An Introduction to the BLAS and LAPACK

Sven Hammarling
NAG Ltd, Oxford
&
University of Manchester

sven@nag.co.uk

With thanks to Jim Demmel, Jack Dongarra and the LAPACK team & contributors
Outline of Talk

• An introduction to the BLAS and LAPACK
• Motivation for the BLAS
• Efficiency of the BLAS
• Efficiency and reliability in LAPACK
• Availability of the BLAS and LAPACK
The Basic Linear Algebra Subprograms (BLAS)

- Specifications for the computational kernels that form the basic operations of numerical linear algebra
- Building blocks for higher level linear algebra
- Implemented efficiently by vendors (and others) on most machines
Discussion of linear equation solvers on the Pilot ACE (1951+):

“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)

This History of Computing in the 20th Century.
The Three Levels of BLAS

The Level 1 BLAS are concerned with scalar and vector operations, such as

\[ y \leftarrow \alpha x + y, \quad \alpha \leftarrow x^T y, \quad \alpha \leftarrow \|x\|_2 \]

the Level 2 BLAS with matrix-vector operations such as

\[ y \leftarrow \alpha Ax + \beta y, \quad x \leftarrow T^{-1}x \]

and the Level 3 BLAS with matrix-matrix operations such as

\[ C \leftarrow \alpha AB + \beta C, \quad X \leftarrow \alpha T^{-1}X \]
LAPACK

Linear Algebra PACKage for high-performance computers

- Systems of linear equations
- Linear least squares problems
- Eigenvalue and singular value problems, including generalized problems
- Matrix factorizations
- Condition and error estimates
- The BLAS as a portability layer

Dense and banded linear algebra for Shared Memory
Motivation for the BLAS
Motivation for the Level 1 BLAS

The Level 1 BLAS were intended to:

- be a design tool for the development of software in numerical linear algebra
- improve readability and aid documentation
- aid modularity and maintenance, and improve robustness of software calling the BLAS
- promote efficiency
- improve portability, without sacrificing efficiency, through standardization
Examples

CALL DAXPY ( N, ALPHA, X, INCX, Y, INCY )

\( y \leftarrow \alpha x + y \)

\( C = \text{ZDOTC} ( N, X, \text{INCX}, Y, \text{INCY} ) \)

\( c \leftarrow x^H y \)

\( \text{XNORM} = \text{DNRM2} ( N, X, \text{INCX} ) \)

\( x_{norm} \leftarrow \| x \|_2 \)
Vector Arguments

The $n$ element vector $x$ is represented by the arguments

$\text{N, X, INCX}$

$\text{N}$ - $n$

$\text{X}$ - name of the array

$\text{INCX}$ - increment or stride

$\text{INCX} \geq 0$

$$x_i \equiv X(1 + (i - 1) \times \text{INCX})$$

$\text{INCX} < 0$

$$x_i \equiv X(1 + (N - i) \times \text{INCX})$$
Examples of use of increments

REAL A( LDA, N )

jth column of A

..., A( 1, J ), 1, ...

ith row of A

..., A( I, 1 ), LDA, ...

leading diagonal of A

..., A( 1, 1 ), LDA + 1, ...
Paging

Virtual memory  Main memory

What to do? Avoid swapping pages in and out! That is, reference neighbouring data
Strategy for Paging - (Fortran)

At least arrange innermost loops to reference by column (in Fortran)

LINPACK (1978)

Block strategies can do significantly better
\[ A = U^T U - \text{Cholesky} \]

\[
\begin{pmatrix}
A_{11} & a_j & A_{13} \\
. & a_{jj} & \alpha_j^T \\
. & . & A_{33}
\end{pmatrix}
= \begin{pmatrix}
U_{11}^T & 0 & 0 \\
. & v_j^T & u_{jj} \\
U_{13}^T & u_j & U_{33}^T
\end{pmatrix}
\begin{pmatrix}
U_{11} & v_j & U_{13} \\
0 & u_{jj} & u_j^T \\
0 & 0 & U_{33}
\end{pmatrix}
\]

\[ a_j = U_{11}^T v_j \]

\[ a_{jj} = v_j^T v_j + u_{jj}^2 \]

