Life as a Developer of Numerical Software

Sven Hammarling
NAG Ltd, Oxford
&
University of Manchester

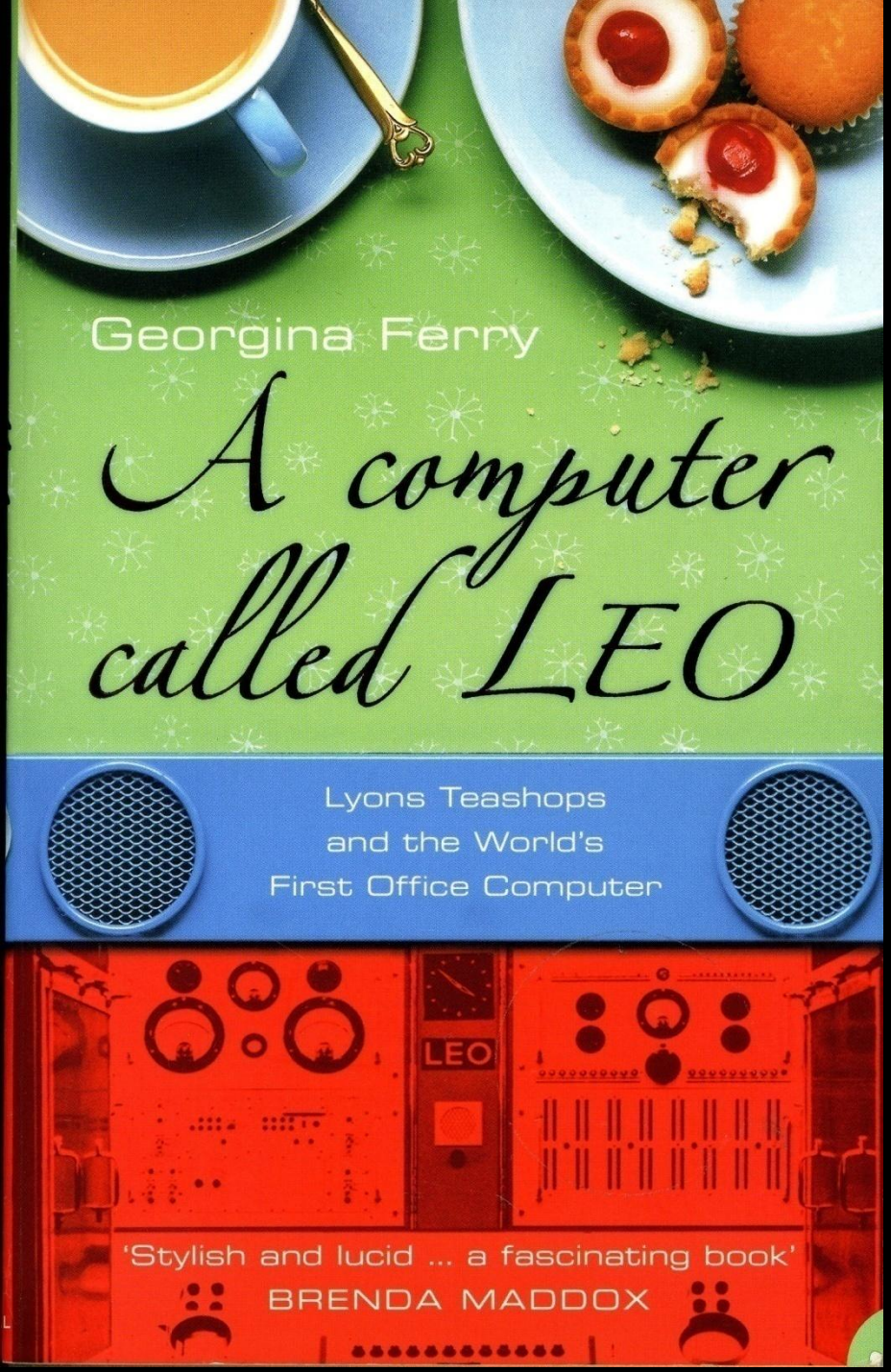
Significant Topics in the story

- Architecture
- Floating Point Arithmetic
- Languages and Tools
- Software and Coding

Personal Programming History

- Mechanical computation, 1964+
- First program as student in 1965 on Elliot 803
- Middlesex Polytechnic, 1968 – 1979
- National Physical Laboratory, 1979 – 1982
- NAG Ltd, 1982 – date
- Level 2 and Level 3 BLAS, 1988 & 1990
- LAPACK, 1992 – date
- ScaLAPACK, 1997

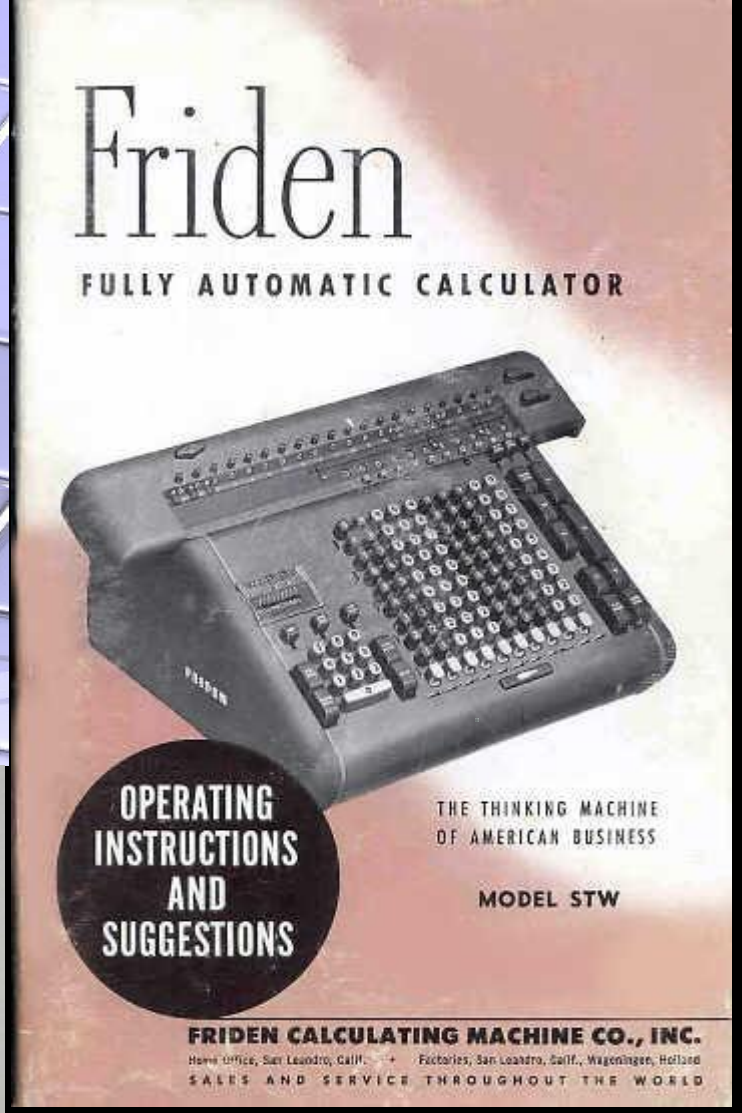
LEO – Lyons Electronic Office


Georgina Ferry
A computer called LEO
Lyons Teashops and the World's First Office Computer
LEO
'Stylish and lucid ... a fascinating book'
BRENDA MADDOX

