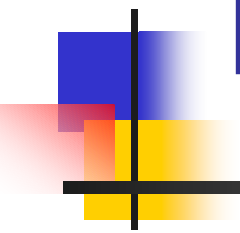
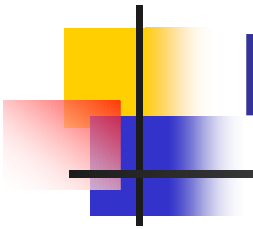


LAPACK

Linear Algebra PACKage

Janice Giudice
David Knezevic

1



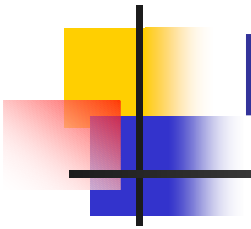
Motivating Question

Recalling from last week…

- Level 1 BLAS: vectors ops
- Level 2 BLAS: matrix-vectors ops
  - $O(n^2)$ flops on $O(n^2)$ data
- Level 3 BLAS: matrix-matrix ops
  - $O(n^3)$ flops on $O(n^2)$ data

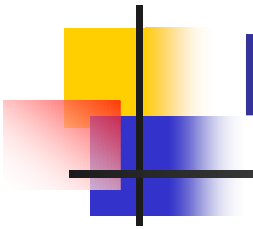
Maximize ratio flops to memory refs.

- Can we provide **portable** software for computations in dense linear algebra that is **efficient** on a wide range of modern high-performance computers?



LINPACK and EISPACK

- LINPACK (1970s and early 1980s) is a numerical subroutine library for solving linear equations, least-squares problems, and for finding singular values written in Fortran.
- EISPACK (1972-1973) is a collection of double-precision Fortran subroutines that compute the eigenvalues and eigenvectors of several classes of matrices.
- MATLAB started its life in the late 1970s as an interactive calculator built on top of LINPACK and EISPACK, which were then state-of-the-art Fortran subroutine libraries for matrix computation.
- LINPACK and EISPACK **portable**... but **inefficient**, why? ...



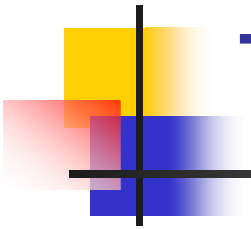
LINPACK and EISPACK

- … Because their memory access patterns disregard the multi-layered memory hierarchies of the machines, (recall Stef's mountain) thereby spending too much time moving data instead of doing useful floating-point operations…

- i.e. They are based on the vector operation kernels of the Level 1 BLAS

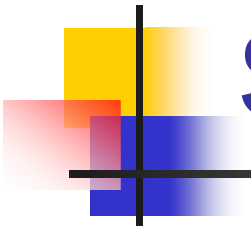
What is LAPACK?

- LAPACK, designed to supersede LINPACK and EISPACK, addresses inefficiency by reorganizing the algorithms to use block matrix operations in the innermost loops, i.e. max calls to LEVEL 3 BLAS
  - efficiency, functionality, algorithms
  - vectorization, data movement, parallelism
- LAPACK is a portable, modern (1990) library of Fortran77 subroutines for solving the most commonly occurring problems in numerical linear algebra.
  - linear systems of equations
  - eigenvalues and eigenvectors
  - linear least squares
  - singular value decomposition
- Freely available on *netlib*
  - Pre-built libraries for a variety of architectures



The main point is…

The (higher Level) BLAS are the building blocks of LAPACK.



Structure of LAPACK

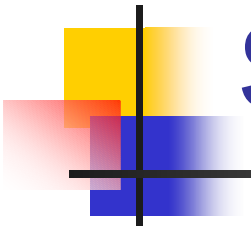
- **Driver routines:**
- **Computational routines:**
- **Auxiliary routines:**

Structure of LAPACK

- **Driver routines:**
  - Solve standard types of problems.
    - *Linear systems.*
- **Computational routines:**
- **Auxiliary routines:**

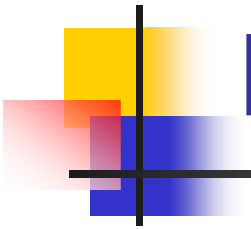
Structure of LAPACK

- **Driver routines:**
  - Solve standard types of problems.
    - *Linear systems.*
- **Computational routines:**
  - Performs a distinct computational task.
    - *LU factorisation.*
- **Auxiliary routines:**



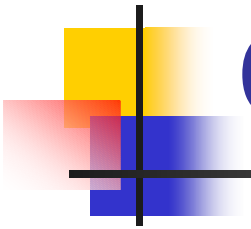
Structure of LAPACK

- **Driver routines:**
  - Solve standard types of problems.
    - *Linear systems.*
- **Computational routines:**
  - Performs a distinct computational task.
    - *LU factorisation.*
- **Auxiliary routines:**
  - Subtask or common low-level computation.
    - Matrix norm.



