How to build DFTFringe

Notes by Franz Hagemann, franz.hagemann@gmx.de

This tutorial is meant for people with little programming experince (like me), too.
Warning: It may take several days to do all this.

First things first: do not use blank spaces in any folder name!

Download and install Cmake. (I have version 3.20.0)
Download and install Code::blocks (I have version 20.03)
Download Qt 5.15.2. (https://download.qt.io/official_releases/qt/5.15/5.15.2/)
Install Qt 5.15.2 with MinGW toolchain. (I selected and used the MinGW 8.1 64 bit toolchain, because for the MinGW 11.2 toolchain, Qmake was missing.)
Do not launch Qt!!! Read on first.
A problem with QT was, that Avast antivirus was screwing it up. I added an exception in Avast's settings for the Qt installation folder. I had to do this prior to the first launch of Qt, otherwise it would get screwed up forever. Maybe other antivirus software will harm Qt too. I don't know.

Build OpenCV 3.4.12 with OpenCL

Download OpenCV 3.4.12 source, unzip and copy to OpenCV root folder of your choice.
Download OpenCV modules:
https://github.com/opencv/opencv_contrib/tree/3.4.12
Unzip and copy into the OpenCV root folder. Add this to system path.

If „add to system path" worked, can be checked with the command
$env:PATH
in Windows powershell; powershell has to be restarted after adding a path.

Start Cmake.

Select your OpenCV root folder as source: opencv-3.4.12/sources.
Define some build folder („Where to build the binaries").

Now Cmake will ask for the compilers. **Select the ones you will use for the entire DFTFringe build.**
I selected the ones that came with Qt, because I used these for the Qt builds too.
In my case, they are located here:
C:\Qt-5.15.2\Tools\mingw810_64\x86_64-w64-mingw32\bin
C: gcc.exe
C++: g++.exe
Fortran: gfortran.exe
Add the path of your chosen compilers to the system path.
Maybe Cmake will not work until you restart it, after adding the path of your chosen compilers to the system path. So restart Cmake.

Click configure
Click configure (until nothing is red anymore)

In Cmake configuration:
uncheck OPENCV_ENABLE_ALLOCATOR_STATS
(this is to avoid the error *gcc: error: long: No such file or directory* during the Code::Blocks build)
check WITH OPENCL
enable BUILD_TESTS
enable BUILD_EXAMPLES (optional, not needed, but maybe also useful to see if the build works)
click configure
click configure (until nothing is red anymore)
click generate…
A Code::Blocks project is created in the build folder (OpenCV.cbp).

You are done with Cmake.

Now, go to your build folder and open the Code::Blocks project.
Click build.
If it looks like Code::blocks isn't doing anything, Press F2 to enable logs panel. Select "Build log" in it.
In my case the build took 46 minutes.

Build qwt

Download qwt.
I ended up using **qwt 6.1.6** because 6.1.3 and 6.2.0 build would not build or the builds were faulty.
Extract.

Open qwt.pro in QT Creator.
In project configuration, define a build folder.
Select a toolkit. The toolkit should use the same compilers as previously selected in Cmake.
I selected a toolkit that uses minGW 8.1 64 bit.
Click configure. Wait for actions in bottom roght corner to finish.
Click build
There was one error during compilation, in qwt_mml_document.cpp
Qt Creator offers a fix: #include "qregularexpression.h"
I clicked the bulb to use the fix. Then clicked build again.
The build was successful.

LAPACK

download from here:

https://github.com/Reference-LAPACK/lapack/releases

I built version 3.9.1 (I could not build 3.10.1 or 3.8)

