LAPACK Usage

LAPACK was made available on April 1, 1992. It has been picked up by over 8000 people from netlib at ORNL alone. (In addition to netlib, a number of sites have it available via FTP and NAG provides LAPACK on magnetic media.) In addition, Cray, IBM, Convex, DEC, Fujitsu, NEC, NAG, IMSL have adopted the software. In doing so the vendors distribute, support, and answer questions from users of this software. Recently we asked the users what applications LAPACK was being used on. We were surprised by the number of users and the different problems LAPACK was being used on. A partial list follows:

Cardio-magnetism
Wave guide propagation
Image processing
Quantum chemistry/physics
Atomic structure calculations
Electromechanics
Radiative transfer
Geophysics/seismology
Plasma physics
Helicopter flight control
Chemically reactive flows
Nonlinear mechanics
Antenna analysis

Singular value decomposition routines and have found them to be superior to those given in Numerical Recipes in C.

Astronomical data processing. LAPACK routines are mostly used on smaller problems. However we do have large image processing problems.

SVD routine to compute least squares solutions for difference stencils in a spherical geometry.

Build a package called Conlab (a kind of parallel algorithm laboratory) from Umea University.

Mainly for systems obtained from finite element method for partial differential equation problems.

Ab Initio Molecular Dynamics, Quantum Calculation of Complex Materials

Applied for some of the dense sub-matrices that arise in the solution of Medium/Large Sparse Linear Systems.Interior Point Algorithms for Linear Programming.

Incorporating them into C++ matrix classes.

Computer models operating on datasets collected from Sattelite or other atmospheric monitoring equipment/endeavors.

We are writing "Scientific and Engineering C++", J. J. Barton and L. R. Nackman, Addison Wesley, 1993. We use LAPACK as an example of a systematic FORTRAN subroutine library to show how C++ can be used to embed the systematics into code.

Numerical methods class Small to medium

Analysing benchmarks and evaluating different architectural and micro architectural options for higher performance on these benchmarks. Not solving real problems.

Nonsymmetric eigenvalue problems. Stability testing of most of the LAPACK routines. Have also used the condition estimation. Very small. Usually n ¡ 10.

Fluid dynamic problems optimal mathematical techniques to determine the existence of structure in a flow. Matrices are 16 x 16.

Robot-dynamics simulation to solve linear systems, perform SVD, and find eigenvalues/eigenvectors. "small", ranging from 2 to 30 degrees of freedom.

Non-linear least-squares regression utility Small. (less than 10 X 10)

QR routine is used in a research code. Underdetermined systems of small order (say 20) are solved. We are planning to use part of it for a course on numerical analysis.

20x20 wanted to look at the dependence of the determinant of a matrix as we changed N.

Diagonalization of complex hermitian matrices. 20x20), but will be somewhat larger in the future (say 1000x1000).

Electromagnetic scattering and soon spin-wave turbulence. Currently small (20 x 20 complex single precision). Will do much larger (finite difference code, size TBD) with spin-wave.

Lapack will be used to solve a non linear problem iteratively. about 20(parameter size) by 200(data size)

Solving problems of dynamic modelling of structures using time series. orders of 5-30.

I'm still using eispack. Eigenvalue solver for planetary atmospheric wave modes. Modest matrix ¡ 50x50.

My main application is solving full linear systems ad nauseum N might typically be 100 and M might be 25 or so.

Determination of a few lowest eigenpairs of real symmetric matrices. Order of matrix ¡ 100

Simulation of multibody systems. Problem size is generally in the range 5-100.

Non-linear optimization engines for control systems applications. large but block diagonal, with the blocks rarely exceeding 100x100 or so.

Newton-Raphson iterative solution of a nonlinear system of equations, describing the statistical equilibrium of excited state populations of ions in a partially ionized gas. 50-100 equations are involved for each of 10 ionic species and several (many) iterations are required to get a converged solutions.

Eigen system problems, variable up to 200 by 200

Programs for solving the inverse problem in CardioMagnetism. We currently measure the magnetic fields generated by the cardiac currents in human subjects, and are interested in developing software and algorithms to determine the position and direction of these cardiac currents. pical sizes ... solving matrix systems of about 200 equations in 50-500 unknowns. Always ill-posed and often singular.

