arpackpp installation

The arpackpp library consists of header files and can be installed without compiling. However, to compile the examples or a program that includes these headers, it is necessary to install some libraries.

System Libraries: GFORTRAN, BLAS, LAPACK, ARPACK

These libraries can be installed via a package manager, for example when using APT:

```
$ sudo apt-get update -qq
$ sudo apt-get install -y gfortran libopenblas-dev liblapack-dev libarpack2-dev
```

Currently there is no package for SuperLU 5.0 so you need to install that separately (see below).

BLAS:

Alternatively OpenBLAS can be obtained from GitHub: https://github.com/xianyi/OpenBLAS The script

```
$ ./install-openblas.sh
```

will install OpenBLAS into the ./external directory.

ARPACK:

The actively maintained "new generation" package from GitHub https://github.com/opencollab/arpack-ng can be installed via

```
$ ./install-arpack-ng.sh
```

into the external directory.

The BLAS and LAPACK routines required by the ARPACK FORTRAN package are distributed along with the software.

Most classes defined by arpackpp do not require BLAS and LAPACK routines other than those distributed with ARPACK (classes for band and dense matrices are the only exception). However, many examples included in the arpackpp "examples" directory require some routines from these two packages that are NOT included in the ARPACK library.

Since arpackpp is a collection of class templates in c++ (which means that one can install it without compiling even a single file), and since one should use vendor-optimized versions of the BLAS and LAPACK if they are available, these libraries were not included in the ARPACK++ distribution.

LAPACK can be obtained from the URL: http://www.netlib.org/lapack/. The BLAS is included in the LAPACK distribution. The LAPACK library on your system must be the public release (the current release is version 3.0). Since LAPACK includes a subset of the BLAS files, the user must take some care while installing this libraries to avoid code duplication.

SUPERLU (version 5.0):

When installing SuperLU, the user must specify what BLAS library is used in the "make.inc" file (when using the old Makefiles). The script

```
$ ./install-superlu.sh
```

downloads and installs SuperLU5.0 from http://crd-legacy.lbl.gov/~xiaoye/SuperLU/superlu_5.0.tar.gz into the external directory. It will search for BLAS in the external directory. You can pass the environment variable BLAS=SYSTEM to the install script to use the system BLAS. Note, you should use the same BLAS for compiling SuperLU, that you will use when compiling the arpackpp examples (or your own code).

UMFPACK and CHOLMOD:

These libraries are now part of the SuiteSparse package http://faculty.cse.tamu.edu/davis/SuiteSparse/SuiteSparse-4.4.5.tar.gz The script

```
$ ./install-suitesparse.sh
```

installs these together with the METIS package from http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/OLD/metis-4.0.3.tar.gz into the external directory.

Compile Examples (cmake):

Arpackpp supports cmake for the compilation of the examples. To build some examples, including the ones that depend on SuperLU, do

```
$ mkdir build
$ cd build
$ cmake -D SUPERLU=ON ..
$ make
```

For this to work all dependencies need to be installed (either on the system or in the ./external subdirectory). See above for details. This will first find the system BLAS and use it. To point cmake to a different package, e.g. OpenBLAS, do:

```
$ cmake -D SUPERLU=ON -D BLAS_goto2_LIBRARY=../external/libopenblas.a ../
```

Compilation of CHOLMOD and UMFPACK examples can be switched-on via:

```
$ cmake -D CHOLMOD=ON -D UMFPACK=ON ../
```

You can also use ccmake instead of cmake to see all variables and manually overwrite specific paths to ensure the right libraries are being used.

Compile Examples (Makefiles in-source build):

Currently we still support standard Makefiles and in-source build:

Arpackpp example directories contain Makefiles that should be used to compile the examples. For example, to compile example "symsimp" (that can be found in the examples/product/simple directory, you just need to write

```
$ make symsimp
```

File symsimp.cc will be compiled and linked to arpackpp libraries, and an executable file named symsimp will be created.

Compiler-dependent instructions

These compiler-dependent instructions were supplied originally, it is unclear if they are still relevant:

Some compiler-dependent functions and data types used by arpack++ were grouped in the file include/arch.h. Thus, this file should be changed to reflect the characteristics of your system. Because at the present time the library was only compiled with the GNU g++ compiler and tested in a SUN SparcStation, further work must be done in order to allow the use of ARPACK++ in other environments.

Moreover, arpack++ also includes a file, include/arcomp,h, that contains the definition of a class template called arcomplex, created to emulate the g++ complex class when other compilers are being used. arcomplex is the only complex type referenced by other ARPACK++ files, so you must change the definition of this class in order to work with complex numbers if g++ (or CC) is not being used.

UMFPACK instructions

The following details on UMFPACK were supplied originally. It is unclear if they are still relevant:

Presently, the UMFPACK library does not allow the user to supply matrices in compress sparse column (CSC) format, but only matrices in coordinate format. However, once supplied, the matrix is converted internally by the package to the CSC format.

Since arpack++ stores matrices in csc format, a few modifications on the UMFPACK code are needed in order to compatibilize the libraries. These modifications are listed below.

1. If single precision real matrices are to be used, it is necessary to comment out lines 579 and 580 of the ums2fa.f file (to avoid calling subroutine UMS2CO from UMS2FA).
2. If double precision real matrices are to be used, it is necessary to comment out lines 579 and 580 of the umd2fa.f file (to avoid calling subroutine UMD2CO from UMD2FA).
3. If single precision complex matrices are to be used, it is necessary to comment out lines 582 and 583 of the umc2fa.f file (to avoid calling subroutine UMC2CO from UMC2FA).
4. If double precision complex matrices are to be used, it is necessary to comment out lines 582 and 583 of the umz2fa.f file (to avoid calling subroutine UMZ2CO from UMZ2FA).