DRIVER ROUTINES

- systems of linear equations
- least-squares solutions of linear systems
  - overdetermined
  - underdetermined
  - rank deficient
- eigenvalue problems
  - symmetric
  - nonsymmetric
- singular value problems



COMPUTATIONAL ROUTINES

- matrix factorizations
  - LU
  - Cholesky
  - QR (many variants; generalised, implicitly shifted, root free...)
  - LQ
  - SVD
  - Schur
  - generalized Schur
- back subsitution
- condition number estimation
- error bound computation

COMPUTATIONAL ROUTINES

- eigenvalues and vectors
  - Cuppen's divide and conquer
  - relatively robust representaion (RRR)
  - bisection
  - inverse iteration
- Matrix balancing
- invariant subspaces & condition numbers
  - solve Sylvester matrix equation
  - condition numbers of eigenval/vecs of a matrix in Schur form
  - move eigenvalues in matrix of Schur form

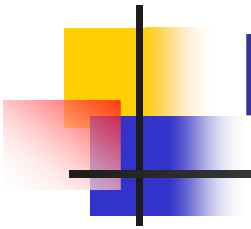
Data Types and Precision

- Real
- Complex

} Most computations have matching routines

- Single
- Double

} All computations have matching routines



Matrix Types

- Dense
- Banded

But…

- not general sparse matrices

Routine Naming

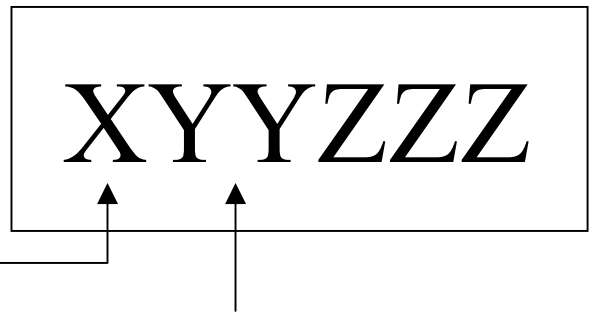
XYYZZZ

Routine Naming

Data Type

- S single
- D double
- C complex
- Z double complex

Routine Naming



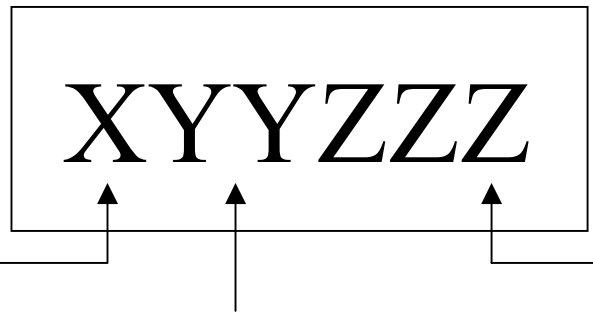
Data Type

- S single
- D double
- C complex
- Z double complex

Matrix Type

- DI diagonal
- BD bidiagonal
- GE general
- OR orthogonal

Routine Naming



Data Type

- S single
- D double
- C complex
- Z double complex

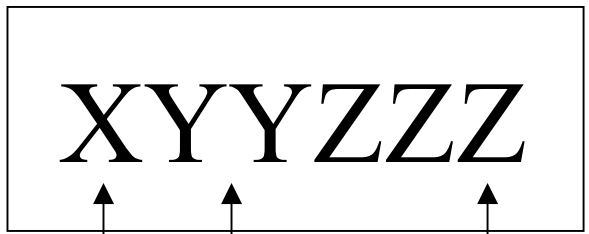
Matrix Type

- DI diagonal
- BD bidiagonal
- GE general
- OR orthogonal

Computation

- SVD singular value decomp
- LLS linear least squares
- CON condition number
- TRF factorize

Routine Naming



Data Type

- S single
- D double
- C complex
- Z double complex

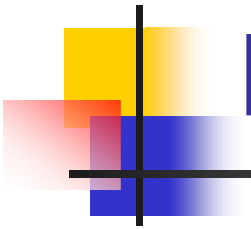
Matrix Type

- DI diagonal
- BD bidiagonal
- GE general
- OR orthogonal

Computation

- SVD singular value decomp
- LLS linear least squares
- CON condition number
- TRF factorize

DGESVD

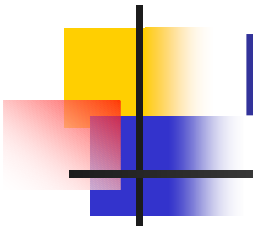


Driver Routines

- Simple
  - Performs the basic operation requested
    - solve a complete problem, like finding the eigenvalues of a matrix or solving a set of linear equations

- Expert (storage x 2 simple)
  - Additional options

    e.g. condition number, checks for near-singularity, refinement, error bounds.

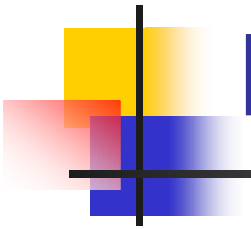


Documentation

- Just like the BLAS routines, the LAPACK routines are virtually self-explanatory. Details of the input and output parameters for any given subroutine are contained in the header section of the file.

David now takes you through the handout….

- `SUBROUTINE SGESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )`

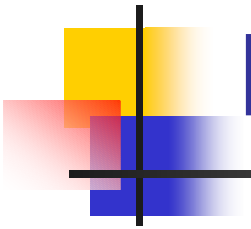


Benchmarks

- Linear Equations
  - Dense
  - Tridiagonal
- Linear Least Squares
  - Overdetermined
- Singular Value Decomposition
  - For square matrices
- Eigenproblem
  - Nonsymmetric

Benchmarks

- Linear Equations
  - Dense — DGESV
  - Tridiagonal — DGTSV
- Linear Least Squares
  - Overdetermined — DGELS
- Singular Value Decomposition
  - For square matrices — DGESVD
- Eigenproblem
  - Nonsymmetric — DGEEV



