Installing SuperLU

Julio C.

May 12, 2020

Contents

If you find any problem or something in this guide makes no sense please contact the author.

Please also refer to the original documentation in case you are having troubles with the installation.

1 Requirements

Before installing this version of SuperLU please check that you have not any other version of SuperLU already installed in your system. If that is the case then proceed to uninstall it.

Once you ensure that SuperLU is not already installed in your system then proceed to check that you have the following packages installed in your system.

- In `Ubuntu 16.04 LTS 64 bits`
  - `cmake 3.5.1-1ubuntu3`
- In `Ubuntu 18.04.2 LTS 64 bits`
  - `cmake 3.10.2`

2 Installation

- Extract the compressed file `/external_src/superLU/superlu_5.2.0.tar.gz` in a folder of your preference. We recommend you to extract it out of the chapchom project folder to avoid adding the files to the git repository. If you do extract it inside the chapchom project folder please extreme precautions when adding your files to the git repository.

- Go to the folder where you extracted the files and type

```
mkdir build
cd build
cmake .. -DCMAKE_INSTALL_PREFIX=../lib
```

```
mkdir build
cd build
cmake .. -DCMAKE_INSTALL_PREFIX=../lib
```

the last line indicates where to perform the installation, I use to install it in the `lib` directory of the SuperLU folder, that is why that value. However, if you have root privileges then you may not need to specify a value for the `CMAKE_INSTALL_PREFIX` variable.

NOTE: If you are installing `Armadillo` with `SuperLU` support then you may need to install `SuperLU` with the flag `-fPIC` (which stands for 'Position Independent Code'), to do so open the `CMakeLists.txt` file in the main folder of `SuperLU` and edit the line where `CFLAGS` are added (in `SuperLU 5.2.0` it is in line `68`). This is how it looks for `version 5.2.0`.

```
set(CMAKE_C_FLAGS "-fPIC -DPRNTlevel=0 -DAdd_ ${CMAKE_C_FLAGS}")
```

- When the process finish then type

```
make
```

You can try `make -j # of processors` instead of `make` to use more processors at compilation time.

- Install it ...

```
make install
```

- And finally, run the test by typing

```
ctest
```

You can check the results of the testing process in the following files

```
build/TESTING/s_test.out  single precision, real
build/TESTING/d_test.out  double precision, real
build/TESTING/c_test.out  single precision, complex
build/TESTING/z_test.out  double precision, complex
```