

Recent Results on Accuracy and Stability of Numerical Algorithms

Nick Higham
Department of Mathematics
University of Manchester

higham@ma.man.ac.uk
<http://www.ma.man.ac.uk/~higham/>



THE UNIVERSITY
of MANCHESTER

Contents

- ★ Area of a Triangle by Heron's Formula
- ★ Fused Multiply-Add Instruction
- ★ Gaussian Elimination with Rook Pivoting
- ★ Newton's Method
- ★ The Exchange Operator

Floating Point Arithmetic: Standard Model

$$fl(x \text{ op } y) = (x \text{ op } y)(1 + \delta),$$
$$|\delta| \leq u, \quad \text{op} = +, -, *, /.$$

Here, u is the **unit roundoff**, or machine precision.

In IEEE double precision arithmetic $u = 2^{-53} \approx 1.1 \times 10^{-16}$.

In IEEE, all arithmetic ops performed *as if* first calculated to infinite precision, then rounded.

Default: round to nearest, round to even in case of tie.

Example

$$fl(abc) = ab(1 + \delta_1) \cdot c(1 + \delta_2) = abc(1 + \delta_1)(1 + \delta_2)$$
$$\approx abc(1 + \delta_1 + \delta_2).$$

Area of a Triangle

Heron's formula for area A of triangle with sides of length a , b , and c :

$$A = \sqrt{s(s-a)(s-b)(s-c)}, \quad s = (a+b+c)/2.$$

Inaccurate for needle-shaped triangles, e.g., $a \approx b + c$.

Kahan (1983): let $a \geq b \geq c$, then evaluate $4A$ as

$$\sqrt{(a+(b+c))(c-(a-b))(c+(a-b))(a+(b-c))}.$$

Gives an accurate answer for all data.

Examples

In IEEE double:

| a | 10 | 1 | 1 |
|-------|----------------------|-----------------------|-----------------------|
| b | 11 | 1 | $\sin(\text{eps})$ |
| c | 12 | $1\text{e-}13$ | $\cos(\text{eps})$ |
| Exact | $5.152123\text{e+}1$ | $5.000000\text{e-}14$ | $1.110223\text{e-}16$ |
| Kahan | $5.152123\text{e+}1$ | $5.000000\text{e-}14$ | $1.110223\text{e-}16$ |
| Heron | $5.152123\text{e+}1$ | $4.996004\text{e-}14$ | $0.000000\text{e+}0$ |

Subtraction

Theorem 1 (Sterbenz) *Let x and y be floating point numbers with $y/2 \leq x \leq 2y$. Then $x - y$ is computed exactly (assuming $x - y$ does not underflow).*

Recall $a \geq b \geq c$,

$$\sqrt{(a + (b + c))(c - (a - b))(c + (a - b))(a + (b - c))}.$$

Examine $c - (a - b)$. We have

$$b \leq a \leq b + c \leq 2b$$

so by the theorem, $a - b$ is exact. Hence

$$fl(c - (a - b)) = (c - (a - b))(1 + \delta), \quad |\delta| \leq u.$$

Fused Multiply-Add Instruction

Intel IA-64 architecture has a fused multiply-add instruction with just one rounding error:

$$fl(x + y * z) = (x + y * z)(1 + \delta), \quad |\delta| \leq u.$$

With an FMA:

- ★ Inner product $x^T y$ can be computed with half the rounding errors.
- ★ The algorithm

$$w = b * c$$

$$e = w - b * c$$

$$x = (a * d - w) + e$$

computes $x = \det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right)$ with high relative accuracy (Kahan).

Fused Multiply-Add Instruction (cont.)

But

★ What does $a*d + c*b$ mean?

★ The product

$$(x + iy)^*(x + iy) = x^2 + y^2 + i(xy - yx)$$

may evaluate to non-real with an FMA.

★ $b^2 - 4ac$ can evaluate negative even when $b^2 \geq 4ac$.

