# Effects of Underflow on Solving Linear Systems

*James Demmel*

Computer Science Division
University of California
Berkeley, CA

*ABSTRACT*

Software to solve systems of linear equations by Gaussian elimination has in the past ignored the effects of underflow. But when underflows are replaced by zeroes, this software can give spurious though plausible results much worse than could be blamed on roundoff. When underflow is gradual, as in the proposed IEEE standard for floating point arithmetic, the same software gives provably more reliable results. To achieve the same reliability without gradual underflow, but with underflows set to zero, complicated tests must be inserted into the software.

## 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 [1] 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 [2] states in its introduction that "Any underflows which occur are harmless." This statement is almost always true, since underflow's status as a borderline phenomenon means it may be ignored safely in most situations, but not always. The advantage of gradual underflow over store zero is that a cleaner line can be drawn between problems where underflow can be ignored and those where it cannot, thus helping 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 by the IEEE Microprocessor Standards Committee on how to handle underflow. The decision to use gradual underflow instead of the usual store zero approach came after much argument about which one made reliable numerical software easier to write [1]. In this paper we show that gradual underflow makes writing reliable linear system solvers significantly easier than store zero.

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 no entries of $L$ exceed 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 residual turn out to be underflows in the final solution vector $x$. Intermediate gradual underflows will be shown to 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 second two goals 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 can introduce significant errors, possibly producing reasonable looking results whose error greatly exceeds the uncertainty attributable to roundoff alone (see the Examples).

The second and third goals of reliable software can be achieved, but not easily, with store zero. 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. To tell which small class of possible gradual underflows can contribute significant errors and require a warning is easy: they are the underflows in the answer itself.

It is important to understand how the unavoidable uncertainty due to roundoff can affect our ability to meet out first goal: produce accurate results. Guaranteeing



CH1630-3/81/0000/0113$00.75 

results $x$ to within a certain accuracy is not possible for a price everybody is willing to pay. Unless we pay the price, by doing iterative refinement for example (discussion of iterative refinement is beyond the scope of this paper), we can only guarantee that our computed answer $x$ 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 $x$ of the new problem may be very different from the solution $x$ of the original problem, and there is no way, at negligible cost in time and storage, 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 $Ax-b$." This is an achievable goal. (Discussion of iterative refinement is beyond the scope of this paper.)

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.

## 2. Examples

We use G.U. to denote arithmetic performed with gradual underflow, and S.Z. to denote that performed with underflows set to zero; we assume the reader is acquainted with both kinds of arithmetic [1,3,4]. In particular, we perform G.U. in the "normalizing mode" rather than the "warning mode". We will present our model of arithmetic briefly here, and give a more precise model of rounding and underflow errors in the next section.

Let $\varepsilon$ be the rounding error of the arithmetic, and $\lambda$ 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 $e_{min} \le e \le e_{max}$. Then $\varepsilon = 2^{-n}$ (the difference between 1 and the next larger 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 $\lambda$ is generated by arithmetic, underflow is signalled and something special happens; S.Z. replaces the number by zero, and G.U. by a nearest "denormalized" number or zero. Denormalized numbers in G.U. arithmetic consist of an arithmetic progression from 0 to $\lambda$ with common separation equal to $\mu = \varepsilon\lambda$.

We denote the usual condition number of the matrix $A$ by

$$k(A) = \|A\|_\infty \cdot \|A^{-1}\|_\infty ,$$

and a new set of condition numbers by

$$\mathrm{Cond}(A,x) = \frac{\| \, |A^{-1}| \, |A| \, |x| \, \|_\infty}{\|x\|_\infty}$$

$$\mathrm{Cond}(A) = \| \, |A^{-1}| \, |A| \, \|_\infty .$$

These new condition numbers, due to Skeel[6], will be discussed more fully in section 3 below. Note $\mathrm{Cond}(A) \ge \mathrm{Cond}(A,x)$ for all $x$.

