Short manual for the configuration of SMART+ for applications in Mechanics of Materials

Dimitris Chatziathanasiou

The following manual gives direct instructions to Windows 7 & Windows 8 users on how to:
- Install the Microsoft Visual Studio environment.
- Generate a CMake build which allows to compile the source code of SMART+ in C++ utilizing the library Armadillo, full with its links to LAPACK and BLAS libraries.

What you will need:
- Administrator rights on your pc.
- About 25 GB of free space on your hard drive.
- To be aware of your Windows configuration: 32-bit or 64-bit.
- To be careful about storing all necessary documents and installing all necessary programs in directories named in Linux style: The names of the respective folders should not contain spaces or accentuated letters/special characters not found in the English alphabet.
- Optionally: Student's account. With a student account, you can probably download and install the "Professional" version of Visual Studio. Otherwise, you opt for the "Express" version. The steps to follow will be identical for both versions.

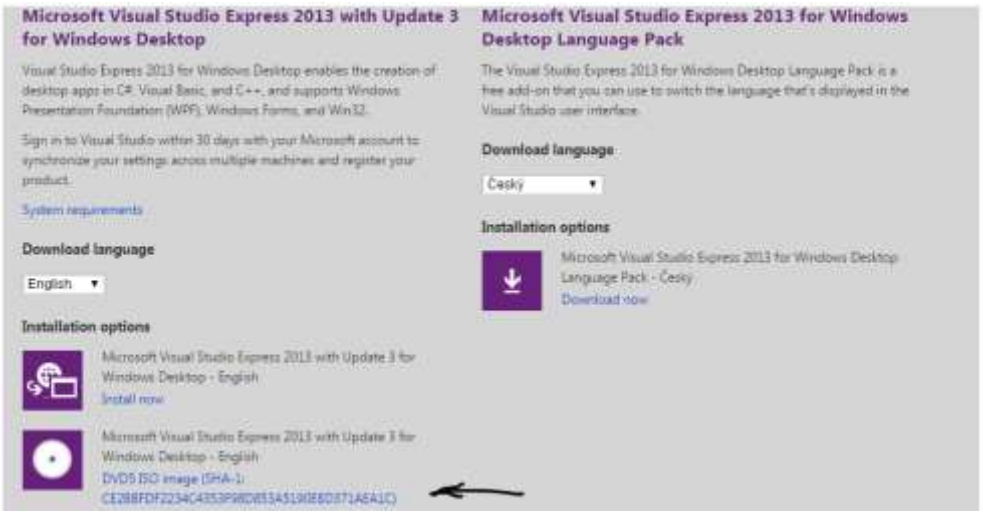
Before the direct instructions, the function of every component in the process of compilation is described:
- The contents downloaded from the github repository is the source code. When compiled, it will yield an executable file, which is the end product of all the work.
- Microsoft Visual Studio executes the compilation of the source code. It needs a "build", which is the set of configurations that link all the parts of the code towards a workable "solution".
- CMake is the tool that builds this solution.
- Armadillo is a library called by the source code.
- BLAS and LAPACK are libraries linked with the solution, but not directly called.

1. Install your version of Microsoft Visual Studio.

If you have your student account, navigate through the website of your academic institution to the site of software resources. Look for the Microsoft Dreamspark link and connect to that page. If you have navigated correctly, the first page to arrive will look like the following screenshot:

In the search bar, look for Visual Studio and then select the latest version (say 2012 or 2013) of Visual Studio Professional and make sure it is tagged as "for Windows Desktop" and "32-bit". Select your preferred language and click on "Add to shopping cart".

