A Brief History of Numerical Libraries

Sven Hammarling
NAG Ltd, Oxford
&
University of Manchester
First – Something about Jack
Jack's thesis  
(August 1980)  
30 years ago!
ALGORITHM 589
SICEDR: A FORTRAN Subroutine for Improving the Accuracy of Computed Matrix Eigenvalues

JACK J. DONGARRA
Argonne National Laboratory

Categories and Subject Descriptors: G.1.3 [Numerical Analysis]: Numerical Linear Algebra—eigenvalues, G.4 [Mathematics of Computing]: Miscellaneous—FORTRAN
General Terms: Algorithms
Additional Key Words and Phrases: Matrix eigensystems, iterative method; eigensystem improvement

1. INTRODUCTION

SICEDR is a FORTRAN subroutine for improving the accuracy of a computed real eigenvalue and improving or computing the associated eigenvector. It is first used to generate information during the determination of the eigenvalues by the Schur decomposition technique. In particular, the Schur decomposition technique results in an orthogonal matrix Q and an upper quasi-triangular matrix T, such that

\[ A = QTQ^T. \] (1.1)

Matrices A, Q, and T and the approximate eigenvalue, say \( \lambda \), are then used in the improvement phase. SICEDR uses an iterative method similar to iterative improvement for linear systems to improve the accuracy of \( \lambda \) and improve or compute the eigenvector \( x \) in \( O(n^2) \) work, where \( n \) is the order of the matrix \( A \). The method used in SICEDR is described in [1, 5].

2. USAGE

For a description of the calling sequence, see the listing presented at the end of this paper.

SICEDR factors the matrix into its Schur decomposition, and this is termed the pre-SICE phase. A modification of the EISPACK routine HQR2 is used for this purpose (see [4, pp. 100–101] for details).
IMPROVING THE ACCURACY OF COMPUTED SINGULAR VALUES

by

J. J. Dongarra

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Prepared for the U. S. DEPARTMENT OF ENERGY
under Contract W-31-109-Eng-38
Unrolling Loops in FORTRAN*

J. J. Dongarra and A. R. Hinds
Argonne National Laboratory, Argonne, Illinois 60439, U.S.A.

SUMMARY
The technique of 'unrolling' to improve the performance of short program loops without resorting to assembly language coding is discussed. A comparison of the benefits of loop 'unrolling' on a variety of computers using an assortment of FORTRAN compilers is presented.

KEY WORDS: Unrolled loops FORTRAN Loop efficiency Loop doubling
Parallel Scientific Computing
First International Workshop, PARA '94
Lyngby, Denmark, June 1994
Proceedings

Recent Advances in Parallel Virtual Machine and Message Passing Interface
4th European PVM/MPI Users’ Group Meeting
Cracow, Poland, November 1997
Proceedings
Small Selection of Jack’s Projects

- Netlib and other software repositories
- NA Digest and na-net
- PVM and MPI
- TOP 500 and computer benchmarking
- NetSolve and other distributed computing projects
- Numerical linear algebra
Onto the Rest of the Talk!
Rough Outline

- History and influences
- Fortran
- Floating Point Arithmetic
- Libraries and packages
- Proceedings and Books
- Summary
Ada Lovelace
(Countess Lovelace)

Born Augusta Ada Byron

1815 – 1852

The language Ada was named after her
“Is thy face like thy mother’s, my fair child! Ada! sole daughter of my house and of my heart? When last I saw thy young blue eyes they smiled, And then we parted,—not as now we part, but with a hope”

Childe Harold’s Pilgrimage,
Lord Byron
Sketch of
The Analytical Engine
Invented by Charles Babbage

By L. F. MENABREA
of Turin, Officer of the Military Engineers

from the Bibliothèque Universelle de Genève, October, 1842, No. 82

With notes upon the Memoir by the Translator
ADA AUGUSTA, COUNTESS OF LOVELACE
## Program for the Bernoulli Numbers