In this section we present four examples of the effects of underflow on performing Gaussian elimination. 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 nor pivot growth in this example.) The second example shows that G.U. produces the correct decomposition of a well conditioned matrix which S.Z. incorrectly decides is singular. Third, 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. Finally, we present an example which shows that regardless of whether we use G.U. or S.Z., Gaussian elimination can only guarantee small residuals, not an accurate answer, even when the matrix $A$ is well-conditioned in the sense that $\mathrm{Cond}(A)$ is small.

### 2.1 Example 1

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

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

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

$$L^{G.U.}(x) \cdot U^{G.U.}(x) = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ .5 & .5 & .5 & .5 & 1 \end{bmatrix} \cdot \lambda \cdot \begin{bmatrix} 2 & & & & 1 \\ & 2 & & & 1 \\ & & 2 & & 1 \\ & & & 2 & 1 \\ & & & & x-2 \end{bmatrix} \qquad (2)$$

$= A(x)$ exactly, and by S.Z. is

$$L^{S.Z.}(x) \cdot U^{S.Z.}(x) = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ .5 & .5 & .5 & .5 & 1 \end{bmatrix} \cdot \lambda \cdot \begin{bmatrix} 2 & & & & 1 \\ & 2 & & & 1 \\ & & 2 & & 1 \\ & & & 2 & 1 \\ & & & & x \end{bmatrix} \qquad (3)$$

$= A(x) + E$, where the error matrix $E$ equals

$$E = \lambda \cdot \begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & -2 & \end{bmatrix} . \qquad (4)$$

We see S.Z. causes a relatively large error in the $U(x)_{55}$ entry, whereas G.U. gives the correct decomposition. When $x=2$, using S.Z. leads us to conclude that 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 2, and if $x$ is a small integer no rounding errors occur in either decomposition.

### 2.2 Example 2

Let

$$A = \begin{bmatrix} 2\lambda & 3\lambda \\ \lambda & 2\lambda \end{bmatrix} , \qquad (5)$$

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

$$L^{G.U.} \cdot U^{G.U.} = \begin{bmatrix} 1 & \\ .5 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2\lambda & 3\lambda \\ & \lambda/2 \end{bmatrix} = A , \qquad (6)$$

but by using S.Z. we obtain

$$L^{S.Z.} \cdot U^{S.Z.} = \begin{bmatrix} 1 & \\ .5 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2\lambda & 3\lambda \\ & 0 \end{bmatrix} . \qquad (7)$$

Thus, G.U. correctly decomposes the matrix $A$, whereas S.Z. incorrectly makes the matrix look singular.

### 2.3 Example 3

Consider the ordinary differential equation

$$\dot{x}(t) = \frac{1-(t/T)^M}{T-t} x(t) \ , \quad x(T_0) = c \ . \qquad (8)$$

We try to solve this equation numerically by replacing $x(t)$ by the truncated power series $\sum_{n=1}^{N} x_n t^n$, the function $(1-(t/T)^M)/(T-t)$ by its (finite) power series, and then 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

$$A = \begin{bmatrix} N & -1/T & -1/T^2 & \cdots & & -1/T^N \\ & N-1 & -1/T & \cdots & & -1/T^{N-1} \\ & & N-2 & \cdots & & -1/T^{N-2} \\ & & & \ddots & & \vdots \\ & & & & 1 & -1/T \\ 1 & 1/T_0 & 1/T_0^2 & \cdots & 1/T_0^{N-1} & 1/T_0^N \end{bmatrix} , \qquad (9)$$

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

We chose $M=15$, $N=14$, $T=512.$, $T_0=500.$, and $c=100.$ for this example. We used an implementation of the IEEE Floating Point Standard [4] on a VAX 11/780 [6]. $\varepsilon$ was $2^{-23}$ and $\mu$ 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.

