This document will outline the settings and libraries required to build and execute QRFactor in either Visual Studio or Nsight Eclipse. It is assumed that the CUDA Toolkit (which includes Nsight Eclipse) has already been installed, as well Microsoft Visual Studio if required. The read-in of the system matrix A requires the data to be in Matrix Market format (see the NIST page for more information on MMF, as well as to download the library routines `mmio.h` and `mmio.c`).

1 Running `QRFactor` in Visual Studio 2019 (VS2019)

Place the system matrix file `sysMatA.mtx`, the library routines `mmio.c` and `mmio.h`, the wrapper `mmio_wrapper.cpp`, the source file `QRFactor.cpp`, and the VS2019 files `QRFactor.vcxproj` and `QRFactor.vcxproj.user` in a single directory. The time-dependent input data files can be located in a separate directory that will be set later.

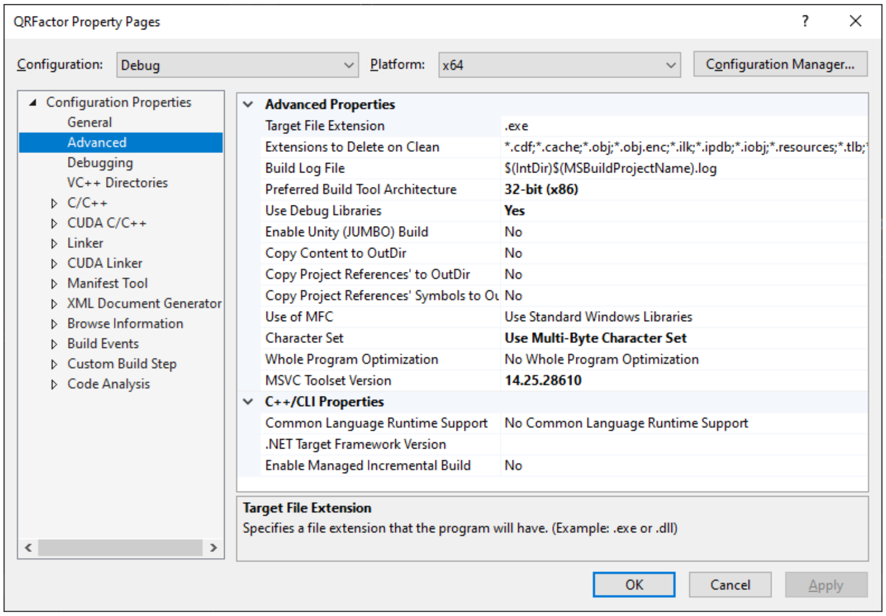
There are several project settings that will need to be changed before the program can be built. These will be discussed below, and screen shots of the relevant settings will be included in appendix A. The following settings will ensure that VS2019 is able to find the correct library files:

- Under Advanced Properties, ensure **MSVC Toolset** is set to 14.25.28610.
- Under C++/General, ensure **Additional Include Directories** contains ./;$(CudaToolkitIncludeDir);$(CudaToolkitIncludeDir)/include;C:/ProgramData/NVIDIA Corporation/CUDA Samples/v10.2/common/inc.
- Under C++/Optimization, ensure **Optimization** is set to Maximum Optimization (Favour Speed) (/O2).
- Under Linker/General, ensure **Additional Library Directories** is set to $(CudaToolitLibDir).
- Under Linker/Input, ensure **Additional Dependencies** includes cusolver.lib;cusparse.lib; cudart_static.lib;kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdlg32.lib;advapi32.lib; shell32.lib;ole32.lib;oleaut32.lib;uuid.lib;odbc32.lib;odbccp32.lib;%(AdditionalDependencies).

With these settings in place, the project can be built by selecting Build → Build QRFactor (Ctrl + B). `QRFactor.exe` can then be run either through VS2019 or via the command line.

