

Re: Reported any bugs in C-LAPACK routine DSPEVX?

Source: <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2004-10/0450.html>

From: Andreas Krebs (krebs.andreas_at_gmx.de)

Date: 10/25/04

Date: Mon, 25 Oct 2004 15:56:29 +0200

Javier Almeida wrote:

> *Andreas Krebs <krebs@math.tu-cottbus.de> wrote in message news:<4178B06B.4070108@math.tu-cottbus.de>...*

>

>>Hi,

>>I do not know much about CLAPACK since I prefer to declare
>>and call the latest Fortran LAPACK routines directly. The only problem
>>with this is that Fortran stores matrices in columns and C stores
>>matrices in rows. So, you have to call the matrix routines of
>>LAPACK exchanging 'n' by 't' and 't' by 'n'. The first thing I would
>>do in your case is one of the following.

>>

>>1. Expand the SP-matrix to a GE-matrix, compute the eigenvalues with
>> dgeevx.

>>

>>2. Call the dspevx fortran-routine directly. Drop me a line if you
>> need an example.

>>

>>Regards, Andreas

>>

>

>

> Hello, thanks for your suggestions.

> The fact that fortran stores matrices column-wise and C row-wise
> is no longer a problem for me since CLAPACK (and LAPACK++) also works
> like fortran and I've get used to store matrices this way. I've used
> before routine dgeevx but as far as you involve generic matrices in
> your problem you have to be aware of the possibility of getting
> complex eigenvalues/eigenvectors (sometimes with null precision
> machine imaginary parts, but imaginary after all).

Yes, but if your general input matrix is symmetric, then all imaginary parts should be neglectable, i.e., very close to 0, and one can work around this problem. But

I agree that a direct call is better.

sci.math.num-analysis: Re: Reported any bugs in C-LAPACK routine DSPEVX?

- > *So I think the*
- > *solution may come from your second suggestion, but I'm not really sure*
- > *I know how to link fortran routines directly from C++, maybe I should*
- > *include a header in my programs (with correct type conversions) and*
- > *link against lapack_F77.a ?.*
- >
- > *I would be greatly thankful if you could help me with an*
- > *example.*

For the dstevx routine I do it like this. Remark the additional _ behind the routine name needed by C:

Declaration in src.c:

```
void dstevx_(char *, char *, int *, double *, double *,
            double *, double *, int *, int *,
            double *, int *, double *, double *, int *, double *, int *,
            int *, int *);
```

Since Fortran knows only call by reference, all dummy arguments are declared as pointers.

Of course, the declaration can be done also in a header.h file which is included into src.c. This should be preferred if you need the routine in more than one source file or if plan to create your own reusable lapack.h header file.

Maybe you need an additional
extern C

when declaring the routine in C++. I am not sure about this.

Calling the routine:

When calling the routine, we need the addresses of all arguments, i.e. the &var for the scalars and var = &var[0][0] for the matrices and var = &var[0] for the vectors. Probably, you get compiler warnings for using &var[0][0]. That's why I prefer var, here diag, ndiag.

```
dstevx_(&jobz, &jobz, &gqnm1,
        diag, ndiag, &m_one, &zero, &il, &iu, &zero,
        &nfound, &gqk[1], evecs, &gqnm1, work, iwork,
        ifail, &info);
```

Of course, all arguments of dstevx have to be declared firstly and be assigned appropriately, e.g. jobz='N'; zero=0.; m_one=1.; etc.

Linking lapack and blas:

Using intel's mkl (their lapack and blas version for linux)

```
-L/opt/intel_fc_80/mkl70/lib/32 -lmkl_lapack -lmkl_ia32 -lguid -lpthread
```

works fine. Using gcc and g77

probably

```
-L$(LAPACKDIR) -lblas -llapack -lblas
```

sci.math.num-analysis: Re: Reported any bugs in C-LAPACK routine DSPEVX?

will not be enough since the program needs to link some fortran77 specific libraries, e.g. g2c or f2c. To get these you can proceed as follows:

Write a small fortran test program which uses a lapack routine, compile and link it successfully. Use the linker `g77 -v` to make the linker eloquent. It will show you all included libraries. Then you copy these libraries into the C linker command, i.e.,
`gcc -o out src.o -L$(LAPACKDIR) -lblas -llapack -lblas -L.... -L... -L... -L...`

where ... is replaced by the fortran specific libraries.

This method works also with the intel compilers `ifort`, `ifc` and `icc`, of course, with different fortran libraries.

There exist other ways of including lapack routines into C. But I think this is the most direct approach. Hope this helps. Maybe google gives you more detailed information.

Regards, Andreas