Table of Contents

[1. About 2](#_Toc76472495)

[2. Getting Started 3](#_Toc76472496)

[2.1 GCC/G++ 3](#_Toc76472497)

[2.2 MS Visual Studio (2019) 3](#_Toc76472498)

[3. Input Files 5](#_Toc76472499)

[3.1 Pointer File 5](#_Toc76472500)

[3.2 Simulation Files 6](#_Toc76472501)

[3.2.1 Material File 6](#_Toc76472502)

[3.2.4 Beam File 6](#_Toc76472503)

[3.3 Option Files 6](#_Toc76472504)

[3.4 Utility Files 6](#_Toc76472505)

# 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:

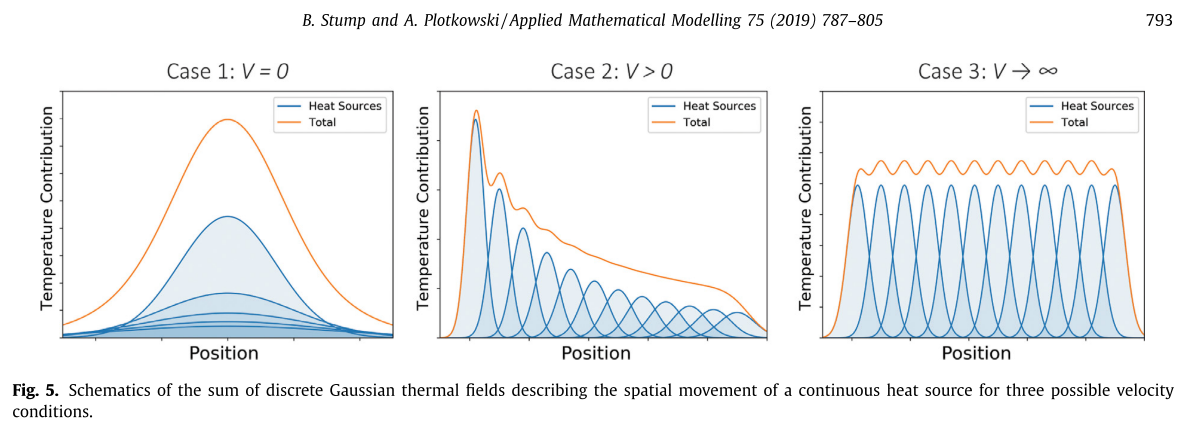