$L$ 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.

$$L = \begin{bmatrix} 1 & & & & & \\ 0 & 1 & & & & \\ 0 & 0 & & & & \\ \vdots & \vdots & & & & \\ 0 & 0 & \cdots & 1 & & \\ L_{15,1} & L_{15,2} & \cdots & L_{15,14} & 1 \end{bmatrix} \qquad (10)$$

$$U = \begin{bmatrix} N & -1/T & -1/T^2 & \cdots & & -1/T^N \\ & N-1 & -1/T & \cdots & & -1/T^{N-1} \\ & & N-2 & \cdots & & -1/T^{N-2} \\ & & & \ddots & & \vdots \\ & & & & 1 & -1/T \\ & & & & & U_{15,15} \end{bmatrix} \qquad (11)$$

$A$'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, barely above the underflow threshold. S.Z. and G.U. compute all elements of $L$ and $U$ identically except for $U_{15,15}$. In fact, all additions in the computation of $L$ add normalized numbers with like magnitudes and like signs, so no cancellation, loss of significance, nor underflows occur. If the exponent range were unbounded, so underflow never happened, the correct value $U_{15,15} \approx 2.09281_{10}-37$ would be computed. This is the value computed using G.U. But when S.Z. is used instead, the computed value is $U_{15,15}^{S.Z.} \approx 1.72763_{10}-37$, a relative difference of .174 from the correct value. All additions in the computation of $U_{15,15}$ involve numbers of like magnitude and sign, so cancellation cannot be blamed for the discrepancy. This relative difference in the last entry of $U$ is very important, because one divides by $U_{15,15}$ in the course of solution. Thus, the computed solution $x^{G.U.}$ is very close to the true $x$, and the relative difference in solution vectors is

$$\frac{\| x^{G.U.} - x^{S.Z.} \|_\infty}{\| x^{G.U.} \|_\infty} \approx .211 \ .$$

Thus, G.U. obtains markedly better results than S.Z. This example is very interesting because there is nothing obviously wrong with the matrix. All its entries are unexceptional normalized numbers, and every row and every column contains at least one number no tinier than $1/T \approx .00195$ and none larger than $N=14$, 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.

This example was chosen to be simple and realistic; even though it can be solved analytically, it could be changed easily into a two dimensional problem without an explicit solution, but with the same sensitivity to underflow.

We repeat that even though $A$ appears very ill conditioned, since $k(A) \approx 1/\lambda$ (i.e. near the overflow threshold in most arithmetics), it is also well conditioned in the sense that $\mathrm{Cond}(A,x) \approx 5.5$. We will discuss the significance of this example later in section 4.

### 2.4 Example 4

Let

$$A = \begin{bmatrix} G & G \\ g & 2g \end{bmatrix} \ , \quad A^{-1} = \begin{bmatrix} 2/G & -1/g \\ -1/G & 1/g \end{bmatrix} \ ,$$

where $g/G$ underflows to 0 using either S.Z. or G.U. The $L$ obtained is thus the identity matrix since $L_{2,1}=fl(g/G)=0$, and so the $L$ and $U$ obtained are the exact factors of the matrix

$$A+E = \begin{bmatrix} G & G \\ 0 & 2g \end{bmatrix} \ ,$$

which is a very different matrix than $A$. If $b^T = (G,0)$, then $x = A^{-1}b = (2,-1)^T$, whereas $\hat{x} = (A+E)^{-1}b = (1,0)^T$, so $\hat{x}$ does not resemble $x$ at all. The residual $r$ is however guaranteed to be small, in the sense that $\| r \|_\infty / \| \, |A| \, |\hat{x}| + |b| \, \|_\infty$ is small:

$$\frac{\| r \|_\infty}{\| \, |A| \, |\hat{x}| + |b| \, \|_\infty} = \frac{\| E\hat{x} \|_\infty}{\| \, |A| \, |\hat{x}| + |b| \, \|_\infty} \le \frac{g|\hat{x}_1|}{G|\hat{x}_1| + G|\hat{x}_2|} \le \frac{g}{G} \le \lambda\varepsilon/2 \ .$$

Of course $A$ is an exceedingly ill conditioned matrix in the sense that $k(A) \approx 2G/g$ is beyond the reciprocal of the underflow threshold, so we would be inclined not to trust our results anyway. However, Cond($A$) is only 7. This is true because Cond($A$) = Cond($DA$) for any nonsingular diagonal matrix $D$, so $A$ has the same condition number as the utterly tame matrix

$$\begin{bmatrix} G^{-1} & \\ & g^{-1} \end{bmatrix} A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

Needless to say, in the absence of underflow we would compute a very accurate solution. We will return to this example later to explain why we can get inaccurate results from a matrix with a small condition number Cond($A$).