Gaussian Elimination with Rook Pivoting

On k th stage of GE, swap rows k, r and columns k, s , where

$$|a_{rs}^{(k)}| = \max_{k \leq i \leq n} |a_{is}^{(k)}| = \max_{k \leq j \leq n} |a_{rj}^{(k)}|.$$

Intermediate between partial and complete pivoting.

| | | | | | |
|---|----|---|---|---|----|
| 1 | 10 | 1 | 2 | 4 | 5 |
| 0 | 5 | 2 | 7 | 8 | 2 |
| 2 | 0 | 3 | 1 | 9 | 4 |
| 3 | 2 | 4 | 2 | 1 | 0 |
| 1 | 4 | 5 | 6 | 3 | 1 |
| 1 | 0 | 3 | 4 | 0 | 12 |

Properties of Rook Pivoting

- **Cost of pivot search:** worst-case $O(n^3)$, but in practice $O(n^2)$. Foster (1997): if all $a_{ij}^{(k)}$ are iid random variables then $O(n^2)$.
- **Growth factor bounds:**

$$\rho_n \leq \begin{cases} 2^{n-1}, & \text{partial (Wilkinson),} \\ 1.5n^{\frac{3}{4}} \log n, & \text{rook (Foster),} \\ cn^{1/2} n^{\frac{1}{4}} \log n, & \text{complete (Wilkinson).} \end{cases}$$

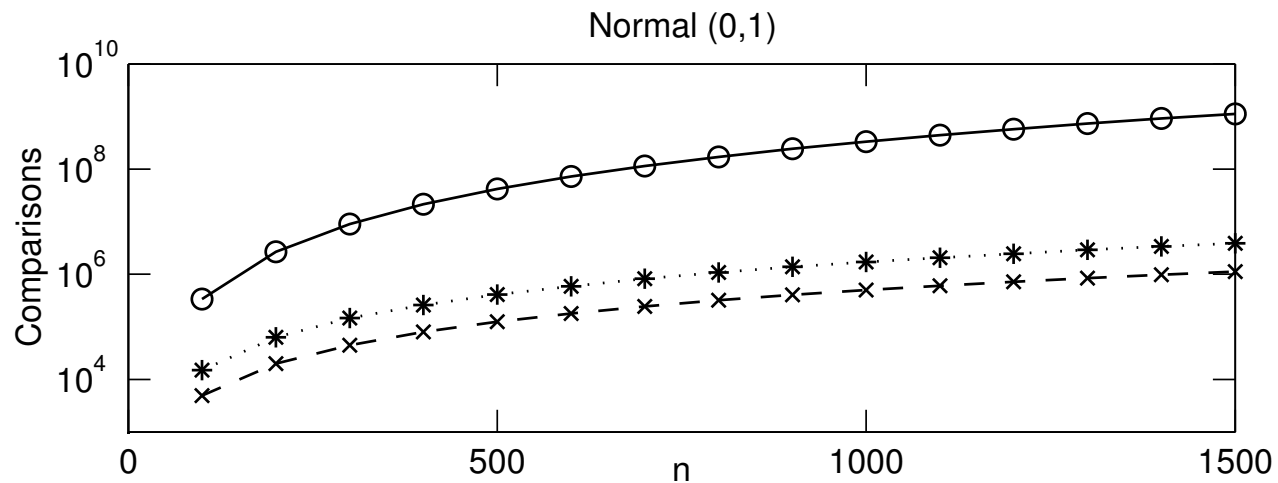
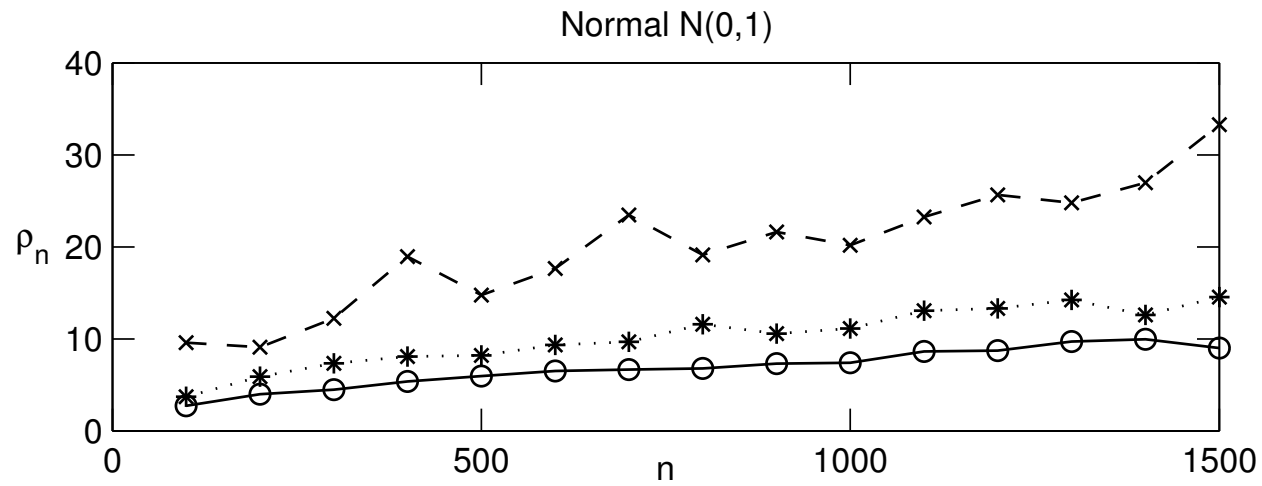
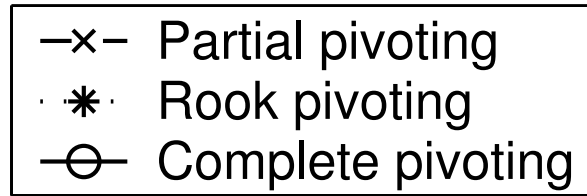
- **LU factors:** in $PAQ = LU$,