|  |
| --- |
|  |

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++

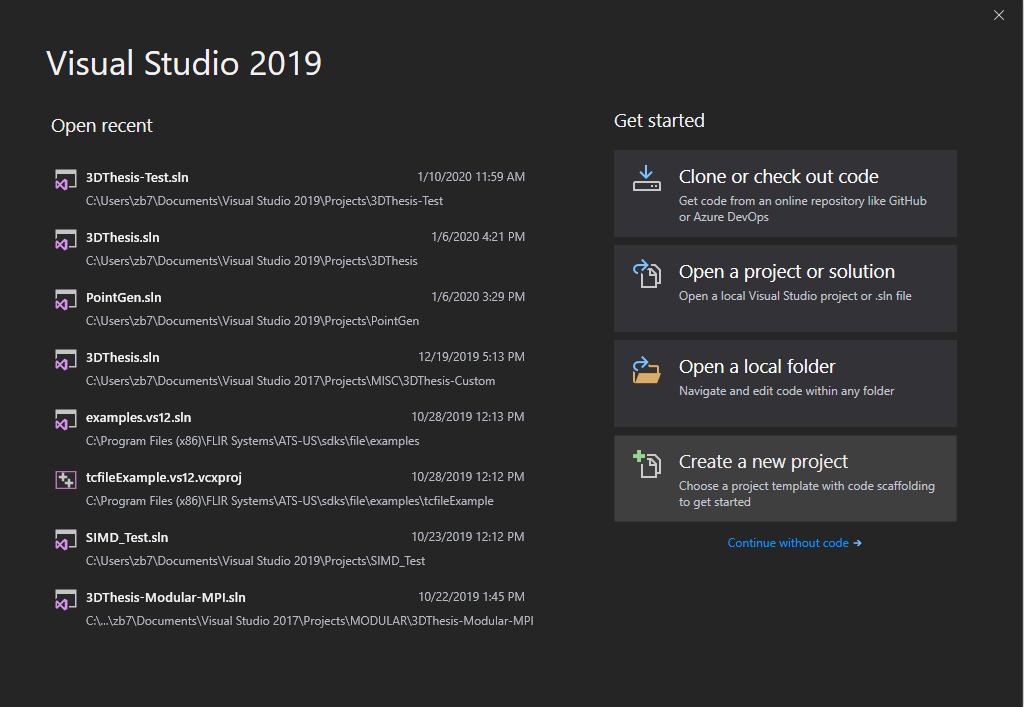
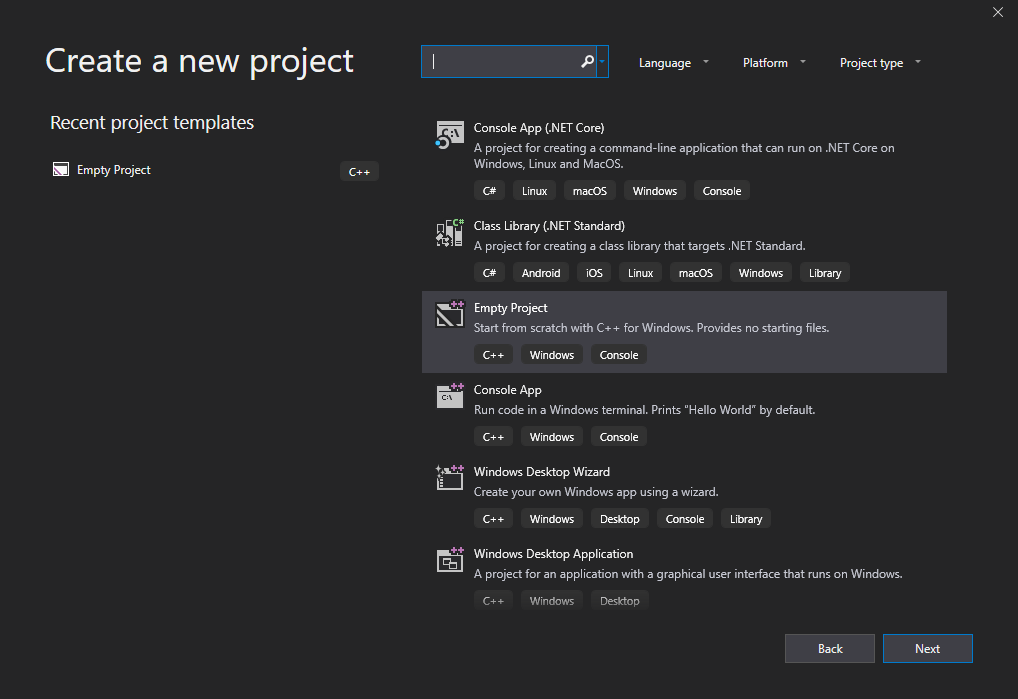
This is the easiest approach, especially for users of a Mac of Linux machine. Simply download gcc/g++ (Google search), navigate to where the makefile is located, open the terminal, command prompt, or bash shell, and type “make.” The resulting application will be in folder …\build\application. To run 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.