If \( U_{11} \) has been found then

\[ U_{11}^T v_j = a_j \]

\[ u_{jj}^2 = a_{jj} - v_j^T v_j \]
Code for Cholesky factorization - Linpack routine DPOFA (using Level 1 BLAS)

DO 30 J = 1, N
   INFO = J
   S = 0.0D0
   DO 10 K = 1, J-1
      T = A(K,J) - DDOT(K-1,A(1,K),1,A(1,J),1)
      T = T/A(K,K)
      A(K,J) = T
      S = S + T*T
   10 CONTINUE
   S = A(J,J) - S
   C ......EXIT
   IF (S .LE. 0.0D0) GO TO 40
   A(J,J) = SQRT(S)
30 CONTINUE
Vector Machines

Gain their speed by pipelining and having vector instructions

Need to keep the vector registers or pipelines busy

Raise the level of granularity
Motivation for Level 2 BLAS

Level 1 BLAS are too low a level of granularity for vector machines.

They hide the matrix-vector nature of operations from the compiler.

Level 2 BLAS offer more scope for optimization.
Example

CALL DGEMV ( TRANS, M, N,
             ALPHA, A, LDA, X, INCX,
             BETA, Y, INCY )

\[ y \leftarrow \alpha Ax + \beta y \]
\[ y \leftarrow \alpha A^T x + \beta y \]

Note: TRANS is CHARACTER*1, so that calling program can use

CALL DGEMV ( 'Transpose', M, N, … )
\[ A = U^T U \] - Cholesky

\[
\begin{pmatrix}
A_{11} & a_j & A_{13} \\
. & a_{jj} & \alpha_j^T \\
. & . & A_{33}
\end{pmatrix}
= \begin{pmatrix}
U_{11}^T & 0 & 0 \\
. & v_j^T & u_{jj} \\
U_{13}^T & u_j & U_{33}^T
\end{pmatrix}
\begin{pmatrix}
U_{11} & v_j & U_{13} \\
0 & u_{jj} & u_j^T \\
0 & 0 & U_{33}
\end{pmatrix}
\]

\[ a_j = U_{11}^T v_j \]

\[ a_{jj} = v_j^T v_j + u_{jj}^2 \]

If \( U_{11} \) has been found then

\[ U_{11}^T v_j = a_j \]

\[ u_{jj}^2 = a_{jj} - v_j^T v_j \]

(_TRSV)

(_DOT + square root)
Code for Cholesky factorization
LAPACK-style (using Level 2 BLAS)

DO 10 J = 1, N
   CALL DTRSV('Upper','Transpose','Non-unit',
   J –1,A,LDA,A(1,J),1)
   S = A(J,J) – DDOT(J –1,A(1,J),1,A(1,J),1)
   IF (S .LE. ZERO) GO TO 20
   A(J,J) = SQRT(S)
10  CONTINUE
Motivation for Level 3 BLAS

Level 2 BLAS are too low a level of granularity for many hierarchical memory and parallel machines.

Matrix-matrix operations allow:
- full re-use of cache and/or local memory
- vector and scalar operations to be performed in parallel within blocks
- parallel operations on distinct blocks
Performance Improvements

Efficiency in LAPACK has been principally achieved by restructuring algorithms so that as much computation as possible is performed by calls to:

Level 3 and Level 2 BLAS

In order to exploit the Level 3 BLAS we needed:

**BLOCK-PARTITIONED ALGORITHMS**
$$A = U^T U \, - \, \text{Block Cholesky}$$

$$\begin{pmatrix}
A_{11} & A_{12} & A_{13} \\
. & A_{22} & A_{23} \\
. & . & A_{33}
\end{pmatrix}
= \begin{pmatrix}
U_{11}^T & 0 & 0 \\
U_{12}^T & U_{22}^T & 0 \\
U_{13}^T & U_{23}^T & U_{33}^T
\end{pmatrix}
\begin{pmatrix}
U_{11} & U_{12} & U_{13} \\
0 & U_{22} & U_{23} \\
0 & 0 & U_{33}
\end{pmatrix}$$

$$A_{12} = U_{11}^T U_{12}$$

$$A_{22} = U_{12}^T U_{12} + U_{22}^T U_{22}$$

$$U_{11}^T U_{12} = A_{12}$$

$$U_{22}^T U_{22} = A_{22} - U_{12}^T U_{12}$$

(_TRSM)

(_SYRK + point Cholesky)
Code for Cholesky factorization
LAPACK-style (using Level 3 BLAS)

DO 10 J = 1, N, NB
   JB = MIN(NB,N-J+1)
   CALL DTRSM('Left','Upper','Transpose','Non-unit',
               J-1,JB,ONE,A,LDA,A(1,J),LDA)
   CALL DSYRK('Upper','Transpose',JB,J-1,-ONE,
              A(1,J),LDA,ONE,A(J,J),LDA)
   CALL DPOTF2(JB,A(J,J),LDA,INFO)
   IF (INFO .NE. 0) GO TO 20
10 CONTINUE
Example

CALL DGEMM (TRANSA, TRANSB, M, N, K, ALPHA, A, LDA, B, LDB, BETA, C, LDC )

\[ C \leftarrow \alpha AB + \beta C \]

\[ C \leftarrow \alpha A^T B + \beta C \]

\[ C \leftarrow \alpha AB^T + \beta C \]

\[ C \leftarrow \alpha A^T B^T + \beta C \]

C is always m by n
Efficiency of the BLAS
### Why Higher Level BLAS?

<table>
<thead>
<tr>
<th>BLAS</th>
<th>Mem Refs</th>
<th>Ops</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>$3n$</td>
<td>$2n$</td>
<td>$3 : 2$</td>
</tr>
<tr>
<td>$y \leftarrow \alpha x + y$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level 2</td>
<td>$n^2$</td>
<td>$2n^2$</td>
<td>$1 : 2$</td>
</tr>
<tr>
<td>$y \leftarrow \alpha Ax + \beta y$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level 3</td>
<td>$4n^2$</td>
<td>$2n^3$</td>
<td>$2 : n$</td>
</tr>
<tr>
<td>$C \leftarrow \alpha AB + \beta C$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The so-called ‘surface to volume’ effect

Key to high performance is effective use of memory hierarchy. True on all architectures.
IBM RS/6000-590 (66 MHz, 264 Mflop/s Peak)

- Level 3 BLAS
- Level 2 BLAS
- Level 1 BLAS

Order of vector/Matrices

Mflop/s
Data Movement

“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
Availability of the BLAS

Standard portable versions of the **BLAS** are available from

[www.netlib.org/blas/index.html](http://www.netlib.org/blas/index.html)

together with test and timing routines

The **BLAS** are included in the F06 Chapter of the NAG Fortran Library

Vendors supply efficient versions of the **BLAS** for most machines
AMD Core Maths Library (ACML) from AMD:

developer.amd.com/acml.aspx

Intel Maths Kernel Library (MKL):

Availability of the BLAS (cont’d)

Efforts to aid users and vendors include:

• ATLAS (Automatically Tuned Linear Algebra Subprograms)
  math-atlas.sourceforge.net

• GEMM based BLAS
  www.cs.umu.se/research/parallel/blas/
To consider the BLAS in the light of modern software, language, and hardware developments

Main chapters:

- Introduction
- Dense and Banded BLAS
- Sparse BLAS
- Extended and Mixed Precision BLAS
- Appendix: C interface to the Legacy BLAS

Specifications for F77, F95 and C
More on LAPACK
LAPACK

Users’

Guide

E. Anderson, Z. Bai, C. Bischof
S. Blackford, J. Demmel, J. Dongarra
J. Du Croz, A. Greenbaum, S. Hammarling
A. McKenney, D. Sorensen

SIAM, 1999 (3rd Edition)

Proceeds to SIAM student travel fund

www.netlib.org/lapack/lug/lapack_lug.html
“In Addition to providing faster routines than previously available, LAPACK provides more comprehensive and better error bounds. Our goal is to provide error bounds for most quantities computed by LAPACK.”