## 3. Error Analysis

### 3.1 Assumptions

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

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

$$fl(a \circ b) = (a \circ b)(1 + e) \qquad (12)$$

unless $a \circ b$ underflows or overflows. To take underflow into account, we write[3]

$$fl(a \circ b) = (a \circ b)(1 + e) + \eta \qquad (13)$$

unless $a \circ b$ overflows. In the case of G.U. we have the following constraints on $e$ and $\eta$:

(1) $|e| \le \varepsilon$ and $|\eta| \le \lambda\varepsilon$,

(2) $\eta e = 0$,

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

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

(1) $|e| \le \varepsilon$ and $|\eta| \le \lambda$, and

(2) $\eta e = 0$.

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

We assume no overflow occurs.

We ignore terms which are $O(\varepsilon^2)$ or $O(\varepsilon\mu)$. This allows us to replace $1/(1+e_1)$ by $1-e_2$ (where $|e_i| \le \varepsilon$), or $\eta_1 * (1+e)$ by $\eta_2$ (where $|\eta_i| \le \lambda$ for S.Z. or $\lambda\varepsilon$ for G.U.) when convenient.

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 matrix and vector norms. $\|b\|_\infty$ denotes the infinity norm of $b$, namely $\max_j |b_j|$, and $\|A\|_\infty$ denotes the infinity norm of $A$, $\|A\|_\infty = \max_{x \neq 0} \|Ax\|_\infty / \|x\|_\infty = \max_i \sum_j |A_{ij}|$.

Then $k(A) = \|A\|_\infty \|A^{-1}\|_\infty$ denotes usual 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 Gaussian elimination is used to solve

$$Ax = b \qquad (14)$$

for $x$ it generates instead an approximation $\hat{x} = x + \delta x$ which satisfies some perturbed problem

$$(A + \delta A)\hat{x} = b + \delta b \ . \qquad (15)$$

The task of backwards error analysis is to infer bounds on $\delta A$ and $\delta b$ from the details of the arithmetic used to implement the elimination process. These bounds can be used in turn to bound the residual

$$r = A\hat{x} - b = -\delta A\hat{x} + \delta b = A\delta x \qquad (16)$$

and then the error $\delta x$.

We proceed by breaking the elimination process into three steps:

1) Decomposition : Try to decompose $A$ into $LU$ and obtain $L$ and $U$ where $A = LU - \Delta A$; bound $\Delta A$ in terms of $A$.

2) Forward Substitution : Try to solve $Ly = b$ and obtain $\hat{y}$ where $(L+\delta L)\hat{y} = b + \Delta b$; bound $\delta L$ in terms of $L$ and $\Delta b$ in terms of $b$.

3) Backward Substitution : Try to solve $Ux = \hat{y}$ and obtain $\hat{x}$ where $(U+\delta U)\hat{x} = \hat{y} + \delta\hat{y}$; bound $\delta U$ in terms of $U$ and $\delta\hat{y}$ in terms of $\hat{y}$.

Thus, combining the above three steps we obtain

$$(A + \Delta A + L\delta U + \delta L U + \delta L \delta U)\hat{x} = b + \Delta b + (L+\delta L)\delta\hat{y} \ . \qquad (17)$$

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

$$r = -\delta A\hat{x} + \delta b = -(\Delta A + L\delta U + \delta L(U + \delta U))\hat{x} + \Delta b + (L + \delta L)\delta\hat{y} \qquad (18)$$

Equation (18) shows that we may choose $\delta A$ and $\delta b$ to be

$$\delta A = \Delta A + L\delta U + \delta L(U + \delta U) \qquad (19)$$

and

$$\delta b = \Delta b + (L + \delta L)\delta\hat{y} \ . \qquad (20)$$

The usual error analyses attribute the various perturbations $\Delta A$, $\delta L$, $\Delta b$, $\delta U$, and $\delta\hat{y}$ directly to roundoff errors during the elimination process. Underflow contributes additional perturbations thus:

(1) during decomposition $\cdots$ to $\Delta A$ and $\delta L$,

(2) during forward substitution $\cdots$ to $\Delta b$,

(3) during backward sustitution $\cdots$ to $\delta\hat{y}$ and $\hat{x}$.