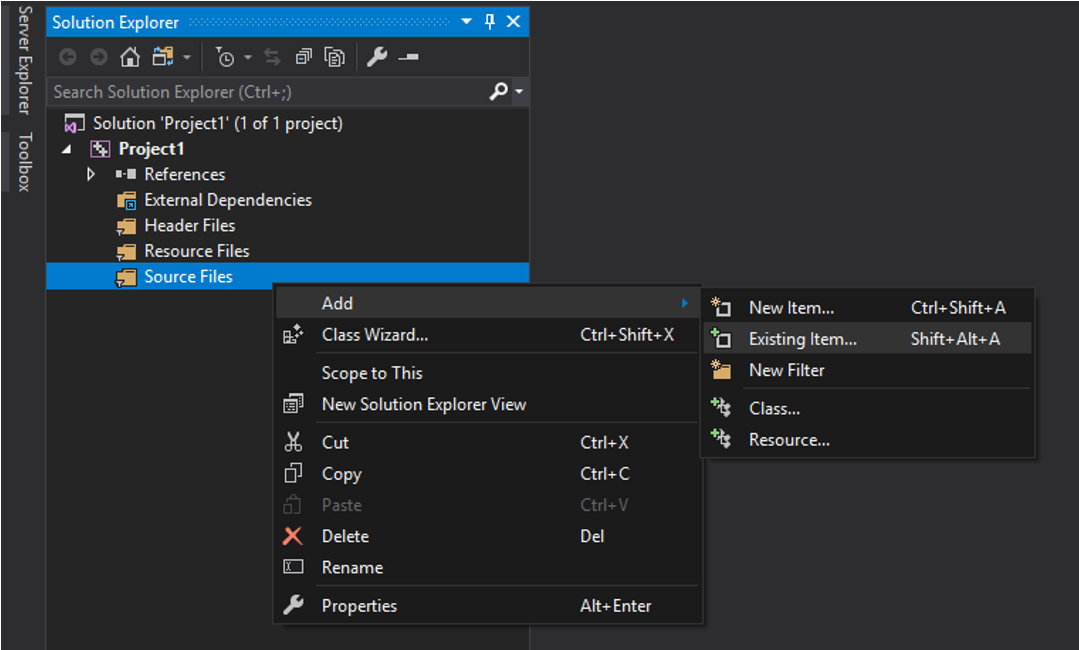
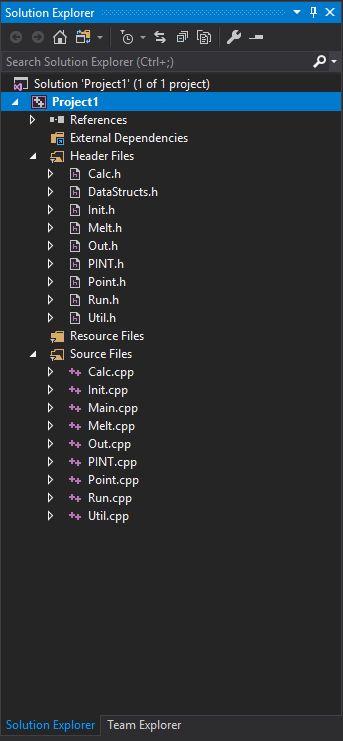
Note: G++ is unable, using the currently written code, to vectorize the temperature calculation function for points. This leads to it being a factor of 2 slower than if the code were compiled using Visual Studio and, conceivably, the intel compiler.