2 Running `QRFactor` in Nsight Eclipse

A Visual Studio Settings



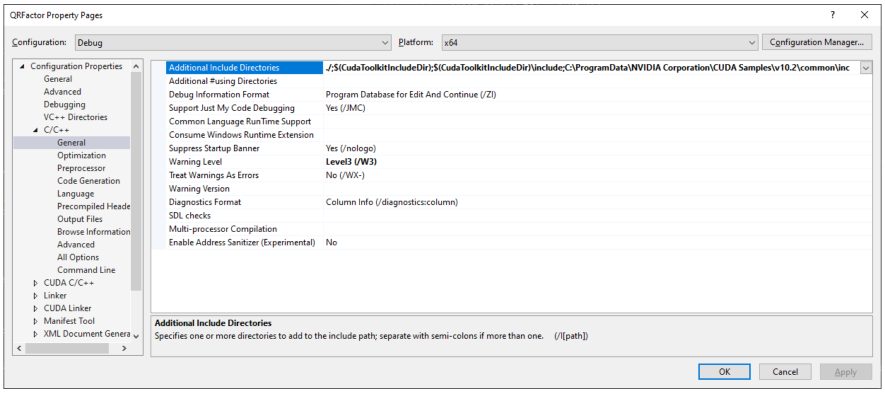
QRFactor Property Pages

Configuration: Debug Platform: x64

Advanced Properties	
Target File Extension	.exe
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;*.resources;*.tlb;
Build Log File	$(IntDir)$(MSBuildProjectName).log
Preferred Build Tool Architecture	**32-bit (x86)**
Use Debug Libraries	**Yes**
Enable Unity (JUMBO) Build	No
Copy Content to OutDir	No
Copy Project References' to OutDir	No
Copy Project References' Symbols to Ou	No
Use of MFC	Use Standard Windows Libraries
Character Set	**Use Multi-Byte Character Set**
Whole Program Optimization	No Whole Program Optimization
MSVC Toolset Version	**14.25.28610**
C++/CLI Properties	
Common Language Runtime Support	No Common Language Runtime Support
.NET Target Framework Version	
Enable Managed Incremental Build	No

Target File Extension
Specifies a file extension that the program will have. (Example: .exe or .dll)

Figure 1: QRFactor Properties → Advanced



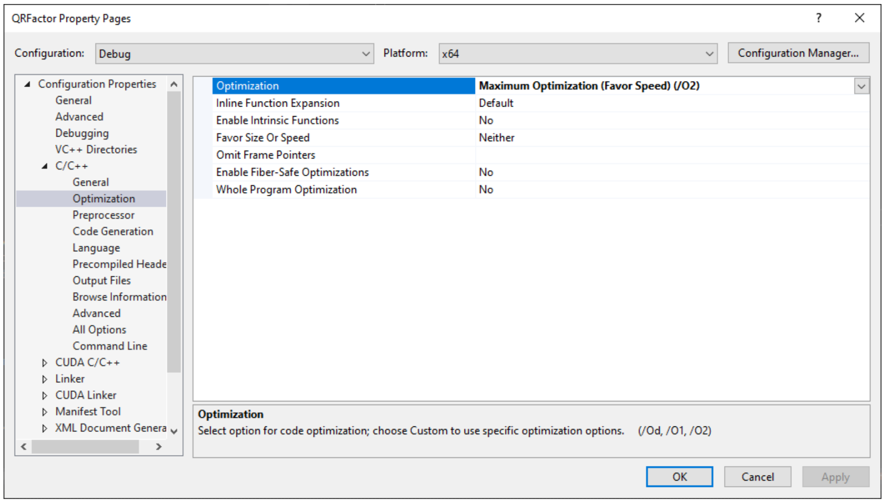
QRFactor Property Pages

Configuration: Debug Platform: x64

Additional Include Directories	**./;$(CudaToolkitIncludeDir);$(CudaToolkitIncludeDir)\include;C:\ProgramData\NVIDIA Corporation\CUDA Samples\v10.2\common\inc**
Additional #using Directories	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Support Just My Code Debugging	Yes (/JMC)
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
Warning Version	
Diagnostics Format	Column Info (/diagnostics:column)
SDL checks	
Multi-processor Compilation	
Enable Address Sanitizer (Experimental)	No