$$\text{cond}(U) = \| |U^{-1}| |U| \|_{\infty} \leq 2^n - 1,$$

because $|u_{ij}| \geq |u_{ii}|$. For GE with any pivoting scheme,

$$\frac{\|x - \hat{x}\|_{\infty}}{\|x\|_{\infty}} \leq 3nu \text{cond}(A) \text{cond}(U) + O(u^2).$$

Numerical Experiment



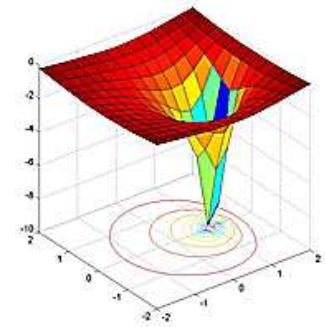
History of Rook Pivoting

- Introduced for nonsymmetric matrices by Neal & Poole (1992): “A Geometric Analysis of Gaussian Elimination”.
- Related strategy for symmetric indefinite matrices introduced by Fletcher (1976).
- Ashcraft, Grimes & Lewis (1998) noted that Bunch–Kaufman pivoting strategy does not bound L factor in $PAP^T = LDL^T$. Motivated their definition of *symmetric rook pivoting*.

Rook Pivoting Summary

- ★ For nonsymmetric A , mainly of pedagogical interest. May give insight into growth factor for partial pivoting.
- ★ *Symmetric* rook pivoting of real practical interest. E.g., when bounded L required: modified Cholesky factorization of Cheng & H (1998).

Matrix Computation Toolbox (H, 2002)



<http://www.ma.man.ac.uk/higham/mctoolbox>
Functions `gep`, `ldlt_symm`.

