Using the "Multi-scale cardiac simulation framework" on Windows

Documentation by Jakub Tomek, University of Oxford, June 2019

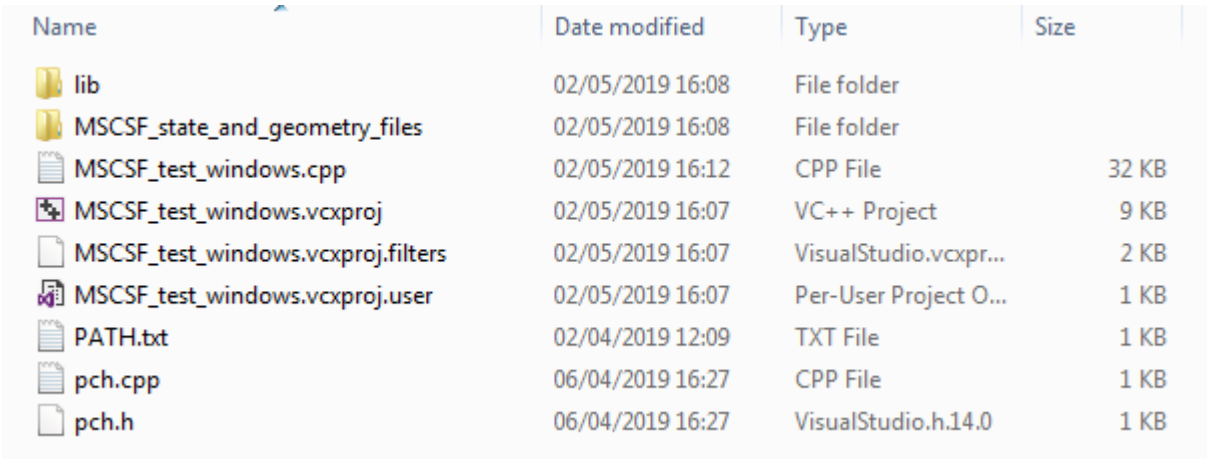
This document goes over steps necessary to run the MSCF library on Windows using Visual Studio. The steps were created based on Windows 7 and Visual Studio 2017, using `Single_cell_3D_main.cc` as the main function. These instructions apply to all code in this format (including the hAM_WL human atrial cell models code) – simply use the appropriate main file (e.g. `Single_cell_native.cc; Tissue_integrated.cc)` and follow the instructions below.

1. Create project

Create a new project in Visual Studio (VS), selecting 'Visual C++/Windows Console Application'. I call the project 'MSCSF_test_windows' for the purpose of this step-by-step. 'Location' at the bottom of the screen gives the location where the project codes are stored. Now, open this folder with the project (the folder with MSCF_test_windows'.cpp code, NOT with MSCF_test_windows'.sln) and copy there:

a) 'lib' folder from MSCSF/CODE
b) 'MSCSF_state_and_geometry_files' folder from MSCSF_distribution (the archive in which the library is distributed).
c) 'PATH.txt. from MSCSF_distribution.

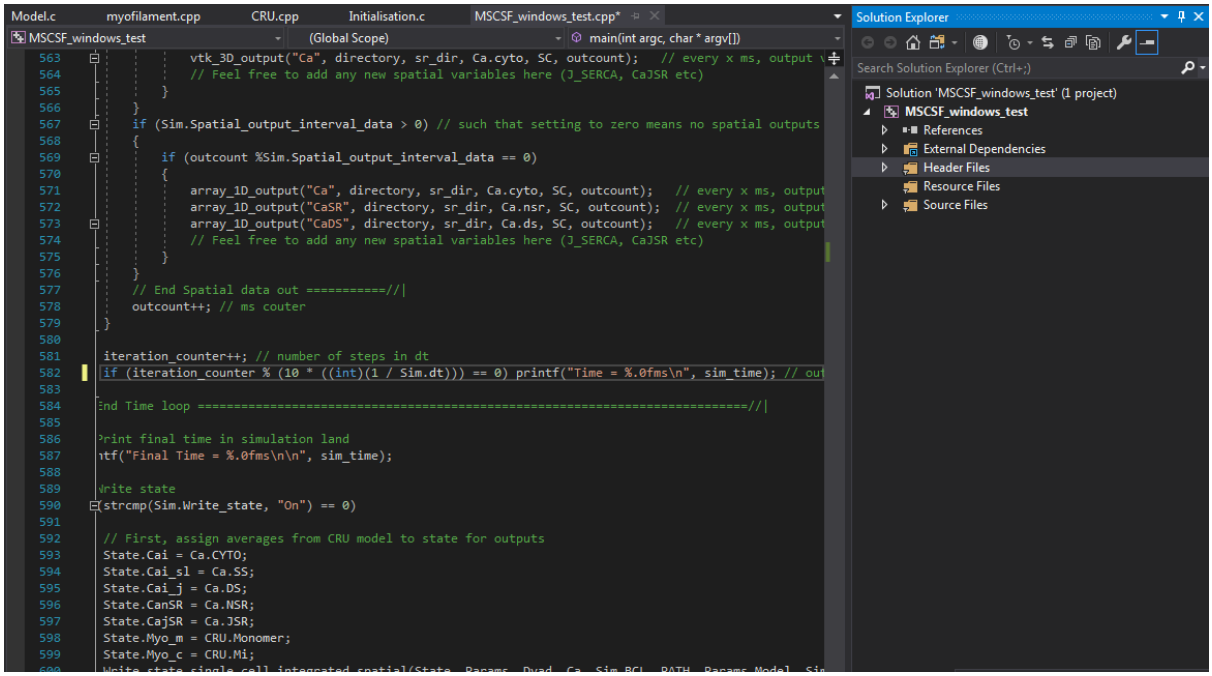
The folder should look as follows:



Name	Date modified	Type	Size
lib	02/05/2019 16:08	File folder	
MSCSF_state_and_geometry_files	02/05/2019 16:08	File folder	
MSCSF_test_windows.cpp	02/05/2019 16:12	CPP File	32 KB
MSCSF_test_windows.vcxproj	02/05/2019 16:07	VC++ Project	9 KB
MSCSF_test_windows.vcxproj.filters	02/05/2019 16:07	VisualStudio.vcxpr...	2 KB
MSCSF_test_windows.vcxproj.user	02/05/2019 16:07	Per-User Project O...	1 KB
PATH.txt	02/04/2019 12:09	TXT File	1 KB
pch.cpp	06/04/2019 16:27	CPP File	1 KB
pch.h	06/04/2019 16:27	VisualStudio.h.14.0	1 KB

2. Add source and header files to the project

Right-click 'Header Files' in the Solution Explorer (pane at the right part of VS):



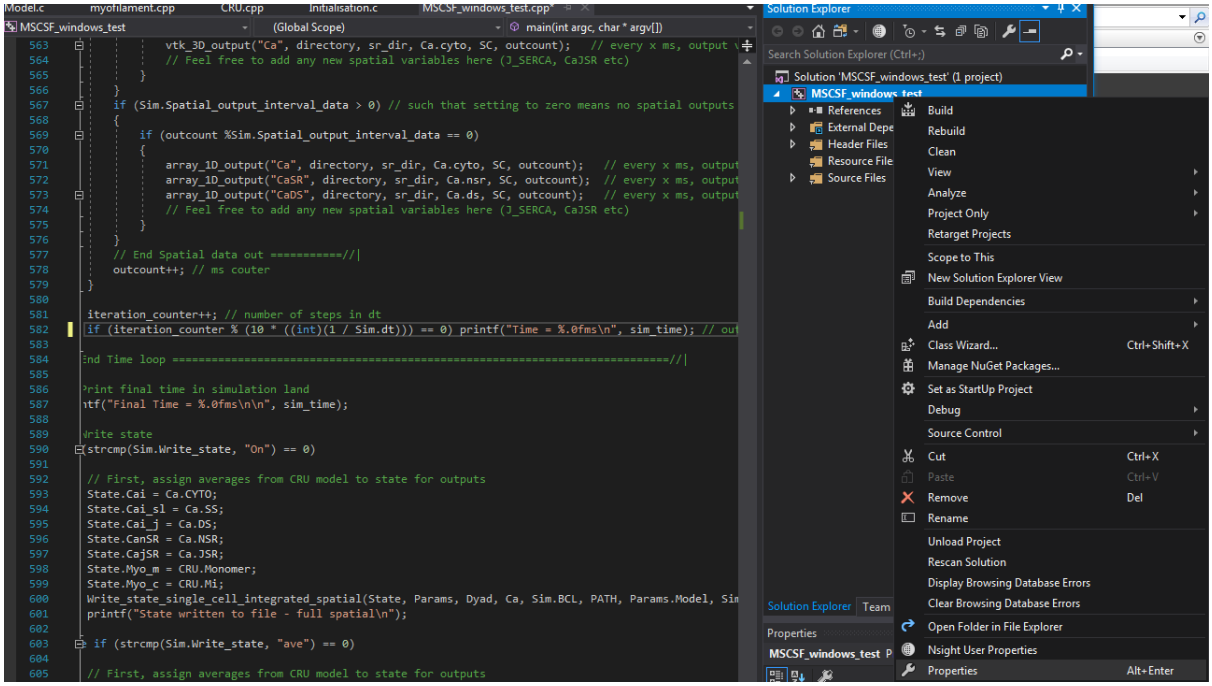
Click Add-Existing Item, and navigate to your project folder/lib, and select all files with extension .h or .hpp (if you sort the folder by Type, c/cpp and h/hpp become separated). Then right-click 'Source Files' and likewise select and add all .c or .cpp files.

