$$

When using S.Z. we get

$$|E_{ij}| \leq (2j-1)\epsilon_s \max(\frac{\lambda_s}{\epsilon_s}, \hat{g}_{ij}^{SZ}) + \epsilon |U_{jj}| \max(|L_{ij}|, \frac{\lambda}{\epsilon}) \qquad (50)$$

$$\leq (2j-1)\epsilon_s \hat{g}_{ij}^{SZ} + \epsilon |U_{jj}| \max(|L_{ij}|, \frac{\lambda}{\epsilon}) \quad \text{if} \quad \hat{g}_{ij}^{SZ} \geq \frac{\lambda_s}{\epsilon_s} \ .$$

Case $i \leq j$: We compute U_{ij} using the following formula (see Appendix 1):

$$U_{ij} = float(A_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj}) \ . \qquad (51)$$

As above, we let $\hat{g}_{ij}$ be the magnitude of the largest intermediate value:

$$\hat{g}_{ij} = \max_{1 \leq i' \leq i} (float(A_{ij} - \sum_{k=1}^{i'-1} L_{ik} U_{kj}), float(L_{ii'} U_{i'j})) \ . \qquad (52)$$

There are two subcases, depending on whether $\epsilon_s = \epsilon$ or $\epsilon_s < \epsilon$. We deal with the $\epsilon_s = \epsilon$ case first. We rewrite equation (51) as

$$A_{ij} = \sum_{k=1}^{i} L_{ik} U_{kj} + E'_{ij} \qquad (53)$$

and apply Proposition 1 to bound $|E'_{ij}|$.

In the absence of underflow

$$|E'_{ij}| \leq (2i-1)\epsilon_s \hat{g}_{ij} \ . \qquad (54)$$

When using G.U.

$$|E'_{ij}| \leq (i-1)\epsilon_s \hat{g}_{ij}^{GU} + i\epsilon_s \max(\lambda_s, \hat{g}_{ij}^{GU}) \qquad (55)$$
$$\leq (2i-1)\epsilon_s \hat{g}_{ij}^{GU} \quad \text{if} \quad \hat{g}_{ij}^{GU} \geq \lambda_s \ .$$

When using S.Z.

$$|E'_{ij}| \leq (2i-1)\epsilon_s \max(\frac{\lambda_s}{\epsilon_s}, \hat{g}_{ij}^{SZ}) \qquad (56)$$

$$\leq (2i-1)\epsilon_s \hat{g}_{ij}^{SZ} \quad \text{if} \quad \hat{g}_{ij}^{SZ} \geq \frac{\lambda_s}{\epsilon_s} \ .$$

When $\epsilon_s < \epsilon$, we make one more rounding error when storing U_{jj}. Thus, equation (51) becomes

$$U_{ij} = float[1 \times [A_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj}]] \qquad (57)$$

which we rewrite as

$$U_{ij} = \frac{1}{1+e}(A_{ij} - \sum_{k=1}^{i-1} L_{ik}U_{kj} + E'_{ij}) + \eta' \tag{58}$$

or

$$\begin{aligned} A_{ij} &= \sum_{k=1}^{i} L_{ik}U_{kj} - E'_{ij} + eU_{ij} - \eta \\ &= \sum_{k=1}^{i} L_{ij}U_{kj} + E_{ij} \quad . \end{aligned} \tag{59}$$

where $\eta = \eta'(1+e)$. $|E'_{ij}|$ has the same bound as above, so in the absence of underflow, we get

$$\begin{aligned} |E_{ij}| &\le |E'_{ij}| + \epsilon\hat{g}_{ij} \\ &\le ((2i-1)\epsilon_s + \epsilon)\hat{g}_{ij} \quad . \end{aligned} \tag{60}$$

When using G.U. we get

$$\begin{aligned} |E_{ij}| &\le |E'_{ij}| + \epsilon\max(\lambda,\hat{g}_{ij}^{GU}) \\ &\le (i-1)\epsilon_s\hat{g}_{ij}^{GU} + i\epsilon_s\max(\lambda_s,\hat{g}_{ij}^{GU}) + \epsilon\max(\lambda,\hat{g}_{ij}^{GU}) \\ &\le ((2i-1)\epsilon_s + \epsilon)\hat{g}_{ij}^{GU} \quad \text{if} \quad \hat{g}_{ij}^{GU} \ge \lambda \quad . \end{aligned} \tag{61}$$

When using S.Z., we get

$$\begin{aligned} |E_{ij}| &\le |E'_{ij}| + \epsilon\max(\frac{\lambda}{\epsilon},\hat{g}_{ij}^{SZ}) \\ &\le (2i-1)\epsilon_s\max(\frac{\lambda_s}{\epsilon_s},\hat{g}_{ij}^{SZ}) + \epsilon\max(\frac{\lambda}{\epsilon},\hat{g}_{ij}^{SZ}) \\ &\le [(2i-1)\epsilon_s + \epsilon]\hat{g}_{ij}^{SZ} \quad \text{if} \quad \hat{g}_{ij}^{SZ} \ge \frac{\lambda}{\epsilon} \quad . \end{aligned} \tag{62}$$

This completes the proof of Lemma 1. Q.E.D.

To see what this lemma means, we compare the bounds in the absence of underflow (which come from roundoff contributions only) to the bounds in the presence of underflow.

Part of the bounds to compare is a multiple of $\epsilon_s\hat{g}_{ij}$ in the absence of underflow, $\epsilon_s\max(\lambda_s,\hat{g}_{ij})$ using G.U., and $\epsilon_s\max(\lambda_s/\epsilon_s,\hat{g}_{ij})$ using S.Z. Comparisons involve $\hat{g}_{ij}$s with identical superscripts, which we omit for simplicity. If $\hat{g}_{ij} \ge \lambda_s$, these bounds using G.U. are the same as the bounds without underflow. The condition $\hat{g}_{ij} \ge \lambda_s$ is trivially satisfied as long as the original datum (or some intermediate result) contains a nonzero normalized number. The analogous condition $\hat{g}_{ij} \ge \lambda_s/\epsilon_s$ for S.Z., on the other hand, requires a large danger region above the underflow threshold λ_s where data may not lie.

Similarly, when $i \ge j$ we must compare the bounds $\epsilon|U_{jj}L_{ij}|$ without underflow, $\epsilon|U_{jj}|\max(|L_{ij}|,\lambda)$ using G.U., and $\epsilon|U_{jj}|\max(|L_{ij}|,\lambda/\epsilon)$ using S.Z. Again, the bound for G.U. is identical to the bound for roundoff alone as long as $|L_{ij}| \ge \lambda$, but $|L_{ij}| \ge \lambda/\epsilon$ is needed for S.Z. to be the same as roundoff.

In other words, with G.U., as long as the original datum itself is normalized, we expect an error no worse than we would have expected with roundoff alone. In contrast, with S.Z. the original datum (or some intermediate result) must be larger than the underflow threshold λ_s or λ by a factor of $1/\epsilon_s$ or $1/\epsilon$ (2^{24} on a machine with 24 bit precision) for the same bound as roundoff alone to apply.

It is possible, however, even with G.U. for $\hat{g}_{ij}$ to be $\le\lambda_s$, or $|L_{ij}| \le \lambda_s$ to occur, with apparently disastrous changes to the solution. For example, consider the matrix

$$A = \begin{bmatrix} B & B \\ b & 2b \end{bmatrix} \tag{63}$$

where both B and b are normalized numbers, larger than λ/ϵ if desired, but so that b/B underflows to zero even using G.U. Then the factors L and U obtained are the exact factors of the utterly different looking matrix

$$A' = \begin{bmatrix} B & B \\ 0 & 2b \end{bmatrix} = A + E = A + \begin{bmatrix} & \\ -b & \end{bmatrix} . \tag{64}$$

The solution of a system $A'x = b$ can be utterly different from solution of $Ax = b$, but the important point is that the residual will remain small, in the sense that $||r||_\infty / || \, |A| \, |x| \, ||_\infty$ is small:

$$\frac{||r||_\infty}{|| \, |A| \, |x| \, ||_\infty} = \frac{||Ex||_\infty}{|| \, |A| \, |x| \, ||_\infty} = \frac{|bx_1|}{B|x_1| + B|x_2|} \leq \frac{b}{B} \leq \lambda\epsilon \tag{65}$$

for any computed solution x.