Newton's Method

$F : \mathbb{R}^m \mapsto \mathbb{R}^m$ continuously differentiable.

$J(v)$: Jacobian matrix of F .

Assume that J is Lipschitz:

$$\|J(\bar{v}) - J(v)\| \leq \beta \|\bar{v} - v\| \quad \text{for all } v, \bar{v} \in \mathbb{R}^m.$$

Attempt to solve $F(v) = 0$ by Newton's method.

Assume there is v_* such that $F(v_*) = 0$ and $J_* = J(v_*)$ is nonsingular.

Newton's Method in Floating Point

Implement Newton as

$$\text{Solve } J(v_i)d_i = -F(v_i), \quad \text{let } v_{i+1} = v_i + d_i.$$

In floating point arithmetic,

$$\hat{v}_{i+1} = \hat{v}_i - (J(\hat{v}_i) + E_i)^{-1} (F(\hat{v}_i) + e_i) + \varepsilon_i.$$

- e_i is error in computing residual $F(\hat{v}_i)$ at precision \bar{u} ,

$$\|e_i\| \leq u \|F(\hat{v}_i)\| + \psi(F, \hat{v}_i, \bar{u}).$$

- E_i is error in forming $J(\hat{v}_i)$ and solving linear system for d_i ,

$$\|E_i\| \leq u \phi(F, \hat{v}_i, n, u).$$

- ε_i is error made when adding correction \hat{d}_i to \hat{v}_i ,

$$\|\varepsilon_i\| \leq u (\|\hat{v}_i\| + \|\hat{d}_i\|).$$

Limiting Forward Error & Residual

Theorem 2 (Tisseur, 2001) *If*

- $J(v_*)$ *not too ill conditioned,*
- J -*evaluation and solver not too inaccurate/unstable,*
- $\|v_0 - v_*\|$ *not too large,*
- Lipschitz constant β *not too large,*

*then Newton in floating point arithmetic yields a sequence $\{\hat{v}_{i+1}\}$ whose **normwise relative error** & **residual** decrease until*

$$\frac{\|\hat{v}_{i+1} - v_*\|}{\|v_*\|} \approx \frac{\|J(v_*)^{-1}\|}{\|v_*\|} \psi(F, v_*, \bar{u}) + u,$$

$$\|F(\hat{v}_{i+1})\| \approx \psi(F, \hat{v}_i, \bar{u}) + u \|J(\hat{v}_i)\| \|\hat{v}_i\|.$$

Applications

Can apply to **iterative refinement**: solving $F(v) = 0$ with a “good” starting value, e.g.

- linear system $Ax = b$,
- eigenvalue problem $Ax = \lambda x$,
- generalized eigenproblem $Ax = \lambda Bx$, and higher degree ...
- in **fixed precision**: $\bar{u} = u$, or
- in **mixed precision**, e.g., $\bar{u} = u^2$.