Solution of linear equations (sgetr[fs]) in Rheinboldt's Continuation package typical problem sizes range from 20 - 200

Not yet, but we are planning on using it extensively to replace older linpack/eispack calls. linear systems, eigenvalue problems, eigenvector determination (all matricies are large, complex, and dense (260x260 and larger).

Boundary element approximations of guided waves propagation, leading to eigenvalue problems. 300 or so,

Free electron laser 400x400 simplest case we will need to solve a matrix which size is n000xn000.

Control systems analysis and design. Hardly larger then several hundred.

Benchmark to use in microprocessor design. 500x500 matrixes

Some research projects use LAPACK (in particular BLAS level 1,2,3 to test new algorithms). We use also LAPACK for teaching.

Image Processing,esspecially image understanding. 512x512 images, and some feature matrix of images. (64x1 - 64x64 - 256x256)

Scientific computations for my research in Numerical Analysis. 4X4 to approximatly 512X512.

Doing research on variants of gram-schmidt orthogonalization. I use the factorization routines, condition number routines, LAPACK test matrices, dlarnv, dlarnd, dlamch, etc. Order 500 matrices (and smaller).

Testing of eigenvalue codes, QR factorizations. Up to order 500 now. We'll go to 1000

Complex eigenvalue problems. 600 x 600 matrix complex

DIAGONALIZATION OF COMPLEX MATRICES UP LINEAR SYSTEMS OF EQUATIONS. TO N=800 ON

Electrical engineering problems. 800x15 matricies.

Optimization; Solving System of Equations (thesis) order 100 to 1000

My application is analysis of dynamical systems. Matrix sizes about 1000 by 1000.

Diagonalize symmetric matrices. quantum chemistry calculation in organic molecules for applications in non-linear optics. The method involves configuration interaction with a semi-empirical hamiltonian. The matrices are up to 1000x1000.

Liquid state statistical mechanics, solution of integro-differential equations. Typical matrix rank is 200-1000

Matrix equation that arises in Control Theory by a parallel algorithm. size ¡= 1024.

SVD, Linear systems, Generalized Eigenproblems (QZ algorithm) Roughly 1000x1000

Eigenvalues and condition numbers of nodal admittance matrices of electric circuits. Up to this point the matrices have had order up to 1000.

Limited to benchmarking 1000x1000

Diagonalizing the matrix to get the eigenvalue and corresponding eigenvectors of real general matrix. various size from 80x80 to about 1600x1600

Sparse Matrices, Atomic Structure Calculations 1,000 - 10,000

Matrix structural analysis and structural dynamics applications. Using eigen value problems and solution for AX=B 1650 X 90 matrices

Linear least squares. Used for calibrating various instruments used in the collection of meteorological data. mxn m: 5000-10000 n:3-5

Calculation of electronic structure in amorphous solids, and simulation of electronic devices. 2000x2000, with a bandwidth of about 50

Solving for a few eigenvectors of a large symmetric matrix. ”A relativistic configuration-interaction calculation of the low-lying states of heliumlike ions.” Two or three eigenvalues and eigenvectors for matrices of 2048 x 2048 stored in symmetric mode.

Still using linpack, general scientific computation – largely linear regression with sometimes eigenvalue problems 10-100 X 1000-10000

Linear equation group need to be solved. 1891 variables.

Writing a MATLAB clone (well almost) to aid me in my dynamics and controls education/research.In MATLAB lingo, I use eig(), hess(), lu(), inv(), svd(), qr() quite frequently. 10000

Benchmark other algorithms being developed and Fluid Dynamics. In both cases, probably 1000x1000, at least.

Linear algebra in support of solving the systems of equations arising from finite-differencing of elliptic PDE's. Small. I am solving 2-D linear, separable PDE's on 180x140 grids. I need about 10$\hat{6}$ steps per run, so the total work requirement is quite large.

Large-scale, semi-linear, elliptic PDE systems in Liquid Crystals Actual problems are around 10,000–100,000 unknowns. Solution procedures are iterative in nature; LAPACK and BLAS are relied upon for a variety of the numerical linear algebra (Cholesky factorizing blocks in block-Jacobi pre-conditioner, block matrix-vector products, vector dot products, etc.).