More precisely, we can show

Lemma 2: Wilkinson style error analysis of decomposition in the presence of underflow

Let $a_{max} = \max_{i,j} |A_{ij}|$, and $g = \max_{i,j} \hat{g}_{ij}/a_{max} \geq 1$. g is Wilkinson's pivot growth factor, and depends on the arithmetic. We choose $\epsilon_s = \epsilon$ for this lemma; an analogous lemma holds for $\epsilon_s < \epsilon$.

Then in the absence of underflow,

$$||E||_\infty \leq n^2 \epsilon g a_{max} + \text{smaller terms} \tag{66a}$$

or

$$\frac{||E||_\infty}{||A||_\infty} \leq n^2 \epsilon g \ . \tag{66b}$$

Using G.U.,

$$||E||_\infty \leq \frac{n^2}{2}(\epsilon g^{GU} a_{max} + \epsilon \max(\lambda, g^{GU} a_{max})) \tag{67a}$$

$$\leq n^2 \epsilon g^{GU} a_{max} \text{ if } g^{GU} a_{max} \geq \lambda ,$$

or

$$\frac{||E||_\infty}{||A||_\infty} \leq n^2 \epsilon g^{GU} \text{ if } g^{GU} a_{max} \geq \lambda \ . \tag{67b}$$

Using S.Z.

$$||E||_\infty \leq n^2 \epsilon \max(g^{SZ} a_{max}, \frac{\lambda}{\epsilon}) \tag{68a}$$

$$\leq n^2 \epsilon \max(g^{SZ} a_{max}, \frac{\lambda}{\epsilon}) \text{ if } g^{SZ} a_{max} \geq \frac{\lambda}{\epsilon} ,$$

or

$$\frac{||E||_\infty}{||A||_\infty} \leq n^2 g^{SZ} a_{max} \text{ if } g^{SZ} a_{max} \geq \frac{\lambda}{\epsilon} \ . \tag{68b}$$

Proof: We replace $\hat{g}_{ij}$ by $g a_{max}$ in the bounds of Lemma 1 and sum. In the absence of underflow we get

$$\text{if } i > j \quad |E_{ij}| \leq 2j \epsilon g a_{max} \tag{69a}$$

$$\text{if} \quad i \geq j \quad |E_{ij}| \leq 2i\epsilon g a_{\max} \ . \tag{69b}$$

Then

$$\begin{aligned} ||E||_\infty &= \max_i \sum_{j=1}^{n} |E_{ij}| \\ &\leq \sum_{j=1}^{n} 2j\epsilon g a_{\max} \\ &= (n^2 + n)\epsilon g a_{\max} \ . \end{aligned} \tag{70}$$

Since $||A||_\infty \geq a_{\max}$,

$$\frac{||E||_\infty}{||A||_\infty} \leq (n^2 + n)\epsilon g \ . \tag{71}$$

If G.U. is used,

$$\begin{aligned} \text{if} \quad i > j \quad |E_{ij}| &\leq j\epsilon g^{GU} a_{\max} + j\epsilon \max(\lambda, g^{GU} a_{\max}) \\ &\leq 2j\epsilon g^{GU} a_{\max} \quad \text{if} \quad g^{GU} a_{\max} \geq \lambda \end{aligned} \tag{72a}$$

$$\begin{aligned} \text{if} \quad i \leq j \quad |E_{ij}| &\leq (i-1)\epsilon g^{GU} a_{\max} + (i+1)\epsilon \max(\lambda, g^{GU} a_{\max}) \\ &\leq 2i\epsilon g^{GU} a_{\max} \quad \text{if} \quad g^{GU} a_{\max} \geq \lambda \ . \end{aligned} \tag{72b}$$

Then

$$\begin{aligned} ||E||_\infty &\leq \sum_{j=1}^{n} [j\epsilon g^{GU} a_{\max} + j\epsilon \max(\lambda, g^{GU} a_{\max}) \\ &= \frac{n^2 + n}{2} (\epsilon g^{GU} a_{\max} + \epsilon \max(\lambda, g^{GU} a_{\max})) \\ &= (n^2 + n)\epsilon g^{GU} a_{\max} \quad \text{if} \quad g^{GU} a_{\max} \geq \lambda \ . \end{aligned} \tag{73}$$

If S.Z. is used,

$$\begin{aligned} \text{if} \quad i > j \quad |E_{ij}| &\leq 2j\epsilon \max(\frac{\lambda}{\epsilon}, g^{SZ} a_{\max}) \\ &\leq 2j\epsilon g^{SZ} a_{\max} \quad \text{if} \quad g^{SZ} a_{\max} \geq \frac{\lambda}{\epsilon} \end{aligned} \tag{74a}$$

$$\begin{aligned} \text{if} \quad i \leq j \quad |E_{ij}| &\leq 2j\epsilon \max(\frac{\lambda}{\epsilon}, g^{SZ} a_{\max}) \\ &\leq 2j\epsilon g^{SZ} a_{\max} \quad \text{if} \quad g^{SZ} a_{\max} \geq \frac{\lambda}{\epsilon} \ . \end{aligned} \tag{74b}$$

Then

$$\begin{aligned} ||E||_\infty &\leq \sum_{j=1}^{n} 2j\epsilon \max(\frac{\lambda}{\epsilon}, g^{SZ} a_{\max}) \\ &= (n^2 + n)\epsilon \max(\frac{\lambda}{\epsilon}, g^{SZ} a_{\max}) \\ &= (n^2 + n)\epsilon g^{SZ} a_{\max} \quad \text{if} \quad g^{SZ} a_{\max} \geq \frac{\lambda}{\epsilon} \ . \end{aligned} \tag{75}$$

This completes the proof of Lemma 2. Q.E.D.

The condition $g^{GU} a_{max} \geq \lambda$ for the bound on $||E||_\infty / ||A||_\infty$ to be the same for G.U. as for no underflow is trivially satisfied as long as one datum in the original matrix is normalized. The constraint $g^{SZ} a_{max} \geq \lambda/\epsilon$ for S.Z. is much stronger, and when it does not hold results can be disastrous, as seen in the examples of section 2.

Similarly, we can prove

Lemma 3: Skeel style error analysis of decomposition in the presence of underflow

Let $a_j = \max_i |A_{ij}|$, $g_j = \max_i \hat{g}_{ij}/a_j$, and $g_C = \max_j g_j$. g_j is a pivot growth factor for column j only. As usual, $\hat{g}_{ij}$, g_j, and g_C depend on the arithmetic. We choose $\epsilon_s = \epsilon$ for this lemma; an analogous one holds for $\epsilon_s < \epsilon$.

Then in the absence of underflow

$$||\hat{Ex}||_\infty \leq 2n^2 \epsilon g_C \, || \, |A| \, |\hat{x}| \, ||_\infty \, . \tag{76}$$

To get the same inequality for G.U. we need $g_C^{GU} a_j \geq \lambda$, and for S.Z. we need $g_C^{SZ} a_j \geq \lambda/\epsilon$.

We need the following proposition for the proof of Lemma 3.

Proposition 3.

Let $P \geq 0$ be an n by n matrix and $z \geq 0$ be an n vector. Denote the column norms of P by $p_j = \max_i p_{ij}$. Then

$$||Pz||_\infty \leq \sum_j p_j z_j \leq n ||Pz||_\infty \, . \tag{77}$$

Proof:

$$||Pz||_\infty = \max_i \sum_j p_{ij} z_j \leq \sum_j \max_i p_{ij} z_j = \sum_j p_j z_j \, .$$

This proves the first inequality, and shows it is sharp if, for example, P has all rows zero but one. Let $p_{j_0} z_{j_0} = \max_j p_j z_j$, so $\sum_j p_j z_j \leq n p_{j_0} z_{j_0}$. Let $p_{i_0 j_0} = p_{j_0}$. Then

$$\begin{aligned} \sum_j p_j z_j &\leq n p_{j_0} z_{j_0} = n p_{i_0 j_0} z_{j_0} \\ &\leq n \sum_j p_{i_0 j} z_j \leq n \cdot \max_i \sum_j p_{ij} z_j \\ &\leq n ||Pz||_\infty \, . \end{aligned}$$

This inequality is sharp if, for example, $P = I$, the identity matrix, and z has all entries equal. Q.E.D.