If you do not have a student account, go to http://www.visualstudio.com/downloads/download-visual-studio-vs and select Microsoft Visual Studio Express (latest version with latest updates) for Windows Desktop. Select the stand-alone installation files:



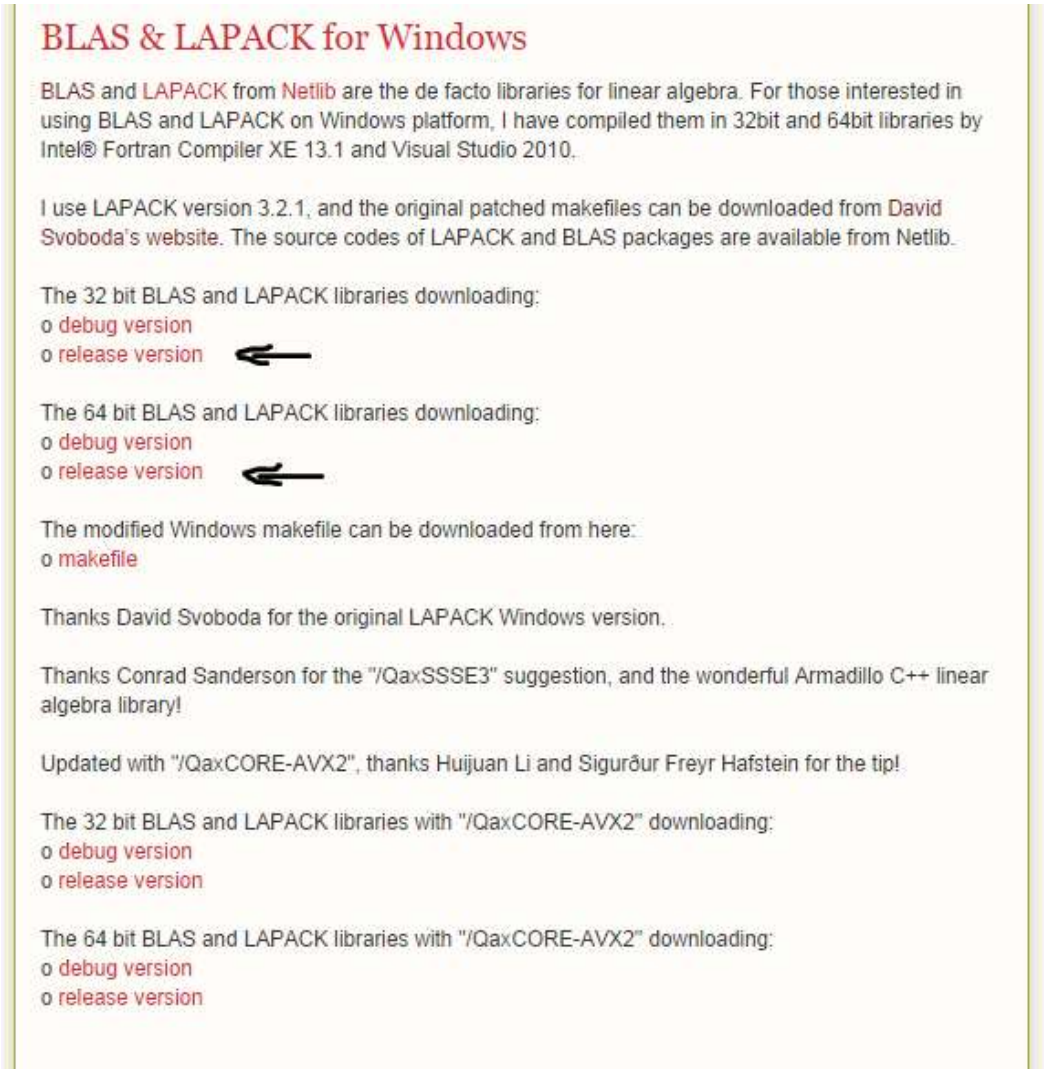
Finalize your download command through the utilities of the website. The download process guides you through the installation of a download manager for Windows applications (Secure Download Manager). Just follow the simple steps given to you through the process and you should end up with a .iso file.

Extract this file preferably with "7-Zip" software. Launch the executable found in the root directory of the folder you just created and follow the instructions given to complete the installation. If you feel uncomfortable with unselecting the various tools suggested by the install wizard, opt for installing all of them.

