

Re: Who uses clapack?

Source: <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2004-12/0317.html>

From: Jentje Goslinga (goslinga_at_telus.net)

Date: 12/10/04

Date: Fri, 10 Dec 2004 23:04:59 GMT

Victor Eijkhout wrote:

- > *The authors of lapack/scalapack are starting to work on a new release of*
- > *these packages. One of the things we want to address is the silliness*
- > *that C users, instead of linking to the binaries of the Fortran version,*
- > *use an automatically translated C version.*
- >
- > *Therefore we'd like to know precisely what the reasons for this are.*

I am developing a suite of C functions consisting of the Symmetric Eigenvalue problem, the SVD and the Block Reflector. I have used Lapack as a reference to write a Divide and Conquer solver for the Symmetric Eigenvalue problem and am now doing the D&C SVD. I am quite well informed.

Many people nowadays perform numerical computations in C or C++ which are just as suitable as Fortran.

There are many engineering applications with substantial number crunching which are written in C.

Most individuals and many smaller Engineering companies do not want to be perpetually burdened with having to purchase a license for a Fortran compiler for their workstations and would rather spend a few man days converting the Lapack modules which their application requires to C. Have you ever seen those bills for a Fortran compiler for an AIX machine? Have you ever had to fight SUN Fortran?

Those mentioned would love to have the main factorizations in C code rather than being burdened with having to purchase a license for a Fortran compiler if one is at all available, and having to cross language link yet another huge library of which only a fraction is used.

I estimate there must be a fair contingent of people who have extracted the CLapack modules required by their application, found out what auxiliary modules are required

and absorbed these into a C application.

The problem with Lapack is that you can't read the code, that it is neither commented nor documented with a Technical Manual and that the algorithms referred to are detailed in obscure internal reports at certain universities.

This is fine if there is reaaaally no maintenance required ever but this is never the case.

And Fortran to C conversion make the code even worse.

Although it is certainly possible to write code of almost the same quality as good C in Fortran, the Lapack code uses 60's style programming:

No dynamic memory allocation but the eternal WORK and IWORK arrays which are partitioned up in intricate ways.

Does Lapack really need to cater to prehistorical compilers which do not support dynamic memory allocation?

The functions generally are too large with the familiar gobs of variables at the start and no attempt at restricting scope.

There are no Modules containing related functions but every function is separate.

The single precision versions are useless in my opinion but others may disagree, at the very least relegate them to recognizable modules so users who do not want them can leave them out.

There is more: insightful remarks on how to re-use arrays to conserve memory, fear of copying arrays resulting in a milliped matrix multiply functionality and so on.

"Lame comments" and huge headers abound but the intricacies of the algorithms are not mentioned.

If you want to sink more time into the Fortran Lapack, you might want to look into the temporary storage issue, allocating the memory instead so the user is not burdened with having to calculate the "work storage" required.

Next get rid of the single precision versions or at least modularize them and generally aggregate the code into a few dozen modules of related functions.

For the QR type methods, store the Givens transformations and allow the user to pass a function to select Eigenvectors or Singular Vectors for computation rather than having to estimate their index for computation beforehand which really sucks.

Just some pointers.

Lapack's use of a BLAS is great but Lapack is tailored to the traditional BLAS rather than the other way around and the functionality of these BLAS functions is not optimal.

Simple Example: BLAS provides DNORM2 and DSCALE functions but 90% plus of the use of these is to Unitize a vector which should be a single function. Currently this requires two function calls between which the all important norm information is truncated from 80 bits accuracy to 64 on a PC for example. Likewise with

sci.math.num-analysis: Re: Who uses clapack?

the computation of the Householder vector. Is this a big deal?
Well, there are quite a few of these instances, and in the end it all adds up. The point is that you might want to look beyond the traditional BLAS which is just historical accident.

Lapack is quite accurate and fast although the accuracy of some of the order two factorizations and some of the orthogonal transformations can still be improved here and there.

- > *Naively one would say, compile the Fortran version, append an underscore*
- > *(when appropriate) to routine names, and just link those libraries to*
- > *your object files.*
- >
- > *What are the gotchas, and what are the real stumbling blocks here?*
- > *Please spell it out in as much detail as you can muster. This situation*
- > *really needs to improve.*

I have spelled it out; this is not intended as an attack but as constructive criticism.

Lapack is free and has been useful to me as a reference and for accuracy testing my own code.

I realize that a lot of people have spent considerable effort in its design and I am grateful for that. And it works.

Thanks,

> V.

Jentje Goslinga

my home site is:

<http://www.numerical-algorithms.com/>