Proof of Lemma 3: We replace $\hat{g}_{ij}$ by $g_C a_j$ in the bounds of Lemma 1, so in the absence of underflow we get

$$\text{if } i > j \quad |E_{ij}| \leq 2j \epsilon g_C a_j \tag{78a}$$

$$\text{if } i \leq j \quad |E_{ij}| \leq 2i \epsilon g_C a_j \, . \tag{78b}$$

For these bounds to hold with G.U. we need $g_C^{GU} a_j \geq \lambda$, and for S.Z. we need $g_C^{SZ} a_j \geq \lambda/\epsilon$. Then

$$||\hat{Ex}||_\infty \leq || \, |E| \, |\hat{x}| \, ||_\infty \tag{79}$$

$$\begin{aligned} &= \max_i \Big(\sum_{j=1}^{i-1} 2j \epsilon g_C a_j |\hat{x}_j| + \sum_{j=i}^{n} 2i \epsilon g_C a_j |\hat{x}_j| \Big) \\ &\leq \sum_{j=1}^{n} 2j \epsilon g_C a_j |\hat{x}_j| \\ &\leq 2n \epsilon g_C \sum_{j=1}^{n} a_j |\hat{x}_j| \end{aligned}$$

$$\leq 2n^2\epsilon g_C || \ |A| \ |\hat{x}| \ ||_\infty$$

by Proposition 3. Q.E.D.

The condition $g_C^{GU} a_j \geq \lambda$ is trivially satisfied as long as one normalized number appears in each column of A. $g_C^{SZ} a_j \geq \lambda/\epsilon$ is a much stronger condition.

We now turn to the error in the solution of the two triangular systems $Ly=b$ and $Ux=y$.

Lemma 4: Error Analysis of Solving a Triangular System

Consider the lower triangular system $Ly=b$ where L does not necessarily have ones on the diagonal. (The analyses when only ones are on the diagonal or L is upper triangular are similar.) Then in solving $Ly=b$ using the program in Appendix 2 (where inner products are accumulated with roundoff error ϵ_s and smallest number μ_s) we obtain $\hat{y} = y + \delta y$ which actually satisfies $(L + \delta L)\hat{y} = b + \delta b$, where

$$|\delta L_{ij}| \leq |L_{ij}| \cdot \begin{cases} \epsilon & \text{if } i=j=1 \\ (i-1)\epsilon_s & \text{if } i>j=1 \\ (i+1-j)\epsilon_s & \text{if } i>j>1 \\ \epsilon + \epsilon_s & \text{if } i=j>1 \end{cases} \tag{80}$$

independent of underflow.

In the absence of underflow

$$\delta b = 0 \ . \tag{81}$$

Using G.U.

$$\begin{array}{lll} \text{if } i=1 & |\delta b_i| \leq \lambda\epsilon |L_{11}| \ UN(y_1) & \text{if } y_1 \text{ underflows} \\ \text{if } i>1 & |\delta b_i| \leq i\cdot\lambda_s\epsilon_s + & \text{intermediate underflows} \\ & \quad \lambda\epsilon |L_{ii}| \ UN(y_i) & \text{if } y_i \text{ underflows} \end{array} \ . \tag{82}$$

Using S.Z.

$$\begin{array}{lll} \text{if } i=1 & |\delta b_i| \leq \lambda |L_{11}| \ UN(y_1) & \text{if } y_1 \text{ underflows} \\ \text{if } i>1 & |\delta b_i| \leq i\cdot\lambda_s + & \text{intermediate underflows} \\ & \quad \lambda |L_{ii}| \ UN(y_i) & \text{if } y_i \text{ underflows} \end{array} \ . \tag{83}$$

If L_{ii} is known to be 1, the only changes in the above bounds (besides substituting 1 for L_{ii}) are that $E_{11} = \delta b_1 = 0$ independent of underflow.

Proof: We apply Proposition 2 to the program in Appendix 2. We have two cases.

Case $i=1$: In this case,

$$y_1 = float(b_1/L_{11}) = \frac{b_1}{L_{11}(1 + E_{11})} + \nu_1 \tag{84}$$

so

$$L_{11}(1 + E_{11})y_1 = b_1 + \nu_1 L_{11}(1 + E_{11})UN(y_i) \tag{85}$$

where

$$|E_{11}| \leq \epsilon \quad \text{and} \quad |\nu_1| \leq \lambda \text{ or } \lambda/\epsilon \tag{86}$$

Case $i>1$: Now

$$y_i = float((b_i - \sum_{j=1}^{i-1} L_{ij}y_j)/L_{ii}) \tag{87}$$

$$= |(1+ e_i)(b_i - \sum_{j=1}^{i-1} L_{ij}(1 + E_{ij})y_j - \rho_i) + \sigma_i|/(L_{ii}(1 + \varsigma_i)) + \nu_i$$

where

$$|E_{ij}| \leq \begin{cases} (i-1)\epsilon_s & \text{if } j=1 \\ (i+1-j)\epsilon_s & \text{if } j>1 \end{cases} \tag{88a}$$

$$|e_i| \leq \epsilon_s \tag{88b}$$

$$|\varsigma_i| \leq \epsilon \tag{88c}$$

and

$$|\rho_i| \leq \begin{cases} (i-1)\lambda_s\epsilon_s & \text{using G.U.} \\ (i-1)\lambda_s & \text{using S.Z.} \end{cases} \tag{89a}$$

$$|\sigma_i| \leq \begin{cases} \lambda_s\epsilon_s & \text{using G.U.} \\ \lambda_s & \text{using S.Z.} \end{cases} \tag{89b}$$

$$|\nu_i| \leq \begin{cases} \lambda\epsilon & \text{using G.U.} \\ \lambda & \text{using S.Z.} \end{cases} \tag{89c}$$

In the absence of underflow, $\rho_i = \sigma_i = \nu_i = 0$. Rearranging equation (87), we obtain

$$L_{ii}(1 + \varsigma_i)y_i = (1 + e_i)(b_i - \sum_{j=1}^{i-1} L_{ij}(1 + E_{ij})y_j - \rho_i) + \sigma_i + \nu_i L_{ii}(1 + \varsigma_i) \tag{90}$$

or

$$b_i = \sum_{j=1}^{i-1} L_{ij}(1 + E_{ij})y_i + L_{ii}\left(\frac{1 + \varsigma_i}{1 + e_i}\right)y_i + \rho_i - \frac{\sigma_i}{1 + e_i} - \nu_i L_{ii}\left(\frac{1 + \varsigma_i}{1 + e_i}\right) . \tag{91}$$

Using the bounds in (88) and (89) in (91), we obtain the bounds in the statement of the lemma. Q.E.D.

Since the bound on δL is independent of the presence of underflow, the following two lemmas are true no matter how underflow is handled (we assume $\epsilon = \epsilon_s$ and $\mu = \mu_s$ for simplicity).

Lemma 5: Wilkinson style error analysis of forward and back substitutions

If g is the pivot growth factor of Lemma 2, then

$$|| L\delta U + \delta L(U + \delta U) ||_\infty \leq \frac{n^3}{2}\epsilon g a_{max} \quad + \text{ smaller terms.} \tag{92}$$

Proof:

$$\begin{aligned} || L\delta U + \delta L(U + \delta U)||_\infty &\leq || L\delta U||_\infty + || \delta L(U + \delta U)||_\infty . \\ &\leq || \,|L|\,|\delta U|\,||_\infty + || \,|\delta L|\,|U + \delta U|\,||_\infty . \end{aligned} \tag{93}$$

Bound each entry of L below the diagonal by 1, and bound each entry of δL using equation (80) as though L had all ones below the diagonal. Bound each entry of U (and $U + \delta U$) above the diagonal by ga_{max}, and bound δU using equation (80) as though U had all its entries equal to ga_{max} above the diagonal. A simple computation shows

$$|| \,|L|\,|\delta U|\,||_\infty \leq \frac{n^3+ 3n^2+ 2n}{6}\cdot\epsilon g a_{max} , \tag{94}$$

and

$$|| \,|\delta L|\,|U + \delta U|\,||_\infty \leq \frac{2n^3+ 3n^2-5n+ 6}{6}\cdot\epsilon g a_{max} . \tag{95}$$

Q.E.D.