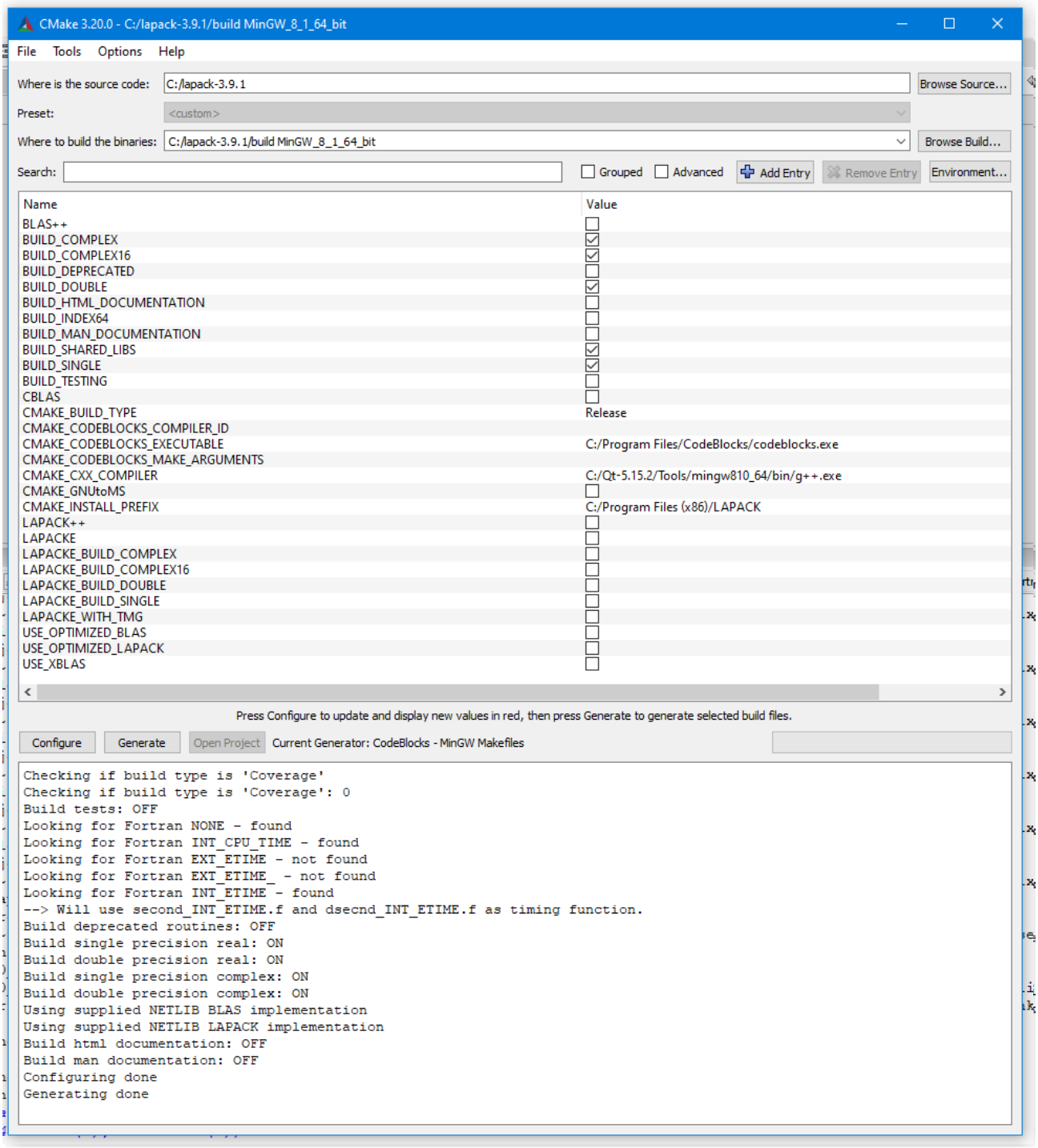
unzip to some folder.

Open **Cmake**, enter the path of the Lapack source folder in field „Where is the source code"

Select a build folder.

Configure. Configure again.

Check: see screenshot (With LAPACKE or CBLAS or BUILD_TESTING checked, the build failed.)



I had to enter the path of my Code::blocks.exe for CMAKE_CODEBLOCKS_EXECUTABLE because it was not found. In my case C:\Program Files\CodeBlocks\ùcodeblocks.exe.

Configure again.

Generate.

There is a known issue with Lapack 3.9, and a serious one. The build will fail.