Benchmarks

Linear Equations		
Dense	DGESV	`backslash` (LU)
Tridiagonal `backslash`	DGTSV	`sparse`
Linear Least Squares		
Overdetermined	DGELS	`qr`
Singular Value Decomposition		
For square matrices	DGESVD	`svd`
Eigenproblem		
Nonsymmetric	DGEEV	`eig`

Benchmark Procedure

```
for k = 2 : maxdim
        n = k^2
        loops = 0
        start timing
        while time < maxtime
              loops = loops + 1
              CALL PROCEDURE (n)
        end
        T(k) = time/loops
end
```

Linear Equations

$$\mathbf{Ax = b}$$

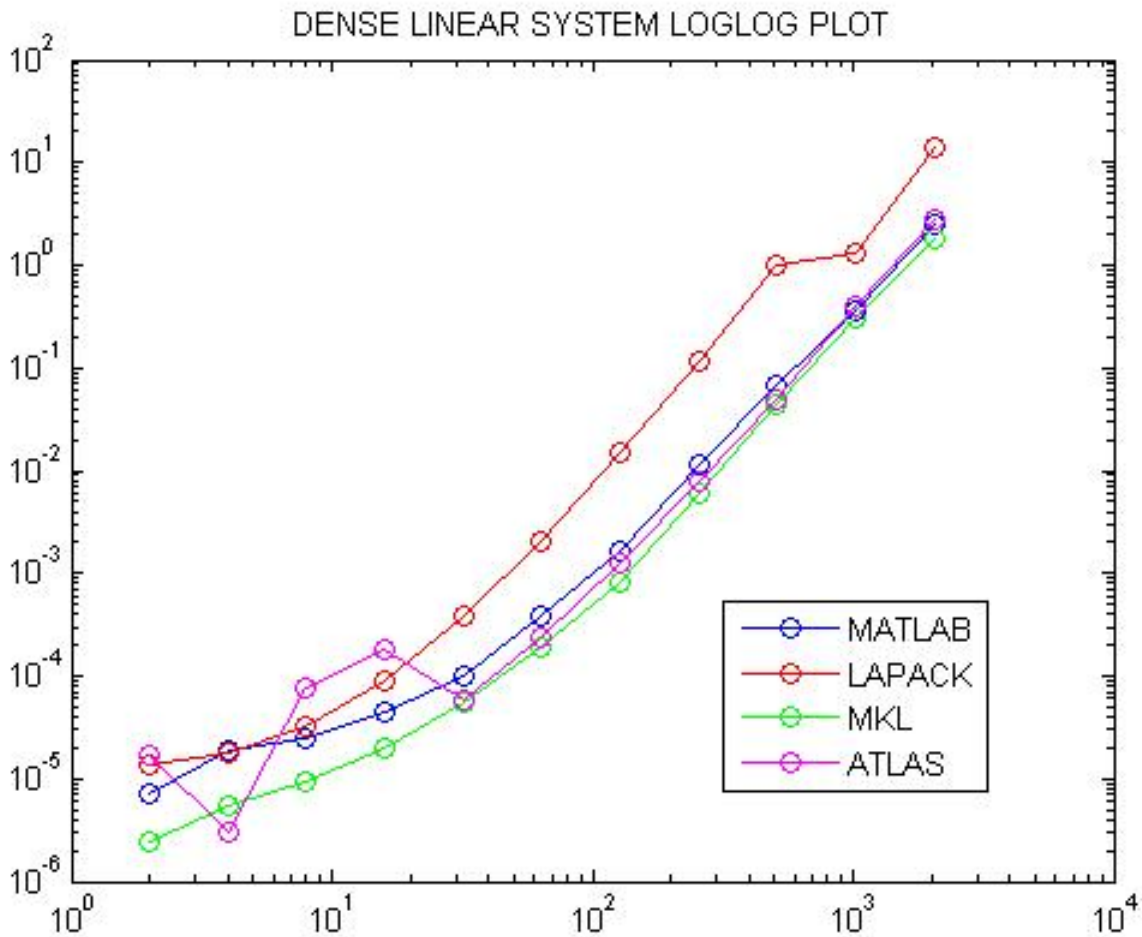
A is an *n*-by-*n* matrix,

b is an *n*-by-*1* vector,

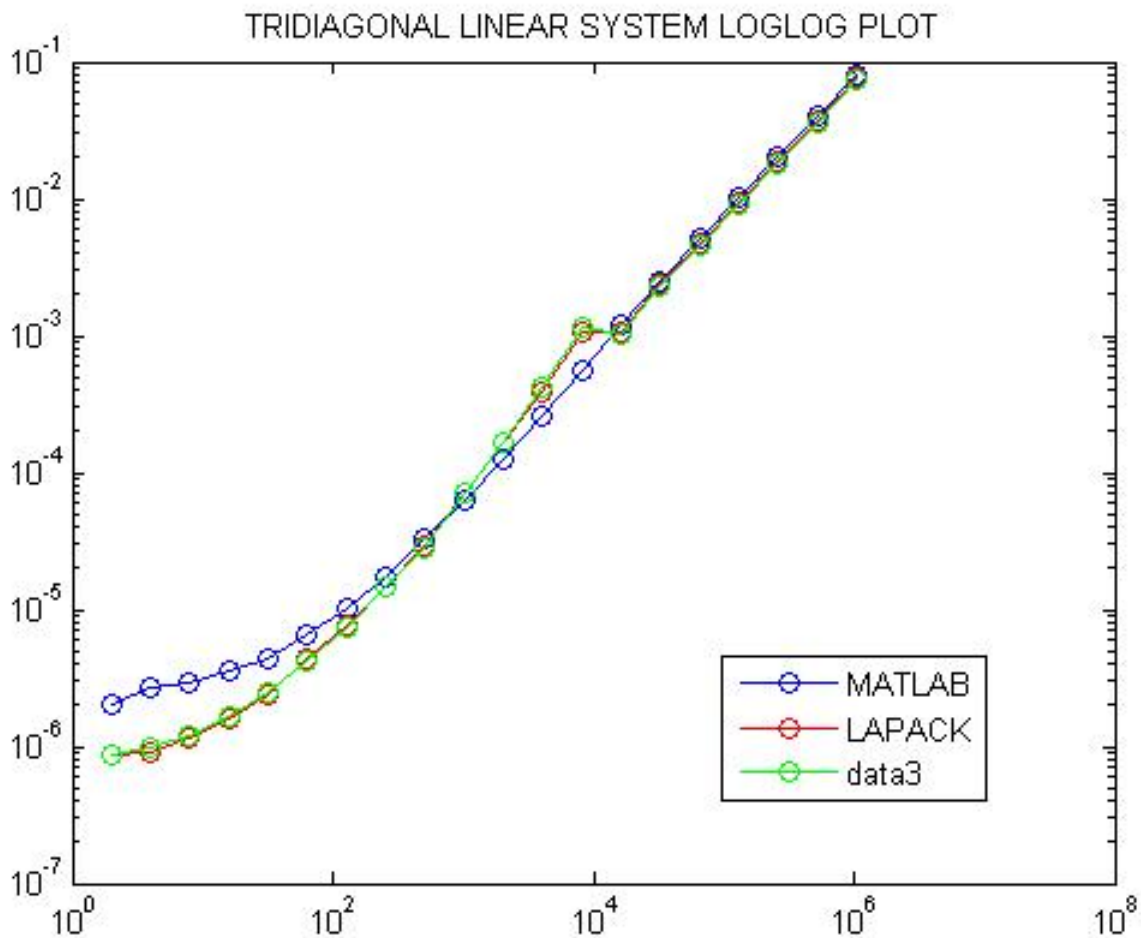
x is an *n*-by-*1* vector.

DGESV – LU factorisation

DGESV RESULTS



DGTSV RESULTS



Linear Least Squares

$$\min_x \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2$$

A is an *m*-by-*n* matrix,

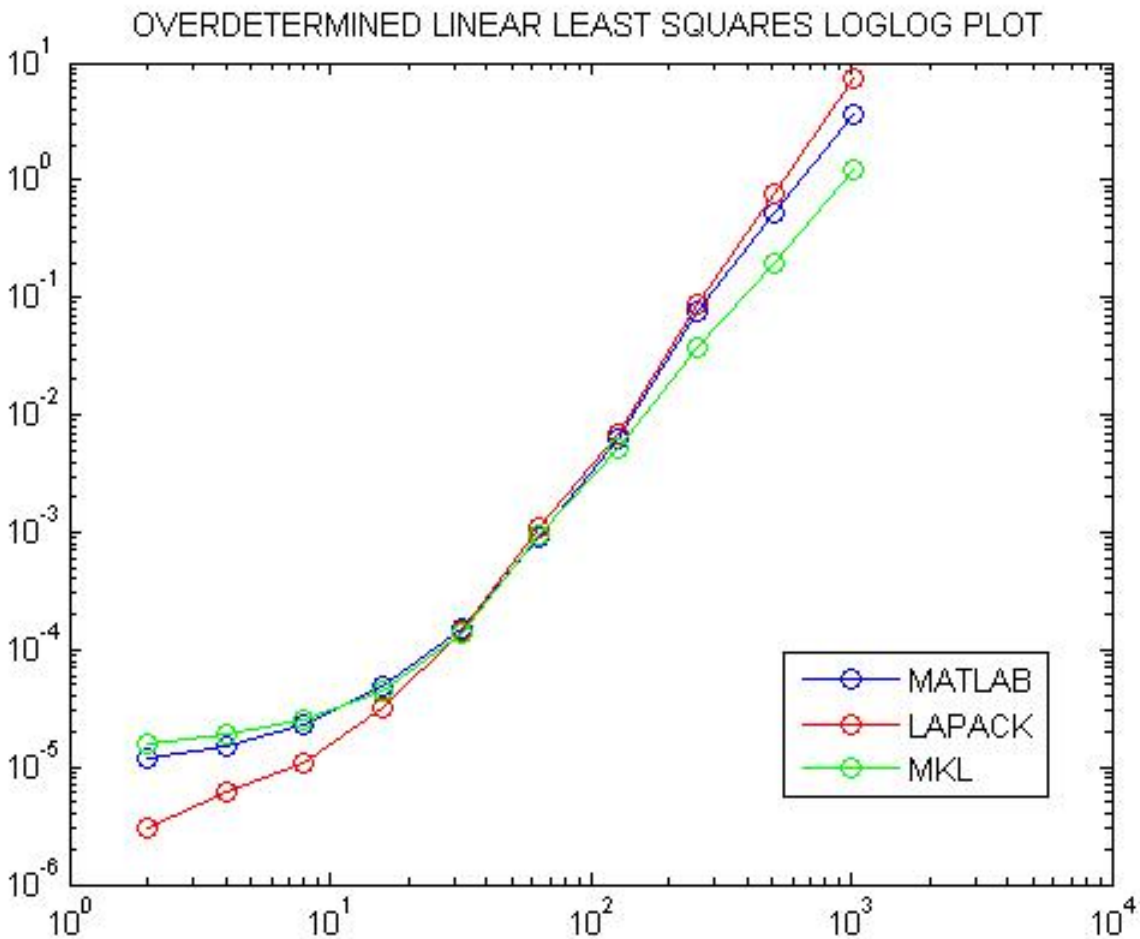
b is an *m*-by-*1* vector,

x is an *n*-by-*1* vector.