These additional perturbations, of the order of $\lambda$ for S.Z. and $\mu = \lambda\varepsilon$ for G.U., all turn out to be comparable with or less than the respective terms to which they contribute. Our task is to summarize quantitatively the perturbations' cumulative effect.

Wilkinson's approach[7] is to determine a bound $\omega_w$ on the errors

$$\|\delta A\|_\infty \le \omega_w \|A\|_\infty \quad \text{and} \quad \|\delta b\|_\infty \le \omega_w \|b\|_\infty \qquad (21)$$

whence

$$||r||_\infty \le \omega_w [||A||_\infty ||\hat{x}||_\infty + ||b||_\infty] \qquad (22)$$

and then it will follow that the error $\delta x$ is bounded:

$$\frac{||\delta x||_\infty}{||x||_\infty + ||\hat{x}||_\infty} \le \omega_w \, k(A) \quad . \qquad (23)$$

The detailed derivation of $\omega_w$ from the details of the arithmetic is given elsewhere [9]. Theorem 1 below states simple requirements on $A$ and $b$ that ensure $\omega_w$ will be scarcely worse if underflow occurs then if it does not.

Skeel's approach [5], modified slightly here, is to determine a bound $\omega_s$ on the relative error in each entry of $A$ and $b$:

$$|\delta A| \le \omega_s |A| \quad \text{and} \quad |\delta b| \le \omega_s |b| \quad . \qquad (24)$$

From these inequalities follows a bound upon the error $\delta x$:

$$\frac{||\delta x||_\infty}{||x||_\infty} \le \omega_s \cdot \frac{|| \, |A^{-1}| \, |A| \, |x| + |A^{-1}| \, |b| \, ||_\infty}{(1 - \omega_s || \, |A^{-1}| \, |A| \, ||_\infty) \, ||x||_\infty} \qquad (25)$$

(provided the denominator is positive.) This motivates defining the following condition numbers:

$$\mathrm{Cond}(A,x) \equiv \frac{|| \, |A^{-1}| \, |A| \, |x| \, ||_\infty}{||x||_\infty} \qquad (26a)$$

$$\mathrm{Cond}(A) \equiv || \, |A^{-1}| \, |A| \, ||_\infty \quad . \qquad (26b)$$

$\mathrm{Cond}(A)$ is an upper bound for $\mathrm{Cond}(A,x)$ for all $x$; the error bounds are useful only if $\omega_s \mathrm{Cond}(A) < 1$.

The bound $\omega_s$ in (24) is not inferred directly from equations (19) and (20). Instead, following Oettli and Prager [8] and Skeel [5] we use an expression for $\omega_s$ obtainable from (16) in terms of the residual $r$:

$$\omega_s = \max_i \frac{|r_i|}{(|A| \, |\hat{x}| + |b|)_i} \qquad (27)$$

where the max is over those $i$ for which the denominator is nonzero. Following Skeel, we overestimate $\omega_s$ by analyzing the elimination process to infer an inequality

$$||r||_\infty \le \omega'_s || \, |A| \, |\hat{x}| + |b| \, ||_\infty \qquad (28)$$

from which we compute the overestimate $\bar{\omega}_s$ as

$$\bar{\omega}_s = \frac{\max_i (|A| \, |\hat{x}| + |b|)_i}{\min_i (|A| \, |\hat{x}| + |b|)_i} \cdot \omega'_s \qquad (29)$$

(where the min in the denominator is over the nonzero values of $(|A| \, |\hat{x}|)_i$ only). Unfortunately $\bar{\omega}_s$ can be a gross overestimate of $\omega_s$, as we will see when we return to example 3 later.