The fix is:

After running CMake, open the SRC\CMakeFiles\lapack.dir**link.txt** in the LAPACK build folder.

It should have 3 long lines.

Add the following sentence as an option of x86_64-w64-mingw32-gfortran.exe:
-Wl,--allow-multiple-definition

Just paste it at the end, the third line of the link.txt file. Save.

(Fix from: https://github.com/Reference-LAPACK/lapack/issues/305#issuecomment-642780550)

Now, go to your LAPACK build folder and open the Code::blocks project.

Click build.

If it looks like Code::blocks isn't doing anything, Press F2 to enable Logs panel. Select "Build log" in it.

The build took 4 minutes.

I added following Lapack folders to the system path, but maybe it's not needed.

C:\lapack-3.9.1\build_MinGW_8_1_64bit\bin

C:\lapack-3.9.1\build_MinGW_8_1_64bit\include

C:\lapack-3.9.1\build_MinGW_8_1_64bit\lib

Build Armadillo

download from here:

https://sourceforge.net/projects/arma/

Cmake settings: see screenshot

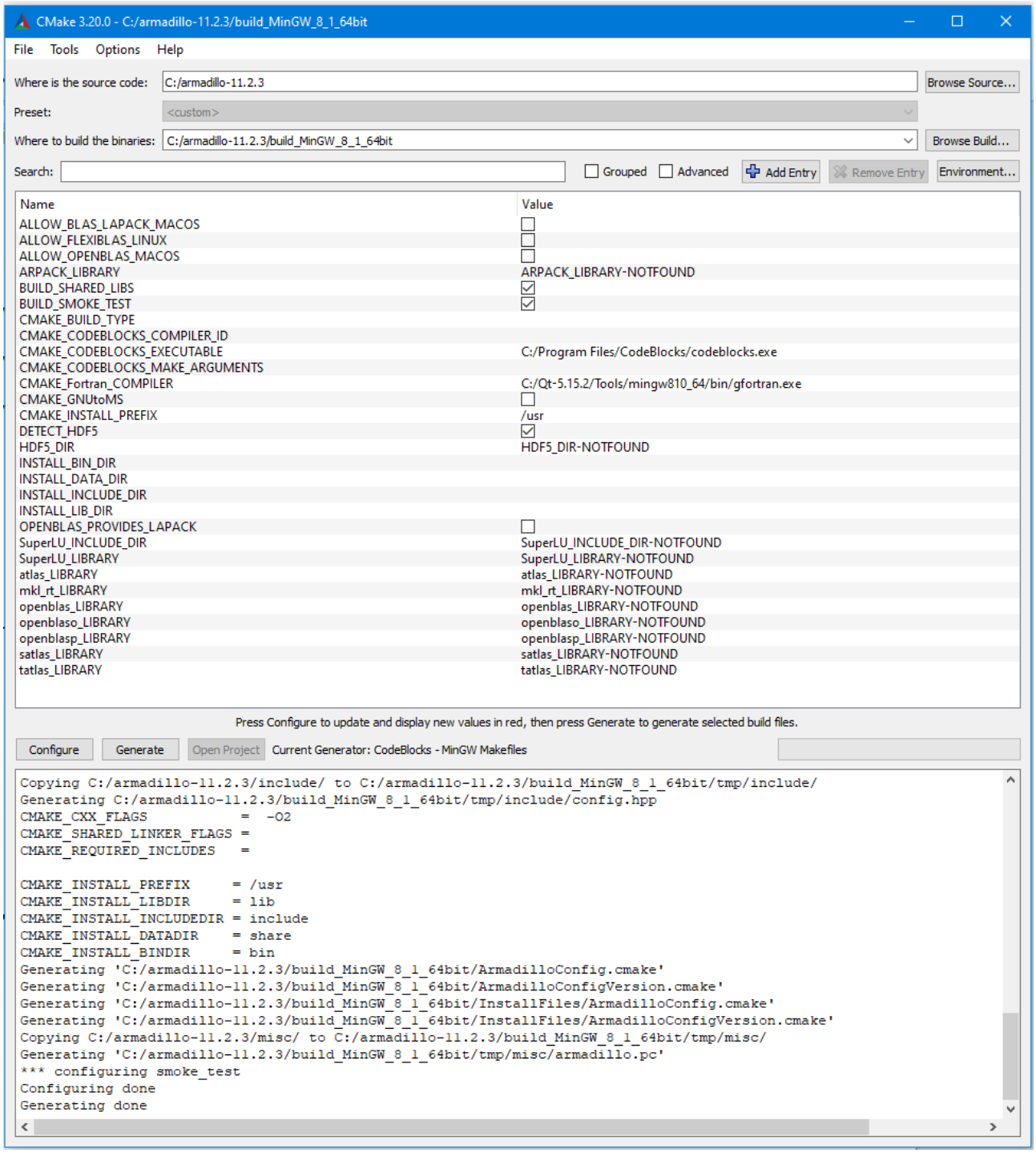
The build in Code::blocks took only 5 seconds.