Lemma 6: Skeel style error analysis of forward and back substitutions

If g_C is the columnwise pivot growth factor of Lemma 3, then

$$|| \, |L\,\delta U + \delta L(U + \delta U)|\hat{x} \, ||_\infty \leq n^3 \epsilon g_C \, || \, |A| \, |\hat{x}| \, ||_\infty \quad + \text{smaller terms.} \tag{96}$$

Proof:

$$|| \, |L\,\delta U + \delta L(U + \delta U)|\hat{x}||_\infty \leq || \, |L| \, |\delta U| \, |\hat{x}| \, ||_\infty + || \, |\delta L| \, |U + \delta U| \, |\hat{x}| \, ||_\infty \tag{97}$$

As in Lemma 5, we bound $|L_{ij}|$ by 1, but now we bound $|U_{ij}|$ by $g_C a_j$, where $a_j = \max_i a_{ij}$. Letting Ω denote the matrix all of whose entries are 1, we have

$$\begin{aligned} || \, |L| \, |\delta U| \, |\hat{x}| \, ||_\infty &\leq g_C \frac{n^2-n+2}{2} \cdot || \, \Omega \, \text{diag}(a_1, \ldots, a_n) \, |\hat{x}| \, ||_\infty \\ &\leq g_C \frac{n^3-n^2+2n}{2} \cdot || \, |A| \, |\hat{x}| \, ||_\infty \end{aligned} \tag{98}$$

by Proposition 3. Similarly

$$\begin{aligned} || \, |\delta L| \, |U + \delta U| \, |\hat{x}| \, ||_\infty &\leq g_C \frac{n^2+n}{2} \cdot || \, \Omega \, \text{diag}(a_1, \ldots, a_n) \, |\hat{x}| \, ||_\infty \\ &\leq g_C \frac{n^3+n^2}{2} \cdot || \, |A| \, |\hat{x}| \, ||_\infty \end{aligned} \tag{99}$$

by Proposition 3. Q.E.D.

Now we must bound $\delta b = \Delta b + (L + \delta L)\delta y$. From Lemma 4 and the fact that $|L_{ij}| \leq 1$ we see

$$\begin{aligned} || \, \delta b \, ||_\infty \leq \quad & \eta n^2/2 && \text{all underflows from } Ly = b \text{ and} \\ & && \text{intermediate underflows from } Ux = y \\ & + \eta \sum_{j=1}^{n} UN(\hat{x}_j) \, |U_{jj}| && \text{underflows of result } \hat{x} \\ & + \text{ smaller terms} \end{aligned} \tag{100}$$

where $\eta = \lambda\epsilon$ with G.U., and $\eta = \lambda$ with S.Z.

We already know from Lemma 5 that the constant ω bounding $|| \, \delta A \, ||_\infty / || \, A \, ||_\infty$ in equation (15) is at least $\frac{1}{2} n^3 \epsilon g$. If we require that $|| \, \delta b \, ||_\infty / || \, b \, ||_\infty$ be no larger than this bound $\frac{1}{2} n^3 \epsilon g$, i.e. that the bound on the error contributed by underflow be no worse than the bound on the error contributed by roundoff, we get

Lemma 7: Wilkinson style requirements that underflows while solving triangular systems contribute error with a bound no worse than roundoff.

When using G.U. we require

$$\begin{aligned} || \, b \, ||_\infty &\geq \frac{1}{n} \lambda && \text{if there are intermediate underflows} \\ \frac{|| \, b \, ||_\infty}{a_{max}} &\geq \frac{2}{n^2} \lambda && \text{if the solution } \hat{x} \text{ underflows.} \end{aligned} \tag{101}$$

When using S.Z. we require

$$\begin{aligned} || \, b \, ||_\infty &\geq \frac{1}{n} \frac{\lambda}{\epsilon} && \text{if there are intermediate underflows} \\ \frac{|| \, b \, ||_\infty}{a_{max}} &\geq \frac{2}{n^2} \frac{\lambda}{\epsilon} && \text{if the solution } \hat{x} \text{ underflows.} \end{aligned} \tag{102}$$

Proof: If there are intermediate underflows, require that

$$\frac{\frac{n^2}{2}\eta}{\| b \|_\infty} \leq \frac{1}{2} n^3 \epsilon g \tag{103}$$

where we replace g by its lower bound 1. If the solution $\hat{x}$ underflows, require that

$$\frac{\sum_{j=1}^{n} | U_{jj} | \eta}{\| b \|_\infty} \leq \frac{1}{2} n^3 \epsilon g \quad . \tag{104}$$

where we replace $\sum_{j=1}^{n} | U_{jj} |$ by its upper bound $n g a_{max}$. In both equations above, substitute $\eta = \lambda\epsilon$ for G.U. and $\eta = \lambda$ for S.Z. Q.E.D.

We now turn to the Skeel style error analysis. From Lemma 6 we know the constant ω^j in equation (19) is at least $n^3 \epsilon g_C$. If we require that

$$\frac{\| \delta b \|_\infty}{\| \, |A| \, |\hat{x}| + |b| \, \|_\infty} \tag{105}$$

be no larger than this bound $n^3 \epsilon g_C$, i.e. that the bound to the error contributed by underflow be no worse than the bound to the error contributed by roundoff, we get

Lemma 8: Skeel style requirements that underflow while solving triangular systems contribute an error with a bound no worse than roundoff

When using G.U., we require

$$\begin{aligned} \| b \|_\infty &\geq \frac{1}{2n} \lambda && \text{if there are intermediate underflows} \\ \frac{\| b \|_\infty}{a_{max}} &\geq \frac{1}{n^2} \lambda && \text{if the solution } \hat{x} \text{ underflows.} \end{aligned} \tag{106}$$

When using S.Z. we require

$$\begin{aligned} \| b \|_\infty &\geq \frac{1}{2n} \frac{\lambda}{\epsilon} && \text{if there are intermediate underflows} \\ \frac{\| b \|_\infty}{a_{max}} &\geq \frac{1}{n^2} \frac{\lambda}{\epsilon} && \text{if the solution } \hat{x} \text{ underflows.} \end{aligned} \tag{107}$$

Proof: If there are intermediate underflows, require

$$\frac{\frac{n^2}{2}\eta}{\| b \|_\infty} \leq n^3 \epsilon g_C \tag{108}$$

where we replace g_C by its lower bound 1. If the solution $\hat{x}$ itself underflows, require

$$\frac{\eta \sum_{j=1}^{n} | U_{jj} |}{\| b \|_\infty} \leq n^3 \epsilon g_C \tag{109}$$

and replace $\sum_{j=1}^{n} | U_{jj} |$ by its upper bound $n g_C a_{max}$. Let $\eta = \lambda\epsilon$ for G.U. and $\eta = \lambda$ for S.Z. Q.E.D.

3.4. Results

Thus, combining Lemmas 2, 5, and 7 we finally obtain

Theorem 1: Wilkinson style error analysis solving $Ax=b$ in the presence of underflow

Let $a_{max}=\max_{ij}|A_{ij}|$, and $g=$[largest number appearing in decomposition]$/a_{max}$. g is the pivot growth factor of Lemma 2.

Then in the absence of underflow

$$||r||_\infty \leq \frac{n^3}{2}\epsilon g[||A||_\infty||\hat{x}||_\infty + ||b||_\infty] \tag{110}$$

When using G.U., as long as

$$\begin{aligned} ga_{max} &\geq \lambda && \text{if there are any underflows during triangular decomposition} \\ ||b||_\infty &\geq \frac{1}{n}\lambda && \text{if there are any intermediate underflows during forward and back substitutions} \\ \frac{||b||_\infty}{a_{max}} &\geq \frac{2}{n^2}\lambda && \text{if the solution } \hat{x} \text{ itself underflows} \end{aligned} \tag{111}$$

then we have

$$||r||_\infty \leq \frac{3}{2}n^3\epsilon g[||A||_\infty||\hat{x}||_\infty + ||b||_\infty] \ . \tag{112}$$

When using S.Z., the above requirements apply with λ replaced by λ/ϵ.

Proof: Combine Lemmas 2, 5, and 7. Q.E.D.