Cfd Boundary Element Method. on matrices of size 100**2 but plan to increase size in the future to 1000**2

I'm using BLAS-3 but not (yet) LAPACK. Should integrate it in the codes in the next 6 months. Application: Electronic structure of solids, liquids and molecules. Problem size: Hermitian eigenvalues and eigenvectors 500 matrix multiplication 20000 X 100 to 100000 X 200.

Computational electromechanics. MoM solution to scattering problems. LU decomposition of large matrices. 5,000 by 5,000.

Eigenanalysis. 5000 variables.

Radiative transfer (relativistic or non-isotropic scattering) code, multi-level non-LTE codes, stellar atmosphere codes. General band matrix systems: typically 392 unknowns width bandwith 7 (each upper and lower) to 12544 with bandwidth 63, usually between one and several thousand of these systems per run, usually 51 rhs per systems (depending on the problem) DGEEV: typical matrix sizes are 8x8 to 64x64, typically 50 of these matrices per wavelength, typically 1000-5000 wavelength points. tri-diagonal systems: typically 50x50 with 51 rhs, about 1000-5000 per model iteration, typically 40 iterations, general matrices: factorization, inverse, solution, range from 4x4 to 50x50, very large number of these matrices.

Solution of differential equations. Eigenvalues and eigenfunctions. (Hartree-Fock calculations of atoms and systems with spherical symmetry) Diagonalization of banded (mostly, threediagonal matrices) n= 1 000 – 20 000

Building blocks for the arnoldi/lanczos k-step diagonalization procedure for mostly sparse matrices a few hundred to thousand

I am one of Jim Demmel's students at UC Berkeley, and we are investigating how to write LAPACK code on distributed memory machines. Symmetric tridiagonal eigenproblems from 512x512 to 16384x16384.

Matrix inversion. 2000-3000 banded matrix of bandwidth 7, but these dimensions may increase substantially when we consider bigger problems.

Eigen Values for some Quantum Physics problems. In the Fall - FE systems and Stastics will move to LAPACK also. 256*256 upto 10,000*10,000 (where LAPACK solves subproblems on an MIMD machine and my code puts solutionms together)

Solve a symmetric positive-definite matrix system 10,000X10,000 sparse banded matrix system by LAPACk. I tried to solve a 26,000 X 26,000 problem by LAPACK and I failed. The values of some diagonal elements become negative by LAPACK.

Eigenproblems, linear systems 1000 for eigenproblems, 5000 for linear-sys

Finite element analysis of differential equations describing the propagation of electromagnetic waves in plasmas used for fusion research. It varies but currently up to 34 x 3000 in band storage.

Applications in geophysics/seismology, solving large sets of linear equations arising in tomography.dimensions are set to be as large as computers' core memory and the matrix solver software will allow (feasible computation time). Currently the sizes are about 14000 x 14000

To solve a Poisson problem on (almost) a rectangular domain. The solution of the Poisson problem is needed as a part of a solver for the Stokes problem. order 600 – 10,000.

Neural networks and self-organizing systems. We don't usually do "standard" NN (back-prop etc) - most of our stuff involves topographical mappings, eg. Kohonen nets. These are often formulated as systems of coupled non-linear ordinary differential equations. In many cases, the coupling is homogenous, although certainly not all-to-all. These are mostly solved using Euler's method; it has good enough accuracy for us, is easy to implement, and we often use delay differential equations (eg, coupled oscillators) where anything else gets very complicated. As we do mostly vision research, our systems are quite large: image size from 128x128 to 512x512 pixels, with a node attached to every pixel. The interesting thing we've been looking at are systems where the interconnect matrix is sparse and inhomogenous, eg, a node is connected with a certain probability to all others, and that probability falls off with distance, but never becomes non-zero; the matrix is determined randomly at initialization. In this case, it would be nice to have a programming tool - which we've been developing - that takes note of the communications pattern during the first iteration and then provides the necessary values from remote nodes proactively during the following iterations. At the same time, it would take care semi-automatically of the data-partitioning when the hardware (eg, # processors) changes. a fairly large transputer network: currently, up to 60 T800; soon, a 64 processor T9000 system.