<table>
<thead>
<tr>
<th>Number of Operation</th>
<th>Number of Operation</th>
<th>Variables acting upon</th>
<th>Variables receiving results</th>
<th>Indication of change in the value on any Variable</th>
<th>Statement of Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$V_2 \times V_3$</td>
<td>$V_1$, $V_2$, $V_4$</td>
<td>$V_5$, $V_6$</td>
<td>$V_2 = V_1 V_3$</td>
<td>$2n$ $2n$ $2n$</td>
</tr>
<tr>
<td>2</td>
<td>$V_4 - V_1$</td>
<td>$V_4$</td>
<td></td>
<td>$V_4 = V_4 - V_1$</td>
<td>$2n - 1$ $2n - 1$</td>
</tr>
<tr>
<td>3</td>
<td>$+ V_5 + V_1$</td>
<td>$V_5$</td>
<td></td>
<td>$V_5 = V_5 + V_1$</td>
<td>$2n + 1$ $2n + 1$</td>
</tr>
<tr>
<td>4</td>
<td>$V_5 \div V_4$</td>
<td>$V_5$</td>
<td></td>
<td>$V_5 = \frac{V_5}{V_4}$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>5</td>
<td>$V_6 \div V_4$</td>
<td>$V_6$</td>
<td></td>
<td>$V_6 = \frac{V_6}{V_4}$</td>
<td>$2n - 1$ $2n - 1$</td>
</tr>
<tr>
<td>6</td>
<td>$V_7$</td>
<td>$V_7$</td>
<td></td>
<td>$V_7 = V_7 - V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>7</td>
<td>$V_8$</td>
<td>$V_8$</td>
<td></td>
<td>$V_8 = V_8 - V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>8</td>
<td>$V_9$</td>
<td>$V_9$</td>
<td></td>
<td>$V_9 = V_9 - V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>9</td>
<td>$V_10$</td>
<td>$V_10$</td>
<td></td>
<td>$V_10 = V_10 - V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>10</td>
<td>$V_11 \times V_1$</td>
<td>$V_11$</td>
<td></td>
<td>$V_11 = V_11 \times V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>11</td>
<td>$V_12$</td>
<td>$V_12$</td>
<td></td>
<td>$V_12 = V_12 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>12</td>
<td>$V_13$</td>
<td>$V_13$</td>
<td></td>
<td>$V_13 = V_13 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>13</td>
<td>$V_14$</td>
<td>$V_14$</td>
<td></td>
<td>$V_14 = V_14 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>14</td>
<td>$V_15$</td>
<td>$V_15$</td>
<td></td>
<td>$V_15 = V_15 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>15</td>
<td>$V_16$</td>
<td>$V_16$</td>
<td></td>
<td>$V_16 = V_16 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>16</td>
<td>$V_17$</td>
<td>$V_17$</td>
<td></td>
<td>$V_17 = V_17 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>17</td>
<td>$V_18$</td>
<td>$V_18$</td>
<td></td>
<td>$V_18 = V_18 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>18</td>
<td>$V_19$</td>
<td>$V_19$</td>
<td></td>
<td>$V_19 = V_19 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>19</td>
<td>$V_20$</td>
<td>$V_20$</td>
<td></td>
<td>$V_20 = V_20 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>20</td>
<td>$V_21$</td>
<td>$V_21$</td>
<td></td>
<td>$V_21 = V_21 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>21</td>
<td>$V_22$</td>
<td>$V_22$</td>
<td></td>
<td>$V_22 = V_22 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>22</td>
<td>$V_23$</td>
<td>$V_23$</td>
<td></td>
<td>$V_23 = V_23 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>23</td>
<td>$V_24$</td>
<td>$V_24$</td>
<td></td>
<td>$V_24 = V_24 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>24</td>
<td>$V_25$</td>
<td>$V_25$</td>
<td></td>
<td>$V_25 = V_25 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
<tr>
<td>25</td>
<td>$V_26$</td>
<td>$V_26$</td>
<td></td>
<td>$V_26 = V_26 + V_1$</td>
<td>$2n$ $2n$</td>
</tr>
</tbody>
</table>

Here follows a repetition of Operations thirteen to twenty-three
Kilburn/Tootill Program to compute the highest proper factor

\[ 2^{18} \text{ took 52 minutes} \]

1.5 million instructions
3.5 million store accesses
First published numerical library, 1951

First use of the word subroutine?
PART II

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.

