Table of Contents

[1. About 2](#_Toc76478385)

[2. Getting Started 3](#_Toc76478386)

[2.1 GCC/G++ 3](#_Toc76478387)

[2.2 MS Visual Studio (2019) 3](#_Toc76478388)

[3. Input Files 5](#_Toc76478389)

[3.1 Pointer File 6](#_Toc76478390)

[3.2 Simulation Files 6](#_Toc76478391)

[3.2.1 Material File 6](#_Toc76478392)

[3.2.2 Beam File 7](#_Toc76478393)

[3.2.3 Path File 7](#_Toc76478394)

[3.3 Option Files 7](#_Toc76478395)

[3.3.1 Domain File 8](#_Toc76478396)

[3.3.2 Settings File 8](#_Toc76478397)

[3.4 Utility Files 10](#_Toc76478398)

[3.4.1 Points File 10](#_Toc76478399)

[3.4.2 ParBeams File 10](#_Toc76478400)

[3.4.2 InfBeams File(s) 11](#_Toc76478401)

[4. OutputFiles 11](#_Toc76478402)

# About

This program provides a way to quickly simulate the conductive heat transfer found in welding and additive manufacturing. The outputs it can produce include temperature and both solidification gradient and velocity, G and V.

For transient thermal conduction in welding and additive manufacturing, assuming uniform and constant thermophysical properties and neglecting the effects of fluid convection and latent heat, energy conservation can be written as:

|  |
| --- |
|  |

where is density, specific heat, temperature, time, thermal conductivity, and a volumetric heat source. Assuming a semi-infinite domain, Equation 1 may be solved analytically. The solution here for temperature at time t using a volumetric Gaussian heat source is:

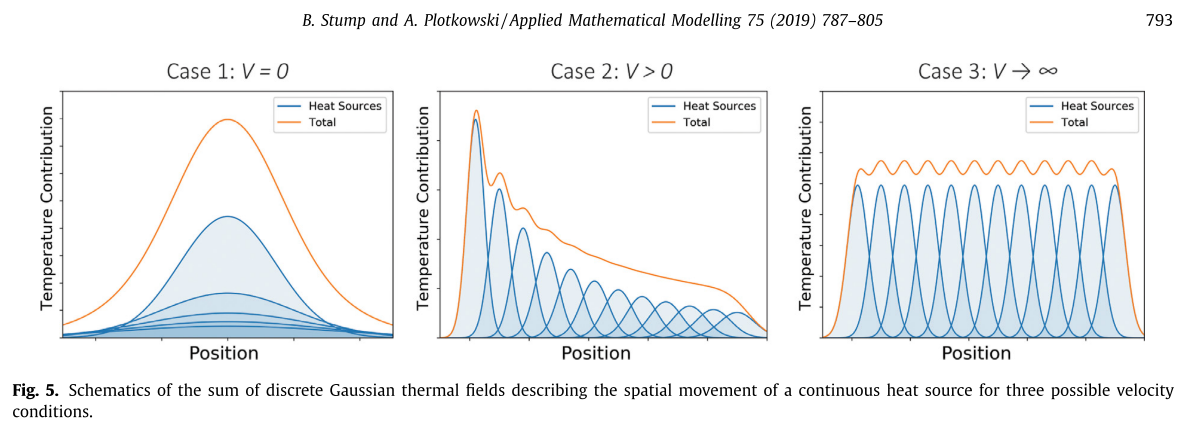
|  |
| --- |
|  |

Where is the vector and is the location of the heat source.

The numerical integration scheme used is an adaptive Gaussian quadrature method based on the following nondimensionalization:

A more detailed explanation of the mathematics can be found in the paper:

B. Stump, A. Plotkowski. "An adaptive integration scheme for heat conduction in additive manufacturing." *Applied Mathmatical Modeling*



# Getting Started

This section is meant to enable someone to both compile and run the code on various operating systems (Windows, Mac, Linux). Using GCC/G++ is more widely used than the MS Visual Studio compiler; however, the MS Visual Studio compiler is superior to the G++ compiler in terms of speed for this project (so far). All methods shown here are completely free.

