Model Based Control of Fused Powder 3D Printing

Samuel A. Stodder

Stodder Engineering Services

ABSTRACT

Powder Bed Fusion 3D printing requires precision control of thermal energy deposited onto polymer powder particles in turn to control polymer phase transition at voxel resolution. With HP's proprietary Multi Jet Fusion process (MJF), such control is delivered through printer parameters that adjust the printer's thermal system with the application of radiant energy and print agents. Currently, identifying a set of optimal printer parameters requires users to develop process expertise. Embedding a MJF process Digital Twin in a model-based control system holds the promise to significantly simplify and automate the process optimization procedure. A thermal simulator of the MJF technology has been developed for this purpose. This simulator runs near real-time, solving a vectorized 3D multi-physics model using explicit finite difference method accelerated by GPU massive parallelism. It can be used as a "virtual sensor" outputting voxel level thermal and material phase states that are otherwise infeasible to measure experimentally.

1. Introduction

This paper describes a simulation of HP's proprietary Multi-Jet-Fusion printing process, commonly known as MJF. The process of melting plastic powder layer by layer in an MJF printer is based on sweeping a fusing lamp wand whose energy is absorbed into a printed fusing agent (FA) and is mostly reflected off of unprinted powder. The area of the bed where a part is to be formed is printed with the energy absorbing ink. This ink causes a rapid rise in temperature until a melt occurs. In addition, a printed detailing agent (DA) is used as an actuator to evaporatively cool where needed.

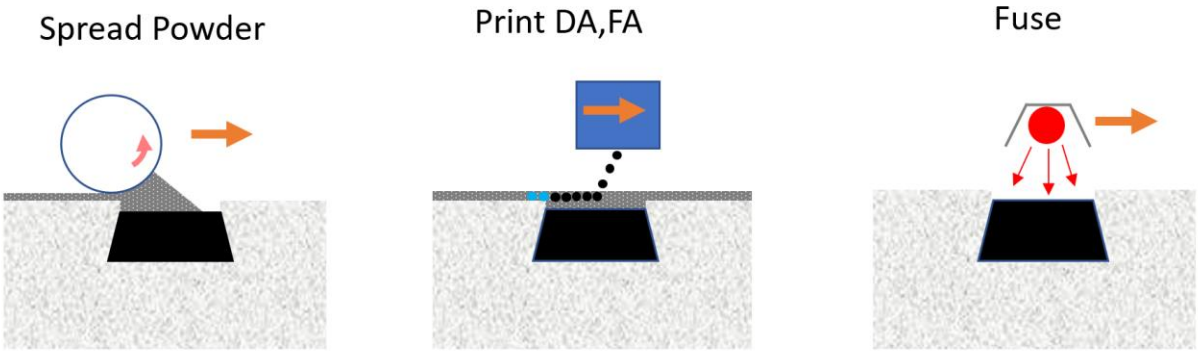

Figure 1 *3D layering process*

A proper powder melt is dependent on three processes to occur; melt coalescence, crystal structure breakdown, and molecular entanglement (reptation). These processes are all very dependent on the temperature vs. time path taken. Voxel by Voxel control of Temperature over time is fundamental to attain consistent geometry and material properties. For example, melt coalescence must complete before the next layer of powder is spread so that full densification occurs. A complete densification allows for the correct volume of powder to be melted for the next part layer. This requires careful control of the melt history during the layer melt process.

If the temperature history of each voxel is controlled, part size and material property variation can be minimized. A big advantage of the MJF technology over Selective Laser Sintering (SLS) is that the discriminate use of detailing agent provides an actuator for fine detail control of the melt process. In this study, model-based control of detailing agent is being explored as a method to automatically control the thermal history of the process voxel by voxel.

2. Methodology

2.1 Model Based Control. If you could measure voxel temperature in real time, then a control system could be utilized to control temperature. However, measurement access for process control is limited both spatially and temporally. Measurement resolution is too low to pick up fine part detail and is also partially blocked by powder spread, and print/fuse carriage motions. However, a 3d thermal prediction engine can be used in a model-based control system to provide the missing state information.