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

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.
Quality Numerical Software

• Should be:
  – Numerically stable, with measures of quality of solution
  – Reliable and robust
  – Accompanied by test software
  – Useful and user friendly with example programs
  – Fully documented
  – Portable
  – Efficient
“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
“Giving business people spreadsheets is like giving children circular saws.

The average spreadsheet programmer does little or no advance planning, has no idea whether his or her algorithms are correct … builds in few or no cross-checks, and does little or no testing.

It is a cast iron certainty that the vast majority of spreadsheets contain errors.”

“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.”

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.
PROGRAM FOR FINDING THE LARGEST VALUE

ATTAINED BY A SET OF NUMBERS

DIMENSION A(999)

FREQUENCY 30(2,1,10), 5(100)

READ 1, N, (A(I), I = 1, N)

FORMAT (I3/(12F6.2))

BIGA = A(I)

DO 20 I = 2, N

IF (BIGA-A(I)) 10,20,20

10 BIGA = A(I)

20 CONTINUE

PRINT 2, N, BIGA

FORMAT (22H1THE LARGEST OF THESE I3, 12H NUMBERS IS F7.2)

STOP 77777
Fortran

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

- What happened to poor Ada?
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
- NAG: Chapter X02
- LAPACK: xLAMCH
### IEEE Arithmetic Formats

<table>
<thead>
<tr>
<th>Format</th>
<th>Precision</th>
<th>Exponent</th>
<th>Approx Range</th>
<th>Approx precision ((u))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>24 bits</td>
<td>8 bits</td>
<td>(10^{\pm38})</td>
<td>(10^{-8})</td>
</tr>
<tr>
<td>Double</td>
<td>53 bits</td>
<td>11 bits</td>
<td>(10^{\pm308})</td>
<td>(10^{-16})</td>
</tr>
<tr>
<td>Extended</td>
<td>(\geq 64)</td>
<td>(\geq 15)</td>
<td>(10^{\pm4932})</td>
<td>(10^{-20})</td>
</tr>
</tbody>
</table>

W. Kahan's self-portrait
T. J. Dekker, W. Hoffmann; *Algol 60 procedures in numerical algebra part 2;* MC Tracts 23, Mathematisch Centrum, Amsterdam (1968)

NAG and IMSL

- The NAG Library and IMSL were both released in 1971
- NAG was a UK University project, started in 1970, but became a not for profit company in 1976
- IMSL was commercial from the outset and is now owned by Rouge Wave
- Both Libraries have comprehensive numerical and statistical routines
Others

• Other long standing libraries are:
  – BCSLIB from Boeing, has its origins in the mid-1960s
  – HSL from Harwell (now developed and supported by RAL) began in 1963
  – SLATEC, an open source collection, has its origins in 1974. Released in 1981?
NAG 40th

- NAG is celebrating 40 years on May 13th

NAG 1970–2010

40th Anniversary

Celebrating 40 years of numerical excellence
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
ACKNOWLEDGMENTS

We are grateful for the contributions that numerous people have made to this project. The Master Test Program was developed by Lawson, with a few modifications by Hanson. The Fortran versions of the BLAS subprograms were written by Lawson, Krogh, Hanson, and J. Dongarra. The assembly coded versions for the Univac 1108 were programmed by Krogh and S. Singletary Gold. The assembly coded versions for the IBM 360/67 were programmed by Hanson and K. Haskell. The assembly coded versions for the CDC 6600 were programmed by Kincaid, J. Sullivan, and E. Williams. Four of these routines were recoded by Hanson and C. Moler. Test runs were made on a variety of machines by P. Fox and E.W. McMahon (Honeywell 6000), P. Knowlton (PDP 10), L. Fosdick (CDC 6000), C. Moler (IBM 360/67), K. Fong (CDC 7600), B. Garbow and J. Dongarra (IBM 370/195), W. Brainerd (Burroughs 6700), and others.

