ATLAS's C-BLAS & C-LAPACK Installation

The SPK core library uses a few linear algebra routines from the BLAS and from the small subset of C LAPACK standard implemented by the ATLAS project. This C implementation should not be confused with the full C implementation of LAPACK that is widely distributed (ex. it is included in RedHat Linux distributions). The ATLAS's implementation of C LAPACK has different routine names and prototypes from the CLAPACK.

Table of Contents

Background

ATLAS

ATLAS stands for Automatically Tuned Linear Algebra Software. ATLAS is both a research project and a software package. This FAQ describes the software package. ATLAS's purpose is to provide portably optimal linear algebra software. The current version provides a complete BLAS API (for both C and Fortran77), and a very small subset of the LAPACK API. For all supported operations, ATLAS achieves performance on par with machine-specific tuned libraries. (ATLAS homepage[1])

BLAS

The BLAS (Basic Linear Algebra Subprograms) are high quality "building block" routines for performing basic vector and matrix operations. Level 1 BLAS do vector-vector operations, Level 2 BLAS do matrix-vector operations, and Level 3 BLAS do matrix-matrix operations. (BLAS homepage[2])

LAPACK

LAPACK (Linear Algebra PACKage) provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision. (LAPACK homepage[3] | CLAPACK homepage[4])

Preparation

The SPK core library uses a few linear algebra routines from the BLAS and the small subset of C LAPACK standard implemented by the ATLAS project. The full LAPACK library is implemented by CLAPACK as well. However, we want the two LAPACK implementations to co-exist peacefully. The two have different interfaces. We use both implementations because: 1) the SPK core library needs only a few functions from the standard, which is covered by the ATLAS's, 2) the ATLAS's functions directly take column major order matrices, which keeps our code clean & simple & efficient, but 3) unfortunately the optimizer later developed by Brad uses eigen value functions which are not yet implemented by ATLAS but provided by CLAPACK. So, in our source code files, we would like to be able to include their headers (something) like this:

```
extern "C"{
  // ATLAS interfaces
  #include <atlas/cblas.h>
  #include <atlas/clapack.h>

  // Standard/widely-distributed CLAPACK interfaces
  #include <f2c.h>
  #include <blaswrap.h>
  #include <clapack.h>
}
```

Here is the outline of what you have to do:

1. Uninstall existing BLAS if any.
2. Build and install the new ATLAS.

ATLAS Installation for The Entire Set of BLAS and a Subset of LAPACK

1. Go to www.netlib.org/atlas[5] and download `atlas3.6.0.tgz` into `/tmp/` on your machine.
2. Unpack the tarball as follows:

   ```
   cd /tmp/
   tar xvzf atlas3.6.0.tgz
   ```

   This will unpack the package into a new subdirectory, `ATLAS`.

   Then, change directory to `ATLAS`.

   ```
   cd ATLAS
   ```

3. Collect information for the complilation/linkage/installation process by specifying a target, `config` which takes an argument that tells the system which ANSII C compiler to use. Enter the following:

   ```
   make config CC=gcc
   ```

   The size of a display page will be asked. Type in an appropriate value.

   ```
      Enter number at top left of screen [0]: <your answer>
   ```

   Say "yes" to the following three questions unless you strongly disagree:

   ```
   Have you scoped the errata file? [y]: y
   Are you ready to continue? [y]: y
   ```

```
use express setup? [y]: y
```

A information gathering process will begin. At the end, you will be asked if you want to proceed with the values that are automatically collected. Inspect the values displayed on screen and let it proceed in building (note: they use "installation" to mean "build") the libraries unless you find something obviously wrong. If you have said "no" to the last question "use express setup?", you'd be prompted to answer to the questions like these:

```
   Enter Architecture name (ARCH) [Linux_P4]:
   Enter File creation delay in seconds [0]:
   Enter Top level ATLAS directory: [/home/Honda/downloads/ATLAS]:
   Enter Directory to build libraries in [$(TOPdir)/lib/$(ARCH)]:
   Enter Library name for ATLAS primitives [libatlas.a]:
   Enter Library name for the C interface to BLAS [libcblas.a]:
   Enter Library name for the Fortan77 interface to BLAS [libf77blas.a]:
   Enter F77 Linker [$(F77)]:
   Enter F77 Link Flags [$(F77FLAGS)]:
   Enter ANSI C compiler (CC) [/usr/bin/gcc]:
   Enter C Flags (CCFLAGS) [-fomit-frame-pointer -O3 -funroll-all-loops]:
   Enter C compiler for generated code (MCC) [/usr/bin/gcc]:
   Enter C FLAGS (MMFLAGS) [-fomit-frame-pointer -O]:
   Enter C Linker [$(CC)]:
   Enter C Link Flags [$(CCFLAGS)]:
   Enter Archiver [ar]:
   Enter ARchiver flags [r]:
   Enter Ranlib [echo]:
   Enter BLAS library []:
   Enter General and system libs [-lm]:
```

Upon the completion, you'll see a message like this:

```
Configuration completed successfully.  You may want to examine the make include
file (Make.Linux_P4) for accuracy before starting the install with the command:
make install arch=Linux_P4
```

The value of `arch` at the last line above reflects the architecture of your installing machine. Thus, the actual value displayed may be different from `Linux_P4`.