## 2.1 GCC/G++

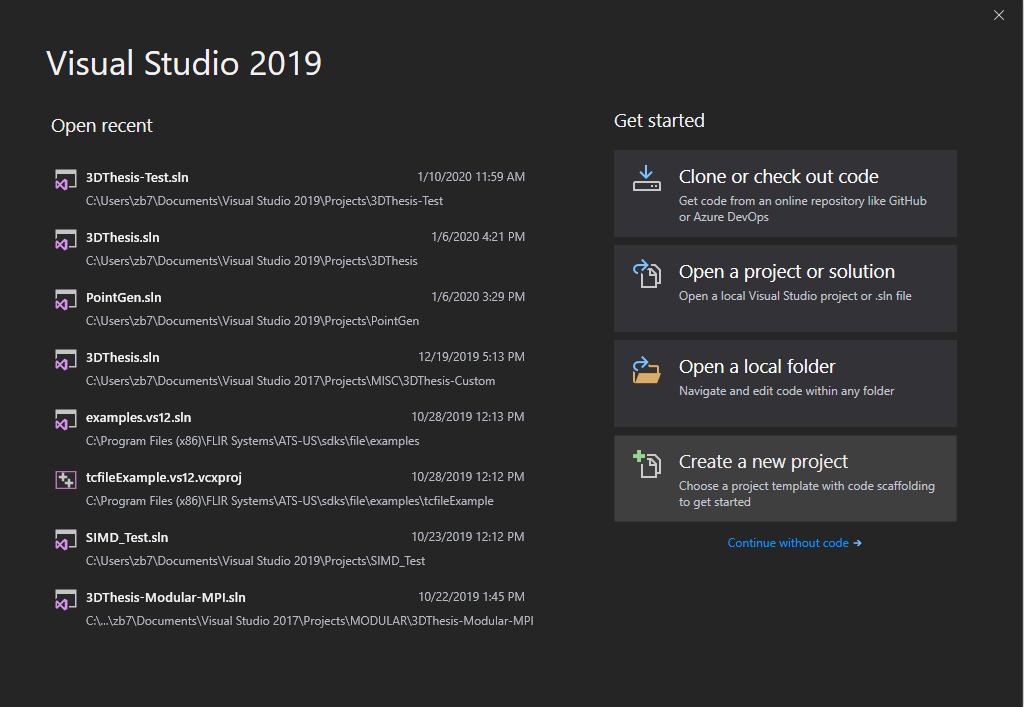
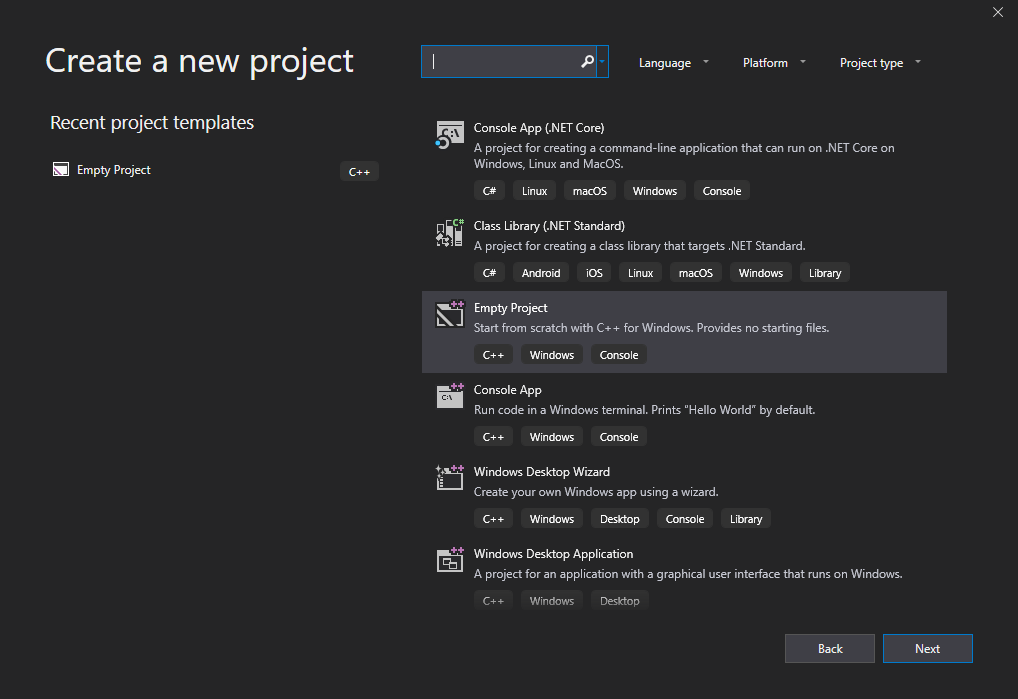
This is the easiest approach, especially for users of a Mac of Linux machine. Simply download gcc/g++ (Google search), navigate to where the makefile is located, open the terminal, command prompt, or bash shell, and type “make.” The resulting application will be in folder …\build\application. To run the example, copy the application to the folder which contains the TestInputs folder and run it. The resulting data file should show up in folder …\Data\TestSim. To view the data, the easiest way is to use ParaVIEW, covered in section 4.

The default argument is “TestInputs/ParamInput.txt” which means when running from the command line, “3DThesis.exe” is the same as “3DThesis.exe TestInputs/ParamInput.txt.” To run the second example, go to the command line and type “3DThesis.exe TestInputs2/ParamInput.txt.” The resulting sequenced data files should show up in folder …\Data\TestSim2. To view the data, the easiest way this sequenced data is to use ParaVIEW, covered in section 4.