Helpful suggestions, based on previous similar work of their own, were given by P.S. Jensen and C. Bailey. Valuable help was also contributed by J. Wisniewski, W. MacGregor, and G. Terrell.

J. Dongarra supplied versions of several Fortran implementations of the subprograms. The choice of coding technique used by J. Dongarra is based on a set of tests that was carried out at over 40 different installations with various machines in operation. The choice of coding technique was made on the basis of superior timing performance at the largest number of these sites [8].
ALGORITHM 656
An Extended Set of Basic Linear Algebra Subprograms: Model Implementation and Test Programs

JACK J. DONGARRA
Argonne National Laboratory

JEREMY DU CROZ and SVEN HAMMARLING
Numerical Algorithms Group, Ltd.

and

RICHARD J. HANSON
Sandia National Laboratory

This paper describes a model implementation and test software for the Level 2 Basic Linear Algebra Subprograms (Level 2 BLAS). Level 2 BLAS are targeted at matrix-vector operations with the aim of providing more efficient, but portable, implementations of algorithms on high-performance computers. The model implementation provides a portable set of FORTRAN 77 Level 2 BLAS for machines where specialized implementations do not exist or are not required. The test software aims to verify that specialized implementations meet the specification of Level 2 BLAS and that implementations are correctly installed.

Categories and Subject Descriptors: F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—computations on matrices; G.1.0 [Numerical Analysis]: General—numerical algorithms; G.1.3 [Numerical Analysis]: Numerical Linear Algebra—linear systems (direct and iterative methods); G.4 [Mathematics of Computing]: Mathematical Software—certification and testing; efficiency; portability; reliability and robustness; verification

General Terms: Algorithms, Measurement, Performance, Reliability, Verification

Additional Key Words and Phrases: Extended BLAS, utilities

1. SCOPE OF THE ALGORITHM

In [7] we have defined the specification of a set of Basic Linear Subprograms for selected matrix-vector operations usually referred to as “Level 2 BLAS” or “Extended BLAS.” They provide a standard framework to develop modular, portable, and efficient FORTRAN 77 code for many computational problems in numerical linear algebra.

J. J. Dongarra’s research was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38; R. J. Hanson’s research was supported by the U.S. Department of Energy.

Authors' addresses: J. J. Dongarra, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439; J. Du Croz and S. Hammarling, Numerical Algorithms Group, Ltd., NAG Central Office, Mayfield House, 256 Banbury Road, Oxford OX2 7DE, England; R. J. Hanson, Applied Dynamics International Corp., 3800 Stone School Road, Ann Arbor, MI 48108.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1988 ACM 009-3-500/88/0300-0018 $0.50


ALGORITHM 679
A Set of Level 3 Basic Linear Algebra Subprograms: Model Implementation and Test Programs

JACK J. DONGARRA
University of Tennessee and Oak Ridge National Laboratory

JEREMY DU CROZ and SVEN HAMMARLING
Numerical Algorithms Group Ltd.

and

IAN DUFF
Harwell Laboratory

This paper describes a model implementation and test software for the Level 3 Basic Linear Algebra Subprograms (Level 3 BLAS). The Level 3 BLAS are targeted at matrix-matrix operations with the aim of providing more efficient, but portable, implementations of algorithms on high-performance computers. The model implementation provides a portable set of FORTRAN 77 Level 3 BLAS for machines where specialized implementations do not exist or are not required. The test software aims to verify that specialized implementations meet the specification of the Level 3 BLAS and that implementations are correctly installed.