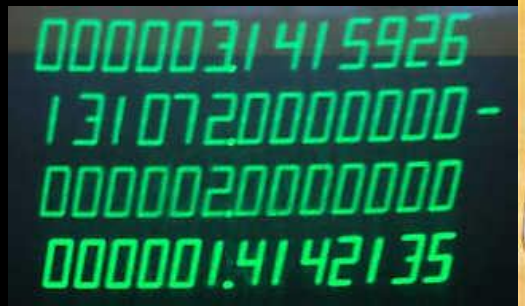

J. LYONS & CO. LTD. No 321

Mechanical Computation


CURTA


Friden
FULLY AUTOMATIC CALCULATOR
OPERATING INSTRUCTIONS AND SUGGESTIONS
THE THINKING MACHINE OF AMERICAN BUSINESS
MODEL STW
FRIDEN CALCULATING MACHINE CO., INC.
SALES AND SERVICE THROUGHOUT THE WORLD


CURTA


000003.141 5926
131 072.0000000 -
000002.0000000
000001.4142135

“Since the use of the punched-card equipment required the use of an operator, it encouraged user participation generally, and this was a distinctive feature of Pilot ACE operation

…

Speaking for myself I gained a great deal of experience from user participation, and it was this that led to my own conversion to backward error analysis.”

Wilkinson (1980) in *A History of Computing in the Twentieth Century*, Academic Press

Architecture

Elliott 803

- 1962 on
- 39 bit machine
- 4K or 8K storage

Loughborough Grammar School, 1970s

Eliot 803 Backing Tape

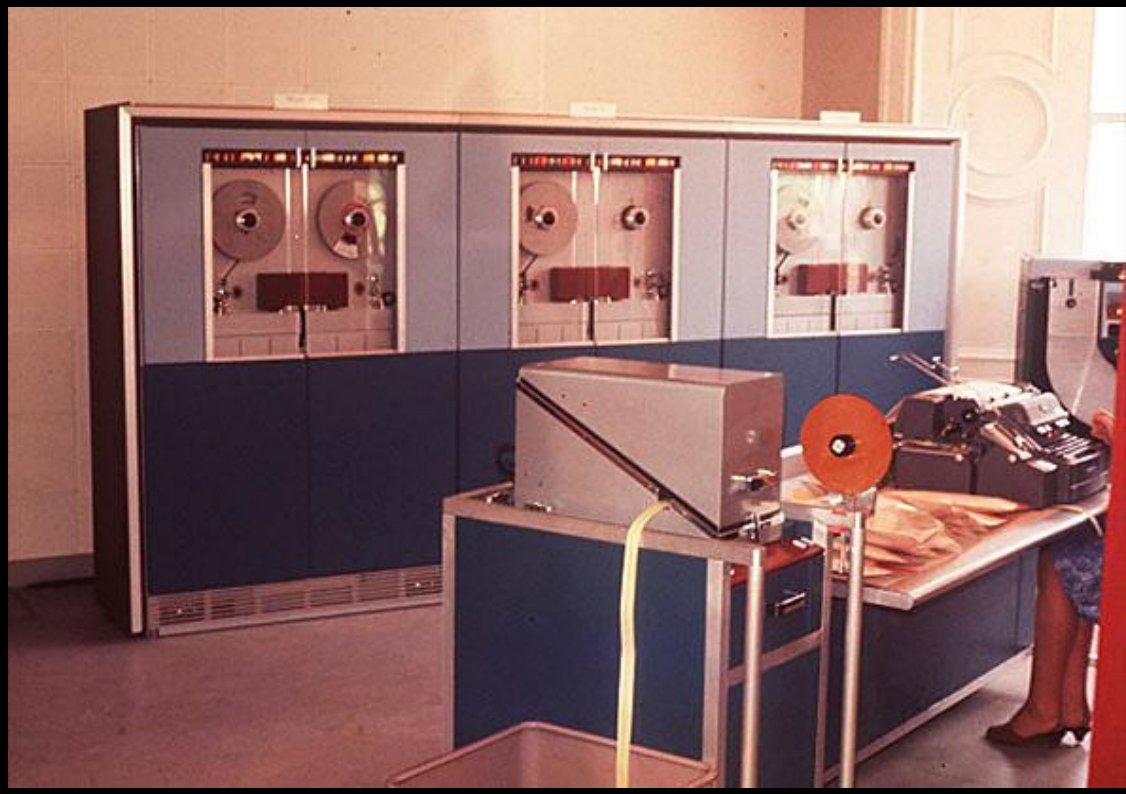
35mm cine film with a magnetic coating, made by Kodak

DEC 10, PDP-10

- Approx 1965 onwards
- 36 bit words
- C. G. Bell, A. Kotok, T. N. Hastings, and R. Hill, *The Evolution of the DECsystem 10*, Comm ACM, 21, 1978