## 2.2 MS Visual Studio (2019)

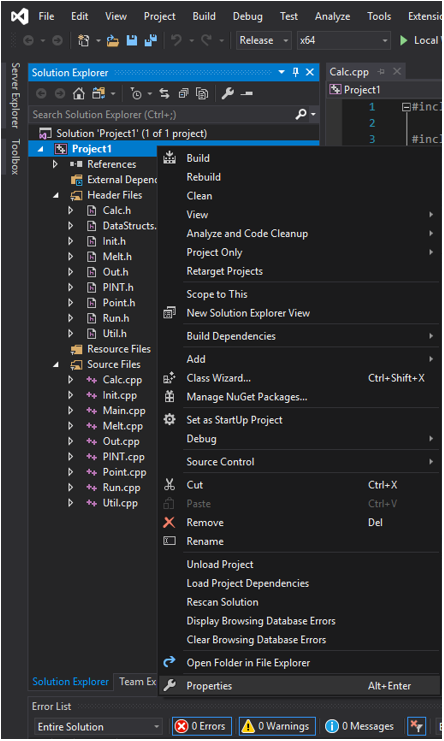
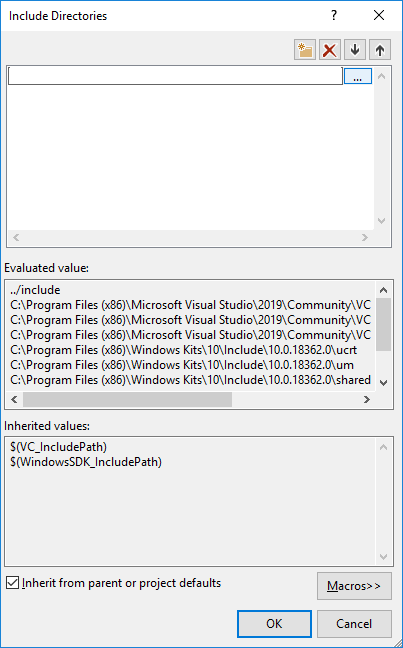
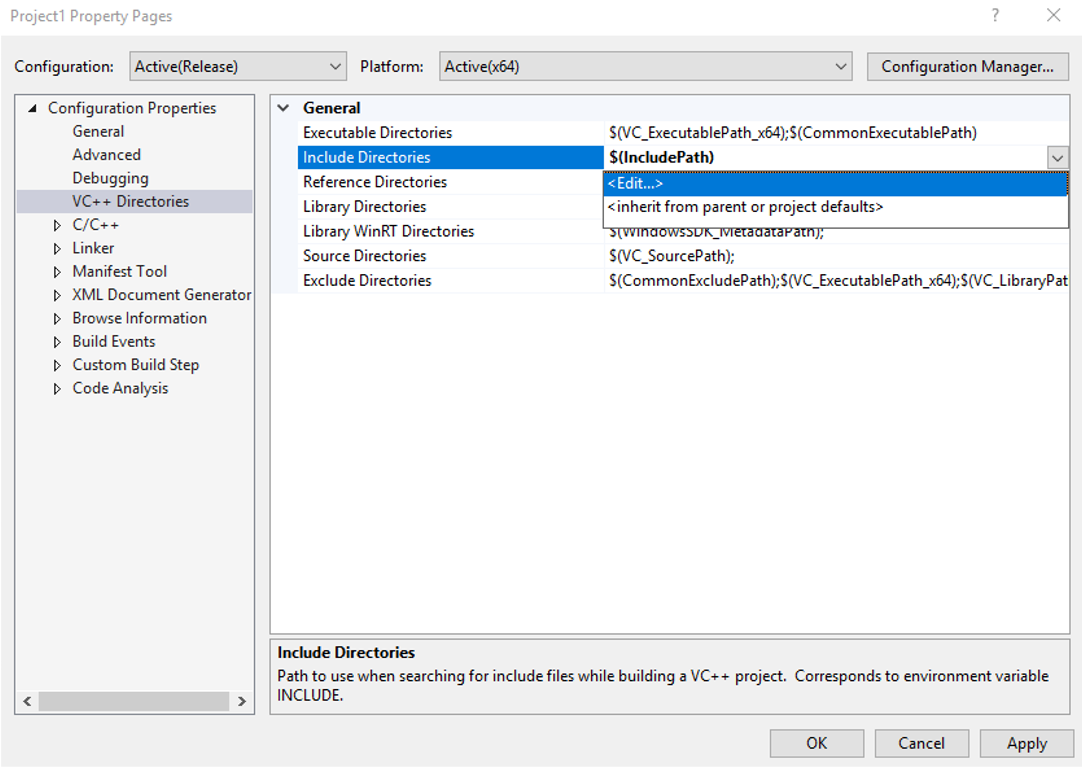
This approach is recommended for Windows users wanting to also edit or understand the code. To set this up, download the latest version of Visual Studio 2019 (Google search) and run it. Go to “Create a new project” under “Get started” and create an “Empty Project.”