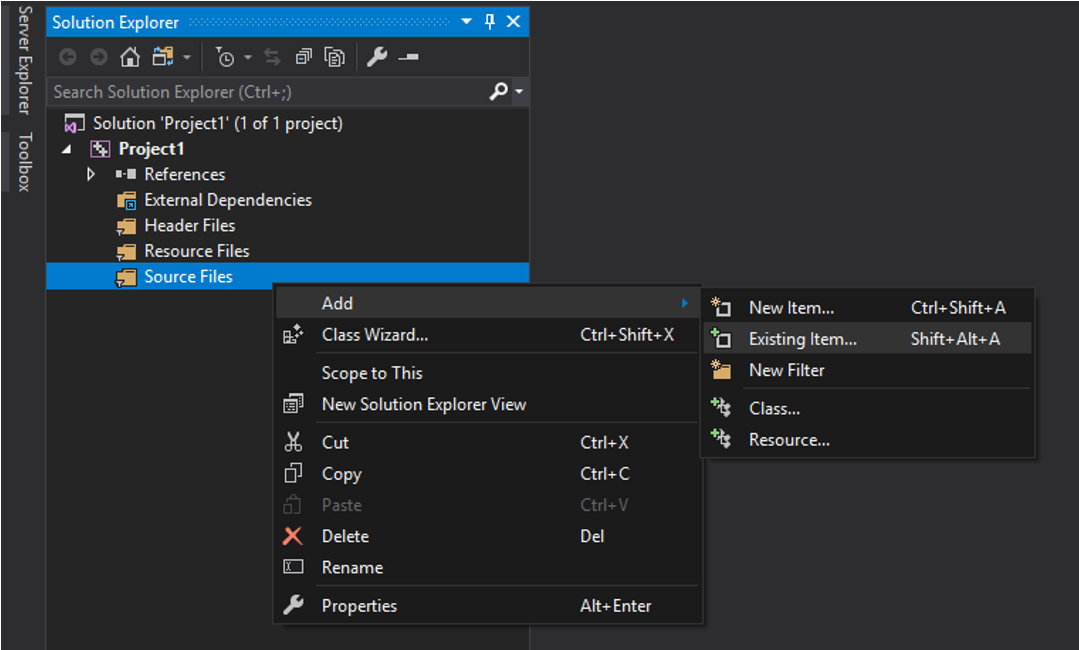
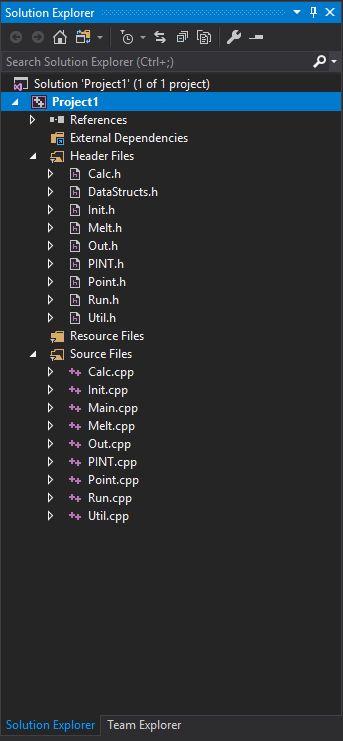
Note: G++ is unable, using the currently written code, to vectorize the temperature calculation function for points. This leads to it being a factor of 2 slower than if the code were compiled using Visual Studio and, conceivably, the intel compiler.