Chemical Reaction Engineering 50 - 30000 Unknowns

Matrix sizes of 1000*1000 are not unusual.

Numerical electromagnetics Up to 15000 (complex symmetric)

Finite Element Code development in Fluid Flow, Heat Transfer, Stress Analysis 3000 finite element in which an element matrix of 27x27 is calculated and assembled into global Matrix, and the element calculations are repeatedly used for millions of times. The applications will be in larger problems in future.

COMPUTATIONAL FLUID DYNAMICS (PANEL METHODS) 10,000 BY 10,000 SINGLE PRECISION ARRAYS NOW, UP TO 30,000 BY 30,000 SOON

Edge plasma code 100 by 100.

I am also fitting it into code being developed under Air Force contract for automated creation of CAD tool models. Eigenvalue solutions of 600x600 double precision systems

Ordinary Differential Equations software 3 to 20 in size.

We use the eigenvector/eigenvalue routines. In conjunction with a suite of programs this is being used to model bio-molecules. At present, matrices from 160x160 to around 800x800.

BENCHMARKING ONLY (FOR NOW), RESULTS USED FOR FUTURE MACHINE DESIGNS

System simulations.

Theoretical plasma physics: I am looking at the structure of magnetic fields produced by particular magnetic field coils, and figuring out a way of modifying the coils so that the magnetic field structure has some desired characteristics. Tiny, up to maybe 10 by 10.

System identification (curve fitting of linear system models) and control design 4000 x 4000 symmetric svd (partial spectrum)

Linear equations and for eigen-analysis. Matrix dimensions are typically 1000 to 5000.

Special-purpose graphics application (in shape theory), about 7500 lines of C code, which includes use of the complex QR decomposition, and, shortly, use of the complex SVD, routines from LAPACK. ¡30 x 3 matrices

Primarily CFD in three dimensions. 10 and 100 usually,

Typically 100 by 100.

CONLAB (CONcurrent LABoratory). CONLAB is an environment for developing algorithms for parallel computer architectures and for simulating different parallel architectures.

I am working on a command language similar to matlab which runs with a data base which has been developed at our computer center. It depends on the user. I am developing on apollo DN3500, but the software will run on multiple

Linear and nonlinear mechanics and dynamics ranging from 1 unknown to 10000 unknowns

Applications are mainly research applications. We are studying methods for large sparse linear systems, and we are using LAPACK for the solution of some subproblems. Typical problem sizes are 100-1000

SVD,least squres, sparse matrices signal processing, estimation, tracking, ops res small to medium (up to n=1000)

Several of the routines for solving linear systems, finding eigenvalues, and finding eigenvectors are linked into a larger code used to compute the activation of materials in the containment wall of a (proposed) fusion reactor.

Helicopter flight control software 6x4 system of equations at every time step.

Tridiagonal equations. (In general we solve finite difference Schrodinger equation in Quantum Mechanics so predict characteristics of quatnum devices.) From complex*16 4x4 to 128x128 matrices.

We are incorporating it within S-Plus, a software environment emphasizing statistical applications Moderate - up to a couple hundred variables

Research purposes, in the area of petroleum reservoir simulation. These problems involve solving systems of nonlinear PDE on several hundreds of thousands of grid cells. The Lapack routines are being used to solve subproblems within each timestep.

3-D TRANSONIC POTENTIAL FLOW FOR TRANSPORT AIRCRAFT UP to 200.000 GRIDPOINTS

Eigenvalues solvers for Real, Symmetric matrices. Inverses of same matrices. Up to 7000x7000 (sparse)

Our group works on developing the Navy's capability for forecasting the global ocean circulation. Numerous subtasks of this project need numerical software and in the past we have used: Matrices are usually order a few hundred or so. The record was probably a 3840x3840 matrix, for which a few of the largest eigenvalues were needed.

We took benchmark programs like the LINPACK and LAPACK to evaluate the performance of our hardware and to compare it with other architectures. Limited up to 1000 x 1000

Computational chemistry 200-1500

FEM analyses. From small sizes up to the maximum machine capacity which is 128Mb memory.