When *m* >= *n*, rank(**A**) = *n*, overdetermined system.

DGELS – QR or LQ factorisation

DGELS RESULTS



Singular Value Decomposition

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$$

A is an *m*-by-*n* matrix,

U is an *m*-by-*m* orthogonal matrix,

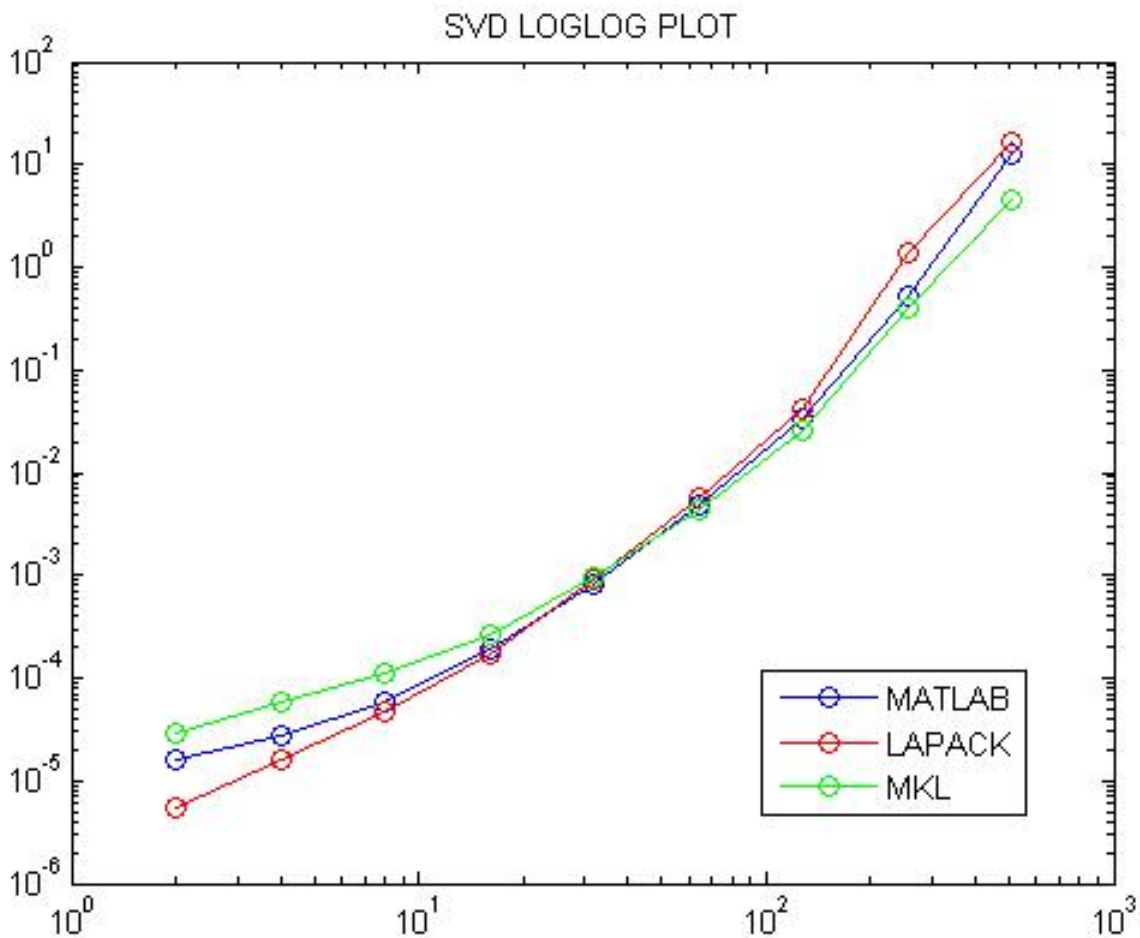
V is an *n*-by-*n* orthogonal matrix.