## 2.2 MS Visual Studio (2019)

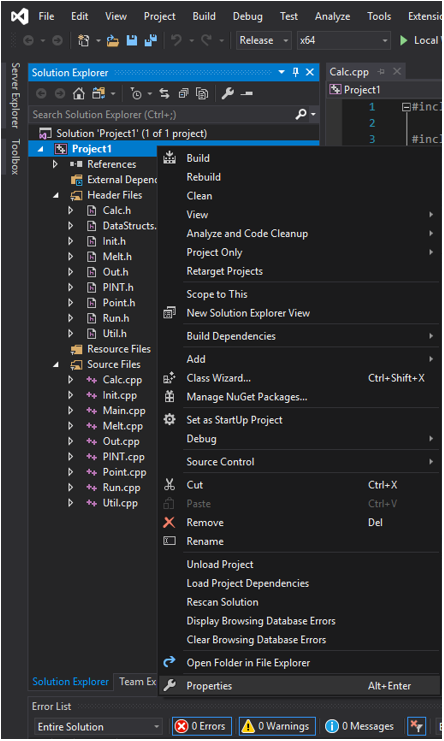
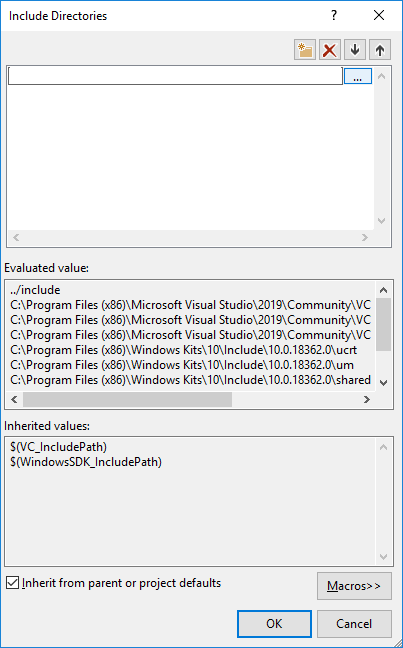
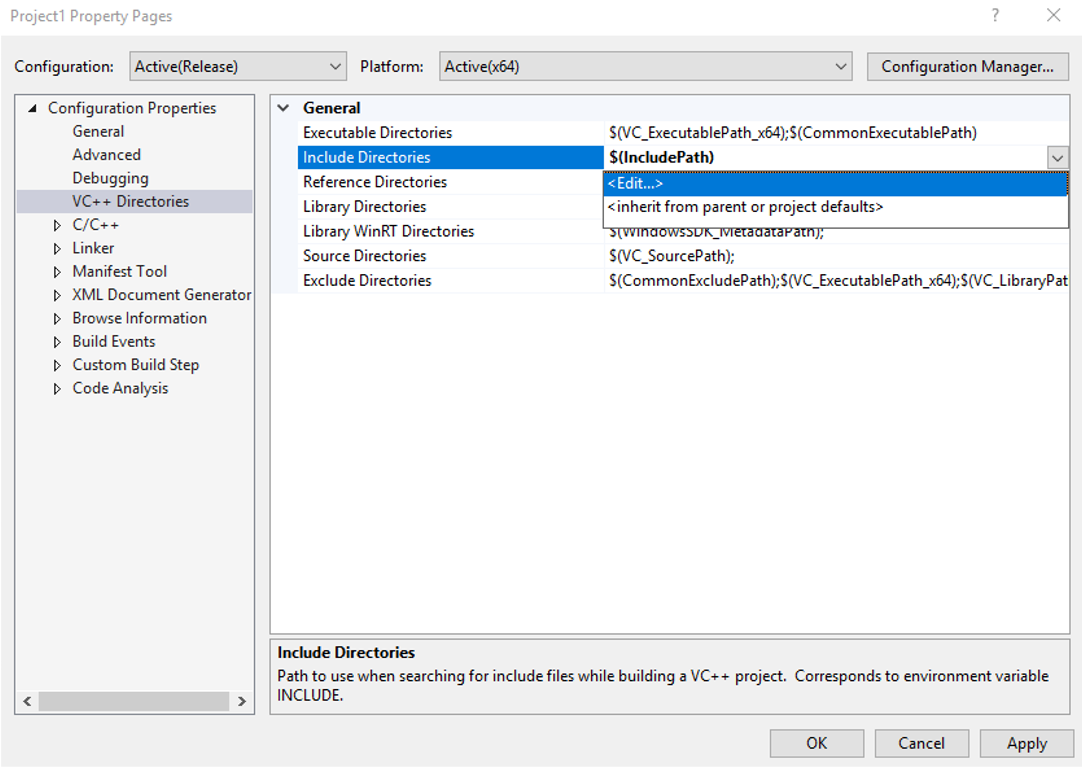
This approach is recommended for Windows users wanting to also edit or understand the code. To set this up, download the latest version of Visual Studio 2019 (Google search) and run it. Go to “Create a new project” under “Get started” and create an “Empty Project.”