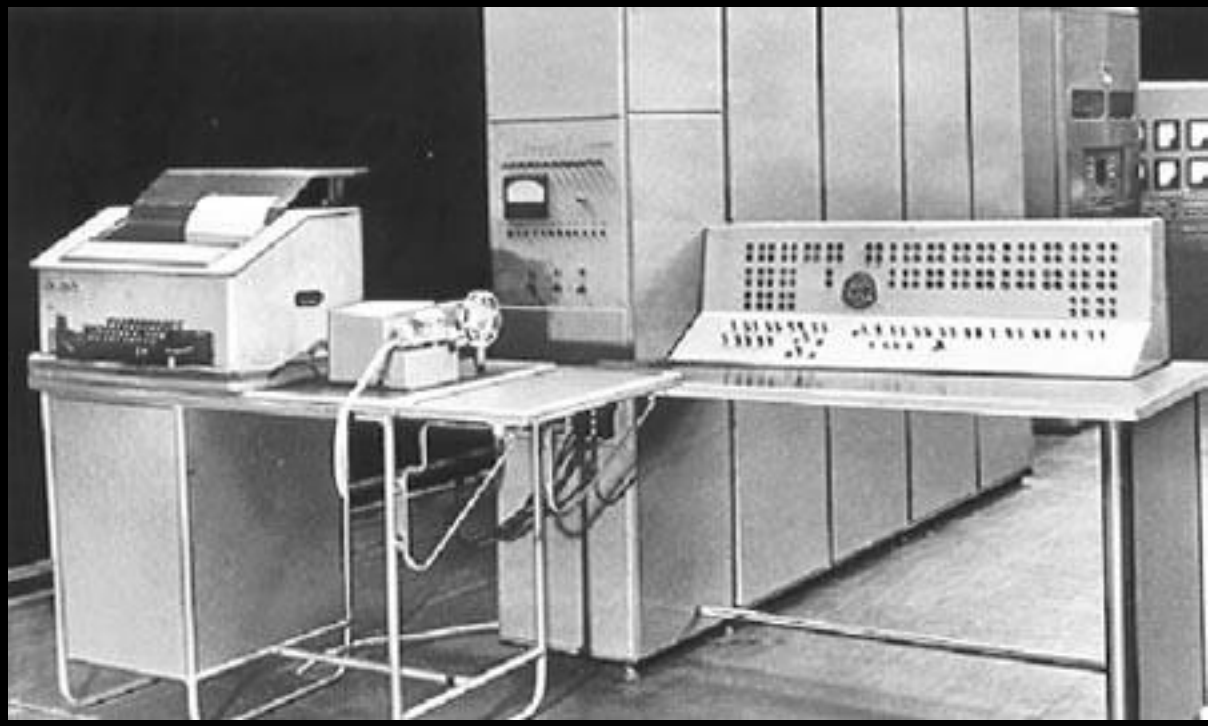
KDF9

- Approx 1963 – 1980
- In the Pilot ACE line
- Multi-user system
- Octal, 48 bit words



SETUN Ternary Computer

- Moscow University
- First ran in 1958
- Ternary arithmetic
- 6 trits (approx 9.5 bits)



I did not use this one!!

IBM 360

1964 on Hexadecimal machine with cache memory on later models

G. C. Stierhoff and A. G. Davis. A History of the IBM Systems Journal. *IEEE Annals of the History of Computing*, Vol. 20, No. 1 (Jan. 1998), pages 29-35.

Cray-1 Vector Machine

1976 Los Alamos

64 bit words, eight 64 word, 64 bit vector registers

130+ Mflops

Cray-1 Vector Machine

Cray X-MP Multiprocessor Vector Machine

1982 successor to the Cray-1

1 – 4 processors

800 Mflops

Fastest machine, 1983 - 1985

CRAY XMP-48 at the EPFL Switzerland

Architectural Change

- Difficult to keep pace with hardware
  - Various word lengths and bases
  - Vector registers and pipelines (Level 2 BLAS)
  - Hierarchical memory (Level 3 BLAS)
  - Shared memory parallelism (Level 3 BLAS)
  - Distributed memory parallelism (parallel BLAS)
  - Multi-core

Floating Point Arithmetic:
model & analysis

Portability

- For portability, needed a model of floating point arithmetic
- IFIP/WG 2.5. B Ford et al. See: http://www.nsc.liu.se/~boein/ifip/projects/p1.txt
- W S Brown *A simple but realistic model of floating-point computation* ACM Trans. Math. Software,7, 445–480, 1981
- NAG: Chapter X02
- LAPACK: xLAMCH

Model Representation

$$x = \pm m \times b^{e-t}, \quad 0 \le m \le b^t - 1$$

b - base (or radix)

t - precision

e - exponent, with exponent range $[e_{\min}, e_{\max}]$

m - mantissa (or significand)

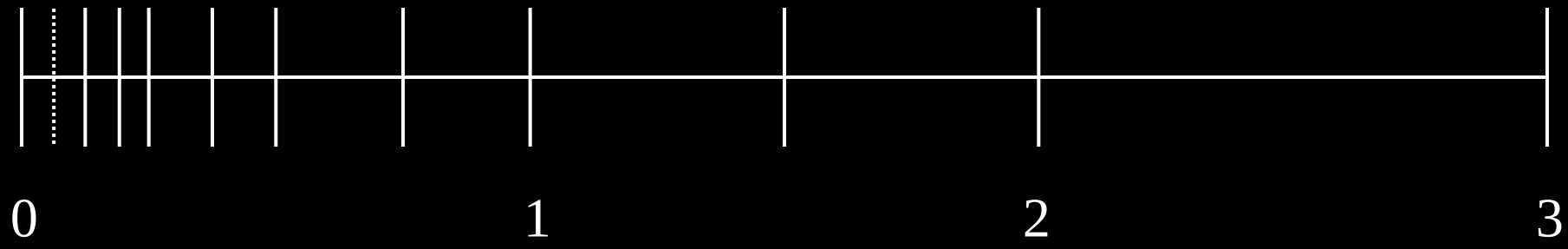
If $x \neq 0$ and $m \geq b^{t-1}$ then x is normalized, otherwise $x \neq 0$ is denormalized. Denormalized numbers between 0 and the smallest normalized number are called subnormal.

$$u = \tfrac{1}{2} \times b^{1-t}$$

is called the relative machine precision

Floating Point Numbers - Example

$$b = 2, \quad t = 2, \quad e_{\min} = -2, \quad e_{\max} = 2$$



The relative machine precision is

$$u = \tfrac{1}{2} \times b^{1-t} = \tfrac{1}{4}$$

See LAPACK routine `_LAMCH`, the Matlab built in `eps`, or the Fortran intrinsic `EPSILON`

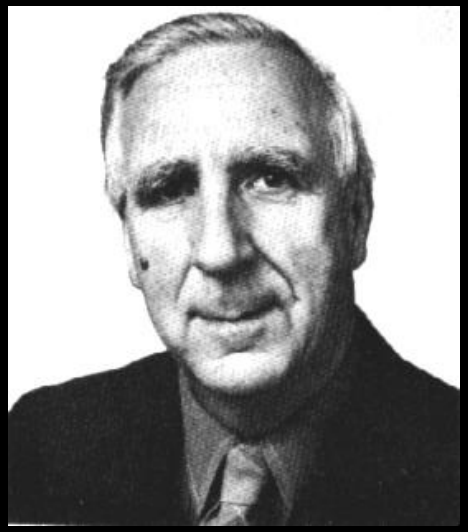
IEEE Arithmetic Formats

Format	Precision	Exponent	Approx Range	Approx precision (u)
Single	24 bits	8 bits	$10^{\pm 38}$	10^{-8}
Double	53 bits	11 bits	$10^{\pm 308}$	10^{-16}
Extended	≥ 64	≥ 15	$10^{\pm 4932}$	10^{-20}

W. Kahan's self-portrait

“I have little doubt that about 80 per cent. of all the results printed from the computer are in error to a much greater extent than the user would believe, ..."