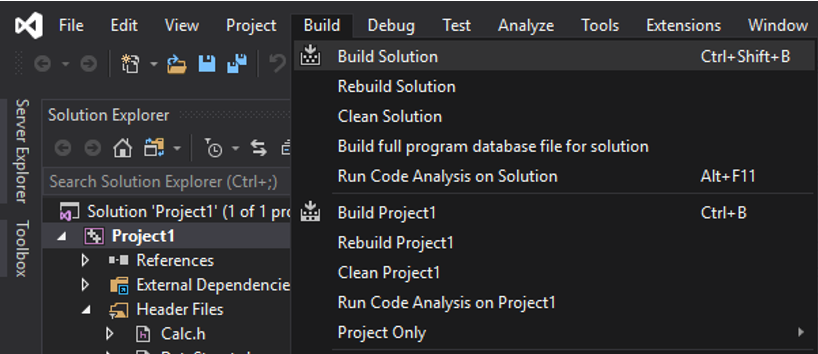
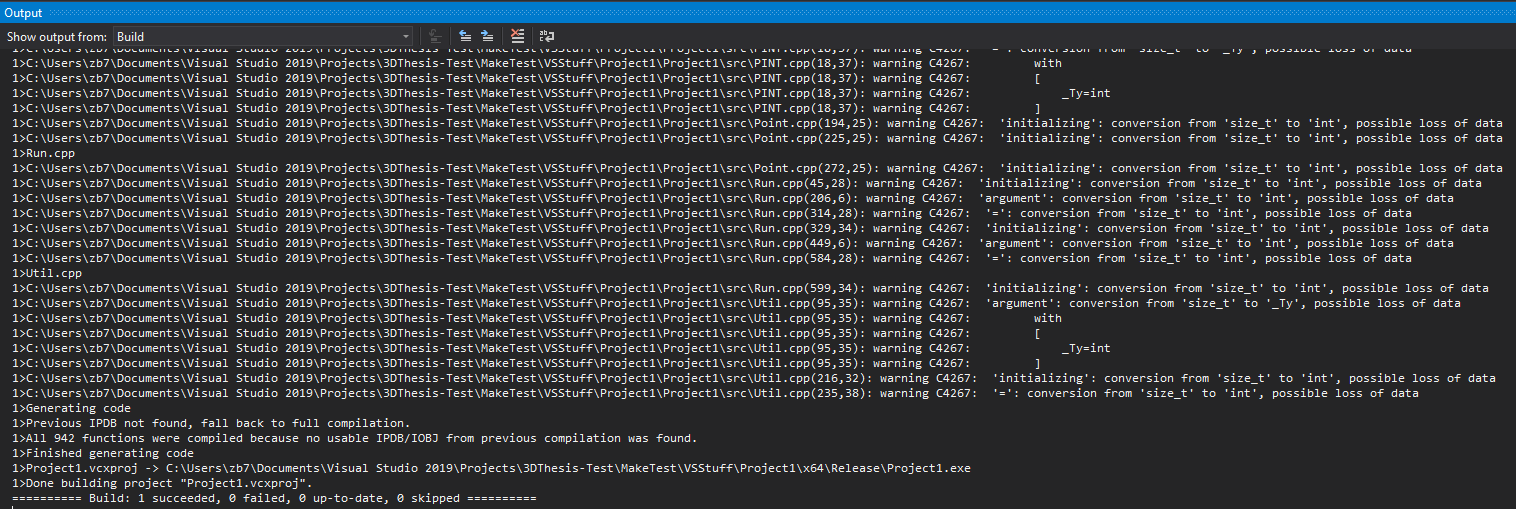
Then navigate to the project directory and double-click on the solution (.sln) file to open it up and then copy the src, include, and TestInputs folders into the folder containing the project (.vcxproj) files. Under source files, add all existing items in the src folder. For the header files, add all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

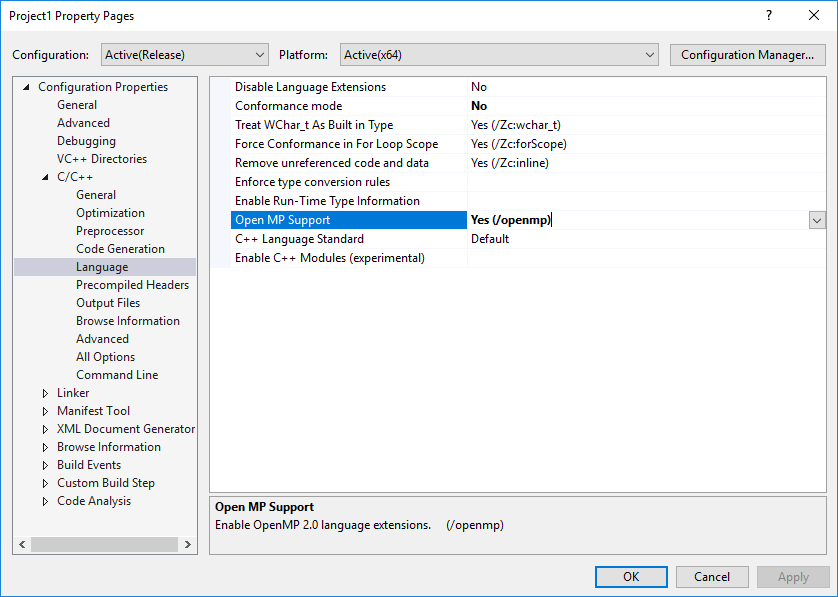
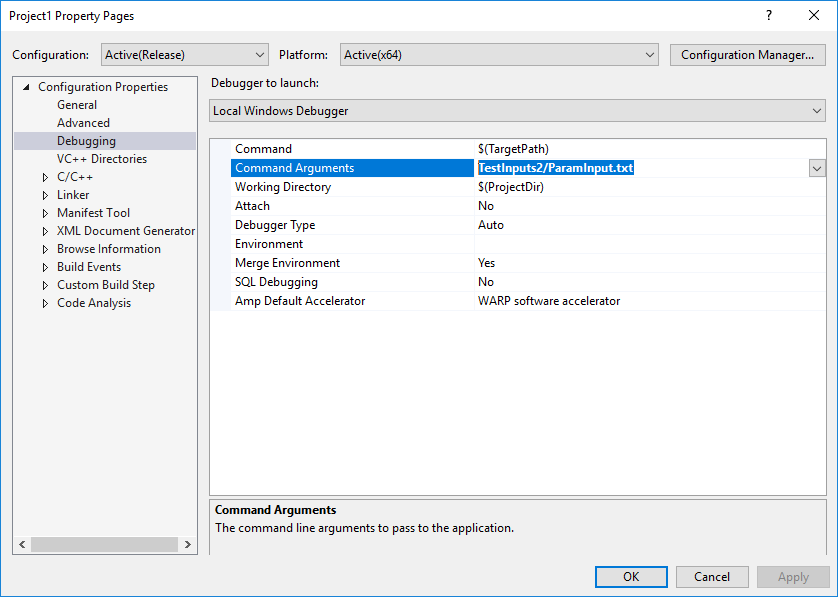
 