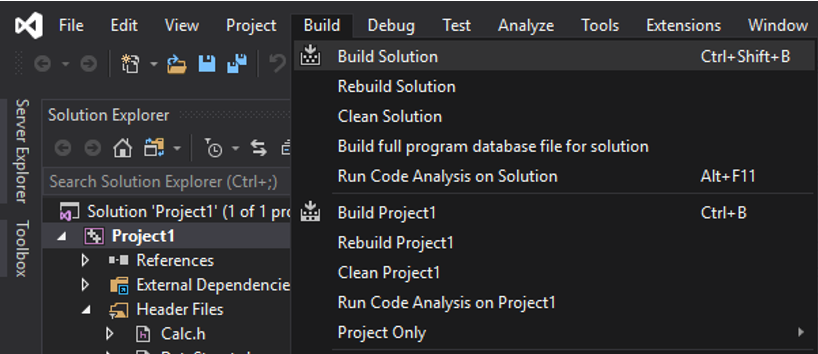
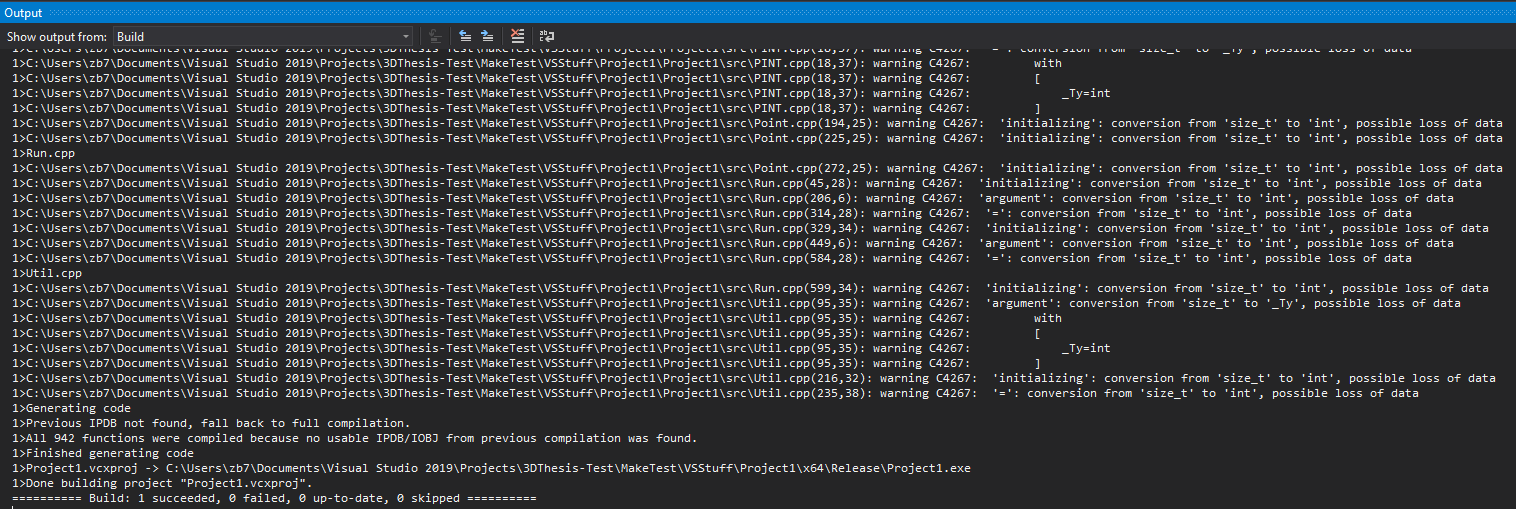
Then navigate to the project directory and double-click on the solution (.sln) file to open it up and then copy the src, include, and TestInputs folders into the folder containing the project (.vcxproj) files. Under source files, add all existing items in the src folder. For the header files, add all existing in the include folder. The solution explorer should then look like the picture on the right.

Lastly, go into to the properties of the project and add the include directory under the include directory settings.

If done correctly, you can now build the solution without any errors (ctrl-shift-B or using the menu build->build solution).

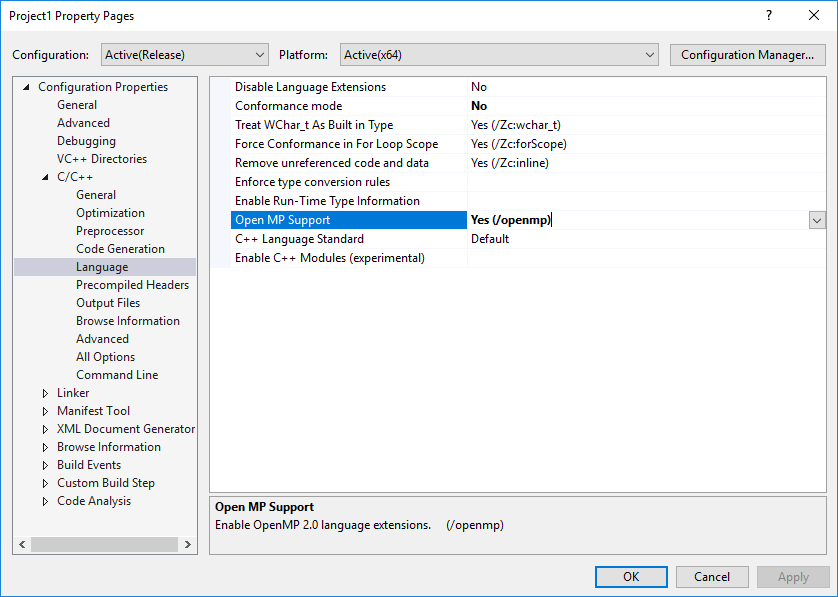
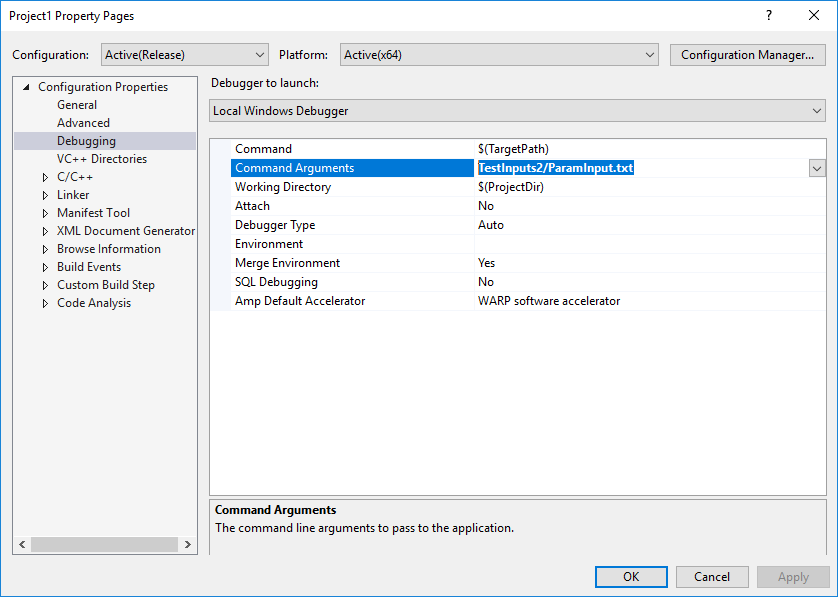
 

