===================

CLAPACK README FILE

===================

============================================================================================

Version 3.1.1

Release date: February 2008

F2C translation of LAPACK 3.1.1

To get revisions info about LAPACK 3.1.0 and 3.1.1,

read LAPACK-revisions3.1.0.info and LAPACK-revisions3.1.1.info

============================================================================================

This README file describes how and how to install the ANSI C translation of the

LAPACK library, called CLAPACK. CLAPACK must be compiled with an ANSI Standard

C compiler. If the C compiler on your machine is an old-style C compiler, you

will have to use gcc to compile the package.

IMPORTANT NOTE:

You \*CANNOT\* just go to www.netlib.org/clapack, download a routine like

sgesv.c and have it work unless you properly install and link to the

f2c and BLAS routines as described below. If your linker complains about

missing functions, you have probably accidentally neglected this step.

Also, you will need the file "f2c.h" (included with the f2c libraries)

in order to compile these routines.

The default BLAS routines included with CLAPACK in the BLAS/SRC

subdirectory may also be used these will most likely be

slower than a BLAS library optimized for your machine. If you do

not have such an optimized BLAS library, you can get one at

http://www.netlib.org/atlas

==============================================================================

For a fast default installation, you will need to

- Untar clapack.tar and modify the make.inc file (see step 1 below)

- Make the f2c libraries (see step 2 below)

- Make the BLAS library (see step 2 below)

- Make the main library, test it, and time it by simply typing

make

If you encounter difficulties, you may find the installation manual for

the FORTRAN version (INSTALL/lawn81.\*) useful.

Procedure for installing CLAPACK:

==============================================================================

(1) 'tar xvf clapack.tar' to build the following directory structure:

CLAPACK/README.install this file

CLAPACK/BLAS/ C source for BLAS

CLAPACK/F2CLIBS/ f2c I/O functions (libI77) and math functions (libF77)

CLAPACK/INSTALL/ Testing functions and pre-tested make.inc files

for various platforms.

CLAPACK/SRC/ C source of LAPACK routines

CLAPACK/TESTING/ driver routines to test correctness

CLAPACK/make.inc compiler, compile flags and library definitions,

included in all Makefiles.

NOTE: It's better to use gcc compiler on some older

Sun systems.

CLAPACK/clapack.h A header file including C prototypes of all the

CLAPACK routines.

You should be sure to modify the make.inc file for your system. Sample

make.inc files for several platforms are included in the INSTALL

subdirectory.

(2) Build the f2c libraries by doing:

make f2clib

##############################################################################

WARNING: 1) If your system lacks onexit() and you are not using an ANSI C

compiler, then you should change your F2CCFLAGS line in

make.inc to

F2CCFLAGS=$(CFLAGS) -DNO\_ONEXIT

On at least some Sun systems, it is better to use

F2CCFLAGS=$(CFLAGS) -Donexit=on\_exit

2) On at least some Sun systems, the type declaration in

F2CLIBS/libI77/rawio.h: extern FILE \*fdopen(int, char\*)

is not consistent with the one defined in stdio.h. In this case

you should comment out this line.

##############################################################################

(3) To run CLAPACK, you need to create a BLAS library.

The performance of CLAPACK largely depends on the performance

of the BLAS library.

You can either use the reference BLAS library included with

this distribution, which is easy to install but not optimized to be

fast on any particular machine, or else find a version of the

BLAS optimized for your machine.

If you want to use the reference BLAS library included with

this distribution, build it by doing

make blaslib

from the main directory.

If you want to find a BLAS library optimized for your machine,

{f2c'd BLAS wrapper

The f2c translated BLAS interfaces used by CLAPACK (and other f2c'd codes)

unfortunately often don't quite match the Fortran interfaces on various

platforms. Consequently, it is difficult to use f2c'd codes with high

performance BLAS routines, which may substantially impede the performance

of these codes. These simple wrappers provide a way around this difficulty

by providing f2c style interfaces (preceded with "f2c\_" to avoid name

collisions) that call through to an underlying CBLAS or F77 BLAS.

f2c.h: The f2c header file

blaswrap.h: A header file to be included in f2c codes that will use the

wrapper. Just #include it at the top of an f2c generated file.

cblaswr.c: A wrapper around the CBLAS interface. This interface is

provided, for instance, by ATLAS (see www.netlib.org/atlas)

fblaswr.c: A sample wrapper around a conventional Fortran BLAS interface.