The Armadillo build does not put the compiled files in a proper build\bin folder. Therefore, I had to collect all the relevant files (armadillo, libarmadillo.dll, libarmadillo.dll.a, and 598 .hpp files) in a folder named bin and a copy of all these files in a folder named armadillo_bits. (I think actually the /armadillo_bits is supposed to contain the .hpp files, and bin the others.)

In my case, the folder structure of my OpenCV build folder and Armadillo bin folder are different from Dale's folders.

For all Armadillo files to be found, I had to create folders named armadillo and armadillo_bits inside my bin folder. Plus, I had to put a copy of the armadillo_bits folder into the armadillo folder.

The bin and the armadillo_bits folders contain all the 604 files except the file named armadillo (no extesion). The folder named armadillo contains only the armadillo file, the .ddl and the .dll.a file.



Build DFTFringe

Qt will most probably not find all the files that are included in the project file.

Some, or maybe all, paths are different than Dale's paths. These paths have to be changed to the corresponding paths on your computer.

For the OpenCV files to be found, I had to include many paths in the DFTFringe.pro file.

After all the changes, that section of my dftfringe.pro fil looks like this:

```
win32 {
    CONFIG( debug, debug|release ) {
      # debug
      LIBS += C:\\qwt-6.1.6\\build_Qt_5_15_2_MinGW_64_bit-Debug\\lib\\qwtd.dll
    } else {
      # release
      LIBS += C:\\qwt-6.1.6\\build_Qt_5_15_2_MinGW_64_bit-Debug\\lib\\qwt.dll
    }
    INCLUDEPATH +=C:\\qwt-6.1.6\\build_Qt_5_15_2_MinGW_64_bit-Debug\\src

    #message("using win32")include
}

INCLUDEPATH += C:\armadillo-11.2.3\build_MinGW_8_1_64bit\bin\armadillo
INCLUDEPATH += C:\armadillo-11.2.3\build_MinGW_8_1_64bit\bin\armadillo_bits

INCLUDEPATH += C:\opencv-3.4.12\build_MinGW_8_1_64_bit\bin
INCLUDEPATH += C:\opencv-3.4.12\build_MinGW_8_1_64_bit
INCLUDEPATH += C:\opencv-3.4.12\sources\include\opencv
INCLUDEPATH += C:\opencv-3.4.12\sources\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\core\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\imgproc\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\photo\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\video\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\objdetect\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\highgui\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\imgcodecs\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\videoio\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\features2d\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\flann\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\calib3d\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\dnn\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\shape\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\stitching\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\superres\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\videostab\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\viz\include
INCLUDEPATH += C:\opencv-3.4.12\sources\modules\ml\include

INCLUDEPATH += C:\qwt-6.1.6\src

LIBS += C:\opencv-3.4.12\build_MinGW_8_1_64_bit\bin\libopencv_core3412.dll
LIBS += C:\opencv-3.4.12\build_MinGW_8_1_64_bit\bin\libopencv_highgui3412.dll
LIBS += C:\opencv-3.4.12\build_MinGW_8_1_64_bit\bin\libopencv_imgcodecs3412.dll
LIBS += C:\opencv-3.4.12\build_MinGW_8_1_64_bit\bin\libopencv_imgproc3412.dll
LIBS += C:\opencv-3.4.12\build_MinGW_8_1_64_bit\bin\libopencv_features2d3412.dll
LIBS += C:\opencv-3.4.12\build_MinGW_8_1_64_bit\bin\libopencv_calib3d3412.dll


LIBS += C:\armadillo-11.2.3\build_MinGW_8_1_64bit\bin\libarmadillo.dll
LIBS += C:\lapack-3.9.1\build_MinGW_8_1_64_bit\bin\liblapack.dll
LIBS += C:\lapack-3.9.1\build_MinGW_8_1_64_bit\bin\libblas.dll
```