2. Download and configure Armadillo source code.

http://arma.sourceforge.net/download.html
Here you will find the latest stable version of Armadillo. Download and extract the files. Keep track of the directory where you saved the extracted files. Let's suppose this is C:\Users\pc_user\armadillo-4.400.2 for now. In this folder, navigate to \include\armadillo_bits and modify the file config.hpp:
Uncomment (remove //) the lines
#define ARMA_USE_LAPACK
#define ARMA_USE_BLAS
#define ARMA_BLAS_UNDERSCORE

Now, you need to have access to BLAS and LAPACK pre-compiled libraries. If those do not appear in C:\Users\pc_user\armadillo-4.400.2\examples
under the \lib_win64 folder, then: visit http://ylzhao.blogspot.com/2013/10/blas-lapack-precompiled-binaries-for.html and download and extract the "release" version of compiled BLAS and LAPACK libraries according to the configuration of your Windows (32-bit or 64-bit system).



BLAS & LAPACK for Windows

BLAS and LAPACK from Netlib are the de facto libraries for linear algebra. For those interested in using BLAS and LAPACK on Windows platform, I have compiled them in 32bit and 64bit libraries by Intel® Fortran Compiler XE 13.1 and Visual Studio 2010.

I use LAPACK version 3.2.1, and the original patched makefiles can be downloaded from David Svoboda's website. The source codes of LAPACK and BLAS packages are available from Netlib.

The 32 bit BLAS and LAPACK libraries downloading:
o debug version
o release version

The 64 bit BLAS and LAPACK libraries downloading:
o debug version
o release version

The modified Windows makefile can be downloaded from here:
o makefile

Thanks David Svoboda for the original LAPACK Windows version.

Thanks Conrad Sanderson for the "/QaxSSSE3" suggestion, and the wonderful Armadillo C++ linear algebra library!

Updated with "/QaxCORE-AVX2", thanks Huijuan Li and Sigurður Freyr Hafstein for the tip!

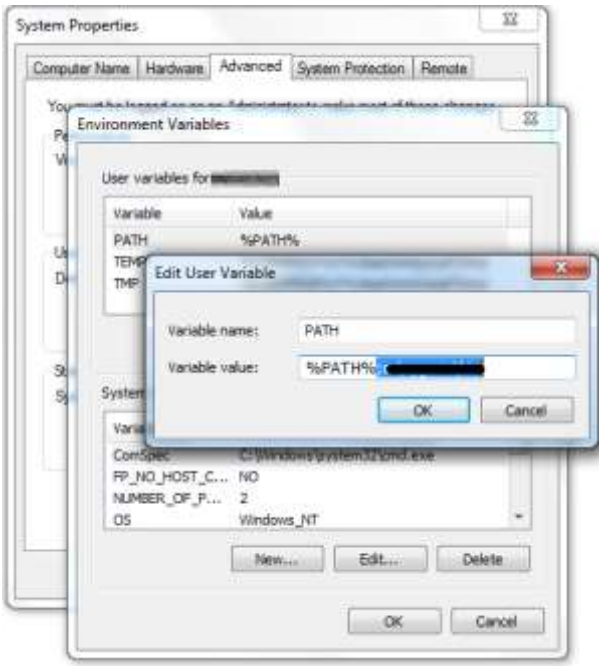
The 32 bit BLAS and LAPACK libraries with "/QaxCORE-AVX2" downloading:
o debug version
o release version

The 64 bit BLAS and LAPACK libraries with "/QaxCORE-AVX2" downloading:
o debug version
o release version

You might as well download the libraries with "/QaxCORE-AVX2" and modify the rest of the procedure accordingly. Make sure to keep track of the directory where you extracted the libraries. Let's suppose it is C:\Users\pc_user\blas_lapack. It should contain two .dll and two .lib files.

Next, you need to modify your system's PATH variable. To do that, go to your System Properties: Control Panel > System (and Maintenance) > System or My Computer > Properties. Both choices lead to the same window. Here, you can also verify the bit-type of your system.