Refinement can

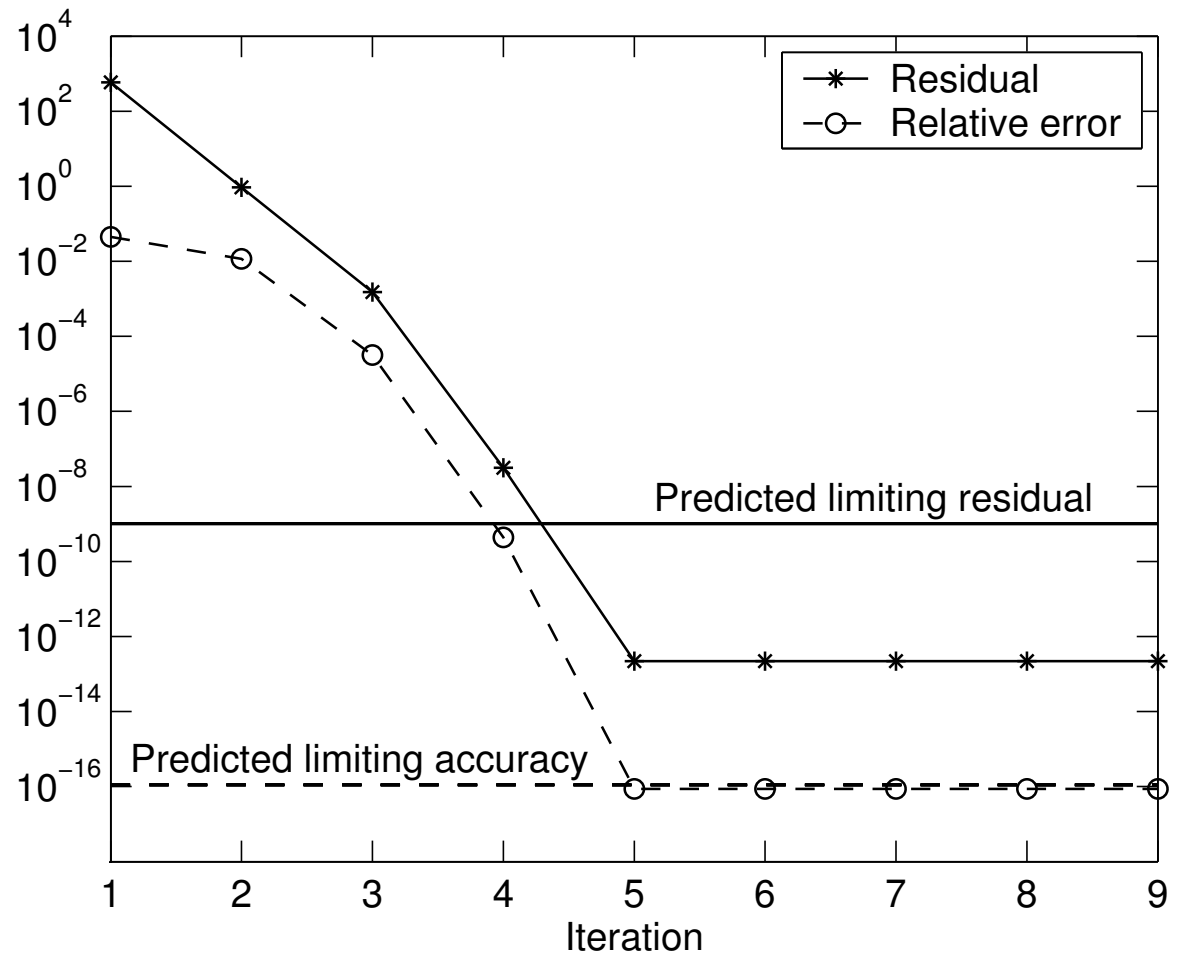
- ★ ameliorate instability of algorithm (**fixed precision**),
- ★ overcome ill conditioning of problem (**mixed precision**).

Example

$$F(v) = \begin{bmatrix} (A - \lambda I)x \\ e_s^T x - 1 \end{bmatrix} = 0, \quad v = \begin{bmatrix} x \\ \lambda \end{bmatrix}.$$

Smallest eigenpair of 10×10 Frank matrix, $\bar{u} = u^2$:

eig:
res $\approx 10^{-10}$
err $\approx 10^{-8}$



Newton's Method for ??

With $J = \text{diag}(\pm 1)$,

$$X_{k+1} = \frac{1}{2}(X_k + JX_k^{-T}J), \quad X_0 = A \in \mathbb{R}^{n \times n}.$$

Newton's Method for ??

With $J = \text{diag}(\pm 1)$,

$$X_{k+1} = \frac{1}{2}(X_k + JX_k^{-T}J), \quad X_0 = A \in \mathbb{R}^{n \times n}.$$

If $X_k \rightarrow X$ then $X = JX^{-T}J$, i.e.,

$$X^T J X = J.$$

Newton's Method for ??