Categories and Subject Descriptors: F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—computations on matrices; G.1.0 [Numerical Analysis]: General—numerical algorithms; G.1.3 [Numerical Analysis]: Numerical Linear Algebra—linear systems (direct and iterative methods); G.4 [Mathematics of Computing]: Mathematical Software—certification and testing; efficiency; portability; reliability and robustness; verification

General Terms: Algorithms, Measurement, Performance, Reliability, Verification

Additional Key Words and Phrases: Extended BLAS, utilities

1. SCOPE OF THE ALGORITHM

In [4] we have defined the specification of a set of Level 3 Basic Linear Algebra Subprograms for selected matrix-matrix operations. They provide a standard framework for developing modular, portable, and efficient Fortran 77 code for many computational problems in numerical linear algebra.

The research of J. J. Dongarra was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38.


Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1990 ACM 009-3-500/90/0300-0018 $0.50

Blas Festival 2010

Moladh na Gàidhealtachd!

Our 2010 festival will run from Friday September 3rd to Sunday September 12th inclusive.
With a stellar lineup, 2 new commissions and at least 3 exciting new venues this years' festival is lining up to be the best yet. Full details will be online soon.

Blas Festival 2009

What a great festival we had in 2009!

Thanks to our sponsors, the artistes, the promoters, the volunteers, our young performers, the Blas team, and above all, each and every person who bought a ticket and came along to one or more of our events! We bucked the credit crunch with a 25% rise in ticket sales. We sold out half of our gigs and 80% of all available seats.
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
Linpack

Users' Guide

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

Here is a guide to LINPACK—a unique collection of Fortran subroutines for analyzing and solving systems of linear algebraic equations and linear least squares problems. LINPACK is designed to be machine-independent, portable, and to operate at optimum efficiency in most cases. The package is intended for both the casual user who simply requires a library subroutine and the specialist who wishes to modify or extend the code to handle special problems. The guide supports these two groups by providing introductory sections with calling sequences and examples followed by more advanced sections spelling out the technical details. The guide should also be useful in the classroom.
LINPACK

- June 1974, ANL
  - Jim Pool’s meeting
- February 1975, ANL
  - Groundwork for project
- January 1976,
  - Funded by NSF and DOE: ANL/UNM/UM/UCSD
- Summer 1976 - Summer 1977
- Fall 1977
  - Software to 26 testsites
- December 1978
  - Software released, NESC and IMSL
Emphasis on error and condition estimates, as well as efficiency
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
Measures of Solution Quality

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
Proceedings of IMA Conference, Loughborough, 1973

Published 1974
Proceedings of IMA Conference, Sussex 1977

Published 1978
Proceedings of IFIP Conference, Boulder 1981

Published 1982

Included “Floating-point Parameters, Models and Standards by Jim Cody
Jim Cody in 1969

Sadly passed away on June 24\textsuperscript{th}, 2009
State of Libraries and Packages in 1984
Articles on:

- Observations on Mathematical Software Effort (Cody)
- LINPACK (Dongarra, Stewart)
- FUNPACK (Cody)
- EISPACK (Dongarra, Moler)
- MINPACK (Moré, Sorensen, Garbow, Hillstrom)
- Software for ODEs (Shampine, Watts)
... (cont’d)

- Sources of Information on Quadrature Software (Kahaner)
- A Survey of Sparse Matrix Software (Duff)
  - included discussion of various sparse matrix packages, including SPARSPACK, ITPACK, FFTPACK and the Harwell Library
- Mathematical Software for Elliptic Boundary Value Problems (Boisvert, Sweet)
  - included discussion of ELLPACK and FISHPACK
... (cont’d)

- The IMSL Library (Aird)
- The SLATEC Common Mathematical Library (Buzbee)
- The Boeing Mathematical Software Library (Erisman, Neves, Philips)
- The PORT Mathematical Subroutine Library (Fox)
- The Evolving NAG Library Service (Ford, Pool)
Today

• BCSLIB, IMSL, HSL and NAG are all alive
• A number of the PACKs are still available; LAPACK is probably the most active (but needs more funding!)
• SLATEC?
• PETSc
• Trilinos (maintained repository)
• ...

Big Challenge for the Future

- Multicore and hybrid chips
  - PLASMA, MAGMA, ...
- Dongarra: “There is no Moore’s law for software”

- Congratulations to Jack and all of ICL, past and present