LAPACK Users’ Guide, Chapter 4 – Accuracy and Stability
Error Bounds in LAPACK: 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
Availability of LAPACK

LAPACK is freely available via netlib:

www.netlib.org/lapack/index.html

Much of LAPACK is included in the NAG Fortran 77 Library and is the basis of the dense linear algebra in the NAG Fortran 90 Library.

Tuned versions of some LAPACK routines are in the NAG Fortran SMP Library.

MATLAB uses LAPACK as of Version 6.
ACML from AMD (free download):
developer.amd.com/acml.aspx

Intel Maths Kernel Library (MKL):

NAG math libraries:
www.nag.co.uk

Many distributions, including Cygwin, have LAPACK (and the BLAS), usually as an ‘extra’
LAPACK 95

- LAPACK 95 is a Fortran 95 interface to LAPACK
- LAPACK 95 contains the functionality of the LAPACK drivers and computational routines
- Makes use of assumed shape arrays, optional arguments, and generic interfaces
LAPACK 95

Users’ Guide

L. Susan Blackford, Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, Minka Marinova, Jerzy Wasniewski, Plamen Yalamov

SIAM, 2001
Proceeds to SIAM student travel fund

www.netlib.org/lapack95/lug95/
There is also a package available for distributed memory parallel machines called ScaLAPACK, which is based upon LAPACK.

www.netlib.org/scalapack/scalapack_home.html

Much of ScaLAPACK is included in the NAG Parallel Library.
ScaLAPACK Users’ Guide

- Susan Blackford, UT
- Jaeyoung Choi, Soongsil U
- Andy Cleary, LLNL
- Ed D’Azevedo, ORNL
- Jim Demmel, UC-B
- Inder Dhillon, UC-B
- Jack Dongarra, UT/ORNL
- Sven Hammarling, NAG
- Greg Henry, Intel
- Antoine Petitet, UT
- Ken Stanley, UC-B
- David Walker, ORNL
- Clint Whaley, UT

SIAM, July ‘97
Proceeds to Student Travel Fund

http://www.netlib.org/scalapack(slug)/scalapack_slug.html
LAPACK Release 3.1

- 3.1.0 released on 12 November 2006
  - Improved numerical algorithms
  - Some further functionality
  - Thread safety
  - Bug fixes (e.g. LAWN 176)

- See: www.netlib.org/lapack/lapack-3.1.0.changes

- 3.1.1 released on 26 February 2007
  - Mainly bug fixes (principally to test software)

- See: www.netlib.org/lapack/lapack-3.1.1.changes
LAPACK Release 3.1 (cont’d)

- CLAPACK 3.1.1 released on 18th February, 2008
  - Produced using f2c, with some tidying of the interface
LAPACK Development

Further releases planned (NSF support)

LAWN 164: NSF Proposal
LAWN 181: Prospectus for the Next
LAPACK and ScaLAPACK Libraries
(February ’07)

LAWN – LAPACK Working Note
www.netlib.org/lapack/lawns/downloads
Using LAPACK
LAPACK Structure

• Driver routines to solve high level problems, such as linear equations and eigenvalue problems
• Computational routines that are used by the driver routines, such as Cholesky factorization and QR factorization
• Auxiliary routines for low level tasks, such as generating a Householder transformation (listed, but not documented in the Users’ Guide)
LAPACK Example

xGELS solves linear least squares problems:

\[ \min_{x} \| b_j - Ax_j \|_2 \quad \text{or} \quad \min_{x} \| b_j - A^H x_j \|_2 , \quad j = 1, 2, \ldots, r \]

where \( A \) is an \( m \) by \( n \) matrix of full rank, usually with \( m \geq n \) so that \( \text{rank}(A) = n \), and \( b_j \) and \( x_j \) are vectors