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

Figure 2: QRFactor Properties → C/C++ → General

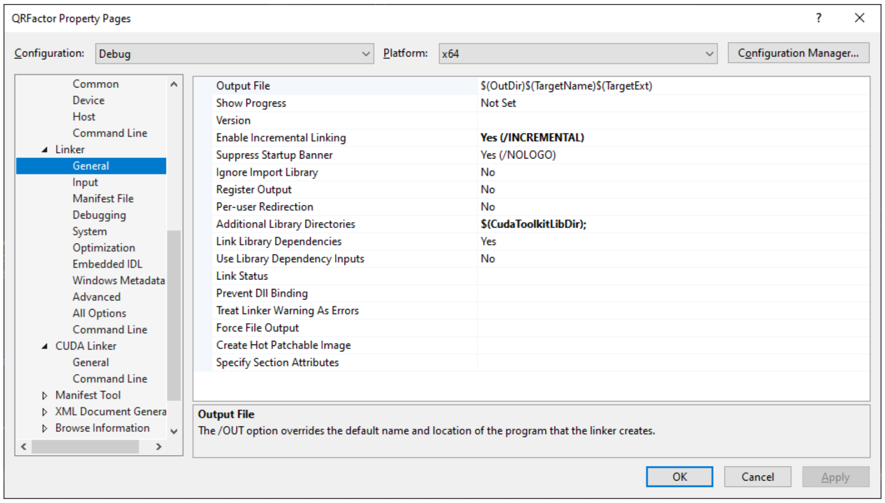


Configuration: Debug Platform: x64

Optimization	Maximum Optimization (Favor Speed) (/O2)
Inline Function Expansion	Default
Enable Intrinsic Functions	No
Favor Size Or Speed	Neither
Omit Frame Pointers	
Enable Fiber-Safe Optimizations	No
Whole Program Optimization	No

Optimization
Select option for code optimization; choose Custom to use specific optimization options. (/Od, /O1, /O2)

Figure 3: QRFactor Properties → C/C++ → Optimization

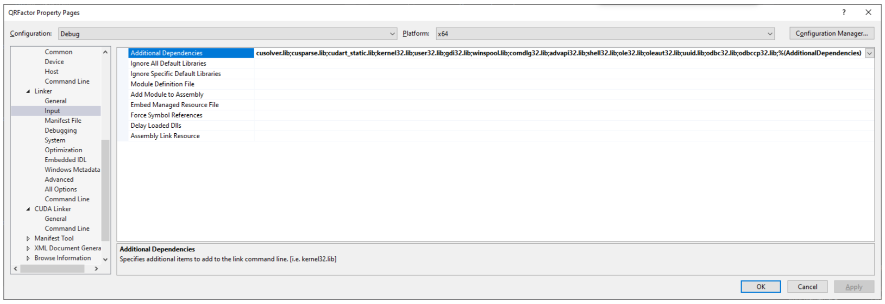


Configuration: Debug Platform: x64

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	Yes (/INCREMENTAL)
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	$(CudaToolkitLibDir);
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

Output File
The /OUT option overrides the default name and location of the program that the linker creates.

Figure 4: QRFactor Properties → Linker → General



Configuration: Debug Platform: x64

Additional Dependencies	cusolver.lib;cusparse.lib;cudart_static.lib;kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdlg32.lib;advapi32.lib;shell32.lib;ole32.lib;oleaut32.lib;uuid.lib;odbc32.lib;odbccp32.lib;%(AdditionalDependencies)
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]

Figure 5: QRFactor Properties → Linker → Input