Come up with a value that best describes your architecture and set it as a shell variable. For the sake of demonstration, I will use `LinuxRH9_P4` to describe RedHat Linux 9.0 on Pentium 4.

```
export ARCH=LinuxRH_P4
```

4. Build the library by entering the following:

   ```
   make install arch=$ARCH
   ```

   The string on the right hand side of `arch` assignment is arbitrary. If you want to change it to something else, do it, but the consequent instructions will assume you replace the text "Linux_P$" with the string you chose here.

5. Install the package (ie. libraries and headers).

   ```
   mv lib/$ARCH/liblapack.a lib/$ARCH/libatlas_lapack.a
   su

   mkdir /usr/local/lib/atlas
   mkdir /usr/local/include/atlas
   ```

```
cp lib/$ARCH/libatlas.a /usr/local/lib/atlas
cp lib/$ARCH/libcblas.a /usr/local/lib/atlas
cp lib/$ARCH/libatlas_lapack.a /usr/local/lib/atlas
cp lib/$ARCH/libf77blas.a /usr/local/lib/atlas
cp include/cblas.h /usr/local/include/atlas
cp include/clapack.h /usr/local/include/atlas
ln -s /usr/local/lib/atlas/libatlas.a /usr/local/lib/libatlas.a
ln -s /usr/local/lib/atlas/libcblas.a /usr/local/lib/libcblas.a
ln -s /usr/local/lib/atlas/libatlas_lapack.a /usr/local/lib/libatlas_lapack.a
ln -s /usr/local/lib/atlas/libf77blas.a /usr/local/lib/libf77blas.a
```

If gcc failed to find the libraries, that's probably because `/usr/local/lib` is not in the linker's normal search path. You can fix this problem by including the library directory in the searc path. Enter the following from the terminal.

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

6. We want to test the above installation as well as testing the sanity of the built libraries. To do that, let's make changes to a base Make input file so that the subsequent tests will be attempted to use libraries in the installed directory rather than the ones in /tmp/ATLAS/. Open `Make.$ARCH` with an editor and edit the lines, where `ATLASlib`, `CBLASlib` and `LAPACKlib` variables are set, to the following new values:

```
ATLASlib  = -latlas
CBLASlib  = -lcblas
LAPACKlib = -latlas_lapack
```

7. Run the tests included in the disctribution:

```
make sanity_test arch=$ARCH
```

If all things go well, you'll see messages like these:

```
DONE BUILDING TESTERS, RUNNING:
SCOPING FOR FAILURES IN BIN TESTS:
fgrep -e fault -e FAULT -e error -e ERROR -e fail -e FAIL \
        bin/Linux_PII/sanity.out
8 cases: 8 passed, 0 skipped, 0 failed
4 cases: 4 passed, 0 skipped, 0 failed
8 cases: 8 passed, 0 skipped, 0 failed
4 cases: 4 passed, 0 skipped, 0 failed
8 cases: 8 passed, 0 skipped, 0 failed
4 cases: 4 passed, 0 skipped, 0 failed
8 cases: 8 passed, 0 skipped, 0 failed
4 cases: 4 passed, 0 skipped, 0 failed
DONE
SCOPING FOR FAILURES IN CBLAS TESTS:
fgrep -e fault -e FAULT -e error -e ERROR -e fail -e FAIL \
        interfaces/blas/C/testing/$ARCH/sanity.out | \
                fgrep -v PASSED
make[1]: [sanity_test] Error 1 (ignored)
DONE
SCOPING FOR FAILURES IN F77BLAS TESTS:
fgrep -e fault -e FAULT -e error -e ERROR -e fail -e FAIL \
        interfaces/blas/F77/testing/$ARCH/sanity.out | \
                fgrep -v PASSED
make[1]: [sanity_test] Error 1 (ignored)
DONE
```

8. Done. You may remove `/tmp/ATLAS/` entirely.

Notes

1. http://www.netlib.org/atlas/
2. http://www.netlib.org/blas/
3. http://www.netlib.org/lapack/
4. http://www.netlib.org/clapack/
5. http://www.netlib.org/atlas/