\( x \) can be one of S (single), D (double), C (complex) or Z (double complex, COMPLEX*16)
LAPACK Example (cont’d)

SUBROUTINE xGELS( TRANS, M, N, NRHS, A, LDA, B, LDB, WORK, LWORK, INFO )

TRANS – ‘N’ or ‘T’ (real) or ‘C’ (complex)
NRHS – r
LDA, LDB – Leading dimensions of A and B
WORK – Workspace of dimension LWORK
INFO – Error argument
LAPACK INFO Argument

- INFO < 0 : the (-INFO)th argument was incorrectly supplied, computation aborts with an error message
- INFO = 0 : No errors detected
- INFO > 0 : computational failure or warning (e.g. matrix was singular or nearly singular for a linear equation solver)
- It is generally essential to test INFO on exit from the routine
LAPACK INFO Argument (cont’d)

- Error handling when INFO < 0 is performed by LAPACK routine XERBLA (which is also used by the BLAS)
- Some vendors provide their own version of XERBLA
- Users may provide their own version of XERBLA
Workspace Arguments

- The array WORK is used for workspace (also RWORK and IWORK)
- Sufficient workspace must be supplied
- The optimal workspace required is returned in WORK(1)
- Some routines have the associated argument LWORK for the dimension of WORK
Workspace

- For many routines the workspace depends upon the block size (NB)
- NB is determined by the LAPACK routine ILAENV
- If in doubt, assume NB = 64
- ILAENV also returns other information, such as cross-over points
Workspace Arguments (cont’d)

• When LWORK is present it can be set to -1 for a workspace query, in which case no computation is performed, except to compute the optimal workspace (returned in WORK(1))

• Especially useful when calling LAPACK from languages with dynamic memory allocation
LAPACK Storage Schemes

- General matrices
- Triangular, symmetric and Hermitian matrices
- Band matrices
- Packed triangular, symmetric and Hermitian matrices
- Tridiagonal and bidiagonal matrices, represented by vectors
Triangular, Symmetric and Hermitian Matrices

- Only the upper or lower triangle need be supplied, the remainder of the array is not referenced.
- For Hermitian matrices, the imaginary part of the diagonal elements are not referenced and, for output arguments, are set to zero.
Option Arguments

- **UPLO** for triangular, symmetric and Hermitian matrices
  - ‘U’ upper triangle of A is stored
  - ‘L’ lower triangle of A is stored

- **DIAG** for triangular matrices
  - ‘U’ matrix is unit triangular
  - ‘N’ matrix is not unit triangular
Conventional Storage
(UPLO = ‘U’)

<table>
<thead>
<tr>
<th>Hermitian matrix $A$</th>
<th>Storage in array $A$</th>
</tr>
</thead>
</table>
| $\begin{pmatrix}
    a_{11} & a_{12} & a_{13} & a_{14} \\
    \bar{a}_{12} & a_{22} & a_{23} & a_{24} \\
    \bar{a}_{13} & \bar{a}_{23} & a_{33} & a_{43} \\
    \bar{a}_{14} & \bar{a}_{24} & \bar{a}_{34} & a_{44}
\end{pmatrix}$ | $\begin{pmatrix}
    a_{11} & a_{12} & a_{13} & a_{14} \\
    * & a_{22} & a_{23} & a_{24} \\
    * & * & a_{33} & a_{34} \\
    * & * & * & a_{44}
\end{pmatrix}$ |
**Conventional Storage**
*(UPLO = ‘L’)*