Σ is an *m*-by-*n* diagonal matrix containing the singular values of A

Let $m = n$

DGESVD

DGESVD RESULTS



Nonsymmetric Eigenproblem

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$$

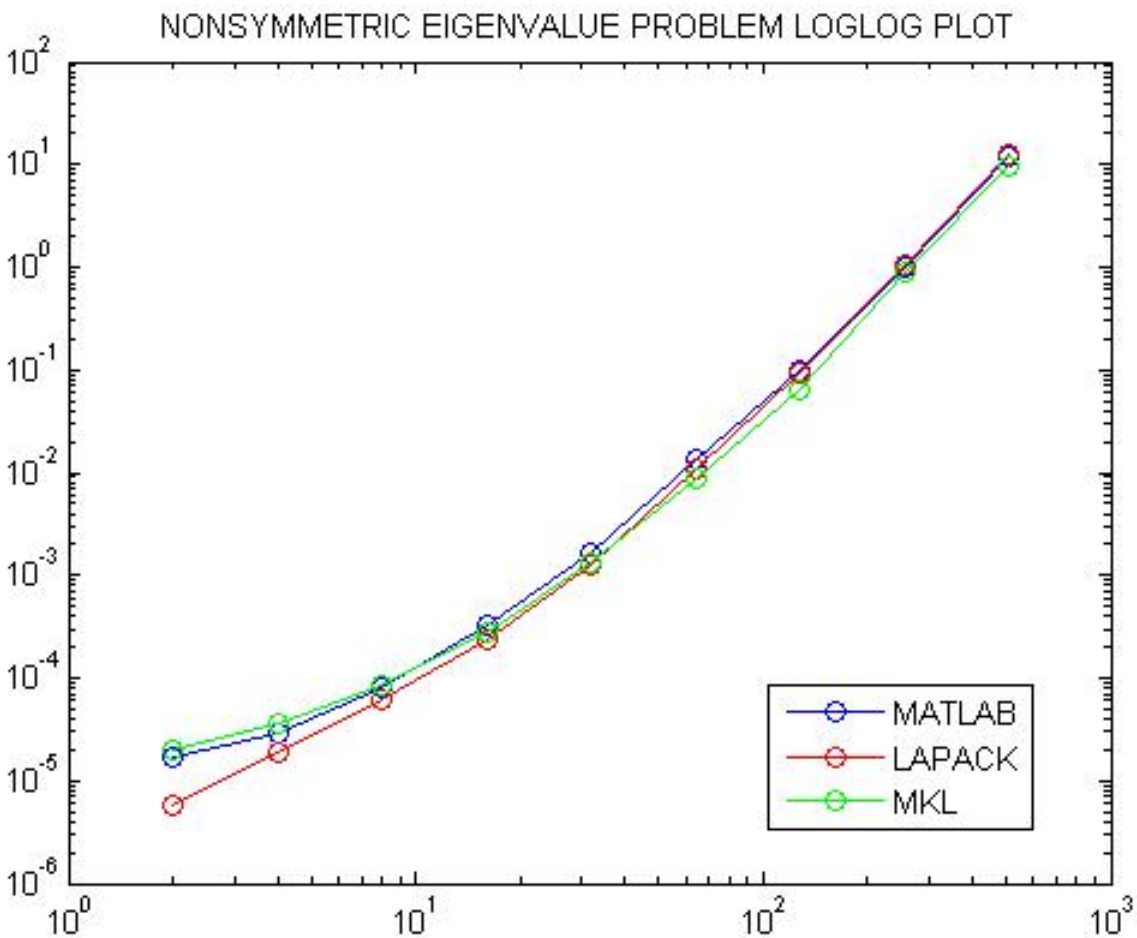
A is an *m*-by-*n* matrix,

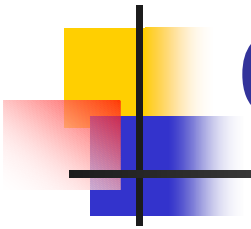
u is an *m* vector; an eigenvector,

λ are the eigenvalues.

DGEEV

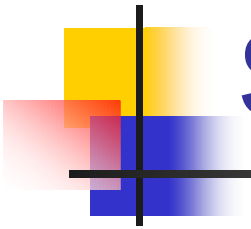
DGEEV RESULTS





Computer Suitability

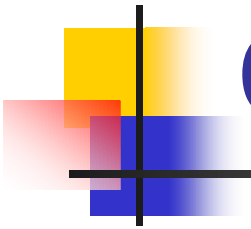
- Designed to give high efficiency on
  - vector processors
  - high performance “super-scalar” workstations
  - shared memory processors
- Satisfactory use on all scalar machines
  - PC’s
  - workstations
  - mainframes



ScaLAPACK

Is a distributed-memory version of LAPACK for parallel architectures

- massively parallel SIMD machines
- distributed memory machines



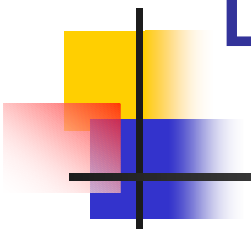
CLAPACK

- CLAPACK's goal is to provide LAPACK for someone who does not have access to a Fortran compiler.
- The CLAPACK library was built using a Fortran to C conversion utility called **f2c**.
- The entire Fortran 77 LAPACK library is run through **f2c** to obtain C code, and then modified to improve readability.

LAPACK Flop Counting

- So far just presented timing results, but how to measure flops and flops/s?
- Easy for some operations, e.g. For NxN matrices A and B, A*B requires $2N^3$ flops, or LU factorisation requires $0.67N^3$
- But flops for eigenvalue computation or SVD depend on input data

LAPACK Flop Counting



- Best way to count flops would be to use hardware counters – standard feature on modern processors for software profiling
- Accessing these counters can be difficult: often poorly documented, unstable or inaccessible to user level software
- Hence cross-platform software for hardware flop counting has traditionally been limited