Combining Lemmas 3, 6, and 8, we obtain

Theorem 2: Skeel style error analysis of solving $Ax=b$ in the presence of underflow

Let $a_j=\max_i|A_{ij}|$, and $g_C=\max_j($[largest number appearing the the decomposition in column j]$/a_j)$. g_C is the columnwise pivot growth factor of Lemma 3.

Then in the absence of underflow

$$||r||_\infty \leq n^3\epsilon g_C||\ |A|\ |\hat{x}| + |b|\ ||_\infty + \text{smaller terms.} \tag{113}$$

When using G.U., as long as

$$\begin{aligned} g_C a_j &\geq \lambda && \text{for all } j\text{, if there are any underflows during triangular decomposition} \\ ||b||_\infty &\geq \frac{1}{2n}\lambda && \text{if there are any intermediate underflows during forward and back substitutions} \\ \frac{||b||_\infty}{a_{max}} &\geq \frac{1}{n^2}\lambda && \text{if the solution } \hat{x} \text{ itself underflows} \end{aligned} \tag{114}$$

we have

$$||r||_\infty \leq 3n^3\epsilon g_C||\ |A|\ |\hat{x}| + |b|\ ||_\infty \ . \tag{115}$$

When using S.Z., the above requirements apply with λ replaced by λ/ϵ.

Proof: Combine Lemmas 3, 6, and 9. Q.E.D.

4. Conclusions

First we apply Theorem 1 to show how to write reliable software for solving $Ax=b$ from the point of view of a Wilkinson style error analysis. We compare the requirements of Theorem 1 for G.U. and S.Z. For G.U. as long as one normalized number appears during the decomposition ($ga_{max} \geq \lambda$), error with underflow has a bound identical to error without underflow. If there are intermediate underflows while solving the triangular systems, as long as some component of b is normalized ($|| b ||_\infty \geq \lambda$), error with underflow has a bound at worst twice the bound without underflow. If the answer x itself underflows, we can either issue an error message (which would be very reasonable since the first goal of reliable software is only to compute an answer if it is representable) or test to see if $|| b ||_\infty / a_{max}$ is not too small.

All these requirements are natural ones to make, since they say that when a problem's inputs are normalized and its computed solution is normalized, we should expect the solution to be scarcely worse with underflow than without. Thus, the only underflows which can cause concern in a problem with normalized inputs are underflows in the solution itself.

In contrast, the bounds for S.Z. are all higher by a factor of $1/\epsilon$. Thus, using S.Z. we can neither solve as many problems as with G.U., nor decide so easily which underflows matter. Thus, from the point of view of a Wilkinson style error analysis, G.U. makes writing reliable software easier.

From Theorem 2 we see the requirements for the bounds on error with underflow to be the same as the bounds without in the Skeel type error analysis differ only by a factor of 1/2 from the Wilkinson style bounds, and so the comments in the above paragraphs still apply. The only difference for G.U. is that a normalized number must appear in the course of computing each column, not just somewhere in the whole matrix.

We wish to emphasize that we have only derived conditions under which error *bounds* with underflow are about the same as without underflow. There is no way using this analysis to say how closely this bound will be approached with and without underflow. Even when the requirements of Theorems 1 and 2 are satisfied, G.U. can sometimes get a better result than S.Z., as in example 3 of section 2. All the columns and rows of the matrix in that example have entries much larger than the underflow threshold, but the matrix appears so ill-conditioned that the last pivot is approximately equal to the underflow threshold, and so can only be computed accurately by G.U. and not S.Z. The backward errors in computing the solution using either G.U. or S.Z. are both extremely small, but the condition number (Skeel's or Wilkinson's) is so large, that one would not want to trust the answer anyway. From the point of view of this paper, it is an "accident" that G.U. got the right answer and S.Z. did not, but we conjecture that G.U. will always get a "better" answer than S.Z., for any reasonable definition of "better".

Appendix 1: Program Listing of Triangular Decomposition

```
array A[1..N,1..N] of real;
array L[1..N,1..N] of real;
array U[1..N,1..N] of real;
/* note that arrays A,L, and U can occupy the same locations in memory;
we use three different arrays here to keep the notation consistent with
the previous analysis */

/* Triangular Factorization: A=LU */
/* As in the analysis, we assume the matrix has been permuted so row or column
exchanges are unnecessary. We indicate the precisions of all computations
except stores into entries of A, L, and U, which are always with
precision (ε,μ). */

for p=1 to N do /* p=pivot number */
      begin
      for j=p to N do  /* compute U[p,j] */
            begin
            SUM=A[p,j];
            for k=1 to p-1 do SUM=SUM - L[p,k]*U[k,j];  /* precision (ε_s,μ_s) */
            U[p,j]=SUM;
            end
      /* test for singularity/divide by zero omitted */
      for i=p+ 1 to N do /* compute L[i,p] */
            begin
            SUM=A[i,p];
            for k=1 to p-1 do SUM=SUM - L[i,k]*U[k,p];  /* precision (ε_s,μ_s) */
            L[i,p]=SUM/U[p,p];  /* precision (ε,μ) */
            end
      end
```

Appendix 2: Program Listing of Solution of Both Triangular Systems Following Triangular Decomposition

```
array B[1..N] of real;
array X[1..N] of real;
array Y[1..N] of real;
/* Again, B,X, and Y could occupy the same locations in memory. Arrays
A,L and U are defined above. As in Appendix 1, precisions of all computations
except entries of B, X, and Y are shown. */

/* Forward Substitution: LY=B */

for i=1 to N do
      begin
      SUM=0;
      for k=1 to i-1 do SUM=SUM + L[i,k]*Y[k]; /* precision (ε_s,μ_s) */
      Y[i]=B[i]-SUM;  /* precision (ε_s,μ_s) */
      end

/* Back Substitution: UX=Y */

for i=N downto 1 do
      begin
      SUM=0;
      for k=i+ 1 to N do SUM=SUM + U[i,k]*X[k]; /* precision (ε_s,μ_s) */
      X[i]=(Y[i] - SUM)/U[i,i]; /* precision (ε_s,μ_s) */
      end
```

Appendix 3: Analysis of Cholesky Decomposition

Introduction

Cholesky is an algorithm for solving positive definite symmetric systems of linear equations $Ax=b$ using Cholesky decomposition ($A=L^TL$ where L is lower triangular) followed by forward ($L^Ty=b$) and back ($Lx=y$) substitutions. Our analysis and conclusions are analogous to those in the body of the report: G.U. makes Cholesky more reliable than S.Z. We will need the following notation in addition to that used already: $||\ Anrm2$ and $||\ bnrm2$ denote the 2-norms of the matrix A and vector b, and $k_2(A)$ the spectral condition number, the condition number with respect to the 2-norm. $\lambda_{max}(A)$ and $\lambda_{min}(A)$ denote the largest and smallest eigenvalues of the matrix A, respectively.

We also assume $\lambda<\epsilon^2$ for the analysis of the Cholesky decomposition in S.Z.

Proposition 4: Cauchy-Schwartz Inequality in Floating Point

In the case of G.U. we have

$$|fl(\sum_{i=1}^{n} a_i b_i)| \le (1+2n\epsilon)\sqrt{fl(\sum_{i=1}^{n} a_i^2) + n\lambda\epsilon}\sqrt{fl(\sum_{i=1}^{n} b_i^2) + n\lambda\epsilon + n\lambda\epsilon}\ .$$

If in addition $fl(\sum_{i=1}^{n} a_i^2) \ge \lambda$ and $fl(\sum_{i=1}^{n} b_i^2) \ge \lambda$, we have

$$|fl(\sum_{i=1}^{n} a_i b_i)| \le (1+4n\epsilon)\sqrt{fl(\sum_{i=1}^{n} a_i^2)}\sqrt{fl(\sum_{i=1}^{n} b_i^2)}\ .$$

In the case of S.Z. we have

$$|fl(\sum_{i=1}^{n} a_i b_i)| \le (1+2n\epsilon)\sqrt{fl(\sum_{i=1}^{n} a_i^2) + n\lambda}\sqrt{fl(\sum_{i=1}^{n} b_i^2) + n\lambda}\ .$$

If we know $fl(\sum_{i=1}^{n} a_i^2) \ge \lambda$ and $fl(\sum_{i=1}^{n} b_i^2) \ge \lambda$ we can only assert

$$|fl(\sum_{i=1}^{n} a_i b_i)| \le (n+1)\sqrt{fl(\sum_{i=1}^{n} a_i^2)}\sqrt{fl(\sum_{i=1}^{n} b_i^2)}$$

but if we know $fl(\sum_{i=1}^{n} a_i^2) \ge \lambda^2$ and $fl(\sum_{i=1}^{n} b_i^2) \ge \lambda^2$ we can write

$$|fl(\sum_{i=1}^{n} a_i b_i)| \le (1+3n\epsilon)\sqrt{fl(\sum_{i=1}^{n} a_i^2)}\sqrt{fl(\sum_{i=1}^{n} b_i^2)}\ .$$

Proof: Since

$$|fl(\sum_{i=1}^{n} a_i b_i)| \le fl(\sum_{i=1}^{n} |a_i|\ |b_i|)$$

we can assume without loss of generality that $a_i \ge 0$ and $b_i \ge 0$.