Regression and time series, and Bayesian analysis problems. Our largest problems maybe 300x300

Calculate the electronic structure of dilute alloys and periodic crystals. This requires to solve complex linear systems of equations and to determine eigenvalues and eigenvectors of complex matrices. 10 and 1000 for the linear systems and between 10 and 30 for the eigenvalue and -vector determinations.

Eigenvalue and singular value problems, systems of equations moderate size, up to 1000 x 1000

The general field is the numerical modelling of chemically reacting flows. Specific problems are: Solution of stiff ODE systems (sparse Jacobians) Mechanism reduction based on eigenvector analysis of chemical reaction systems. Solution of stiff PDEs by the method of lines. Thus, I am especially interested in Linear algebra for large sparse structured matrices Methods for finding and ordering Schur-vectors The computational effort usually limits the size of our problems. The size of the matrices where we have to determine Schurvectors is app. 50 * 50 at the moment, however we aim at problems with 2000*2000 matrices.

What follows is a collection of comments from various people on the package.

```
From: cff@vuse.vanderbilt.edu (Charlotte F. Fischer)

It would be nice if naming conventions could be portable going to Cray
(avoiding the Double to Single Precision conversion).  The
Cray compiler will ignore double precision in source, but
it cannot ignore the names of subroutines.  At least, I think
that still is the case.

------------------------------------------------------------------------------

From: fredrick@acd.ucar.edu (Timothy Fredrick)

It's an impressive library -- I just received the manual yesterday.  Keep
up the good work!

------------------------------------------------------------------------------
From: keenan@cs.rice.edu (Phil Keenan)

I bought the manual and LAPACK looks good for dense problems, but I
have not tried it yet for two reasons:

1) the book makes it sound difficult to get started if you don't buy
the whole tape from NAG --- is there stuff on the tape that is not on
netlib?

and

2) I mainly do PDE's, which involve solving huge (say 1-10 million
unknowns) sparse linear systems, often on parallel machines like the
nCube, Intel Paragon or CM5; LAPACK currently supports neither
sparsity nor parallelism, though I hear you are working on both, which
is very good.

------------------------------------------------------------------------------

From: "GEORGE WHITE, BIOLOGICAL OCEANOGRAPHY DIVISION" <G_WHITE@bionet.bio.dfo.ca>

As usual, the numerical analysis community has lost the PR wars.
If I mention I want to use LAPACK I am told I really should look
into LINPACK, or maybe if the problem is important we might consider
buying IMSL.  If it hasn't been mentioned in Byte, it doesn't exist.
Maybe someone should send Pournell a review copy.

------------------------------------------------------------------------------

From: foo@t13.Lanl.GOV (castor fu)

We're pretty happy with the code, although you might want to point out
that DEC's BLAS library leaves something to be desired.  We ended
up using the fortran coded routines because the DEC BLAS produced
wrong answers.

------------------------------------------------------------------------------

From: ian@ziggy.ds.boeing.com (Ian Searle (Sys-Mngr))
```

Please eliminate ALL FORTRAN I/O and character string passing
in FORTRAN argument lists. We commonly link LAPACK routines to
C, and FORTRAN I/O and character string passing exacerbate the
differences in compilers, and make linking LAPACK to C more
tedious.

From: astfalk@hydra.convex.com (Greg Astfalk)

If I had a wish list I'd like to see general sparse routines
in LAPACK and folowing that iterative solvers. Now that, in
addition to being a lot of work, would be ''quite the package!''

From: ellis@qucis.queensu.ca (Randy Ellis)

Keep up the good work! It is immensely reassuring to find this
quality of software available on the network. Cheers,

From: "Ferhat F. Hatay" <hatay@spot.Colorado.EDU>

Thanks for putting together a much needed LAPACK. I have been
using LINPACK and EISPACK on CRAY-YMP, CRAY-XMP,
SGI 4D35, SGI Indigo, DEC workstations for quite a while.
I have not started playing with LAPACK seriously
since I already made my blackbox units for my problems.

From: "Nitin R. Keskar [ChemE mf12703]" <keskar@s1.msi.umn.edu>

Thank you all for providing us Users with LAPACK!

From: "Lee R. Nackman" <lrn@watson.ibm.com>

The manual is clear and easy to use.

One suggestion: provide header files for C and C++ users. Most C and C++
compilers make it easy to call Fortran subroutines, but declarations are
required. Writing them once and making them available would be a boon
to C and C++ users.