Leslie Fox, IMA Bulletin, 1971



One of my aims has been to give users the means to measure the quality of their results

NATIONAL PHYSICAL LABORATORY

Notes on Applied Science
No. 32

Rounding Errors in Algebraic Processes

by
J. H. WILKINSON, M.A., Sc.D.

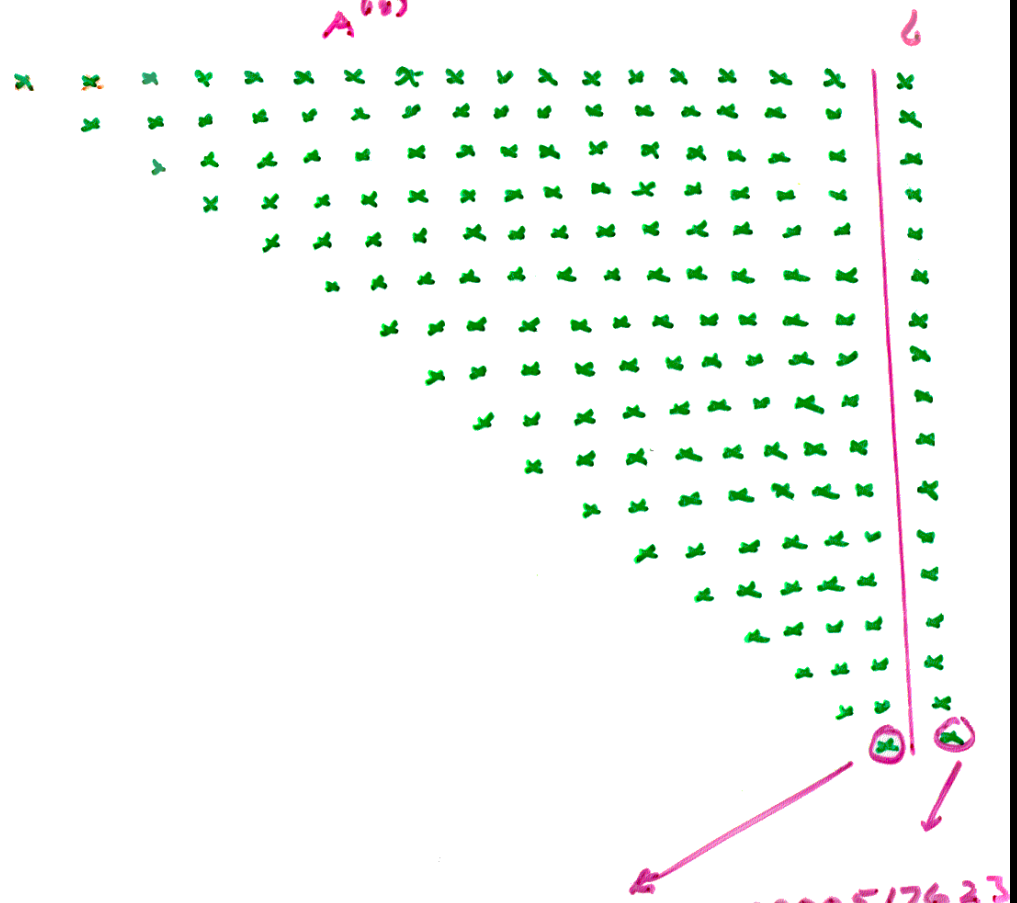
LONDON
HER MAJESTY'S STATIONERY OFFICE
1963

N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, USA, second edition, 2002.

Gaussian Elimination Example

System of 18 equations solved, using Gaussian elimination, by Fox, Goodwin, Turing and Wilkinson

Described in Wilkinson's Turing lecture (1971)



$$\cdot 0000623788_{10} = \cdot 0000517623$$

$$x_{18} = \frac{\cdot 0000517623}{\cdot 0000623788} = \cdot 829805971$$

$r = b - A\tilde{x}$

r_1 b_1 $(A\tilde{x})_1 = 0000000000135\ldots$ (9)

b_2 $(A\tilde{x})_2 = 000000000241$ (9)

$r_{18} = b_{18}$ $(A\tilde{x})_{18} = 00000000000353$ (10)

Wilkinson Model

Let x be a real number. Then we use the notation:

$$fl(x) = \text{ floating point value of } x$$

Fundamental assumption:

$$\boxed{fl(x) = x(1+\varepsilon), |\varepsilon| \le u}$$

where u is the unit rounding error, or relative machine precision. Of course

$$\frac{fl(x) - x}{x} = \varepsilon$$

Floating Point Error Analysis

If x and y are floating point numbers then

$$\boxed{fl(x \otimes y) = (x \otimes y)(1+\varepsilon), \quad |\varepsilon| \leq u}$$

where

$$\otimes \equiv +, -, \times, \div$$

Forward and Backward Error

$$Ax = b, \quad r = b - A\tilde{x}$$

$$A = \begin{pmatrix} 99 & 98 \\ 100 & 99 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, x = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad (\kappa_1 \ A \ = 199^2 \simeq 4 \times 10^4)$$

Consider $\tilde{x} = \begin{pmatrix} 2.97 \\ -2.99 \end{pmatrix}$

$$\tilde{x} - x = \begin{pmatrix} 1.97 \\ -1.99 \end{pmatrix}, \quad \text{but } r = \begin{pmatrix} -0.01 \\ 0.01 \end{pmatrix}$$

Now consider $\tilde{x} = \begin{pmatrix} 1.01 \\ -0.99 \end{pmatrix}$,

$$\tilde{x} - x = \begin{pmatrix} 0.01 \\ 0.01 \end{pmatrix}, \quad \text{but } r = \begin{pmatrix} -1.97 \\ -1.97 \end{pmatrix}$$

Condition and Error Analysis

Approximately:

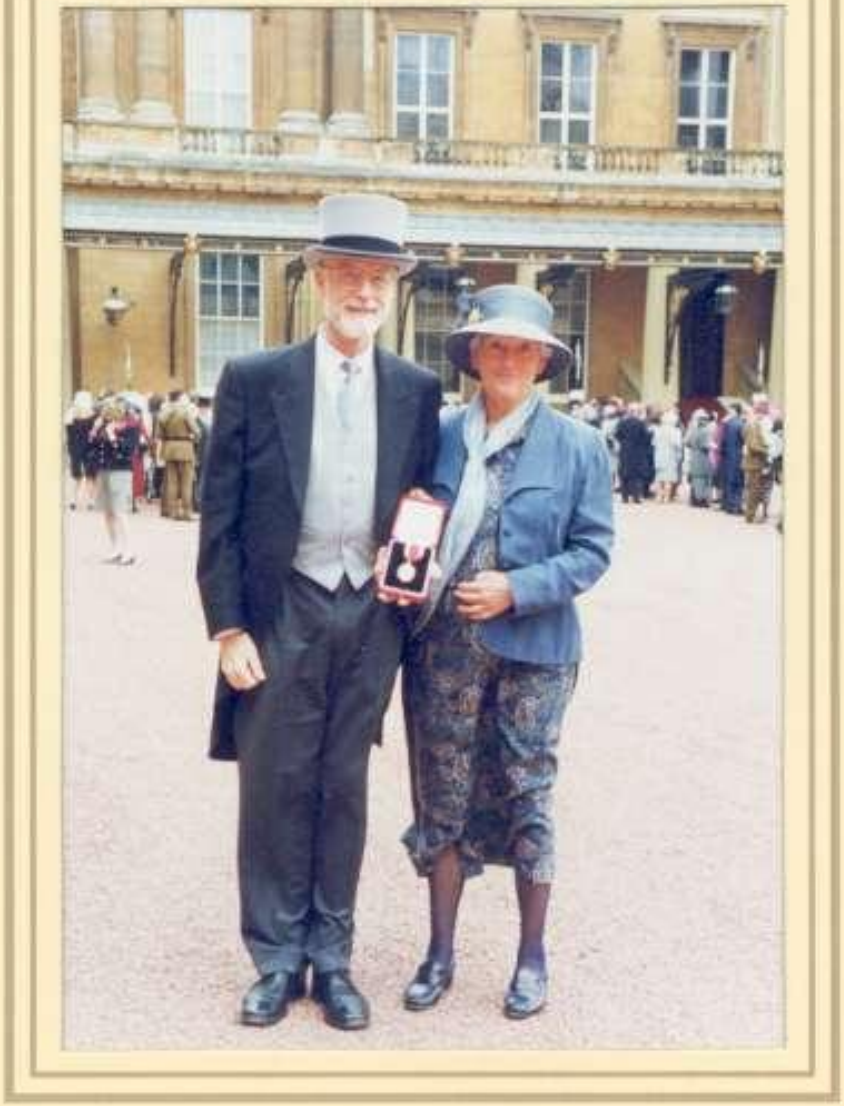
forward error $\leq$

condition number $\times$ backward error

Languages and Tools

Elliott and Algol 60

- On the 8K version, by 1965 Algol 60 was available
- Written by a team led by Tony Hoare; see *The Emperor's Old Clothes*, Comm. of the ACM, 24, 1981 (1980 Turing lecture)



Obsolete Languages

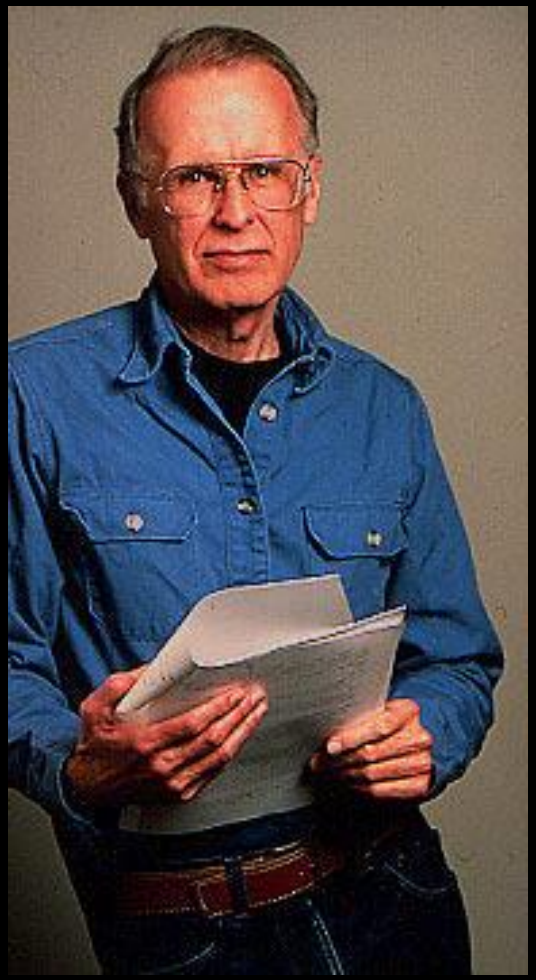
- Some languages have come and more or less gone
  - Algol 60 and Algol 68
  - Basic
  - Occam
  - Pascal
  - …
  - Ada (apart from embedded systems?)

Fortran

- Fortran lives on – now Fortran 2003
- Fortran 2008 under discussion

John Backus

1924 - 2007



Best regards, Walter
John Backus

Programmer's
Reference Manual
October 15, 1956

THE FORTRAN AUTOMATIC CODING SYSTEM FOR THE IBM 704 EDPM®

This manual supersedes all earlier information about the FORTRAN system. It describes the system which will be made available during late 1956, and is intended to permit planning and FORTRAN coding in advance of that time. An Introductory Programmer's Manual and an Operator's Manual will also be issued.

APPLIED SCIENCE DIVISION
AND PROGRAMMING RESEARCH DEPT.
International Business Machines Corporation
590 Madison Ave., New York 22, N. Y.