Select Advanced system settings. Click on Environment Variables. Select the PATH variable and click on Edit. In the variable value field, add the latest directory as such:
C:\Users\pc_user\blas_lapack;
and click OK. Make sure to add the semicolon. Close the System Properties by clicking OK consecutively.



http://boost.teeks99.com/
Here you will find Boost++. Open the link of the latest version and download the .exe file which corresponds to your Visual Studio version and your Windows bit system. For example, for Visual Studio 2013 (version 12) and 64-bit Windows, download boost_1_59_0-msvc-12.0-64.exe.
Execute this and you will obtain the Boost++ libraries for your Windows system. Open it and search for the folder containing the pre-compiled libraries. Copy the address of that directory. A valid example would be:
C:\Users\pc_user\boost_1_59_0\lib64-msvc-12.0

Paste this address to the PATH environment variable, just like with Armadillo. Add the semi-colon and click OK.

3. Build the Visual Studio solution for SMART+

If CMake is not already available on your computer, you need to install it. For further information, visit http://www.cmake.org/install/

Open the folder "smartplus", delete the file "CMakeLists.txt" and rename the file "CMakeLists_Windows.txt" to "CMakeLists.txt". Open it and follow the instructions given in the comments. Namely, after "Include Armadillo and Boost++", copy the full path of the "armadillo" library and then its directory path. Copy the directory path of Boost++ too. This time, it's the parent directory of the whole library, not just the folder of the pre-compiled .lib files.

```
set(LIBRARY_OUTPUT_PATH lib/${CMAKE_BUILD_TYPE})

#All actions for Armadillo
#[[
In the first line, copy the full file path for the armadillo
library between the quotation marks. For example, a valid file
path would be:
C:/Documents/armadillo-4.400.2/include/armadillo
Do not delete the quotation marks. Respect the form of the slash
]]
#[[
In the second line, copy the directory path for the armadillo
library between the quotation marks. For example, a valid
directory path would be:
C:/Documents/armadillo-4.400.2/include
Do not delete the quotation marks. Respect the form of the slash
]]
set(ARMADILLO_LIBRARY "")
set(ARMADILLO_INCLUDE_DIR "")
include_directories(${ARMADILLO_INCLUDE_DIR})
```

```
set(LIBRARY_OUTPUT_PATH lib/${CMAKE_BUILD_TYPE})

#All actions for Armadillo
#[[
In the first line, copy the full file path for the armadillo
library between the quotation marks. For example, a valid file
path would be:
C:/Documents/armadillo-4.400.2/include/armadillo
Do not delete the quotation marks. Respect the form of the slash.
]]
#[[
In the second line, copy the directory path for the armadillo
library between the quotation marks. For example, a valid
directory path would be:
C:/Documents/armadillo-4.400.2/include
Do not delete the quotation marks. Respect the form of the slash.
]]
set(ARMADILLO_LIBRARY "C:/Documents/armadillo-
4.400.2/include/armadillo")
set(ARMADILLO_INCLUDE_DIR "C:/Documents/armadillo-
4.400.2/include")
include_directories(${ARMADILLO_INCLUDE_DIR})
```

Then, after "All further actions for BLAS, LAPACK and Boost++", give the name of the libraries for BLAS, LAPACK and Boost++ in the appropriate fields. Save and close the document. You can see in the images above and below the proper way to make your modifications.

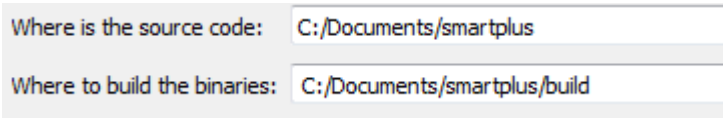
```
endif ()

#All actions for BLAS and LAPACK
#[[
In the first two lines, copy the name of the blas and lapack
libraries before the closing parenthesis, followed by the suffix
.lib. For example, a valid name would be:
blas_win64_MTd.lib
and thus the first line would read:
find_path(BLAS_INCLUDE_DIR blas_win64_MTd.lib)
]]
#[[
In the next two lines, copy the same name before the closing
parenthesis, without the suffix. For example, a valid name would
be:
lapack_win64_MTd
]]
find_path(BLAS_INCLUDE_DIR )
find_path(LAPACK_INCLUDE_DIR )
find_library(BLAS_LIBRARY )
find_library(LAPACK_LIBRARY )
if (BLAS_INCLUDE_DIR AND BLAS_LIBRARY)
```

```
endif ()

#All actions for BLAS and LAPACK
#[[
In the first two lines, copy the name of the blas and lapack
libraries before the closing parenthesis, followed by the suffix
.lib. For example, a valid name would be:
blas_win64_MTd.lib
and thus the first line would read:
find_path(BLAS_INCLUDE_DIR blas_win64_MTd.lib)
]]
#[[
In the next two lines, copy the same name before the closing
parenthesis, without the suffix. For example, a valid name would
be:
lapack_win64_MTd
]]
find_path(BLAS_INCLUDE_DIR blas_win64_MTd.lib)
find_path(LAPACK_INCLUDE_DIR lapack_win64_MTd.lib)
find_library(BLAS_LIBRARY blas_win64_MTd)
find_library(LAPACK_LIBRARY lapack_win64_MTd)
if (BLAS_INCLUDE_DIR AND BLAS_LIBRARY)
```