From: schreiber@schreiber.asd.sgi.com (Olivier Schreiber)

Thanks for bringing us lapack.

From: Ronald Schoenberg <rons@u.washington.edu>

I'm looking forward to not having to fill up my area with LAPACK source code downloaded from NETLIB.

From: reg@sun4d.plk.af.mil (Reg Clemens)

The CRAY2 presents other problems... If you are familiar with the architecture of the machine, but ours has 2gb of real memory shared by the 4 processors, and then each processor has 16kwords of ‘local’ memory. The OS uses these 16kwords for ‘linkage’ info (and some programs will try to use them for ‘fast’ memory) but the problem is that something like 8-10words are used for each subroutine loaded... This means that somewhere arround 1500 routines and it is FULL. At this point it TRIES to overlay local memory, but even when it suceeds, it usually cant do the right thing (nb function names passed as fortran arguments negate any analysis that it does)...

So the bottom line is that the LAPACK QA tests are in general TOO BIG for the CRAY2! Ditto some of the IMSL QA tests... With IMSL in the past I have got in and split things up into more manageable chunks, I have not had the chance to try that with LAPACK... Not even having looked at the code I suspect that it would be more difficult.

So, the libraries are out and available for use, but have not been ‘officially’ announced. Seems that the day after I finished building and QAing the CRAY2 version they put a new CRAY2 Fortran compiler and I need to recompile with it... to be really happy. I did get some impressive results with the CRAY2 multitasking turned on, but as one might expect, those routines that can take advantage of it get even more than the 4x speedup that one would expect (just to confuse the issue) and those that cant take advantage seem to run about 10% slower with it (multitasking) enabled...

Have had at least one user that needed a Double Precision version of ‘RS’ and was able to point him at the equivalent LAPACK routine.

Other than that, since I havent made an official announcement, there has not been a great deal of usage yet, but from what I have seen to date, they (LAPACK) go to the top of my list for recommended routines, then SLATEC, then at the very bottom anything in IMSL.

From: ren@math.berkeley.edu (Huan Ren)

This summer I tried Lapack, eispack, IMSL and some program by myself. Lapack is a very efficient general package, I use some rotines to compute all the eigenvalues and eigenvectors of that general complex matrix. It works very well and very accurate. I use IBMrisc6000.

This is really a good package and I recommend many people in my group to use it.

From: yeti@sara.la.asu.edu (Peter Hauschildt)

I really appreciate the monumental efforts of the LAPACK team creating a portable and high-performance linear algebra package! I found already that LAPACK make the transition from one machine to another VERY simple, just find the correct loader commdand and that's it. I have installed LAPACK on my IBM RS/6000 and had absolutely no problems. I have converted my codes to LAPACK and I am considering remove of all references to LINPACK. New or updated versions will only support ESSL and LAPACK and I hope that many vendors will support optimized versions of LAPACK (or, e.g, interface LAPACK->ESSL) for their machines.

Thank you for your great work,

From: Bill Venables <wvenable@stats.adelaide.edu.au>

This is a major suggestion. I would prefer the software be written in some language other than Fortran 77. My preference is for C or C++. Most software I write will have a front end built from lex/yacc and so will be in C. Using Fortran routines from a C calling program is for me clumsy, error prone and non-portable. Speed is not such a big issue for me but accuracy is.

From: Ricardo <DUARTE@BRLNCC.BITNET>

First of all thank you for a superb job accomplished in dictating a "real standard" for the implementation of the current and future generation of Linear Algebra based codes.

Finally I would like to thank you very much for the attention in my name and in all of the COPPE/UFRJ researchers involved in our Linear Algebra based Optimization projects.

From: zdenko@masc1.rice.edu (Zdenko Tomasic)

keep up a good work!

From: hsong@ardour.bellcore.com (G. Hugh Song)

Thank you for the good work.

From: pauln%physunc.phy.uc.EDU@UTKVM1.UTK.EDU (Paul Nisamannephong)

Have a good time

From: mcgrant@rascals.stanford.edu (Michael C. Grant)

I would like more support for packed symmetric systems, including support for translation between packed and non-packed types. I could also really use a good qz implementation!