WORKING COMMITTEE

J. W. BACKUS
R. J. BEEBER
S. BEST
R. GOLDBERG
H. L. HERRICK
R. A. HUGHES
University of California Radiation Laboratory, Livermore, Calif.
L. B. MITCHELL
R. A. NELSON
R. NUTT
United Aircraft Corp., East Hartford, Conn.
D. SAYRE
P. B. SHERIDAN
H. STERN
I. ZILLER

C ← FOR COMMENT / STATEMENT NUMBER (1–5)	CONTINUATION (6)	FORTRAN STATEMENT (7–72)	IDENTIFICATION (73–80)
C		PROGRAM FOR FINDING THE LARGEST VALUE	
C	X	ATTAINED BY A SET OF NUMBERS	
		DIMENSION A(999)	
		FREQUENCY 30(2,1,10), 5(100)	
		READ 1, N,.(A(I), I = 1,N)	
1		FORMAT (I3/(12F6.2))	
		BIGA = A(1)	
5		DO 20 I = 2,N	
30		IF (BIGA-A(I)) 10,20,20	
10		BIGA = A(I)	
20		CONTINUE	
		PRINT 2, N, BIGA	
2		FORMAT (22H1THE LARGEST OF THESE I3, 12H NUMBERS IS F7.2)	
		STOP 77777	

Solving

$$\sin(x) = e^{-x}$$

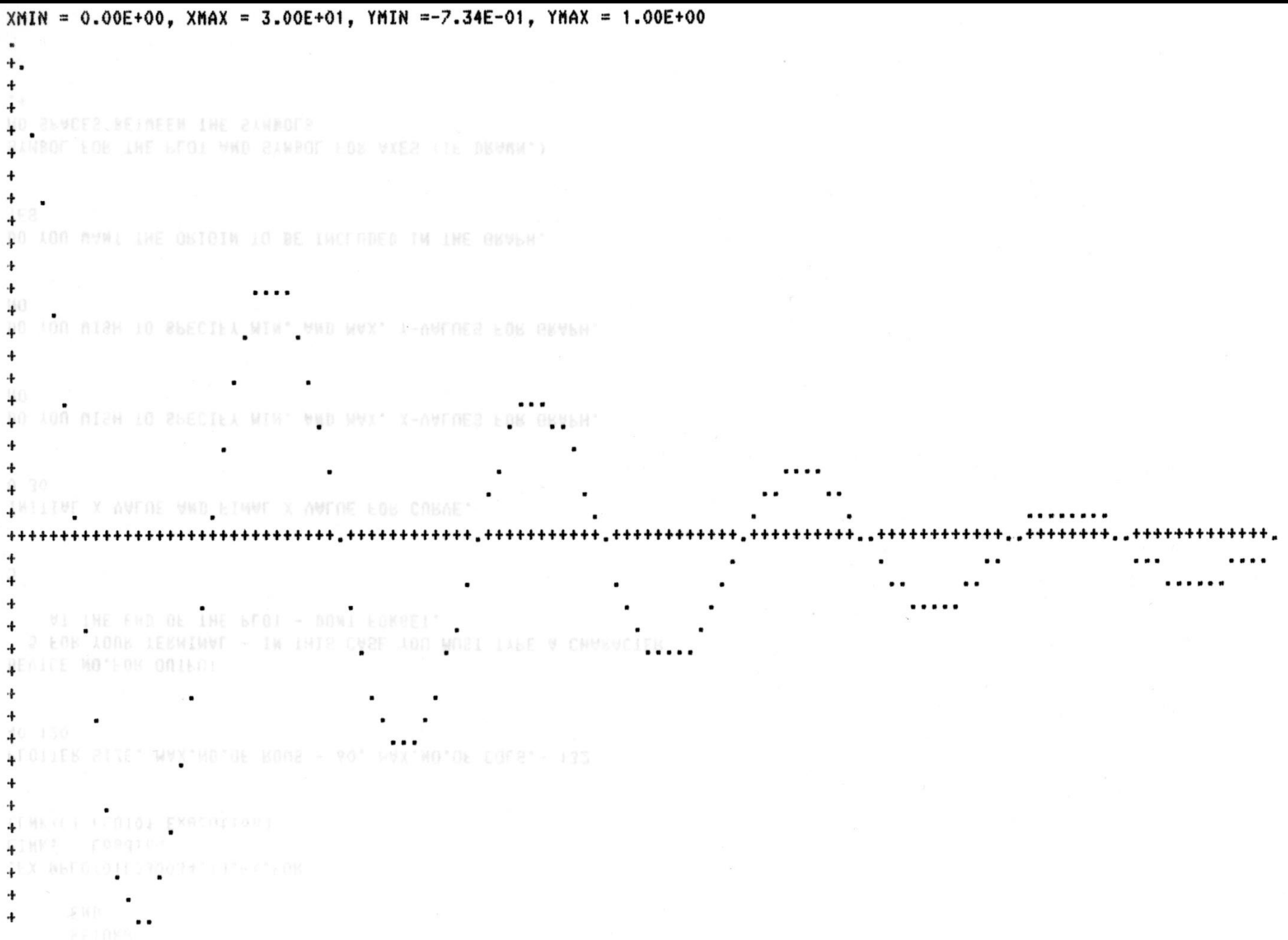
using the NAG Library

```
 TYPE NAG1.FOR

C  EXAMPLE OF USING NAG ROUTINE C05AAF.
C
      EXTERNAL F
      A=0.0
      B=1.0
      EPS=0.0
      ETA=0.0
      IFAIL=0
      X=0.5
C
      CALL C05AAF(A,B,EPS,ETA,F,X,IFAIL)
C
      Y=F(X)
      WRITE(5,1)X,Y
    1 FORMAT(8HOROOT = ,1PE15.7/14HOVALUE OF F = ,1PE15.7///1X)
      STOP
      END
      REAL FUNCTION F(X)
      REAL X
      F=SIN(X)-EXP(-X)
      RETURN
      END

.EX NAG1.FOR,NAG:FOR.REL/SEARCH
FORTRAN: NAG1
MAIN.
F
LINK:   Loading
[LNKXCT NAG1 Execution]


ROOT =   5.8853274E-01

VALUE OF F =  -7.4505806E-09



STOP

END OF EXECUTION
CPU TIME: 0.03  ELAPSED TIME: 0.24
EXIT

.
```

Plotting

$\cos(x)\exp(-x/10)$

```
.TYPE FX.FOR

      FUNCTION F(X)
      F=COS(X)*EXP(-0.1*X)
      RETURN
      END

.EX @PLOT01[230034,1],FX.FOR
LINK:   Loading
[LNKXCT PLOT01 Execution]


PLOTTER SIZE. MAX.NO.OF ROWS - 60, MAX.NO.OF COLS.- 132
40 120


DEVICE NO.FOR OUTPUT
  5 FOR YOUR TERMINAL - IN THIS CASE YOU MUST TYPE A CHARACTER
    AT THE END OF THE PLOT - DONT FORGET.

5


INITIAL X VALUE AND FINAL X VALUE FOR CURVE.
0 30


DO YOU WISH TO SPECIFY MIN. AND MAX. X-VALUES FOR GRAPH.
NO


DO YOU WISH TO SPECIFY MIN. AND MAX. Y-VALUES FOR GRAPH.
NO


DO YOU WANT THE ORIGIN TO BE INCLUDED IN THE GRAPH.
YES


SYMBOL FOR THE PLOT AND SYMBOL FOR AXES (IF DRAWN.)
NO SPACES BETWEEN THE SYMBOLS
.+
```