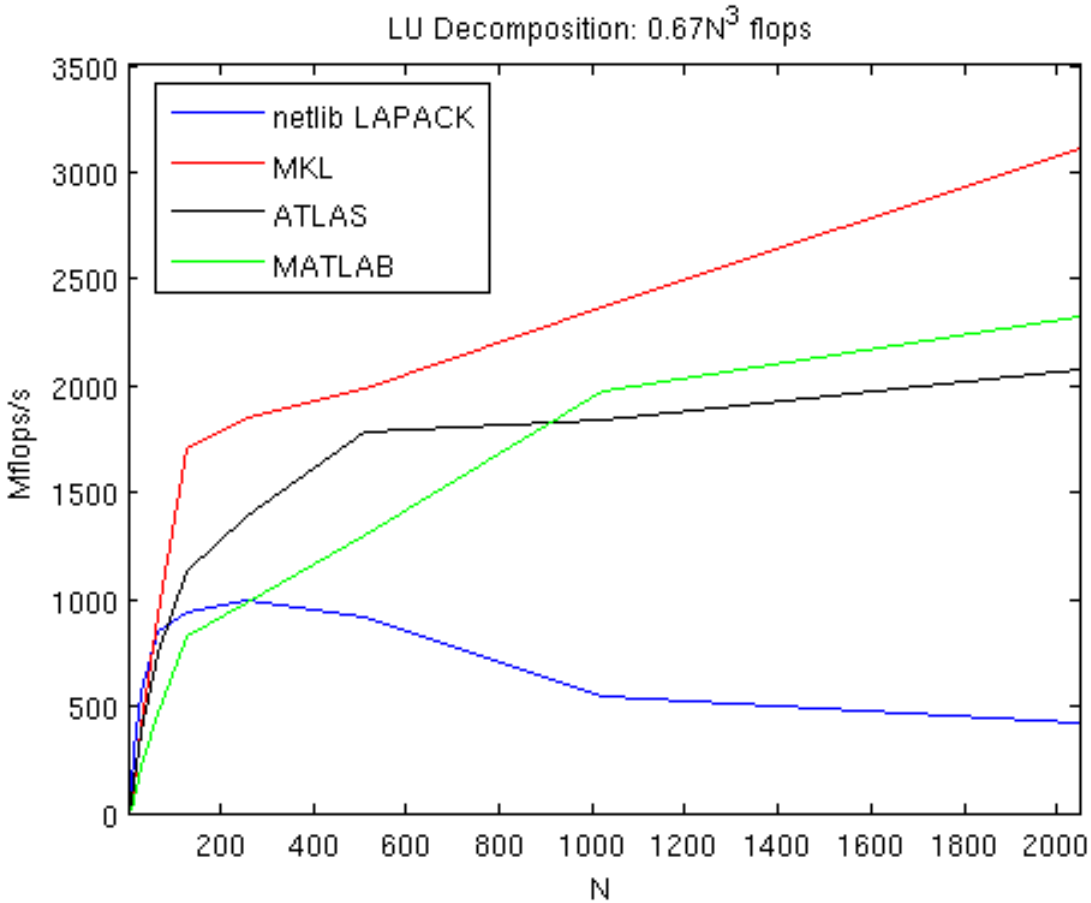
LAPACK Flop Counting

- PAPI project: aims to define standard interface to hardware counters to simplify profiling software
  - See “A Scalable Cross-Platform Infrastructure for Application Performance Tuning using Hardware Counters”, J. Dongarra et. al.
- We did not do direct flop counting, couldn't find any simple software for LAPACK
- e.g. MATLAB: >> help flops
  - “FLOPS Obsolete floating point operation count. Earlier versions of MATLAB counted the number of floating point operations. With the incorporation of LAPACK in MATLAB 6, this is no longer practical.”