<table>
<thead>
<tr>
<th>Hermitian matrix $A$</th>
<th>Storage in array $A$</th>
</tr>
</thead>
</table>
| $\begin{pmatrix}
  a_{11} & \bar{a}_{21} & \bar{a}_{31} & \bar{a}_{41} \\
  a_{21} & a_{22} & \bar{a}_{32} & \bar{a}_{42} \\
  a_{31} & a_{32} & a_{33} & \bar{a}_{34} \\
  a_{41} & a_{42} & a_{34} & a_{44}
\end{pmatrix}$ | $\begin{pmatrix}
  a_{11} & * & * & * \\
  a_{21} & a_{22} & * & * \\
  a_{31} & a_{32} & a_{33} & * \\
  a_{41} & a_{42} & a_{43} & a_{44}
\end{pmatrix}$ |
# Band Storage

<table>
<thead>
<tr>
<th></th>
<th>Band matrix $A$</th>
<th>Band storage in array $AB$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$a_{11}$</td>
<td>$*$</td>
</tr>
<tr>
<td></td>
<td>$a_{12}$</td>
<td>$a_{12}$</td>
</tr>
<tr>
<td></td>
<td>$a_{12}$</td>
<td>$a_{23}$</td>
</tr>
<tr>
<td></td>
<td>$a_{21}$</td>
<td>$a_{21}$</td>
</tr>
<tr>
<td></td>
<td>$a_{22}$</td>
<td>$a_{22}$</td>
</tr>
<tr>
<td></td>
<td>$a_{23}$</td>
<td>$a_{23}$</td>
</tr>
<tr>
<td></td>
<td>$a_{23}$</td>
<td>$a_{33}$</td>
</tr>
<tr>
<td></td>
<td>$a_{31}$</td>
<td>$a_{31}$</td>
</tr>
<tr>
<td></td>
<td>$a_{32}$</td>
<td>$a_{32}$</td>
</tr>
<tr>
<td></td>
<td>$a_{33}$</td>
<td>$a_{33}$</td>
</tr>
<tr>
<td></td>
<td>$a_{34}$</td>
<td>$a_{34}$</td>
</tr>
<tr>
<td></td>
<td>$a_{42}$</td>
<td>$a_{42}$</td>
</tr>
<tr>
<td></td>
<td>$a_{43}$</td>
<td>$a_{43}$</td>
</tr>
<tr>
<td></td>
<td>$a_{44}$</td>
<td>$a_{44}$</td>
</tr>
<tr>
<td></td>
<td>$a_{45}$</td>
<td>$a_{45}$</td>
</tr>
<tr>
<td></td>
<td>$a_{53}$</td>
<td>$a_{53}$</td>
</tr>
<tr>
<td></td>
<td>$a_{54}$</td>
<td>$a_{54}$</td>
</tr>
<tr>
<td></td>
<td>$a_{55}$</td>
<td>$a_{55}$</td>
</tr>
<tr>
<td></td>
<td>$a_{55}$</td>
<td>$*$</td>
</tr>
<tr>
<td></td>
<td>$a_{55}$</td>
<td>$*$</td>
</tr>
</tbody>
</table>
Hermitian Band Storage
(UPLO = ‘U’)

<table>
<thead>
<tr>
<th>Hermitian band matrix $A$</th>
<th>Band storage in array $AB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{11}$ $a_{12}$ $a_{13}$</td>
<td>$<em>$ $</em>$ $a_{13}$ $a_{24}$ $a_{35}$</td>
</tr>
<tr>
<td>$\bar{a}<em>{12}$ $a</em>{22}$ $a_{23}$ $a_{24}$</td>
<td>$*$ $a_{12}$ $a_{23}$ $a_{34}$ $a_{45}$</td>
</tr>
<tr>
<td>$\bar{a}<em>{13}$ $\bar{a}</em>{23}$ $a_{33}$ $a_{34}$ $a_{35}$</td>
<td>$a_{11}$ $a_{22}$ $a_{33}$ $a_{44}$ $a_{55}$</td>
</tr>
<tr>
<td>$\bar{a}<em>{14}$ $\bar{a}</em>{34}$ $a_{44}$ $a_{45}$</td>
<td></td>
</tr>
<tr>
<td>$\bar{a}<em>{15}$ $\bar{a}</em>{45}$ $a_{55}$</td>
<td></td>
</tr>
</tbody>
</table>
### Hermitian Band Storage

