Albany TDM Project
Jacob S. Pessin
Thursday December 10, 2020

Grid Conversion with Consolidation from FEM Unstructured Grid to Regular Structured Grid.

1 Purpose

The objective of this code is to be able to alter the output of the FEM simulation to be used in the phase field simulation for micro structure prediction. This is done by interpolating the temperature field data at each time step of the simulation onto a regular grid of points which encompasses the desired location of the part. This included first consolidating the melted powder of the part, and then doing an initial interpolation onto a coarse regular grid from the FEM grid. A second interpolation is then done onto a grid of the desired grid spacing and is output to a csv file.

2 Consolidation

The consolidation that is performed for this grid conversion is done for a single powder layer and is compatible with any of the single powder layer simulation experiments. It is the same process as the MATLAB consolidation code, and performs a consolidation on the unstructured mesh once, and uses the same consolidated geometry for each of the time steps. This consolidation works by first importing the node locations, element associations, and Ψ_1 data from the last time step of the simulation. The overlapping nodes are first deleted in order to better unify each part of the mesh. The code then gets the boundary nodes in a list from the nodes near the powder substrate interface. These nodes are then used to assemble the ID array. The LM array can then be constructed using the output from the ID array. The sparcity can then be computed using the LM array. Then the stiffness matrix can be calculated using equations for the weakform which gives the vectors K and F. The sparse matrix M is then generated using the vector K and the system of equations can be solved for a_{bar}, by solving $M * a_{bar} = F$. It is then simple to use to a_{bar} to generate the array for the displacements d. The coordinates for the node points can then be consolidated and shifted using the array d.

3 Element Locations

This section of the grid conversion code primarily locates the the element that each grid point resides in.

3.1 Coarse Grid

Here the code generates the x, y, and z locations for the coarse grid that is generated based off of the user input data. The user inputs parameters for the origin of the gird, the number of grid points in each direction, and the spacing of the grid in each direction. A mesh grid is then generated which is used in later interpolation steps.

3.2 Reduced Search List

The code then reduced the size of the element search list by only searching through the elements that are within the range of the regular grid points for the x, y, and z directions.

3.3 Element Search

This section of the code loops through every grid point. For each point it tech checks if the given grid point is within an element from the reduced element search list. The code uses an `inhull` function that checks if a set of given points are within a polygon defined by at least three points. An element association list is then produced as an array with its number of rows equal to the number of grid points and number of columns equivalent to four (the number of nodes that define an element).

4 First Temperature Interpolation

In this part of the code the initial interpolation is done on the coarse grid in order to decrease computation time. The interpolation is relatively simple and take the x, y, and z values for each grid point and uses the `griddata` function to interpolate the temperature values from the nodes of the element onto the given grid point. the temperate values are imported for each time step of the simulation and an output array with the number of rows equal to the number of time steps and the number of columns equal to the number of grid points is produced.

5 Second Temperature Interpolation

This is the last section of the code and performs a linear interpolation from the regular coarse grid onto a much more fine grid using the `interp3` or `interp2` function. Which function that is used depends on if the code is generating a 2D or a 3D grid of regular points. Previous to this the code generates a time array by importing the data for when each time step occurs. A CSV file is created and the time step information as well as the interpolated temperature values are exported to this file for every time step.