FEM Solution Manipulation

Introduction

The FEM Solution Manipulation code is designed to be a multi-purpose program for various tasks that require reading in FE results data, performing computations and filtering, and writing the output in usable formats. Currently, the main directory for this code is `/lore/dolanj/3D_Manufacturing/FEM_results_manipulation`. The `build_with_base.sh` script is a template, where the user can go in an add their top-level C++ file into the build with the base code. The top level C++ file should contain a main function that will perform the desired task. More information on the available functions and associated input values are discussed in the following sections.

Note: When importing data using these functions, if there are missing timesteps, they will automatically be skipped. See function terminal output for details.

The basic steps required to use this code:

1. Create a new C++ file for your specific use case and include the main `FEM_solution_manipulation.hpp` header file
2. Write a main function that defines the relevant input arguments and calls the desired function
3. Modify the `build_with_base.sh` script to compile with the new C++ file and create an executable
4. Run the executable

Function: create_grid_csv_temp_hist_file

The `create_grid_csv_temp_hist_file` function imports solutions from the Albany FEM code, interpolates results from the unstructured FE grid onto a user-specified regular grid, and writes the regular grid results to a proprietary csv file for use as an input to the RPI phase field model. An example of a top level C++ file is shown below. In order to link and compile with the code base, the user must have `#include "FEM_solution_manipulation.hpp"` in their cpp file. The input arguments for the `create_grid_csv_temp_hist_file` function are discussed below.

Usage Example

```cpp
#include "FEM_solution_manipulation.hpp"

int main()
{
    std::string input_filepath = "/lore/dolanj/AMP_Modeling_Data/Thermocouple_Simulation/Thermocouple_Sim_Results/Thermocouple_Parallel_60W_160mms/";
    std::string input_filename = "Thermocouple_Parallel_60W_160mms";
    std::string output_filename = "sol_transfer_code_test.csv";
    double grid_origin[] = {50e-6, 500e-6, 0};
    int num_parts = 4;
    int grid_dir[] = {1,1,1};
    int num_pts[] = {10,10,1};
    double grid_spacing[] = {20e-6, 20e-6, 5e-6};

    create_grid_csv_temp_hist_file(input_filepath, input_filename, num_parts, output_filename, grid_origin, grid_dir, num_pts, grid_spacing);
}
```

Input Arguments

- input _filepath
    - The relative or absolute path to the folder containing the results subfolders for each timestep from an Albany FE simulation
- input_filename
    - The base name of the results subfolders and the simulation pvd file
- num_parts
    - The number of mesh partitions used for the specific simulation. This can be determined by the amount of numbered folders that exist inside the folder for the solutions at each timestep
- output_filename
    - The complete name of the file to which the interpolated regular grid results will be written to. Include the .csv extension
- grid_origin
    - An array of three numbers defining the x, y, and z coordinates respectively of the origin corner of the regular grid
- grid_dir
    - An array of three numbers defining the direction that the regular grid extends, either in the positive or negative x, y, and z directions respectively. These numbers shall be defined as 1 for the positive direction or -1 for the negative direction. **Note: Always set these equal to 1. This is a legacy feature, and the current standard for communication with the phase field model requires positive steps in all grid directions.**
- num_pts
    - An array of three numbers defining the number of regular grid points to be created in the x, y, and z directions respectively. The total number of points generated will be the product of these three values
- grid_spacing
    - An array of three numbers defining the spacing between each grid point in the x, y, and z directions respectively. The total length of the regular grid in each direction is the product of the number of points and the grid spacing in the given direction.

Function Call

Call the function with the input arguments ordered as shown in the figure above.

Function: create_grid_csv_temp_hist_file_consol

This function is identical to the `create_grid_csv_temp_hist_file` function, except that there are several additional input arguments that control the estimated consolidation of the powder layer. These additional input arguments are described below. Note that this function is a temporary replacement until the powder layer consolidation can be computed properly within the 3DM problem. This function will consolidate the entire powder layer uniformly, even regions that have not been melted. Care must be taken to ensure that all points in the regular grid are located inside a fully melted portion of the mesh. In addition, the grid will have to be positioned in the location where the consolidated powder will be, rather than its current location in the input mesh. For example, the unconsolidated powder layer will begin at z=0 in the input mesh. After consolidation, the top of the powder layer will be at a distance equal to the following.

Top of consolidated powder layer = z = powder layer thickness * initial porosity

Usage Example

```
#include "FEM_solution_manipulation.hpp"

int main()
{

    std::string input_filepath = "/lore/dolanj/AMP_Modeling_Data/Thermocouple_Simulation/Thermocouple_Sim_Results/Thermocouple_Parallel_60W_160mms/";
    std::string input_filename = "Thermocouple_Parallel_60W_160mms";
    std::string output_filename = "sol_transfer_code_test.csv";
    double grid_origin[] = {50e-6, 500e-6, 0};
    int num_parts = 4;
    int grid_dir[] = {1,1,1};
    int num_pts[] = {10,10,1};
    double grid_spacing[] = {20e-6, 20e-6, 5e-6};
    double powder_layer_thickness = 50e-6;
    double initial_porosity = 0.7;

    create_grid_csv_temp_hist_file_consol(input_filepath, input_filename, num_parts, output_filename, grid_origin, grid_dir, num_pts, grid_spacing
                                          powder_layer_thickness, initial_porosity);
}
```

Additional Input Arguments

- powder_layer_thickness
    - The measurement from the top of the powder layer (at z=0) to the top surface of the substrate. This value should match both the value used as an input to the 3DM simulation from which the results are being interpolated and the actual z thickness of the powder region in the solid model used in the simulation. (These values should always be the same. If they are not, the 3DM simulation results would be invalid.)
- initial_porosity
    - A decimal value between 0 and 1 that represents the initial packing of the powder layer before consolidation. This value should match the value

used as an input to the 3DM simulation from which the results are being interpolated.

Low-Level Code Information

node_location_tests

The `node_location_tests` file contains functions that are used to select specific nodes from the FE unstructured mesh. The functions are passed data from the nodes, most commonly the x,y, and z coordinate of a node point. The function will perform a test, then return 1 if the test passes or 0 if the test fails.

For example, the function “thermocouple_1” in `node_location_tests.cpp` was used to find nodes at the interface of a thermocouple junction from an early test simulation. The equations describing a thin disc at the modelled thermocouple interface were computed using the coordinates of each node in the FE mesh. If the nodes were found to lie within the thin disc region, they were put into a special group. Once all nodes were searched, the temperature values from the nodes in this special group were averaged to find the simulated gauge temperature of the thermocouple.

In general, previously created functions should not be modified and new functions should be added for each individual new use case. This will allow previous code to remain functional, in case old FE solutions need to be re-interpolated. If there is a similar test function that already exists, copy and rename it.