This works on a Sun platform, but will require substantial tinkering

on platforms with different Fortran calling conventions.

cblas.h: A header file for the CBLAS interface.

fblaswr.h: A header file for the (Sun-style) Fortran BLAS interface

[cz]rotg.c, csrot.c, zdrot.c: Complex Givens rotation routines. For some reason, C interfaces

to these routines were not specified in the CBLAS interface.

}

(4) Compile and run the BLAS TESTING code by doing:

cd CLAPACK/BLAS/TESTING; make -f Makeblat2

cd CLAPACK/BLAS

xblat2s < sblat2.in

xblat2d < dblat2.in

xblat2c < cblat2.in

xblat2z < zblat2.in

cd CLAPACK/BLAS/TESTING; make -f Makeblat3

cd CLAPACK/BLAS

xblat3s < sblat3.in

xblat3d < dblat3.in

xblat3c < cblat3.in

xblat3z < zblat3.in

Inspect the output files \*.SUMM to confirm that no errors occurred.

{NOTE: If a compiling error involving \_atexit appears then see information

within the WARNING above.}

{NOTE: For the highest performance, it is best to use a version of the BLAS

optimized for your particular machine. This may be done by modifying

the line

BLASLIB = ../../blas$(PLAT).a

in CLAPACK/make.inc to point to the optimized BLAS.

Link with BLAS which provides the standard CBLAS interface

==========================================================

If you are using a version of the BLAS which provides the standard

CBLAS interface (e.g. ATLAS), you need to add the appropriate "wrapper" library.

you can make the wrapper library libcblaswr.a by typing

"make cblaswrap" from the main directory. For this setup

(ATLAS with the CBLAS wrapper), the BLASLIB line might look like

Example:

Modification to make.inc

CC = gcc

BLASLIB = ../../libcblaswr.a -lcblas -latlas

Creation of libcblaswr.a : make cblaswrap

Link with BLAS which Fortran calling interface

===============================================

Two possibilities:

- add -DNO\_BLAS\_WRAP to the CC variable to when compiling and no need of a "wrapper" library

Example:

CC = gcc -DNO\_BLAS\_WRAP

BLASLIB = -lgoto -lpthread

- add the sample Fortran calling interface (libfblaswr.a) for systems with

Sun-style Fortran calling conventions is also provided; however,

this interface will need modifications to work on other

architectures which have different Fortran calling convensions.

See the README in the BLAS/WRAP subdirectory for further information.

Example:

CC = gcc

BLASLIB = ../../libfblaswr.a -lgoto -lpthread

Creation of libfblaswr.a : make fblaswrap

}

(5) Build the archive containing lapack source code by doing:

cd CLAPACK/SRC; make

(6) Compile the matrix generation software, the eigenroutine TESTING

code, the linear system TESTING code, and run the LAPACK tests

by doing:

cd CLAPACK/TESTING/MATGEN; make

cd CLAPACK/TESTING; make

Inspect the output files \*.out to confirm that no errors occurred.

I. Compile the matrix generation software, the eigenroutine TESTING code,

the linear system TESTING code, and run the LAPACK tests separately

by doing:

cd CLAPACK/TESTING/MATGEN; make

cd CLAPACK/TESTING/EIG; make

cd CLAPACK/TESTING/LIN; make

cd CLAPACK/TESTING; make

II. After the executable files and libraries have been created for each

of the compiles, the object files should be removed by doing:

make clean

III. Each 'make' may be accomplished just for one or a subset of the

precisions desired. For example:

make single

make single complex

make single double complex complex16

Using make without any arguments will compile all four precisions.