To allow for parallel computing and vectorize calculations, some additional steps are needed (RECOMMENDED). All recommended changes can be found in the project properties (the big menu which contained the ability to add include directories). It is also recommended that the Platform is set to x64 and Configuration is changed to Active (unless debugging).

1. Configuration Properties -> C/C++ -> Code Generation -> Floating Point Model = Fast
2. Configuration Properties -> C/C++ -> Language -> Conformance mode = No
3. Configuration Properties -> C/C++ -> Language -> Open MP Support = Yes

Note: Command line arguments can be entered in VS (figure on the right)

Note: The compiled executable can be found in the “x64” folder located in the same place as the solution (.sln) file.

# 3. Input Files

This section shows how to create a custom simulation along with all the parameters that can be tuned. The general outline for this section will be:

1. File Name
   1. Group Word
      1. Keyword

Where any necessary keyword will have be **bolded**. If a necessary keyword is not included, often the program will default to values found in the test case. Also, When editing files, it is important to begin each group with “{“ and close each group with “}” as seen in the example files.

## 3.1 Pointer File

When running a simulation from the command line, it will be the 2nd argument (ie: “*./3DThesis.exe TestInputs2/ParamInput.txt*”). The default value is “*TestInputs/ParamInput.txt.*” This is the file which tells the program where to find the other input files as well as the name of the simulation. The Data is located within the “Data/” folder found in the same folder as the pointer file. All the files under “Simulation” are necessary, the rest are optional.

NOTE:To use multiple heat sources, use the \* to denote the wildcard (ex: “Beam.\*.txt”). The program will start at “Beam.1.txt” then “Beam.2.txt” and keep going until “Beam.x.txt” does not exist. The same goes for the paths. “Path.x.txt” would be used with the beam parameters found in “Beam.x.txt”

* 1. Simulation
     1. **Name**
     2. **Mode**
     3. **Material**
     4. **Beam**
     5. **Path**
  2. Options
     1. Domain
     2. Output
     3. Settings

## 3.2 Simulation Files

This set of files dictate everything having to do with the physicality of the simulation, such as what is calculated during a simulation, what material it is run on, the heat source, and the path of the heat source. All the files in this section are necessary.

### 3.2.1 Mode File

This file contains all information about how the simulation is run. There are two types of modes: Solidification and Snapshots. Only one mode can be run at a time.

1. Solidification
2. **Tracking** (*None*, *Volume*, or *Surface*)
   * + - *None* always calculates all points.
       - *Volume* always calculates all molten points.
       - *Surface* always calculates the surface of the molten pool. This tracking is fastest if only simulation solidification conditions.
3. **Timestep**
   * + - This is interval between when the temperatures are evaluated. Setting this value too low results in unnecessary computational cost; too high and solidification will be missed.
4. OutputFrequency
   * + - How many timesteps between subsequent outputs. The final result is always output so if intermediate results are not desired, set this value to be a large number.
5. Secondary
   * + - Calculates the secondary solidification characteristics. This is experimental and should never be used but has been left in for research purposes.
6. Snapshots
7. **Times** (can either choose times OR scanFracs but not both)
   * + - Times at which to calculate the temperature snapshots
8. **ScanFracs**
   * + - Times, in terms of percentage of the length of a scan path, at which to calculate the temperature snapshots
9. **Tracking** (*None*, *Volume*, or *Surface*)
   * + - *None* always calculates all points.
       - *Volume* always calculates all molten points.
       - *Surface* always calculates the surface of the molten pool. For temperature snapshots, *Volume* and *Surface* behave identically.

### 3.2.2 Material File

This file contains all the material constants to be used by the simulation. The CET parameters are not necessary and are used to calculate the equiaxed grain fraction according to the model described in Gaumann et al., Acta Materialia, 2001 but are not necessary.