XMIN = 0.00E+00, XMAX = 3.00E+01, YMIN =-7.34E-01, YMAX = 1.00E+00

Compiler Flags

- Use all checking and debugging flags available
  - With NAGWare F95, V5.1, I currently use:

    f95 -C -C=undefined -g -gline -info -u
- Do NOT use flags such as
  - `-Ounsafe` “Perform possibly unsafe optimizations …”
- Use available software tools

Language Support

- Need more language support of the sort being provided in Fortran 2003 and C99
  - Machine (or environment) parameters
  - IEEE arithmetic, such as directed rounding
  - Exception handling
  - Interval arithmetic
- See W Kahan, *How Futile are Mindless Assessments of Roundoff in Floating-Point Computation?* http://www.cs.berkeley.edu/~wkahan/

Software and Coding

First Numerical Library

Wilkes, Wheeler and Gill (1951) "Preparation of Programmes for an Electronic Digital Computer" Addison-Wesley

With special reference to the EDSAC and the use of a library of subroutines

"Short, ready-made programs for performing the more common computing operations … are usually called subroutines" (page 1)

For V:

$z \leftarrow x \pm y$

$y \leftarrow Ax, \quad A = A^T$

SPECIFICATIONS OF LIBRARY SUBROUTINES

Each subroutine is distinguished by a letter denoting its category and a serial number within that category. The categories are as follows.

Category	Subject
A	Floating point arithmetic.
B	Arithmetical operations on complex numbers.
C	Checking.
D	Division.
E	Exponentials.
F	General routines relating to functions.
G	Differential equations.
J	Special functions.
K	Power series.
L	Logarithms.
M	Miscellaneous.
P	Print and layout.
Q	Quadrature.
R	Read (i.e., Input).
S	nth root.
T	Trigonometrical functions.
U	Counting operations.
V	Vectors and matrices.

D

In the specifications on succeeding pages the following information is given in abbreviated form immediately beneath the title of each subroutine:

1. Type of subroutine, i.e., whether open, closed, interpretive, or special.
2. Restriction on address of first order. If the word "even" appears it denotes that the first order must have an even address; if no note appears it indicates that the address may be either odd or even.
3. Total number of storage locations occupied by the subroutine.
4. Addresses of any storage locations needed as working space by the subroutine.
5. Approximate operating time (not possible to state in all cases).

The gaps in the numbering within each category correspond to subroutines which have become obsolete.

Handbook for Automatic Computation

Edited by

F. L. Bauer · A. S. Householder · F. W. J. Olver

H. Rutishauser † · K. Samelson · E. Stiefel

Volume II

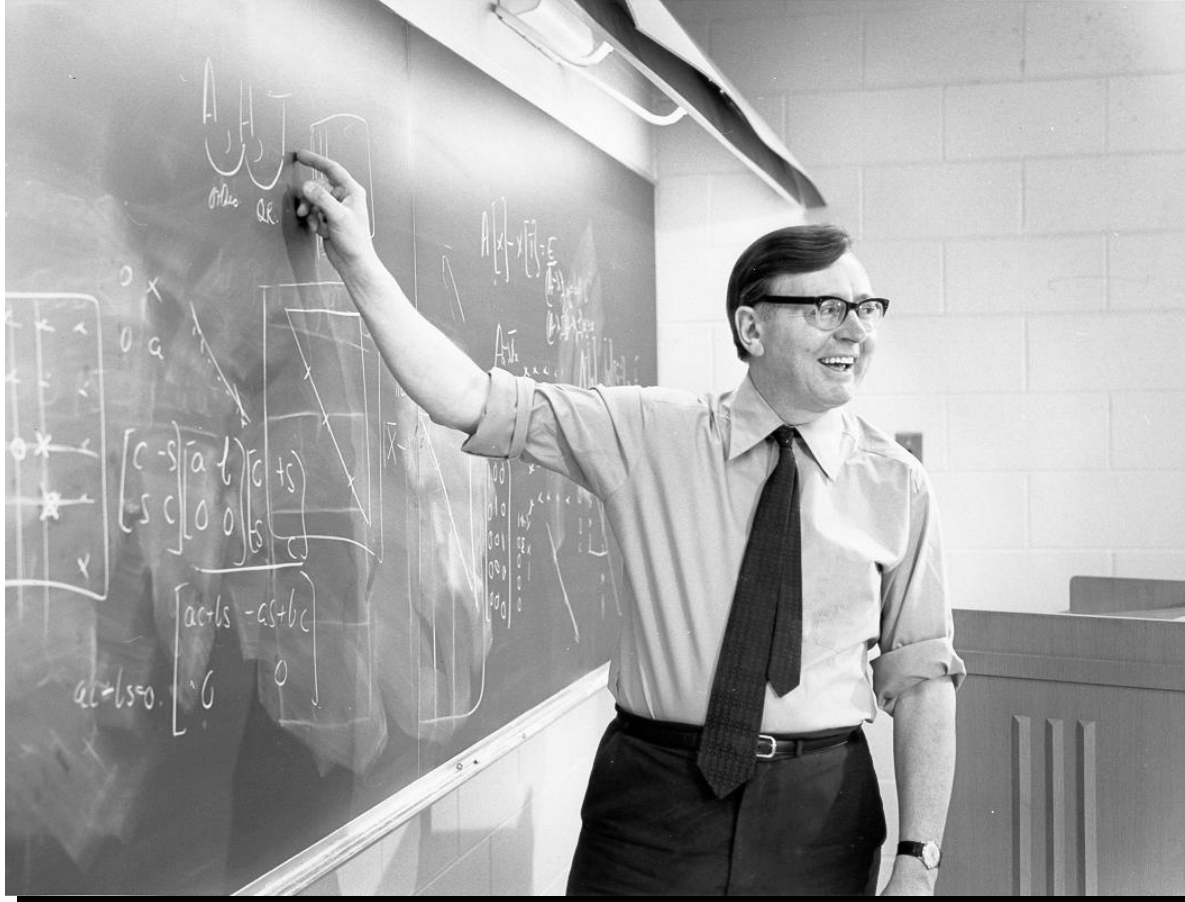
J. H. Wilkinson · C. Reinsch

Linear Algebra

Chief editor

F. L. Bauer

Springer-Verlag Berlin Heidelberg New York 1971



T. J. Dekker, W. Hoffmann; *Algol 60 procedures in numerical algebra part 2;* MC Tracts 23, Mathematisch Centrum, Amsterdam (1968)

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

6

B. T. Smith · J. M. Boyle · J. J. Dongarra
B. S. Garbow · Y. Ikebe · V. C. Klema
C. B. Moler

Matrix Eigensystem Routines – EISPACK Guide

Second Edition

Springer-Verlag
Berlin · Heidelberg · New York

LINPACK INPACK NPACK PACK ACK CK K USERS' GUIDE

J.J. Dongarra C.B. Moler
J.R. Bunch G.W. Stewart

siam

Copyrighted material

Basic Linear Algebra Subprograms

- Level 1 BLAS, 1979
- Level 2 BLAS, 1988
  - Vector machines
- Level 3 BLAS, 1990
  - Hierarchical memory, shared memory parallel
- Dates are for TOMS publication

Discussion of linear equation solvers on the Pilot ACE

"An interesting feature of the codes is that they made a very intensive use of subroutines; the addition of two vectors, multiplication of a vector by a scalar, inner products, etc., were all coded this way."