Now, open DFTFringe.pro in Qt.

In project configuration:

Select the correct toolchain, the same that was already used for qwt.

Select a build folder.

Click configure.

In build settings, I selected Debug (this is default I think)

Click build.

Most certainly, still many files that are included, will not be found.

Add or change those paths in the dftfringe.pro file, like I did, one after the other. After each one, click build again until you find the next missing one.

Errors and fixes that occurred during compilation:

- Qt at some point offered a fix for <QscrollArea>. I took that fix. (click the bulb)
- Qt at some point offered a fix (remove it) for #include "opencv2/highgui.hpp". I took the fix.
- QT: ..\DFTFringe-6.0 sources\imagehisto.cpp:57:5: error: 'imshow' was not declared in this scope

  I took the fix.
- ..\DFTFringe-6.0 sources\profileplot.cpp:501:41: error: 'surfaceAnalysisTools' is not a class, namespace, or enumeration

  surfaceAnalysisTools *saTools = surfaceAnalysisTools::get_Instance();

  I fixed this by adding the following lines to profileplot.cpp:
  #include "surfaceanalysistools.h"

  #include "ui_surfaceanalysistools.h"
- ..\DFTFringe-6.0 sources\dftarea.cpp:385:9: error: 'namedWindow' is not a member of 'cv' → take the fix
- ..\DFTFringe-6.0 sources\dftarea.cpp:385:26: error: 'WINDOW_NORMAL' was not declared in this scope → take the fix
- ..\DFTFringe-6.0 sources\igramarea.cpp:462:13: error: 'namedWindow' is not a member of 'cv' → take the fix
- ..\DFTFringe-6.0 sources\igramarea.cpp:463:13: error: 'moveWindow' is not a member of 'cv' → take the fix
- An error in igramarea.cpp i forgot to copy → take the fix
- gcc: error: sources: No such file or directory

  windres: preprocessing failed.

  mingw32-make[1]: *** [Makefile.Debug:828: debug/DFTFringe_res.o] Error 1

  Fix: I had to remove blank spaces in my DFTFringe source folder names. Also for the build folder in the build settings in Qt
- unwraperrorsview.cpp line 48: QApplication::desktop()→screenGeometry() is deprecated and causes an error.

  I commented out the deprecated stuff, as I found no current substitute, and wrote

  int height = 900;
- I had to copy the DFTFringe icon located here: C:\DFTFringe_6\DFTFringe-6.0_sources\res

  to C:\DFTFringe_6\DFTFringe-6.0_sources and name it icon.ico, because it was not found.

After this fix, the build was successful.

One or two fixes before the build finished successful, i changed the build setting from debug to release.

When the DFTFringe build was succesful, all the neccessary DLLs have to be collected from the Qt, openCV and LAPACK folders.

Look in an existing, recent DFTFringe installation folder and subfolders, which DLLs are needed.

Replicate all this.

Most of the DLLs were found in this folder:C:\Qt-5.15.2\5.15.2\mingw81_64

Take the Dlls from the Qt folder that corresponds to the compiler you used, not some random folder. I always searched for a DLL in the above folder, if it wasn't obviously an OpenCV, qwt or LAPACK one.

I did not find libquadmath-0.dll, but it seems that it is not needed

You are done!