3. Set up the main code

In your main code (MSCF_test_windows.cpp), replace the contents with contents of 'Single_cell_3D_main' which is in 'CODE' subfolder the original MSCSF_distribution archive

4. Set compiler to C++

Go to Project Properties (right click the project – not the Solution – in Solution explorer):



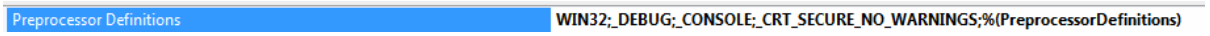
There, go into C/C++ - Advanced, and under ‘Compile As’, select ‘Compile as C++ Code (/TP)’. Without this step, there would be many compiler errors later.

5. Disable precompiled headers

In Project properties, expand C/C++, go to Precompiled headers, and select ‘Not Using Precompiled Headers’ by clicking on the contents of the ‘Precompiled Header’ line, and opening the drop-down menu at the end of the line. Then Apply and OK this change at the bottom.

6. Dealing with fopen/fclose safety

If you try to Build after the previous step, there are still many errors left; most are complaints about fopen/fclose. To deal with this, go to Project properties - C/C++ - Preprocessor, and there, in ‘Preprocessor Definitions’, add ‘`_CRT_SECURE_NO_WARNINGS;`’ after ‘_CONSOLE;’.



Apply, OK.

7. Compiler limit of if/else

NOTE: The Arguments.c file for the whole distribution has now been updated into this format (April 2020). You should not need to do anything for this step as this error should now be removed.

When trying to Build, the number of errors should be getting quite low. The first one (‘compiler limit: blocks nested too deeply’) is the trickiest – the simplest thing is to replace Arguments.c with the function provided **[here]**.

What happens is that VS compiler does not like deep nesting of if-else (256 is the limit). The default structure of argument parsing in Arguments.c does just that, unfortunately. The provided alternative file rewrites the elseifs into a sequence of non-nested ifs with a small modification to how invalid parameter names are detected (which is functionally equivalent).

8. Finding PI

Now, the top error is 'M_PI': undeclared identifier. The compiler says that it doesn't know where to find the definition of π. To remedy this, go to lib/Tissue.cpp, and after #includes, add '`# define M_PI 3.14159265358979323846`'.

9. Treating undefined variables

Unfortunately, the behavior of undefined variables is platform and compiler-dependent, which can cause various sorts of headache. Visual Studio compiler can detect some cases – such as the case of 'ihit' in 'lsoda.hpp'. This is easy to treat by explicitly zero-initializing ihit (after ihit is defined in 'lsoda' function, just include '`ihit = 0;`').

Similarly, in 'Model.c' and 'CRU.cpp', there are instances of 'potentially uninitialized local pointer variable 'I_out_individual'. To solve this, simply set the respective variables to NULL after they are defined in the functions containing the offending lines. In the current version, of the code, such a definition is, e.g., at the line 1337 of 'CRU.cpp'; you can add '`I_out_individual = NULL;`' at the line 1339.

Another change is needed in myofilament.hpp (this one is not picked up by the compiler and would cause havoc if unchanged) – uncomment the line containing '`dt_myof = 0.05;`'.

Now, the project should Build without errors.

10. Making sure that folders can be created.

The last outstanding issue is Linux-style creation of relevant folders. In the main function file ('MSCSF_windows_test.cpp'), change 'mkdir –p' to 'mkdir' (Windows can create nested folders by default). In the current version, there are four such instances on lines 132-143. Subsequently, also on these lines, change forward slashes ('/') which serve as path separators to double backslashes ('\\'). This is actually not necessary for lines after '`printf(">Creating output files...\n");`' (currently on line 147).

11. Specify PATH to data

Find the file PATH.txt in your project folder. There, simply put 'MSCSF_state_and_geometry_files' as the only content of the file.

12. Running the code

Build the project. Now, find the .exe file, which is in the Debug folder. Cunningly, this is not in Debug folder in your *project* folder, but it is in the *solution* folder, i.e., one level above the project folder

(the same folder where ‘MSCSF_windows_test.sln’ is present). Take the exe file and copy/move it to the project folder. There, it can be run from the command line similarly to the Linux version. E.g., ‘MSCSF_windows_test.exe BCL 500 Beats 2’ runs two beats of the model with default parameters at BCL 500 ms.