James Demmel

Xiaoye Li

Chris Puscasiu

Steve Timson

UC Berkeley

Sept 27 1993

{Revised by Susan Ostrouchov and Jude Toth}

{The University of Tennessee at Knoxville}

{October 15, 1993}

{Revised by Xiaoye Li and James Demmel}

{University of California at Berkeley}

{November 22, 1994}

{Revised by David Bindel and James Demmel}

{University of California at Berkeley}

{July 19, 2000}

{Revised by Julie Langou}

{University of Tennessee}

{February 2008}

\*\* Install-with-Visual-C++.txt \*\*

[Copyright (C) 2004 SDPA Project]

This file instructs how to install SDPA on Windows

by compiling with Visual C++.

---------------------------------------------------------------------

Part A: Compile LAPACK

A-1:

Download http://www.netlib.org/clapack/CLAPACK3-Windows.zip

and unzip the file in the directory

"C:\My Documents\"

to create the directory

"C:\My Documents\CLAPACK"

A-2:

Launch Visual C++ by clicking

"C:\My Documents\CLAPACK\clapack.dsw"

A-3:

Select "clapack file" in FileView.

A-4:

Select [Build(B)]->[Rebuild(R)].

A-5:

Close Visual C++.

---------------------------------------------------------------------

Part B: Compile SDPA

B-1:

Unzip SDPA source file (windows version) in the directory

"C:\My Documents\"

to create the directory

"C:\My Documents\sdpa"

B-2:

Copy a file

"C:\My Documents\CLAPACK\clapack.h"

into the directory

"C:\My Documents\sdpa"

B-3:

Copy a file

"C:\My Documents\CLAPACK\Release\clapack.lib"

into the directory

"C:\My Documents\sdpa"

B-4:

Copy a file

"C:\My Documents\CLAPACK\BLAS\Release\blas.lib"

into the directory

"C:\My Documents\sdpa"

B-5:

Copy files

blaswrap.h, cblas.h, f2c.h and fblaswr.h

from the directory

"C:\My Documents\CLAPACK\BLAS\WRAP"

into the directory

"C:\My Documents\sdpa"

B-6:

Copy files

libF77.lib and libI77.lib

from the directory

"C:\My Documents\CLAPACK\F2CLIBS\Release"

into the directory

"C:\My Documents\sdpa"

B-7:

Edit

"C:\My Documents\sdpa\f2c.h"

line 10

from: typedef long int integer;

to : typedef int integer;

B-8:

Launch Visual C++ by clicking

"C:\My Documents\sdpa\sdpa.dsw"

B-9:

Select "sdpa file" in FileView.

B-10:

Select [Build(B)]->[Build(B)].

B-11:

Select "libsdpa file" as active project in FileView.

B-12:

Select [Build(B)]->[Build(B)].

B-13:

Find a console executable

"C:\My Documents\sdpa\Release\sdpa.exe"

and a static library

"C:\My Documents\sdpa\libsdpa\Release\libsdpa.lib"

B-14:

Compile and execute other example projects (option).

---------------------------------------------------------------------

Part C: NOTE

Some SDPA format files are using UNIX line feed character '\n'.

Such files must be converted into WINDOWS line feed character '\n\r'

in order to be solved by a console executable sdpa.exe.

To convert, the following PERL command is useful.

$ perl -pe 's/\n/\r\n/' unix-file > windows-file

If your Visual C++ is Visual C++ .NET, the single thread DLL

might occur some troubles.

To use the multithread runtime DLL instead of the single thread DLL,

You must also make sure your application is linking with the Debug

multithreaded DLL run-time library or the Multithreaded DLL run-time

library. To check this setting do the following in Visual C++:

1. Select Settings from the Project menu.

2. Click the C/C++ tab.

3. In the Category drop-down list, select Code Generation.

4. In the Use run-time library drop-down list, select Multithreaded

DLL for the Win32 Release configuration, or select Debug

Multithreaded DLL for the Win32 Debug configuration.

Once you have changed this setting, you must rebuild your project.

---------------------------------------------------------------------