Wilkinson, 1980

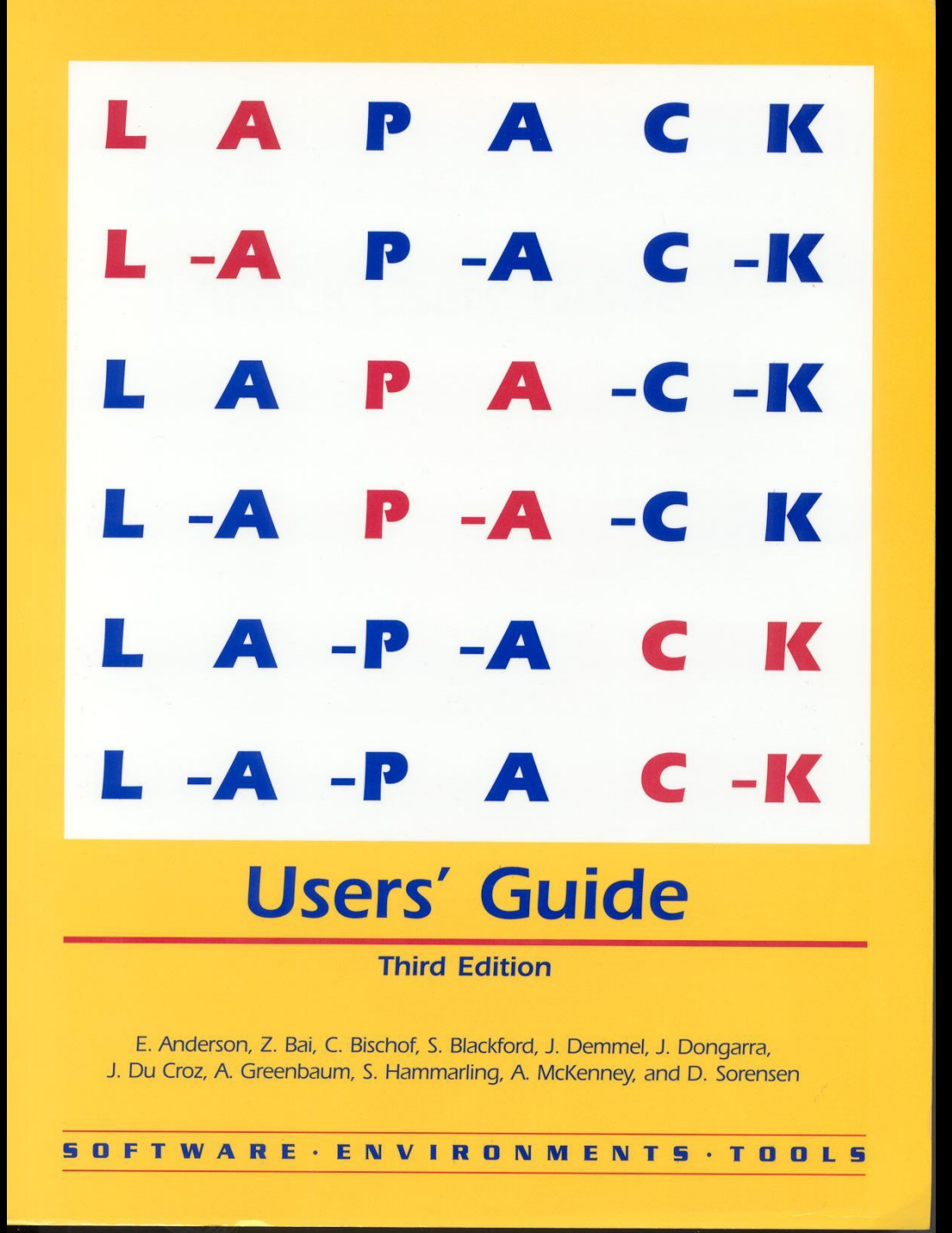
The History of Computing in the 20th Century.

Efficient Use of Data

"Since all machines have stores of finite size often divided up into high speed and auxiliary sections, storage considerations often have a vitally important part to play."

Wilkinson, MTAC, 1955

Emphasis on error and condition estimates, as well as efficiency




LAPACK

Documentation and Testing

- Documentation vital if you want your software to be used
- Testing and quality assurance for reliability and robustness

Software Code Sizes

- NAG Fortran Library, Mark 21
  - Source: **28.3 Mb**
  - Stringent test programs: **42.1 Mb**
  - Example programs: **4.4 Mb**
  - XML documentation: **136 Mb**
- LAPACK 3.0
  - Source: **12.1 Mb**
  - Testing: **10.9 Mb**
  - Timing: **6.5 Mb**
  - Users' Guide: **407 pages**

Quality of Computed Solutions

- Quality numerical software should implement reliable algorithms and should, where appropriate, provide measures of solution quality
  - Error bounds or estimates
  - Condition or sensitivity estimates
- SH *An introduction to the quality of computed solutions*. In B Einarsson, editor, *Accuracy and Reliability in Scientific Computing*, pages 43–76. SIAM, 2005

An Example

DGESVX is an 'expert' driver for solving $AX = B$

`DGESVX(...,RCOND,FERR,BERR,WORK,...,INFO)`

`RCOND` : Estimate of $1/\kappa\ A$

`FERR(j)`: Estimated forward error for X_j

`BERR(j)`: Componentwise relative backward error for X_j

(smallest relative change in any element of A and B

that makes X_j an exact solution)

`WORK(1)`: Reciprocal of pivot growth factor $(1/g)$

`INFO` : > 0 if A is singular or nearly singular

Final Comments

Numerical Software Development

- Numerical software is heavily used
- Taken for granted and not always appreciated
- Hard and time consuming, attention to detail is vital
- Difficult to keep pace with architectural developments
- Still need better language and compiler support, and software tools for floating point