Lapack wrapper for Matlab™

by Rémi Delmas
supervised by Julien Langou

Innovative Computing Laboratory
COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF TENNESSEE

Summary

1. Why a matlab wrapper?
2. Implementation
3. Playing with Matlab

1. Why a matlab wrapper?

- Easy to do : useful information has already been extracted from the fortran source files for the C wrapper.
- Matlab already uses Lapack. With the wrapper I can :
  - Use the optimized functions from the latest release
  - Use the new functions, not implemented in Matlab
- For developers and expert users: debugging, benchmarking, made easy.
- Pedagogic value.

2. Actual implementation

- Very simple wrapper :
  - Each LAPACK function is wrapped by a .c file.

2. Interface with Matlab

- An example of call :

```
[val, vec]= eig(A);
```

```
call dsyev(jobz, uplo, n, a, lda, w, work, lwork, info)
```

```
[jobz, uplo, n, vec, lda, val, work, lwork, info]=    ...
        lapack_dsyev(jobz, uplo, n, a, lda, w, work, lwork, info);
```

2. Actual implementation

- Mex-files : C programs that are pre-compiled and called from within Matlab =>allows for the speed of C, with the Matlab interface on top.
- On input and output of the C function, are mxArrays structures. What we do :
  - Getting the pointer to the data
  - Convert the data if necessary
  - Call the fortran function.
- Design choices :
  - Automatic generation
  - Because of that, no simplification in the list of arguments (N, LDA, etc, are still present).
  - Every parameter is in/out.

2. Actual implementation

- A few issues/remarks :
  - Single precision :

    Some convenient functions do not exist in single precision, so we have to use the underlying, more complicated ones.
  - Complex numbers :
    - In Fortran, each complex number is stored with the real part and the imaginary part at the same location in memory.
    - In matlab, 2 separate matrices are stored.

      => extra cost (memory and cpu). The memory allocation in unavoidable because that writing input parameters is forbidden and leads to segfault.

3. Playing with matlab

- Easy to wrap “by hand” these calls into a nice interface.
  - Ex : syev* : **[val,vec] = syev(a,param)**
    - **val, vec : eigen information**
    - **a : matrix, single or double, real or complex**
    - **param : algorithm used :**
      - **Divide & Conquer**
      - **Multiple Relatively Robust Representations (“Holy Grail”)**
      - Bisection & Reverse Iteration
      - QR
- Using new Lapack functions :
  - MRRR (*syevr), the holy grail.

3. Playing with matlab

- It is easy to change the Lapack library used by matlab.
  => Benchmarking made easy.

- Idea :
  Change the behaviour of existing matlab commands. Eg, I would like “eig” to call the new syevr.

- Problem :
  Testing hard to do. The lapack testing suite cannot/can hardly be used (because of the way the mex function is called).

Conclusion for the MATLAB Wrapper

- Mostly useful :
  - For the dev
  - For who wants to use an “improved Matlab”
  - For people that want to “play” with Matlab

- Some kind of testing needs to be done
- For the user, a nice layer needs to be developped “by hand”

- Webpage : http://icl.cs.utk.edu/~delmas

Conclusion

- 6 weeks of work:
  - 1½ week : script for Matlab + demo code
  - 1½ week : script for C + demo code
  - 3 weeks for testing/debugging the C wrapper
- Still need more documentation though

Conclusion

- Thanks to :
  - Piotr Luszczek for his comments
  - Julie and Julien Langou for their help

- Plans for the future :
  - Maintain the wrappers, improve them, document,
  - Find a way to test the matlab wrapper
  - Certainly change of subject for the next 2½ months (see Julien)
  - Find a job ;)