With $J = \text{diag}(\pm 1)$,

$$X_{k+1} = \frac{1}{2}(X_k + JX_k^{-T}J), \quad X_0 = A \in \mathbb{R}^{n \times n}.$$

If $X_k \rightarrow X$ then $X = JX^{-T}J$, i.e.,

$$X^T J X = J.$$

Such X are J -orthogonal.

Newton's Method for ??

With $J = \text{diag}(\pm 1)$,

$$X_{k+1} = \frac{1}{2}(X_k + JX_k^{-T}J), \quad X_0 = A \in \mathbb{R}^{n \times n}.$$

If $X_k \rightarrow X$ then $X = JX^{-T}J$, i.e.,

$$X^T J X = J.$$

Such X are J -orthogonal.

Can prove: if $\lambda(JA^TJA)$ off negative real axis,

$$X_k \rightarrow A(JA^TJA)^{-1/2}.$$

Downdating Problem

Compute **Cholesky factorization** of a positive definite $C = A^T A - B^T B$, where $A \in \mathbb{R}^{p \times n}$ ($p \geq n$) and $B \in \mathbb{R}^{q \times n}$.

If we can find a **J -orthogonal** Q such that

$$Q \begin{bmatrix} A \\ B \end{bmatrix} \begin{matrix} p \\ q \end{matrix} = \begin{bmatrix} R \\ 0 \end{bmatrix} \begin{matrix} n \\ p+q-n \end{matrix},$$

with $J = \text{diag}(I_p, -I_q)$ and $R \in \mathbb{R}^{n \times n}$ upper triangular, then

$$C = \begin{bmatrix} A \\ B \end{bmatrix}^T J \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix}^T Q^T J Q \begin{bmatrix} A \\ B \end{bmatrix} = R^T R.$$

Indefinite Least Squares Problem

$$\min_x (b - Ax)^T J (b - Ax),$$

where $A \in \mathbb{R}^{m \times n}$, $m \geq n$, and

$$J = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}, \quad p + q = m.$$

Normal equations:

$$A^T J A x = A^T J b.$$

Unique solution iff

$$A^T J A \text{ positive definite.}$$

Methods:

- Chandrasekaran, Gu & Sayed (1998): Cholesky–QR.
- Bojanczyk, H & Patel (2002): hyperbolic QR.

The Exchange Operator

Let $A \in \mathbb{R}^{n \times n}$. The system

$$\begin{array}{c} 1 \\ p \\ q \end{array} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{array}{c} p \\ q \end{array} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{array}{c} 1 \\ p \\ q \end{array} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

where A_{11} is nonsingular, can be rewritten

$$\begin{bmatrix} x_1 \\ y_2 \end{bmatrix} = \text{exc}(A) \begin{bmatrix} y_1 \\ x_2 \end{bmatrix},$$

where

$$\text{exc}(A) = \begin{bmatrix} A_{11}^{-1} & -A_{11}^{-1}A_{12} \\ A_{21}A_{11}^{-1} & A_{22} - A_{21}A_{11}^{-1}A_{12} \end{bmatrix}.$$

Theorem 2 *Let $A \in \mathbb{R}^{n \times n}$ be J orthogonal. Then $\text{exc}(A)$ is orthogonal. Moreover, if A is orthogonal and A_{11} is nonsingular then $\text{exc}(A)$ is J orthogonal.*

History of Exchange Operator

- ▶ Called *gyration operator* in **network analysis** (Duffin et al., 1966).
- ▶ Known as *sweep operator* in **statistics**.
Used to express Gauss–Jordan elimination.
- ▶ Called *principal pivot transform* in **linear algebra**.
 A is a P -matrix implies $\text{exc}(A)$ also a P -matrix.
- ▶ Numerical analysis: Pan & Plemmons (1989), Stewart & Stewart (1998), H (2002).