I really think that these tools could benefit from an implementation in C++. It seems to me that I'm forced to link in a lot of code that I'm not using, because each routine needs to handle several special cases... perhaps with function overloading, each special case can be handled in a separate function---in my case, for example, I often perform eigenvalue calculations where the eigenvectors are of no use... so certainly all of that code I link in to handle the 'save vectors' case is just unnecessary fluff. Using the keywords as template parameters instead of run-time branches might be one approach, e.g.

```
enum { noVecs, Vecs } eigvecONOFF;
enum { allEVS, Interval, Indexed } eigChoices;
enum { upper, lower } whichTriangle;
template < enum eigvecONOFF a, eigChoices b, whichTriangle c >
spevx(  ... etc ... )

spevx< noVecs, allEVS, upper >( ... );
spevx< Vecs, Interval, Indexed >( ... );
```

One can certainly still supply the run-time decision-maker as an envelope for all of these possiblities... I suppose this tends to breed cumbersome notation but MAN would it be readable. I guess it is also more difficult to share the common code in these instances.

On the other hand, let me make it clear that I think LAPACK is a nice package... despite it being in version 1 it's quite dependable and useful. I'm not complaining about its abilities, that's for sure.

From: phil@global.geoph.s.u-tokyo.ac.jp

The netlib software was great very convenient.

From: (Joseph I. Landman) <landman@hal.physics.wayne.edu>

It is a very nice product, and I am very pleased with its predecessors. Please do keep up the good work!

From: Weimin Han <whan@math.uiowa.edu>

It is very nice to use LAPACK.

From: bett@wave1.ece.wisc.edu

Thanks for putting together a nice package.

From: Tomoji KAGATANI <kagatani@rbc.ncl.omron.co.jp>

I am satisfied with your good efforts.

From: meyer@bcstec.ca.boeing.com (Ed Meyer)

I am very impressed with this project - First rate software
Keep up the good work!

From: amashaie@mrco.carleton.ca (Akbar Mashaie)

The emerge of these routines were found a safe, fast, and clear method
in Numerical Analysis. I will contact you as I have any comments.

From: ned@lamont.ldgo.columbia.edu

I really apreciate having access to these and the other routines.
Keep up the great work!

From: kern@masc5.rice.edu (Michel Kern)

I am glad to take this opportunity to thank the whole LAPACK team for
the outstanding job you're doing. something as LAPACK is an invaluable
tool for the entire computational community.

From: COY@PPGENO.MIT.EDU

Within its scope, LAPACK is very nice. Are you considering
expanding the scope?

From: case@scripps.edu (David Case)

Thanks from a satisfied user to all who participated in putting LAPACK
together!

From: Colin Goodall <colin@stat.psu.edu>

I like the systematic approach to linear algebra routines
taken by LAPACK a lot.

From: pof@alamos.cray.com (Paul Frederickson)

Our thanks to all of you for a very nice piece of work.

From: meza@snll-arpagw.llnl.gov (Juan Meza)

Thanks for all the help though. You're doing a great job.

From: mboucher@silver.sdsmt.edu (Mike Boucher)

Although this mail comes from a .edu site, I am an industrial programmer who has worked on large projects before. From that perspective, I did not have high hopes for a project of this size being completed by people who were working at widely separated locations. Clearly, I was mistaken. This is a really nice piece of work and I congratulate you.

From: tsingh%maxwell.ece.uc.EDU@UTKVM1.UTK.EDU (Singh T. Junior)

1 We would like to use C++ but the libraries are in FORTRAN. C++ is very efficient and results in smaller and easily readable code. We would appreciate either LAPACK in C++ or a method to link C++ code with FORTRAN libraries.
2 We are verty happy with LAPACK routines and a lot of research has been done in our group with aid of LAPACK.

From: "Robert J. Harrison" <d3g681@fermi.pnl.gov>

LAPACK is a great and much needed effort ... thanks!

From: IFF081@DJUKFA11.BITNET

1) The LAPACK routines were very helpful for me in order to make my programs portable from the CRAY- via the IBM- to the Intel-computer.

From: Ulrich Maas <FGAE@DS0RUS1I.BITNET>

First of all let me thank you for supplying the LAPACK library which I think is very useful.