In the case of G.U. we may use Proposition 2 and the usual Cauchy-Schwartz inequality to write

$$\begin{aligned} |fl(\sum_{i=1}^{n} a_i b_i)| &= \sum_{i=1}^{n} a_i b_i(1+E_i) + \eta \\ &\le \sum_{i=1}^{n} a_i b_i(1+n\epsilon) + n\lambda\epsilon \\ &\le (1+n\epsilon)\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}\ . \end{aligned}$$

Also

$$fl(\sum_{i=1}^{n} a_i^2) = \sum_{i=1}^{n} a_i^2(1+E_i) + \eta$$

$$\geq \sum_{i=1}^{n} a_i^2(1-n\epsilon) - n\lambda\epsilon$$

so

$$\sum_{i=1}^{n} a_i^2 \leq (1+n\epsilon)(fl(\sum_{i=1}^{n} a_i^2) + n\lambda\epsilon)$$

with an analogous result for $\sum_{i=1}^{n} b_i^2$. Then

$$|fl(\sum_{i=1}^{n} a_i b_i)| \leq (1+2n\epsilon)\sqrt{fl(\sum_{i=1}^{n} a_i^2) + n\lambda\epsilon}\sqrt{fl(\sum_{i=1}^{n} b_i^2) + n\lambda\epsilon} \ .$$

The case for S.Z. is somewhat different. In equation (10) we note $\eta\sum m \leq 0$ because if $\eta_m \neq 0$, $c_m = 0$ so $0 = fl(a_m b_m) = a_m b_m + \eta_m$. Also $z_m = 0$ for all m since the sum is increasing. Thus the total contribution η from underflow in (11) is ≤ 0, so

$$|fl(\sum_{i=1}^{n} a_i b_i)| = \sum_{i=1}^{n} a_i b_i(1+E_i) + \eta$$

$$\leq \sum_{i=1}^{n} a_i b_i(1+n\epsilon) \ .$$

Now

$$fl(\sum_{i=1}^{n} a_i^2) = \sum_{i=1}^{n} a_i^2(1-n\epsilon) - n\lambda$$

so

$$\sum_{i=1}^{n} a_i^2 \leq (1+n\epsilon)(fl(\sum_{i=1}^{n} a_i^2) + n\lambda)$$

and

$$|fl(\sum_{i=1}^{n} a_i b_i)| \leq (1+2n\epsilon)\sqrt{fl(\sum_{i=1}^{n} a_i^2) + n\lambda}\sqrt{fl(\sum_{i=1}^{n} b_i^2) + n\lambda}$$

as desired. Q.E.D.

Error Analysis of Cholesky Decomposition

We assume A is symmetric and positive definite. Writing $A = LL^T + E$, we can bound the error as follows:

In the case of G.U., we let $g = a_{max}$ if $a_{max} \geq \lambda$, and $g = (n+2)a_{max}$ if we can only say $a_{max} \geq \lambda\epsilon$. Then

$$|E| \leq \frac{n^2}{2}(\epsilon g + \max(\lambda\epsilon, \epsilon a_{max})) + \textit{lower order terms}$$

$$\leq n^2\epsilon a_{max} \quad \textit{if} \quad a_{max} > \lambda$$

so

$$\frac{|E|}{|A|} \leq \frac{n^2}{2}(\epsilon + \max(\frac{\lambda\epsilon}{a_{max}}, \epsilon)) + \textit{lower order terms} \qquad (4.1)$$

$$\leq n^2\epsilon \quad \textit{if} \quad a_{max} > \lambda \quad .$$

In the case of S.Z.

$$|E| \leq n^2\max(\epsilon a_{max}, \lambda) + \quad \textit{lower order terms}$$
$$\leq n^2\epsilon a_{max} \quad \textit{if} \quad a_{max} > \frac{\lambda}{\epsilon}$$

so

$$\frac{|E|}{|A|} \leq n^2\max(\epsilon, \frac{1}{a_{max}}\cdot\lambda) + \quad \textit{lower order terms} \qquad (4.2)$$
$$\leq n^2\epsilon \quad \textit{if} \quad a_{max} > \frac{\lambda}{\epsilon} \quad .$$

As with Gaussian elimination, as long as a_{max} is a non-zero normalized number, the error from underflow is no worse than the bound for round off alone, if G.U. is used. The threshold for a_{max} using S.Z. is again higher by a factor of $1/\epsilon$. A listing of the program being analyzed is in Appendix 4.

Proof: As with Gaussian elimination, we have two cases, depending on i and j. We use the first case, $i=j$, to provide a bound g in order to apply Proposition 1 to case 2, $i>j$.

Case 1: $i=j$. We have

$$L_{jj} = fl((A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2)^{1/2})$$
$$= (A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2 + E)^{1/2}/(1+e)$$

so

$$L_{jj}^2(1+e)^2 = A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2 + E$$

or

$$A_{jj} = \sum_{k=1}^{j} L_{jk}^2 - E + 2e_1L_{jj}^2 = \sum_{k=1}^{j} L_{jk}^2 + E_{jj} \quad .$$

We want to apply Proposition 1 to get a bound on $|E|$. We claim the bound g required by Proposition 1 is A_{jj}, because if any term or floating partial sum of the increasing sequence $\sum_{k=1}^{r} L_{jk}^2$ exceeds A_{jj}, the algorithm will terminate because the matrix is (algorithmically) not positive definite.

In the case of G.U. we have

$$|E_{jj}| \leq |E| + |2e_1L_{jj}^2|$$
$$\leq (j-1)\epsilon A_{jj} + j\max(\epsilon A_{jj}, \lambda\epsilon) + 2\epsilon A_{jj}$$
$$\leq (j+1)\epsilon A_{jj} + j\max(\epsilon A_{jj}, \lambda\epsilon)$$
$$\leq (2j+1)\epsilon A_{jj} \quad \textit{if} \quad A_{jj} > \lambda$$
$$\leq (2j+1)\epsilon a_{max} \quad \textit{if} \quad a_{max} > \lambda \quad .$$

In the case of S.Z. we have

$$|E_{jj}| \leq |E| + |2e_1L_{jj}^2|$$
$$\leq (2j-1)\max(\epsilon A_{jj}, \lambda) + 2\epsilon A_{jj}$$

$$\leq (2j+1)\epsilon A_{jj} \quad \text{if} \quad A_{jj} > \frac{\lambda}{\epsilon}$$

$$\leq (2j+1)\epsilon a_{max} \quad \text{if} \quad a_{max} > \frac{\lambda}{\epsilon} \quad .$$

Case 2: $i>j$. We may use Proposition 4 and the result of case $i=j$ to write

$$|fl(\sum_{k=1}^{r} L_{ik}L_{jk})| \leq (1+2r\epsilon)\sqrt{|fl(\sum_{k=1}^{r} L_{ik}^2) + \xi|}\sqrt{|fl(\sum_{k=1}^{r} L_{jk}^2) + \xi|} + \delta$$

$$\leq (1+2i\epsilon)\sqrt{|fl(\sum_{k=1}^{i} L_{ik}^2) + \xi|}\sqrt{|fl(\sum_{k=1}^{j} L_{jk}^2) + \xi|} + \delta$$

$$\leq (1+2i\epsilon)\sqrt{|A_{ii} + \xi|}\sqrt{|A_{jj} + \xi|} + \delta$$

$$\equiv B_{ij}^{GU} \text{ in the case of G.U., and } B_{ij}^{SZ} \text{ in the case of S.Z.}$$

where ξ and δ are bounded as in Proposition 4.
If $A_{ii} > \lambda$ and $A_{jj} > \lambda$ in the case of G.U., or $A_{ii} > \lambda/\epsilon$ and $A_{jj} > \lambda/\epsilon$ in the case of S.Z., then we get

$$B_{ij} \leq (1+4i\epsilon)\sqrt{A_{ii}A_{jj}} \leq (1+4n\epsilon)a_{max} \quad .$$