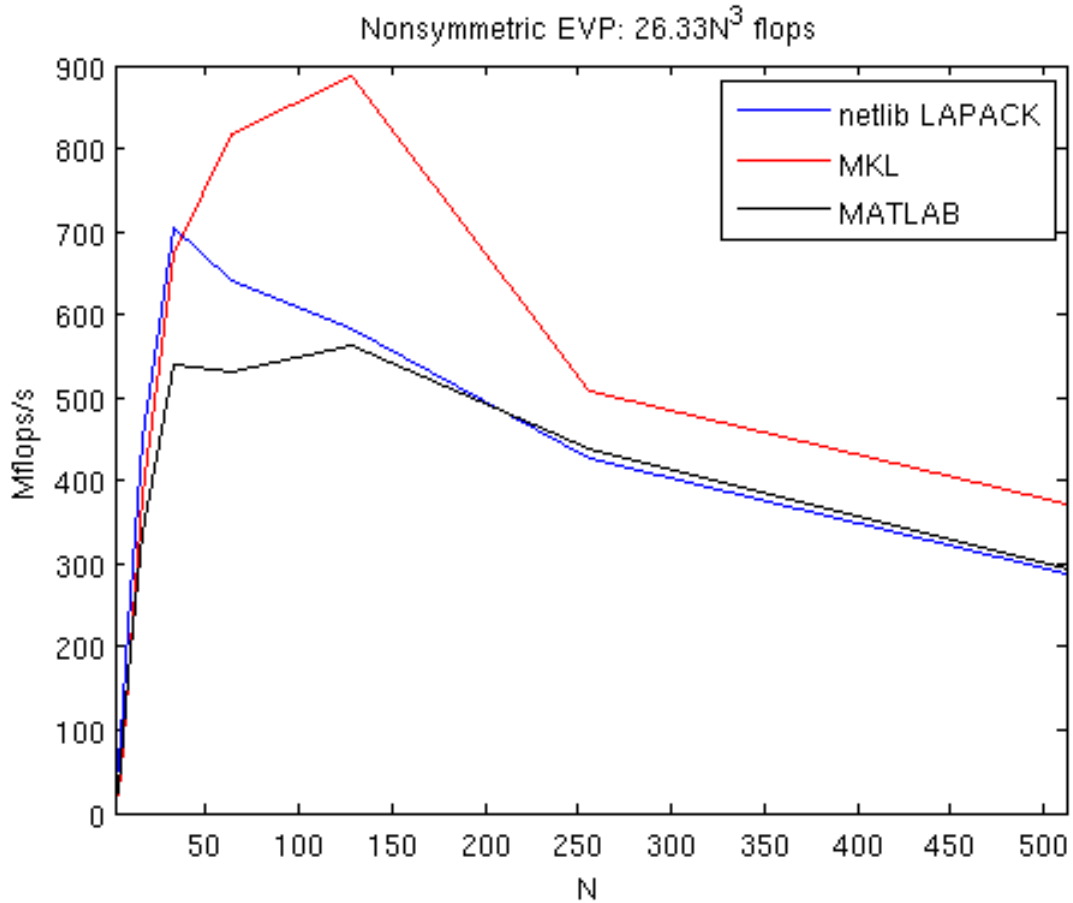
Synthetic Flops

- Alternative approach: use “standard” flop counts for various algorithms
- DGESV (LU factorise): $0.67N^3$
- DGEEV (ew's and ev's): $26.67N^3$
- DGESVD (SVD): $6.67N^3$
  - See http://www.netlib.org/lapack/lug/node71.html

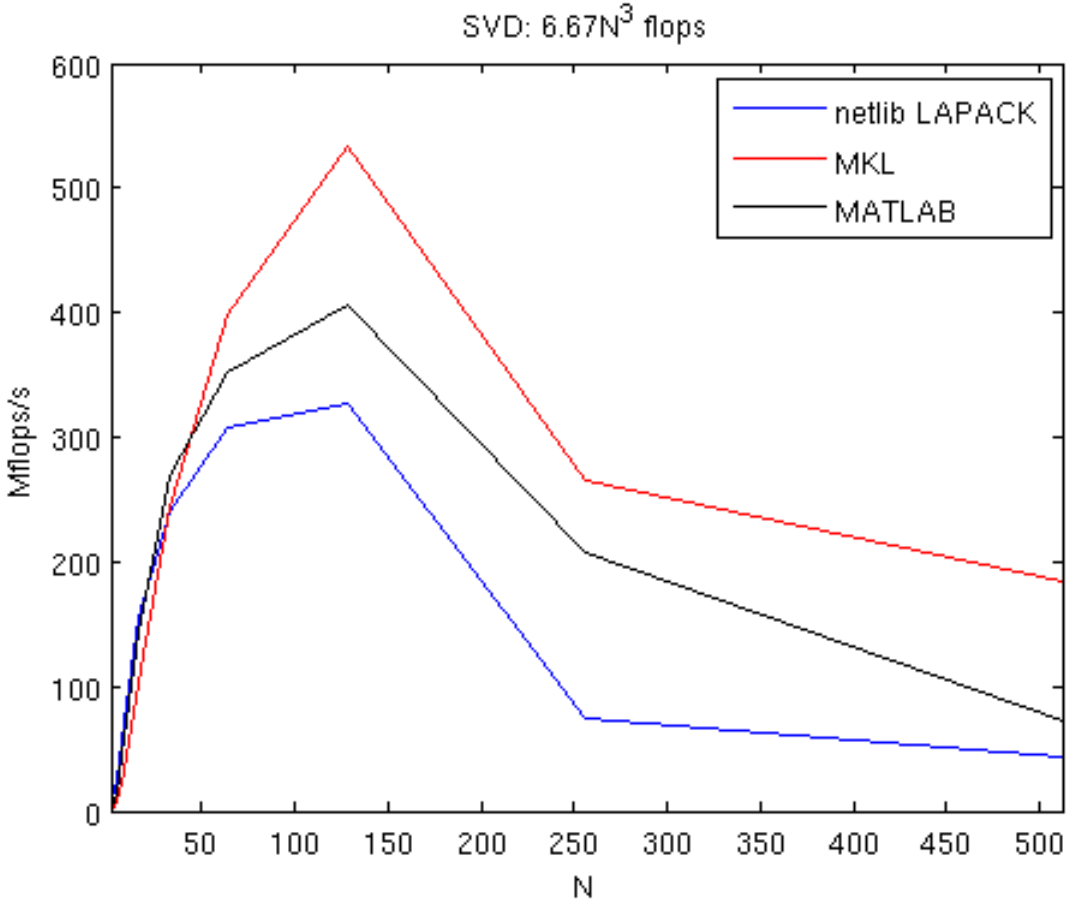
DGESV


LU Decomposition: 0.67N^3 flops
netlib LAPACK
MKL
ATLAS
MATLAB
Mflops/s
N
0
500
1000
1500
2000
2500
3000
3500
200
400
600
800
1000
1200
1400
1600
1800
2000

DGEEV


Nonsymmetric EVP: 26.33N^3 flops
Mflops/s
N
netlib LAPACK
MKL
MATLAB
0
100
200
300
400
500
600
700
800
900
50
100
150
200
250
300
350
400
450
500

DGESVD


SVD: 6.67N3 flops
netlib LAPACK
MKL
MATLAB
Mflops/s
N
0
100
200
300
400
500
600
50
100
150
200
250
300
350
400
450
500

Spectral Methods

- LAPACK capabilities illustrated by Fortran implementation of cheb, p13, p14, p15 from “Spectral Methods in MATLAB” by L.N. Trefethen...