The detailed derivation of $\omega'_s$ is given in reference 9. Theorem 2 below states requirements on $A$ and $b$ that ensure $\omega'_s$ will be scarcely worse if underflow occurs than if it does not. These requirements on $A$ and $b$ are nearly identical to the requirements in the Wilkinson style analysis.

### 3.3 Results

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

Let $a_{max} = \max_{ij} |A_{ij}|$, and $g$ = [largest intermediate result appearing in the decomposition] / $a_{max}$. $g$ is the "pivot growth factor" and is $\le 2^{n-1}$.

Our objective is to estimate a bound $\omega_w$ for which

$$||r||_\infty \le \omega_w [||A||_\infty ||\hat{x}||_\infty + ||b||_\infty] \qquad (22)$$

In the absence of underflow, we can use

$$\omega_w = n^3 \varepsilon \, g / 2 \quad . \qquad (30)$$

If underflow occurs then

$$\omega_w = 3 n^3 \varepsilon \, g / 2 \qquad (31)$$

provided certain conditions are met.

For G.U. these condition are:

$$\begin{array}{lll} g a_{max} \ge \lambda & & \text{if there are any underflows during triangular decomposition} \\ ||b||_\infty \ge \dfrac{\lambda}{n} & & \text{if there are any intermediate underflows during forward and back substitutions} \\ \dfrac{||b||_\infty}{a_{max}} \ge \dfrac{2\lambda}{n^2} & & \text{if the solution } \hat{x} \text{ itself underflows.} \end{array} \qquad (32)$$

For S.Z. the above conditions still apply but $\lambda$ must be increased to $\lambda / \varepsilon$.

Proof: See reference 9.

**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 intermediate result appearing in the decomposition in column $j$] / $a_j$). $g_C$ is the "columnwise pivot growth factor" and is $\le 2^{n-1}$.

Our objective is to estimate a bound $\omega'_s$ for which

$$||r||_\infty \le \omega'_s || \, |A| \, |\hat{x}| + |b| \, ||_\infty \quad . \qquad (28)$$

In the absence of underflow we can use

$$\omega'_s = n^3 \varepsilon \, g_C \quad . \qquad (33)$$

If underflow occurs, then

$$\omega'_s = 3 n^3 \varepsilon \, g_C / 2 \qquad (34)$$

provided certain conditions are met.

For G.U. these conditions are:

$$\begin{array}{lll} g_C a_j \ge \lambda & & \text{for all } j \text{, if there are any underflows during triangular decomposition} \\ ||b||_\infty \ge \dfrac{\lambda}{2n} & & \text{if there are any intermediate underflows during forward and back substutions} \\ \dfrac{||b||_\infty}{a_{max}} \ge \dfrac{\lambda}{n^2} & & \text{if the solution } \hat{x} \text{ itself underflows.} \end{array} \qquad (35)$$

For S.Z. the above conditions apply with except $\lambda$ must be increased to $\lambda / \varepsilon$.

Proof: See reference 9.

The theorems indicate how to write software that will solve $Ax=b$ reliably despite underflow, and how the requirements for G.U. differ from those for S.Z. To keep the residual small in the sense of a Wilkinson style error analysis, we appeal to Theorem 1. With G.U., as long as one normalized number appears during the decomposition ($g a_{max} \ge \lambda$), residual with underflow has a bound not much worse than residual without underflow. If there are intermediate underflows while solving the triangular systems, as long as some component of $b$ is normalized ($||b||_\infty \ge \lambda$), residual with underflow has a bound

scarcely worse than without underflow. If the answer $\hat{x}$ 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 $\| b \| / a_{max}$ is not too small.

All these requirements are natural ones to make, since they say that when a problem's inputs and its computed solution are normalized numbers, we should expect the residual to be scarcely worse with underflow than without. Thus, the only gradual 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 $1/\varepsilon$. 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.

Theorem 2 shows that Skeel style bounds for the residual are scarcely worse with underflow than without provided conditions are satisfied that are almost the same as in Theorem 1. Therefore the previous paragraphs' comments remain valid provided, when underflow is gradual, at least one normalized number appears in each column of $A$, rather than just somewhere is $A$, before or during the decomposition process.