If $A_{ii} > \lambda$ and $A_{jj} > \lambda$, we can only guarantee

$$B_{ij}^{SZ} \leq (n+1)\sqrt{A_{ii}A_{jj}} \leq (n+1)\, a_{max} \quad .$$

The g needed to apply Proposition 1 will be the sum of B_{ij} and a_{max}.
We may now write

$$L_{ij} = fl((A_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{jk})/L_{jj})$$

$$= (A_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{jk} + E)/L_{jj}(1+e) + \eta \quad .$$

where the η appears only if L_{ij} underflows. Rearranging

$$A_{ij} = \sum_{k=1}^{j} L_{ik}L_{jk} - E + eL_{jj}L_{ij} - \eta L_{jj}(1+e)$$

$$= \sum_{k=1}^{j} L_{ik}L_{jk} + E_{ij} \quad .$$

In the case of G.U.

$$|E_{ij}| \leq |E| + \max(|eL_{ij}L_{jj}|, |\eta L_{jj}(1+e)|)$$

$$\leq (j-1)\epsilon g + j\max(\lambda\epsilon, \epsilon g) + \max(\epsilon g, \lambda\epsilon\sqrt{a_{max}})$$

$$\leq j\epsilon g + j\max(\lambda\epsilon, \epsilon g)$$

$$\leq 2j\epsilon a_{max} \quad \text{if} \quad a_{max} > \lambda \quad .$$

We used the fact that $\lambda/\epsilon < 1$ so $\epsilon g < \lambda\epsilon\sqrt{a_{max}}$ only if $a_{max} < \lambda^2 < \lambda\epsilon$, implying $a_{max} = 0$.
In the case of S.Z.

$$|E_{ij}| \leq |E| + \max(|eL_{ij}L_{jj}|, |\lambda\epsilon L_{jj}(1+e)|)$$

$$\leq (2j-1)\max(\epsilon g, \lambda) + \max(\epsilon g, \lambda\sqrt{a_{max}})$$

$$\leq (2j-1)\max(\epsilon g, \lambda\epsilon) + \epsilon g$$

$$\leq 2j\epsilon a_{max} \quad \text{if} \quad a_{max} > \frac{\lambda}{\epsilon} \quad .$$

We used the fact that $\lambda/\epsilon^2 < 1$ so $\epsilon g < \lambda\sqrt{a_{max}}$ only if $a_{max} < (\lambda/\epsilon)^2 < \lambda$, implying $a_{max} = 0$. Adding these bounds for $|E_{i,j}|$ we get the results stated above. Q.E.D.

Numerical Examples

Example 1

Let m^2 be the smallest number $\geq \lambda$ which is the perfect square of the floating point number m, where we assume $m^2 < 2\lambda$. For example, $m^2 = 2^{-126} = \lambda$ in the case of KCS single precision [Coonen, 1979], and 2^{-128} for the PDP-11, VAX, and Payne-Strecker proposal [Payne and Strecker, 1979].

First consider the well-conditioned positive definite symmetric matrix

$$A_1 = m^2 \cdot \begin{bmatrix} 4 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

which has the exact lower triangular factor

$$L_1 = m \cdot \begin{bmatrix} 2 & & \\ 1 & 1 & \\ .5 & .5 & \sqrt{.5} \end{bmatrix} .$$

L_1^{GU}, the factor provided by Cholesky using G.U., is the same as L_1 except for the rounding error incurred by having to represent $\sqrt{.5}$. L_1^{SZ}, the factor provided by S.Z., is

$$L_1^{SZ} = m \cdot \begin{bmatrix} 2 & & \\ 1 & 1 & \\ .5 & 1 & 0 \end{bmatrix}$$

so S.Z. decides that matrix is singular. Note no rounding error occurred using S.Z.

As a second example, consider the well conditioned matrix

$$A_2 = m^2 \cdot \begin{bmatrix} 4 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} .$$

Its exact factor L_1 is

$$L_2 = m \cdot \begin{bmatrix} 2 & & \\ 1 & 1 & \\ .5 & .5 & \sqrt{1.5} \end{bmatrix} .$$

The factor L_2^{GU} provided by G.U. is the same as L_2 except for the rounding required to represent $\sqrt{1.5}$. The factor L_2^{SZ} provided by S.Z., however, is

$$L_2^{SZ} = m \cdot \begin{bmatrix} 2 & & \\ 1 & 1 & \\ .5 & 1 & 1 \end{bmatrix}$$

and

$$(L_2^{SZ})(L_2^{SZ})^T = m^2 \cdot \begin{bmatrix} 4 & 2 & 1 \\ 2 & 2 & 1.5 \\ 1 & 1.5 & 2.25 \end{bmatrix} = A_2 + E_2$$

with a relative error $||E_2||_\infty / ||A_2||_\infty > .1$. Thus S.Z. provides a reasonable looking answer which is totally wrong. Note that no rounding error occurred using S.Z.

Example 2

Let m^2 be as before. Consider the family of matrices

$$A(x) = m^2 \cdot \begin{bmatrix} 4 & & & & 1 \\ & 4 & & & 1 \\ & & 4 & & 1 \\ & & & 4 & 1 \\ 1 & 1 & 1 & 1 & x \end{bmatrix} .$$

Its correct factor $L(x)$, if it exists, is

$$L(x) = m \cdot \begin{bmatrix} 2 & & & & \\ & 2 & & & \\ & & 2 & & \\ & & & 2 & \\ .5 & .5 & .5 & .5 & \sqrt{x-2} \end{bmatrix} .$$

This matrix is positive definite if $x>2$, positive semidefinite if $x=2$, and has both positive and negative eigenvalues is $x<2$. Both G.U. and S.Z. compute all entries of the factor $L(x)$ except the (5,5) entry correctly (using Cholesky). G.U. obtains the correct value $(x-2)m^2$ for its value of L_{55}^2, whereas S.Z. compute xm^2. Thus, as x decreases from 3 to 2 to 1, G.U. correctly decides the matrix is positive definite when $x=3$, and becomes non-positive definite when $x \leq 2$. S.Z., on the other hand, produces an (incorrect) decomposition all the way down to $x=1$. Thus, S.Z. can not only produce an inaccurate decomposition, but produces it after G.U. has correctly decided no such decomposition exists.

S.Z. can produce a decomposition of a matrix when G.U. fails only if the matrix is either 1) so ill-conditioned that the decomposition cannot be trusted, or 2) not positive definite at all. Here is the reason. Assume $a_{max} \geq \lambda$, since otherwise the matrix is identically 0 in S.Z. G.U. fails when its computed value of L_{jj}^2 either rounds to 0 or is negative for some j. L_{jj}^2 rounds to 0 when $L_{jj}^2 < \lambda\epsilon$. It is easy to see that $a_{max} \leq \lambda_{max}(A)$ and $L_{jj}^2 \geq \lambda_{min}(A)$, because

$$\frac{1}{\min_j L_{jj}^2} = (\lambda_{max}(L^{-1}))^2 \leq |L^{-1}|_2^2 = |A^{-1}|_2 = \frac{1}{\lambda_{min}(A)} .$$

Therefore

$$k_2(A) = \frac{\lambda_{max}}{\lambda_{min}} > \frac{a_{max}}{L_{jj}^2} > \frac{1}{\epsilon} ,$$

which means that the matrix is so ill conditioned as to make it difficult to even recognize an accurate inverse, let alone compute one. If L_{jj}^2 is in fact negative, the matrix is not positive definite.