To allow for parallel computing and vectorize calculations, some additional steps are needed (RECOMMENDED). All recommended changes can be found in the project properties (the big menu which contained the ability to add include directories). It is also recommended that the Platform is set to x64 and 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. If Name is set to “TestSim”, then the data for the simulation can be found at “Data/TestSim”. All the files under “Simulation” are necessary, the rest are optional.

* 1. Simulation
     1. **Name**
     2. **Material**
     3. **Beam**
     4. **Path**
  2. Options
     1. Domain
     2. Settings
  3. Utility
     1. Points
     2. ParBeams
     3. InfBeams

## 3.2 Simulation Files

This set of files dictate everything having to do with the physicality of the simulation, such as 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 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 by 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

### 3.2.2 Beam File

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.

1. Shape
2. **Width\_X**
   * + - Width of the beam in the X direction ()
3. **Width\_Y**
   * + - Width of the beam in the Y direction ()
4. **Depth\_Z**
   * + - Penetration depth of the beam ()
5. Intensity
6. **Power**
   * + - Power of energy source ()
7. **Efficiency**
   * + - Absorption efficiency of beam…refer to literature for accurate values. Typically it is 0.35 for L-PBF and 0.85 for E-PBF.

### 3.2.3 Path File

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
     2. Max
     3. Res
        + Resolution in the X-direction of the domain
     4. Num
        + Number of unique X-values in the grid. If this is set to 1, only the Max value is used
  2. Y
     1. Min
     2. Max
     3. Res
     4. Num
  3. Z
     1. Min
     2. Max
        + Typically set to 0 (where the top surface is)
     3. Res
     4. Num
  4. BoundaryConditions
     1. X\_min
     2. X\_max
     3. Y\_min
     4. Y\_max

Note: Not specifying a domain file will result in a preset resolution (50 µm) along with the domain being calculated via the path file with a 1mm buffer on each side and a 1mm simulated depth.

### 3.3.2 Settings File

This file contains all the tunable parameters affecting how the simulation is running. The “*Simulation”* and “*Output”* groups are commonly used; however, the “*Path*,” “*Temperature,”* and “*Neighbors”* groups should remain untouched unless you really understand what you are doing.

1. Simulation
2. 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.
3. Mode
   * + - Mode 0 is a temperature snapshot right after the path file finishes. Mode 1 tracks all points. Mode 2 tracks just the meltpool. Mode 3 just tracks the meltpool perimeter. If only solidification matters and speed is important, then Mode 1 < Mode 2 < Mode 3
4. MaxThreads
   * + - Maximum number of threads the program will use. If it’s not set, it defaults to half the computational threads the computer has.
5. PINT
   * + - Binary toggle (0-off, 1-on) for Parallel IN Time computations. Does not have much benefit unless computing is being done on a cluster. Only Mode 2 and Mode 3 can be used if PINT is enabled.
6. Output
7. Mode
   * + - Can be 0, 1, 2, 3, 4 and control what is in the output file
       - x,y,z,T<0>G,V<1>dTdt,eq\_frac<2>Gx,Gy,Gz<3>H,Hx,Hy,Hz<4>
       - Mode 1 = Mode 0 + G, V
       - Mode 2 = Mode 1 + dTdt, eq\_frac
8. Interval
   * + - How many timesteps between subsequent output files. Defaults to a very large number (only have the final output).
9. Path
10. 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.
11. 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.
12. Temperature
13. 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
14. Sol\_Iter
    * + - Controls the maximum number of iterations for the algorithm used above. Defaults to 10.
15. 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.
16. Cutoff\_T0TL
    * + - Another setting to help control how far back in time the integration is done. Defaults to 1e-9.
17. Neighbors
18. Neighborhood
    * + - For the melt-pool tracking algorithms (mode 1 and 2), defines how large the neighborhood is for every point when seeing if “neighbors” are also liquid. Defaults to 1.

