

Re: Who uses clapack?

Source: <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2004-12/0328.html>

From: Jentje Goslinga (goslinga_at_telus.net)

Date: 12/11/04

Date: Sat, 11 Dec 2004 02:22:13 GMT

beliavsky@aol.com wrote:

>> *Many people nowadays perform numerical computations in C*

>> *or C++ which are just as suitable as Fortran.*

>

>

> *Fortran 95 still has many advantages over C and C++:*

> *(1) true multidimensional arrays, with a wide range of intrinsic*

> *functions like SUM and MAXVAL and the ability to use array sections*

> *like a(1:10,:,2:10:2).*

That is great, now tell me why the part of Lapack, at least the functions for Eigenvalues and SVD I have looked at does not use any of these.

> *(2) the ability to pass one- and multi-dimensional arrays to functions*

> *in a SIMPLE manner and to have functions returning arrays.*

> *(3) arrays that can start at 0, 1, or any other user-specified integer*

> *(4) intrinsic functions like SIN that are generic to single, double,*

> *and often quadruple precision, for both scalar and array arguments*

> *(5) user-defined PURE and ELEMENTAL functions, to facilitate*

> *parallelization and reduce the need for separate functions handling*

> *scalar and array arguments*

> *(6) better handling of complex numbers*

That is great but I rarely need those and can write the required operations in an hour.

> *(7) a form of DO loop that prevents the user from inadvertently*

> *changing the loop variable. WHILE loops and a DO-ENDDO loop with an*

> *EXIT can be used when more flexibility is needed.*

Come on, I am talking to programmers.

> *(8) the ability to EXIT from nested loops*

> *superior facilities for reading and writing formatted tables of numbers*

> *(9) a cleaner, less error-prone syntax. It is natural for the*

> *end-of-line to terminate a statement by default, and to allow*

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

- > continuation lines when needed. A misplaced semi-colon won't
- > drastically change the meaning of a Fortran code.

Sigh.

- >>There are many engineering applications with substantial
- >>number crunching which are written in C.
- >
- >
- > In my company substantial number crunching is done in Excel, because
- > some people refuse to learn anything else. Many C programmers seem to
- > have similar blinders.

I never called myself exclusively a C programmer and may have written a lot more Fortran code than you have.

- >>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?
- >
- >
- > Much scientific computation that used to be done on commercial UNIX
- > platforms is now being done on Linux, where there are many commercial
- > Fortran 95 compilers. The Lahey/Fujitsu Linux compiler costs as little
- > as \$250, and the Intel Fortran compiler is free for non-commercial use.
- > G95 at <http://www.g95.org> is 99.9% of a free, open-source Fortran 95
- > compiler for Linux, Windows, Mac OS X, and other platforms. Gfortran is
- > getting there.

It all depends on the application. Nowadays "systems" usually have a database component and sometimes internet connectivity and much more and it is natural for them to be composed of modules in various different languages. I have seen many engineering applications in the C and C++ languages.

I have written a large system with a Windows user interface, database and internet connectivity in C++.

The beauty of C++ is that the notion that the code will have to connect to modules written in other languages has been an intrinsic part of the design of the language from the beginning.

- >>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.
- >
- >
- > Lapack documentation is at <http://www.netlib.org/lapack/lug/index.html>

Re: Who uses clapack?

> , also as a printed book.

I haven't looked there but assume these are user manuals for people like you. I am talking about design here.

>>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?
>
>
> Fortran 90/95 has dynamic memory allocation.

Precisely, that is what my post implies.

>>The functions generally are too large with the familiar gobs
>>of variables at the start and no attempt at restricting scope.
>
>
> Fortran 90/95 functions and subroutines can have OPTIONAL arguments, so
> that the user may only need to specify a few to get the default
> behavior. The LAPACK95 interface to the LAPACK library already exists,
> and using it, a subroutine call such as
>
> call SGELSD(M, N, NRHS, A, LDA, B, LDB, S, RCOND, RANK, WORK,LWORK,
> IWORK, INFO)
>
> can be replaced with just
>
> CALL LA_GELSD (A, B)
>
>
>>There are no Modules containing related functions but every function
>
> is separate.
>
> Fortran 90/95 has something called a "MODULE" that does exactly this.
> To get compile-time checking of procedure interfaces, the
> "pre-historic" #include statements of C/C++ are not needed.
>
>
>>The single precision versions are useless in my opinion but
>>others may disagree, at the very least relegate them to
>>recognizeable modules so users who do not want them can leave
>>them out.
>
>
> The single precision codes of LAPACK start with the letter S, as
> described at <http://www.netlib.org/lapack/individualroutines.html> , and
> are thus easy to recognize.

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

Yes, I know that. There is rarely an excuse for duplicating code such as for different precisions for example, this is really a design problem. Beware... before you make that copy.

- > *Ideally, each of the BLAS calls could be replaced by calls to Fortran*
- > *90/95 intrinsic functions or array operations, improving readability.*
- >
- > *You have indirectly made a good case for writing the next version of*
- > *LAPACK in Fortran 95.*

I am making lots of useful and constructive comments.

- > *In general, one should try to gain a current understanding of a*
- > *subject, in this case Fortran, before making recommendations about it*
- > *to others.*

Exactly, that's why I sometimes take the liberty to post. I have programmed in Fortran since 1980, in C since 1985 and in C++ and Assembler since 1995.

There are Fortran 90/95 tutorials at

>

http://www.dmoz.org/Computers/Programming/Languages/Fortran/Tutorials/Fortran_90_and_95/

Enough!

The purpose of my post was not to discredit Fortran and I never did, this is not another language war. My article makes some genuinely useful and well researched suggestions for improvement which are based on a lot of thinking. You may think otherwise.

Jentje Goslinga