Uses of Exchange Operator

- ▶ Error analysis of products of J -orthogonal matrices (Stewart & Stewart, 1998; Bojanczyk, H & Patel, 2002):
 - map to orthogonal matrices using `exc`,
 - do error analysis on orthogonal,
 - map back to J -orthogonal using `exc`.
- ▶ Can derive a **hyperbolic CS decomposition** of a J -orthogonal matrix directly from usual CS decomposition of an orthogonal matrix (H, 2002).
- ▶ Can derive an algorithm for constructing **random J -orthogonal** matrices with prescribed norm and condition number (H, 2002).

SECOND EDITION

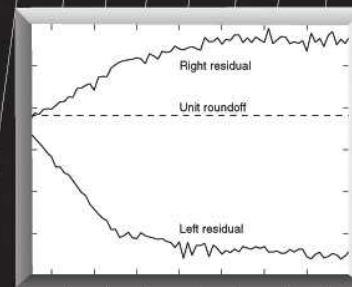
Nicholas J. Higham

Accuracy and Stability
of Numerical Algorithms

Nicholas J. Higham

Accuracy and Stability of Numerical Algorithms

SECOND EDITION



Accuracy and Stability of Numerical Algorithms gives a thorough, up-to-date treatment of the behavior of numerical algorithms in finite precision arithmetic. It combines algorithmic derivations, perturbation theory, and rounding error analysis, all enlivened by historical perspective and informative quotations.

This second edition expands and updates the coverage of the first edition (1996) and includes numerous improvements to the original material. Two new chapters treat symmetric indefinite systems and skew-symmetric systems, and nonlinear systems and Newton's method. An expanded treatment of Gaussian elimination incorporates rank pivoting and additional error bounds. Other new topics include rank-revealing LU factorizations, weighted and constrained least squares problems, and the fused multiply-add operation found on some modern computer architectures.

Although not designed specifically as a textbook, this new edition is a suitable reference for an advanced course. It can also be used by instructors at all levels as a supplementary text from which to draw examples, historical perspective, statements of results, and exercises.

From reviews of the first edition:

"This definitive source on the accuracy and stability of numerical algorithms is quite a bargain and a worthwhile addition to the library of any statistician heavily involved in computing."
— Robert L. Strawderman, *Journal of the American Statistical Association*, March 1999

"This text may become the new Bible about accuracy and stability for the solution of systems of linear equations. It covers 689 pages carefully collected, investigated, and written... One will find that this book is a very suitable and comprehensive reference for research in numerical linear algebra, software usage and development, and for numerical linear algebra courses."
— N. Köckler, *Zentralblatt für Mathematik*, Band B4796

"Nick Higham has assembled an enormous amount of important and useful material in a coherent, readable form. His book belongs on the shelf of anyone who has more than a casual interest in rounding error and matrix computations."
— G.W. Stewart, *SIAM Review*, March 1997



Nicholas J. Higham is Richardson Professor of Applied Mathematics at the University of Manchester, England. He is the author of more than 80 publications and is a member of the editorial boards of *Foundations of Computational Mathematics*, the *IMA Journal of Numerical Analysis*, *Linear Algebra and its Applications*, and the *SIAM Journal on Matrix Analysis and Applications*.

For more information about SIAM books, journals, conferences, memberships, or activities, contact:

siam

Society for Industrial and Applied Mathematics
3600 University City Science Center
Philadelphia, PA 19104-2688
215-381-9800 • Fax 215-386-7999
siam@siam.org • www.siam.org



BKOT0080

siam

OT80

siam

Time to \LaTeX ASNA

| | |
|----------------|----------|
| DX2-33 | 7.5 mins |
| Pentium 120Mhz | 1.3 mins |
| Pentium 500Mhz | 20 secs |
| Pentium 1Ghz | 10 secs |
| Pentium 2.4Ghz | 6 secs |