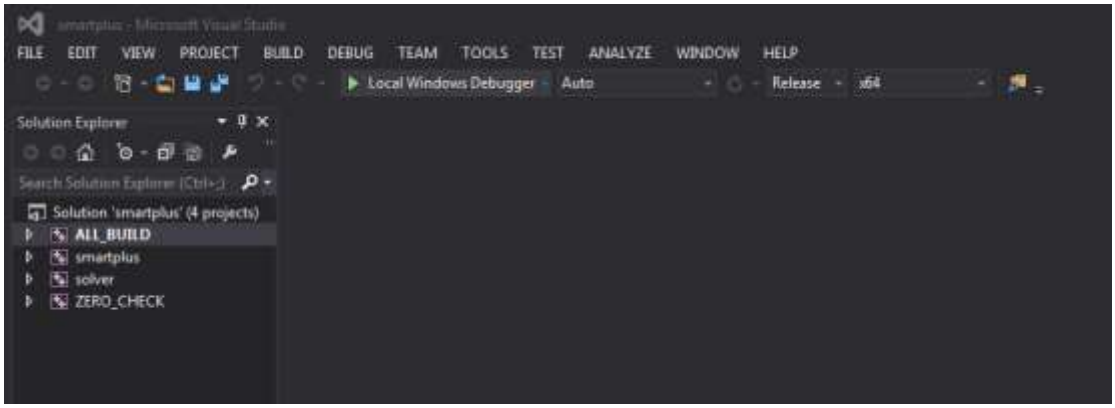
Open CMake. For the source code, give the address of the directory of "smartplus". For the location of the build, give the same directory trailed by the suffix "/build". For example, valid field entries would be:



Click on "Configure". In the new window, choose your version of Visual Studio and bit-type of your Windows. Choose "default native compilers" and click on "Finish". If the solution was built successfully, there should only be one messages in red text in the log, indicating that Armadillo was found. Then, click on Generate.

4. Compile with Visual Studio

Open the file "ALL_BUILD.vcxproj" found in the folder /build with Visual Studio. Make sure that the "Local Windows Debugger" is set to "Auto" (if this field is available) and "Release" and that it indicates the bit-type of your Windows. Click on DEBUG -> Start Debugging. The generated executable should be found in /build/bin/Release/ with the name "solver.exe".



To run the solver, follow the general instructions provided for all operating systems:
"Copy-paste the solver exec to a folder where you want to run the solver executable file for example in a folder exec.
Copy the content of the folder "data" in the same folder."

5. Notes on working with Armadillo and Visual Studio on Windows

- Always write #include <armadillo> at the top of your code.

- There is no need to #include any BLAS or LAPACK libraries.
- Always work with projects when using Microsoft Visual Studio. Do not attempt to compile independent source codes.
- Try to configure the settings of your project to create your executables in the directory you desire. You can do that through the "Property Pages" in Configuration Properties > General. Browse the desired folder in "Output Directory". Give directly your executable the name you like at "Target Name". Then, go execute your programs through the Windows command line.

References

http://www.ru.is/kennarar/sigurdurh/Armadillo64hack.html written by Sigurdur F. Hafstein.

Contact

For further technical issues:

chatziathanasiou.dimitris@gmail.com