1. Constants
2. **T\_0**
   * + - Initial/background temperature ()
3. **T\_L**
   * + - Liquidus temperature ()
4. **k** 
   * + - Thermal conductivity ()
5. **c**
   * + - Specific Heat ()
6. **p**
   * + - Density ()
7. CET
8. N0
9. n
10. a

### 3.2.3 Beam File(s)

This file contains information on the energy source, which is a volumetric gaussian. It should be noted that everything input here is equivalent to . So if the standard deviation of an actual beam is 20µm and radially symmetric, both Width\_X and Width\_Y should be set to 48.9898e-6.

### 3.2.3 Path File(s)

This file dictate where and how the heat source travels. This file and all its components are necessary. An incomplete path file will cause failures. The format is different than other files and is in the form:

Mode X(mm) Y(mm) Z(mm) Pmod Vel(m/s)/Time(s)

* The Mode dictates how the heat source moves.
  + Mode = 0 is a line melt. X, Y, Z controls where the beam travels TO and the last parameter controls the constant speed of this melt.
  + Mode = 1 is a spot melt. X, Y, Z control the location of the spot melt and the last parameter controls the duration of this melt.
* Pmod is a power multiplier to the heat source.
  + This is typically just 0 or 1 to control the beam turning on and off.
  + More advanced scan strategies have variable powers. In this case, it’s best to let Pmod control the power and set the power in the beam file to 1.

NOTE: Make sure that before a line melt, the starting point is correct! A raster pattern is best represented by alternating mode 1 and 0 where mode 1 is only there to set the start point and would have a Pmod of 0 and a short value for Time(s).

## 3.3 Option Files

This set of files dictate everything having to do with the numerics of the simulation, such as the domain considered as well as various settings. These files are optional but it is highly recommended to understand how to tweak for specific uses as they have a large impact on the speed and accuracy of the simulation. Default behavior is specified.

### 3.3.1 Domain File

This file contains information about the domain over which to calculate the temperature solution. It contains ways to control the resolution and bounds of the simulation domain but highly recommended to use. Boundary conditions can be set to provide somewhat nearly insulative boundaries using the method of images (just 1 iteration). For very thins walls (<1mm), this may not be enough to provide a completely insulative effect. Keep in mind that each point is calculated independently, so increasing the resolution by a factor of 2 in each direction will slow down the simulation by a factor of 8.

* 1. X
     1. Min
        + Minimum X value
     2. Max
        + Maximum X value
     3. Res (either Res OR Num can be used)
        + Resolution in the X-direction
     4. Num (either Res OR Num can be used)
        + Number of unique X-values in the grid. If this is set to 1, only the *Max* value is used
  2. Y
     1. Min
        + Minimum Y value
     2. Max
        + Maximum Y value
     3. Res (either Res OR Num can be used)
        + Resolution in the Y-direction
     4. Num (either Res OR Num can be used)
        + Number of unique Y-values in the grid. If this is set to 1, only the Max *value* is used
  3. Z
     1. Min
        + Minimum Z value
     2. Max (typically set to be 0, or wherever the top surface is)
        + Maximum Z value
     3. Res (either Res OR Num can be used)
        + Resolution in the Z-direction
     4. Num
        + Number of unique Z-values in the grid. If this is set to 1, only the *Max* value is used
  4. BoundaryConditions (Z\_max is already a boundary)
     1. X\_min
     2. X\_max
     3. Y\_min
     4. Y\_max
     5. Z\_min
  5. Custom
     1. File
        + File which has a specific set (x,y,z) coordinates to be used for the domain.
        + Note: Using a point file negates any tracking modes (*Volume* and *Surface*), thus all points will always calculated.

Note: Not specifying *Min* or *Max* value results in the unspecified value being calculed via the path file(s) with a domain file 500µm buffer on each side and a 250µm simulated depth. Not specifying a *Res* or *Num* results in a default resolution of 50 µm.

### 3.3.2 Output File

This file contains all variables which can be output. A value of 0 indicated to not output the variable whereas a value of 1 indicates that variable should be output. Most variables default to a value of 0. The memory required to run a simulation increases when more outputs are selected; therefore, it is good practice to output only the necessary or desired information.