The programs in Appendices 1 and 2 show the few extra lines of code needed to implement the test for gradual underflows in the solution $\hat{x}$ in terms of the proposed IEEE standard.

## 4. Examples 3 and 4 Revisited

We wish to emphasize that we have only derived conditions under which *residual 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, or how accurate the computed solution will be.

In example 4 above, the matrix $A$ and vector $b$ satisfy all the conditions of Theorems 1 and 2 for G.U. as well as S.Z., so the residual is small, but the answer $\hat{x}$ is totally inaccurate. This inaccuracy can be explained either by the huge condition number $k(A) \approx$ overflow threshold, or the large backwards error in equation (27): $\omega_s = 1$. In this case $\omega_s$'s upper bound $\bar{\omega}_s$ in equation (29) is also 1. Thus, having a small value of Cond$(A)$ is not sufficient to guarantee accuracy given a small residual $\omega'_s$ ((28)), although a small value of $k(A)$ combined with a small residual $\omega_w$ is enough, as can be seen from (23).

Example 3 is another case where the conditions of Theorems 1 and 2 hold, but now G.U. successfully computes the last pivot $U_{15,15}$ and an accurate solution $\hat{x}$ while S.Z. does not. Again, we have a problem where $k(A)$ is huge and Cond$(A,x)$ is small. Now the $\omega_s$ of equation (27) is $\approx 5.23_{10} -8$, verifying the high accuracy of solution. Unfortunately the bad scaling of the matrix causes the upper bound $\bar{\omega}_s$ of equation (29) to be $2.0_{10}$ 20. This example demonstrates the occasionally intense pessimism of Skeel's approach.

In summary, the significance of examples 3 and 4 is to show that maintaining a small residual in the face of underflow does not guarantee an accurate solution $\hat{x}$, although we conjecture that for not terribly ill conditioned matrices G.U. will provide answers at least as accurate as provided by S.Z.

## 5. Conclusions

We have proven something quite unremarkable: if underflows are gradual, then we continue to get what we have come to expect from Gaussian elimination. That is, we get a small residual as long as the inputs and outputs are all representable (normalized) numbers and there is no indication of singularity or excessive pivot growth. If, however, underflows are handled in the usual way and set to zero, then no such simple guarantee can be made, and some kind of testing on the scaling of the problem is necessary. These results demonstrate that gradual underflow makes it easier to write reliable linear equation solvers than "store zero."

## 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. */
/* Lines prefixed by "G.U.:" are the additions to
the code required for reliability using gradual
underflow. */

/*set mode to prevent spurious warnings about
denormalized numbers */
G.U.: invoke normalizing mode

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];
          U[p,j]=SUM;
          end
     /* insert your favorite test for singularity
       or pivot growth */
     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];
          L[i,p]=SUM/U[p,p];
          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. */
```

```
/* 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;
     end

/* Back Substitution: UX=Y */

/* initialize flag for reporting underflows
in the solution x */
G.U.: xunderflow = false

for i=N downto 1 do
     begin
     SUM=0;
     for k=i+1 to N do SUM=SUM + U[i,k]*X[k];
     temp=Y[i] - SUM
G.U.:underflowflag = false;
     X[i]=temp/U[i,i];
G.U.:if underflowflag then xunderflow = underflowflag;
     end

/* if xunderflow = true, an underflow has
occurred in the solution X */
```

## References


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

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

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

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

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

[6]VAX is a trademark of the Digtal Equipment Corporation.

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

[8]W. Oettli and W. Prager,"Compatibility of Approximate Solution of Linear Equations with Given Error Bounds for Coefficients and Right-hand Sides," Numer. Math. 6(1964), 405-409

[9]J. W. Demmel, "Effects of Underflow on Solving Linear Systems", Computer Science Division, Electronic Research Laboratory Internal Report, U.C. Berkeley, 1981


## Acknowledgement


The author gratefully acknowledges the support of the U. S. Department of Energy (Contract DE-AM03-76SF00034, Project Agreement DE-AS03-79ER10358), the Office of Naval Research (Contract N00014-76-C-0013), and Prof. W. Kahan, whose comment and encouragement have been invaluable.