Error Analysis of Solving Both Triangular Systems Following Cholesky Decomposition

We have

$$(L + dL)(U + dU)x = b + db + (L + dL)dy = b + \Delta b .$$

In the case of G.U. we get

$$|db_i| \leq \lambda\epsilon(i + L_{ii} UN(y_i)) \tag{4.3}$$

$$\leq \lambda\epsilon(i + \sqrt{a_{max}})$$

$$|dy_i| \leq \lambda\epsilon(n-i+1 + L_{ii} UN(z_i)) \tag{4.4}$$

$$\leq \lambda\epsilon(n-i+1+\sqrt{a_{max}})$$

$$|((L+dL)dy)_i| \leq \frac{n^2}{2}\lambda\epsilon\sqrt{a_{max}} + n\lambda\epsilon a_{max} + \textit{lower order terms} \quad . \tag{4.5}$$

The $n^2\lambda\epsilon\sqrt{a_{max}}/2$ term in (4.5) is the result of intermediate underflows in the back substitution, and if no y_i underflows, (4.5) may be replaced by

$$|((L+dL)dy)_i| \leq n\lambda\epsilon a_{max} \quad . \tag{4.6}$$

In the case of S.Z. we get

$$|db_i| \leq \lambda(i + L_{ii}UN(y_i)) \tag{4.7}$$

$$\leq \lambda(i+\sqrt{a_{max}})$$

$$|dy_i| \leq \lambda(n-i+1+L_{ii}UN(z_i)) \tag{4.8}$$

$$\leq \lambda(n-i+1+\sqrt{a_{max}})$$

$$|((L+dL)dy)_i| \leq \frac{n^2}{2}\lambda\sqrt{a_{max}} + n\lambda a_{max} + \textit{lower order terms} \quad . \tag{4.9}$$

The $n^2\lambda\epsilon\sqrt{a_{max}}/2\epsilon$ term in (4.9) is the result of intermediate underflows in the back substitution, and if all y_i terms exceed λ, (4.9) may be replaced by

$$|((L+dL)dy)_i| \leq n\lambda a_{max} \quad . \tag{4.10}$$

A listing of the program being analyzed is given in Appendix 4.

Proof: This is again a straightforward application of the result for solving one triangular system and bound $|L_{ij}| \leq \sqrt{a_{max}}$. Q.E.D.

5. Writing a Reliable Program to Solve linear Systems Using Cholesky Decomposition

As with Gaussian elimination, we translate the condition

$$\frac{|db|}{|b|} \leq kn^2g\epsilon$$

where k is approximately 2, and $g=1$ (because the matrix is symmetric and positive definite) into

$$b_{max} > \lambda \quad \begin{cases} \textit{if there are any intermediate underflows} \\ \textit{in the forward substitution} \end{cases} \tag{4.11}$$

$$\frac{b_{max}}{\sqrt{a_{max}}} > \frac{1}{n^2}\lambda \quad \textit{if some } y_i \textit{ underflows} \tag{4.12}$$

$$\frac{b_{max}}{\sqrt{a_{max}}} > \lambda \quad \begin{cases} \textit{if some } y_i \textit{ underflows and there are} \\ \textit{intermediate underflows in the back substitution} \end{cases} \tag{4.13}$$

$$\frac{b_{max}}{a_{max}} > \frac{1}{n}\lambda \quad \textit{if some } z_i \textit{ underflows} \quad . \tag{4.14}$$

in the case of G.U., and

$$b_{max} > \frac{\lambda}{\epsilon} \quad \begin{cases} \textit{if there are any intermediate underflows} \\ \textit{in the forward substitution} \end{cases} \tag{4.15}$$

$$\frac{b_{max}}{\sqrt{a_{max}}} > \frac{1}{n^2}\frac{\lambda}{\epsilon} \quad \textit{if some } y_i \textit{ underflows} \tag{4.16}$$

$$\frac{b_{max}}{\sqrt{a_{max}}} > \frac{\lambda}{\epsilon} \quad \begin{cases} \textit{if some } y_i \textit{ does not exceed } \lambda \textit{ and} \\ \textit{there are any underflows in the back substitution} \end{cases} \tag{4.17}$$

$$\frac{b_{max}}{a_{max}} > \frac{1}{n}\frac{\lambda}{\epsilon} \quad \textit{if some } x_i \textit{ underflows} \ . \tag{4.18}$$

in the case of S.Z.

These results follow from (4.3) through (4.10).

When to Raise a Warning using G.U.

From (4.1) we know that underflows cannot hurt the triangular factorization as long as a_{max} is normalized. As discussed before, this is a reasonable requirement, so no tests need to be made during decomposition.

From (4.11) we know intermediate underflows in the forward substitution cannot hurt as long as b_{max} is normalized, again a reasonable requirement.

Conditions (4.12) to (4.14) apply only if some y_i or x_i underflows. Thus one could simply raise a warning if any x_i or y_i underflowed, or if b_{max} and a_{max} were available, raise a warning only if some y_i underflows and (4.12) and (4.13) are not satisfied, or if some x_i underflows and (4.13) is not true. Computing b_{max} requires n comparisons, as does computing a_{max} (since it lies on the diagonal of A), so the cost is small but not negligible. As discussed before, raising a warning because the final result x_i is not representable is not necessarily bad.

The event that y underflows without x underflowing can be used to detect ill-conditioning, as discussed below.

If underflow is not signaled during the computation at any time, then we know that $a_{max} > \lambda$ and $b_{max} > \lambda$ with no comparison needed in our programs; this is the usual case.

A reliable program should be able to detect nonpositive-definite and very ill-conditioned matrices (which could cause a divide by zero) and warn the user. This topic will not be discussed further here.

When to Raise a Warning Using S.Z.

From (4.2) we see there is a more stringent requirement on a_{max} that with G.U., and that intermediate underflows can significantly affect the accuracy of the result. In contrast to G.U., tests must be made during the decomposition, possibly including computing a_{max} can comparing it to λ/ϵ (at the cost of n comparisons).

From conditions (4.15) and (4.17) we see intermediate underflows can contribute significantly to the error, and the conditions on b_{max}, $b_{max}/\sqrt{a_{max}}$, and b_{max}/a_{max} are more stringent than the corresponding conditions for G.U.

Thus it is significantly easier to write reliable software using G.U. than S.Z.

Appendix 4: Program Listings

Cholesky Decomposition

```
array A[1..N,1..N] of real;
array L[1..N,1..N] of real;
/* as with Gaussian elimination, A and L could occupy the same locations. */

/* Triangular Factorization: A=LL^T */

for j=1 to N do /* j=column number */
      begin
      for i=j to N do /* i=row number */
            begin
            SUM=A[i,j];
            for k=1 to j-1 do SUM=SUM - L[i,k]*L[j,k];
            if (i=j)
                  L[i,i]=sqrt(SUM);
                  /* test for nonpositive-definiteness/divide by zero omitted */
            else
                  L[i,j]=SUM/L[j,j];
            end
      end
```

Solution of Both Triangular Systems following Cholesky Decomposition

```
array B[1..N] of real;
array X[1..N] of real;
array Y[1..N] of real;
/* As before, B,X, and Y can occupy the same memory locations. Arrays
A and L were defined above. */

/* Forward Substitution: LY=B */

for i=1 to N do
      begin
      SUM=0;
      for k-1 to i-1 do SUM=SUM + L[i,k]*Y[k];
      Y[i]=(B[i] - SUM)/L[i,i]
      end

/* Back Substitution: L^T X=Y */

for i=N downto 1 do
      begin
      SUM=0;
      for k=i+ 1 to N do SUM=SUM + L[k,i]*X[k];
      X[i]=(Y[i] - SUM)/L[i,i];
      end
```

References

M. Payne, W. Strecker, "Draft Proposal for a Binary Normalized Floating Point Standard", SIGNUM Newsletter, October 1979

W. Kahan, J. Palmer, "On a Proposed Floating-Point Standard", SIGNUM Newsletter, October 1979

J.H. Wilkinson,"Rounding Errors in Algebraic Processes", Prentice Hall, 1963

J. Coonen, et al.,"A Proposed Standard for Binary Floating Point Arithmetic", SIGNUM Newsletter, October 1979

J. J. Dongarra, J. R. Bunch, C. B. Moler, G. W. Stewart, LINPACK User' Guide, Society for Industrial and Applied Mathematics, Philadelphia, 1979

J. T. Coonen, "Underflow and the Denormalized Numbers", Computer, vol. 14, no. 3, March 1981, pp 75-87

R. D. Skeel, "Scaling for Numerical Stability in Gaussian Elimination", Journal of the ACM, vol. 26, no. 3, July 1979