1. Grid
2. x
   * + - x-coordinate
3. y
   * + - y-coordinate
4. z
   * + - z-coordinate
5. Temperature
6. T
   * + - Temperature
7. T\_hist
   * + - Temperature history
       - Note: This will create a separate file for each points. DO NOT USE if many points are in the domain
8. Solidification
9. tSol
   * + - Solidification time
10. G
    * + - Magnitude of the thermal gradient at solidification
11. Gx
    * + - x-component of normalized gradient
12. Gy
    * + - y-component of normalized gradient
13. Gz
    * + - z-component of normalized gradient
14. V
    * + - Velocity of solidification front
15. dTdt
    * + - Cooling rate
16. eqFrac
    * + - Equiaxed fraction
17. RDF
    * + - Export results in “*Reduced Data Format”* compatible with ExaCA
18. numMelt
    * + - Number of times a point melted and solidified
19. Solidification+
20. H
    * + - Magnitude of the orthogonal differential change in the solidification gradient in the direction of the solidification gradient
21. Hx
    * + - x-component of normalized H
22. Hy
    * + - y-component of normalized H
23. Hz
    * + - z-component of normalized H

### 3.3.3 Settings File

This file contains all the tunable parameters affecting how the simulation is running. For the most part, the only parameter which should be changed is *MaxThreads* which speeds up the simulation (if the computer running it has at least that many available threads).

1. Temperature
2. Sol\_Tol
   * + - Controls how close the temperature must be to the liquidus temperature (as a fraction) before the search algorithm stops. As an example, if T\_L = 10K, timestep=0.1s, T(t=1.7s)=12K, T(t=1.8s)=9K, and Sol\_Tol=1e-2 THEN the search algorithm won’t stop trying to find the EXACT time the point solidified until it finds a time where 9.9K<T<10.1K. Defaults to 1e-3
3. Sol\_Iter
   * + - Controls the maximum number of iterations for the algorithm used above. Defaults to 10.
4. Cutoff\_Peak
   * + - Helps control how far back in time the integration is done. Defaults to 1e-9. For example, if a simulation runs for 30s, maybe the first 5s have so little temperature contribution that they aren’t worth integrating. This setting rarely gets used since the introduction of the path compression algorithm.
5. Cutoff\_T0TL
   * + - Another setting to help control how far back in time the integration is done. Defaults to 1e-9.
6. Path
7. Buffer
   * + - Control how far outside the domain the scan path is considered (in meters). For example, if a scan file contains information for 6 cubes, but you only want to analyze one of them. Set the domain to include the 1 cube and set the buffer to 0 or 0.001 (1mm) or something small. This will result in a large speedup.
8. Compression
   * + - Binary toggle (0-off, 1-on) for using a path compression algorithm. This is primary useful for LARGE scan paths (1 Mb or larger) with smooth-ish movement of the heat source (ex: raster or point raster) rather than discontinuous movement (ex: random point fill). It does so by compressing multiple nearby diffuse heat sources into a single source. There is a small loss in accuracy but a large speedup for certain simulations.
9. Compute
10. MaxThreads
    * + - Number of threads to use for the simulation.
11. PINT
    * + - Allow for Parallel-IN-Time calculations
        - Note: This has been temporarily removed during the code overhaul

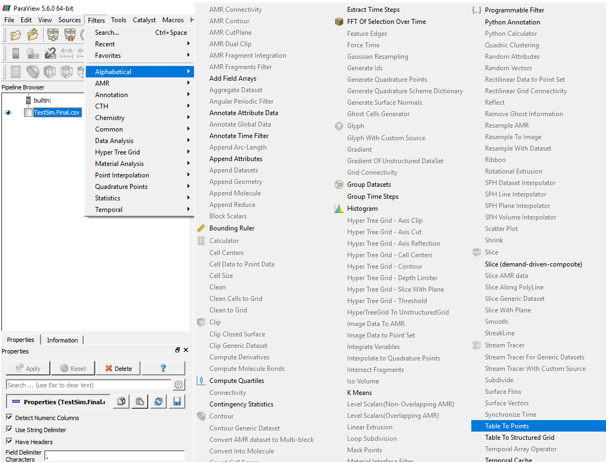
# 4. OutputFiles

Data analysis can be done using a variety of methods (Python’s Pandas module comes to find) but for data visualization, ParaVIEW is a nice, free tool to use. This section is meant to help someone get a head start visualizing the data in ParaVIEW (Google search).

**ParaVIEW**

To get a head start visualizing data in ParaVIEW, first make sure that the code ran the case found in *“…/TestInputs/ParamInput.txt”* and the data can be found in the folder *“…/TestInputs/Data/”*. It should be titles *“TestSim.Solidification.Final.csv”.*

To view this file in ParaVIEW, open it through the folder icon in the top left and then click the apply button. If this is done correctly, a table of data should appear on the right side of the screen. To convert this data to something visual, click on the file on the right side of the screen to highlight it, then apply the tables to points filter. It can be found in Filters->Alphabetical.



On the left side, three dropdown menus should appear titled “X Column”, “Y Column”, and “Z Column”. Simply change these to be x, y, and z, click apply, and exit out of the tabular view of the data. Now at the top of the screen, locate the dropdown menu that says, “Solid Color.” Change this to G (the thermal solidification gradient). The point size and color scale can be changed using other options in the left menu (under the “Coloring” and “Styling” groups respectively). In the end, the result should look something like this (Note: The coloring is done on a log scale here):