**(UPLO = ‘L’)\)**

<table>
<thead>
<tr>
<th>Hermitian band matrix $A$</th>
<th>Band storage in array $AB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{11}$, $\overline{a}<em>{21}$, $\overline{a}</em>{31}$</td>
<td>$a_{11}$, $a_{22}$, $a_{33}$, $a_{44}$, $a_{55}$</td>
</tr>
<tr>
<td>$a_{21}$, $a_{22}$, $\overline{a}<em>{32}$, $\overline{a}</em>{42}$</td>
<td>$a_{21}$, $a_{32}$, $a_{43}$, $a_{54}$, $*$</td>
</tr>
<tr>
<td>$a_{31}$, $a_{32}$, $a_{33}$, $\overline{a}<em>{43}$, $\overline{a}</em>{53}$</td>
<td>$a_{31}$, $a_{42}$, $a_{53}$, $<em>$, $</em>$</td>
</tr>
<tr>
<td>$a_{42}$, $a_{43}$, $a_{44}$, $\overline{a}_{54}$</td>
<td></td>
</tr>
<tr>
<td>$a_{53}$, $a_{54}$, $a_{55}$</td>
<td></td>
</tr>
</tbody>
</table>

*Note: The Hermitian band matrix $A$ is stored in band storage in array $AB$, with the elements above and below the diagonal being stored in reverse order.*
# Packed Storage (UPLO = ‘U’)

<table>
<thead>
<tr>
<th>Triangular matrix $A$</th>
<th>Packed storage in array $AP$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| $\begin{pmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} \\
  a_{22} & a_{23} & a_{24} \\
  a_{33} & a_{34} \\
  a_{44}
\end{pmatrix}$ | $\begin{pmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} \\
  a_{22} & a_{23} & a_{33} \\
  a_{34} & a_{44}
\end{pmatrix}$ |
# Packed Storage (UPLO = ‘L’)

<table>
<thead>
<tr>
<th>Triangular matrix $A$</th>
<th>Packed storage in array $AP$</th>
</tr>
</thead>
</table>
| $\begin{pmatrix}
  a_{11} \\
  a_{21} & a_{22} \\
  a_{31} & a_{32} & a_{33} \\
  a_{41} & a_{42} & a_{34} & a_{44}
\end{pmatrix}$ | $a_{11} a_{21} a_{31} a_{41} a_{22} a_{32} a_{42} a_{33} a_{43} a_{44}$ |
Further Information

• Example programs for the LAPACK drivers can be found at:
  www.nag.co.uk/lapack-ex/lapack-ex.html

• See also the NAG documentation for Chapters F06, F07 and F08:
  www.nag.co.uk/numeric/FL/FLdocumentation.asp
Further Information (cont’d)

- The LAPACK forum is at: icl.cs.utk.edu/lapack-forum/
- The main LAPACK page is: www.netlib.org/lapack/
- The main ScaLAPACK page is: www.netlib.org/scalapack/
Final Comments
Benefits of LAPACK

- Comprehensive and consistent set of routines for dense and banded linear algebra
- Efficient on a wide range of modern computers
- Additional functionality compared to previous packages
- Improved numerical algorithms
- Error and sensitivity estimates
- Portability layer between users applications and the machines
- LAPACK is an ongoing project
BLAS

- Efficient implementations of the BLAS were essential to the success of LAPACK.
- There have been successful projects to aid the development of efficient, machine versions of the BLAS, e.g. ATLAS; the GEMM based BLAS, recursive BLAS.
Hierarchy of Building Blocks

- Users
- Applications Packages
- LAPACK
- BLAS
- Machines
References - BLAS


References - LAPACK


References - General