## 3.4 Utility Files

This set of files are for specific use cases or research interests. For most cases, these will not be used. Additionally, some of these files do interfere with each other (ex: ParBeams and InfBeams) so it is still good to understand what they do.

### 3.4.1 Points File

This file defines a specific set of points to do temperature calculations on. It will replace the calculation points constructed by the “Domain File.” Since this set of points is not structured (unlike the rectangular, structured grid created by the “Domain File”), meltpool tracking will no longer work.

x(mm) y(mm) z(mm)

* x(mm), y(mm), and z(mm) are the coordinates of the points

### 3.4.2 ParBeams File

This file allows for a set of beams moving parallel to the path. It will overwrite the original beam so if there are 2 beams specified in the ParBeams file, only 2 beams will be simulated (instead of 3). They all share the shape and intensity parameters of the original beam (as defined in the “Beam File”) but an additional Pmod parameter (which acts as a power multiplier) allows for an uneven distribution of energy. If the total energy input is meant to be the same as in the “Beam File,” then the sum of all Pmods should be 1. The format is different than other files and is in the form:

Xr(m) Yr(m) Pmod

* Xr and Yr dictate the position of the heat source relative to the path file.
* Pmod is a power multiplier to the heat source.

### 3.4.2 InfBeams File(s)

This group of files allows for an arbitrary number of heat sources (potentially with different shapes) to move independently. When specifying the InfBeam Files in the “Pointer File,” it should point to the last file. Additionally, the files should be numbered sequentially. For example, if it is desired to have 5 independent beams, the argument for “InfBeams” in “Pointer File” should be “Directory/FileName.5.txt.” The files should then be named “FileName.1.txt, FileName.2.txt …. FileName.5.txt” and all located in “Directory.”

The files themselves have 2 different headers:

Power Efficiency Shape

* Power and Efficiency have the same meaning as in the “Beam File”
* Shape dictates whether or not to use the contant beam shape from the “Beam File” or have it become variable during the path as specified in the second header (0-BeamFile, 1-ThisFile). to use

Mode X(mm) Y(mm) Z(mm) Pmod Time(s) sX(m) sY(m) sZ(m)

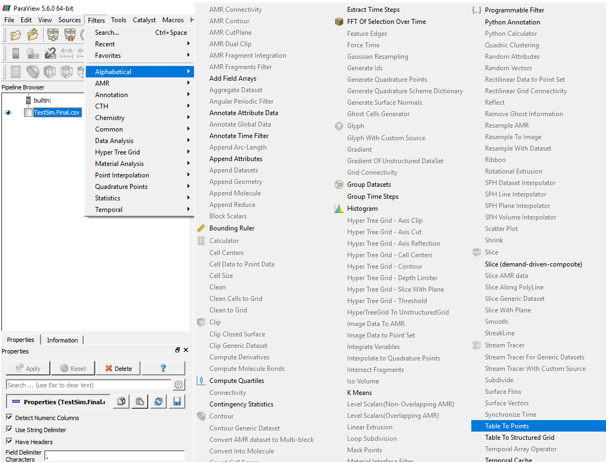
* Mode, X(mm), Y(mm), Z(mm), Pmod, Time(s) have the same meaning as in the “Path File”
* sX(m), sY(m), sZ(m) have the same as the Width\_X, Width\_Y, Depth\_Z variables in “Beam File” respectively; however, the will vary over the course of a segment like X(mm), Y(mm), and Z(mm) (ie: linearly for mode 1 but instant for mode 0)

# 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 …\Data\TestSim. It should be titles “TestSim.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):