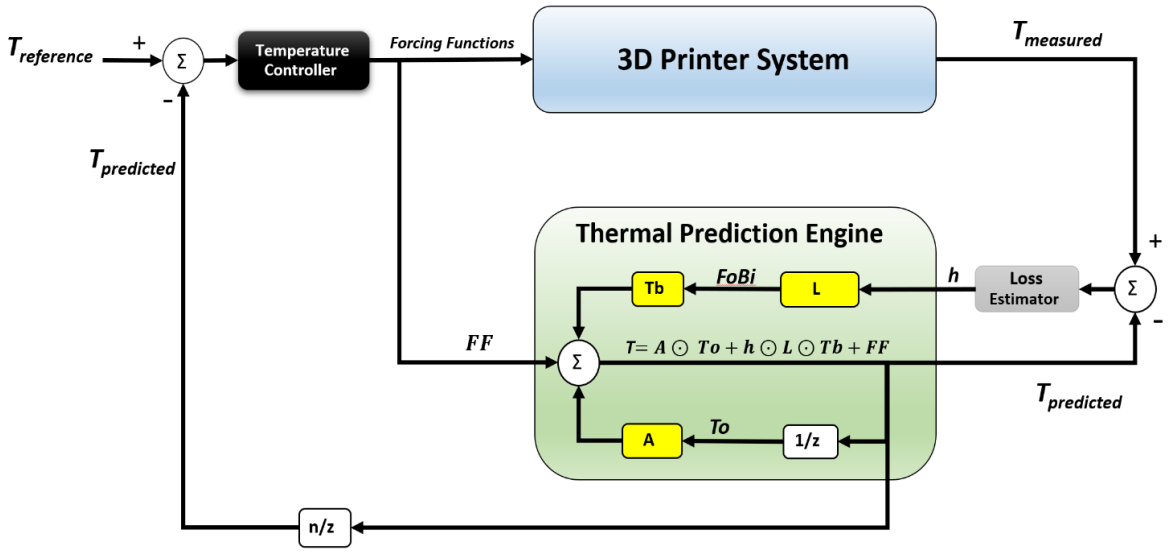

Figure 2 *Model Based Control*

In figure 2 shown above, the thermal prediction engine acts as a digital twin to the 3d printer system. The same time dependent forcing functions that are fed into the 3d printer are fed into the model. The model predicts the next Temperature iteration at a high temporal and spatial resolution. Limited thermal measurements from the plant are compared to the model prediction to modify loss coefficients. In this way the model is prevented from drifting away from the MJF plant due to variable loss behavior across the bed. And finally, the predicted temperature is

compared with the reference to enable a correction to the physical actuators (Detailing Agent and Heat Lamp) for the next layer.

2.2 Thermal Prediction Engine. The thermal prediction engine is a heat diffusion solver that predicts temperature, & material state of a 3D array of voxels at each time step. The model is constructed to utilize 25 z axis node layers. The 22 top layer nodes have a thickness in z equal to one build layer (0.08 mm). At the beginning of a new layer, the temperature and material state (powder or melt) for each voxel is shifted down by one layer and a fresh layer of powder is added to the top of the 3D bed array. To prevent the array from increasing in the number of z axis layer nodes, an accumulator at node layer 23 absorbs the energy of the layer directly above it and increases its thickness by one layer each time a top layer powder spread is added. Node layer 24 is 4 mm thick and node layer 25 is 8 mm thick. The z axis can be limited to 25 node layers because the lower layer temperatures have a small effect on the top layers. In this way, more nodes are not needed to get a good result for top level temperature prediction. The result is a much simpler faster model.

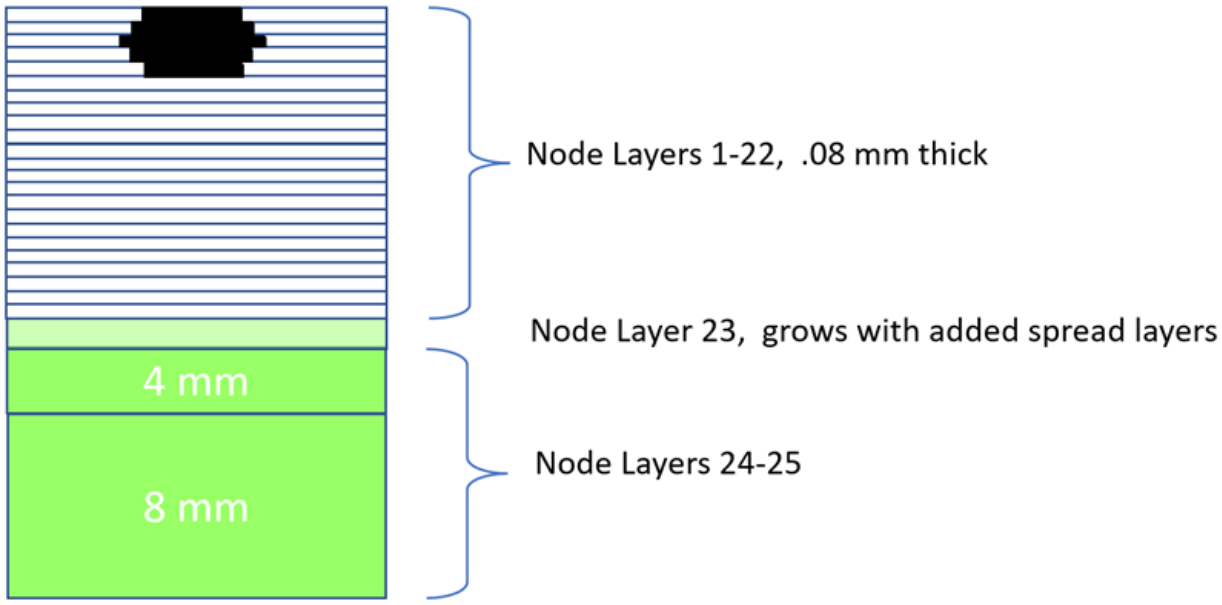

Figure 3 *Z axis node layers*

In figure 4b, the image shows the xy top layer bed temperature field. In figure 4d, the image shows the xz temperature slice of the bed. Figure 4d shows the z axis with a total of 25 nodes. A whole bed array has many nodes in x and y but only 25 in z. Typically the xy nodes are set at 0.24 mm spacing.

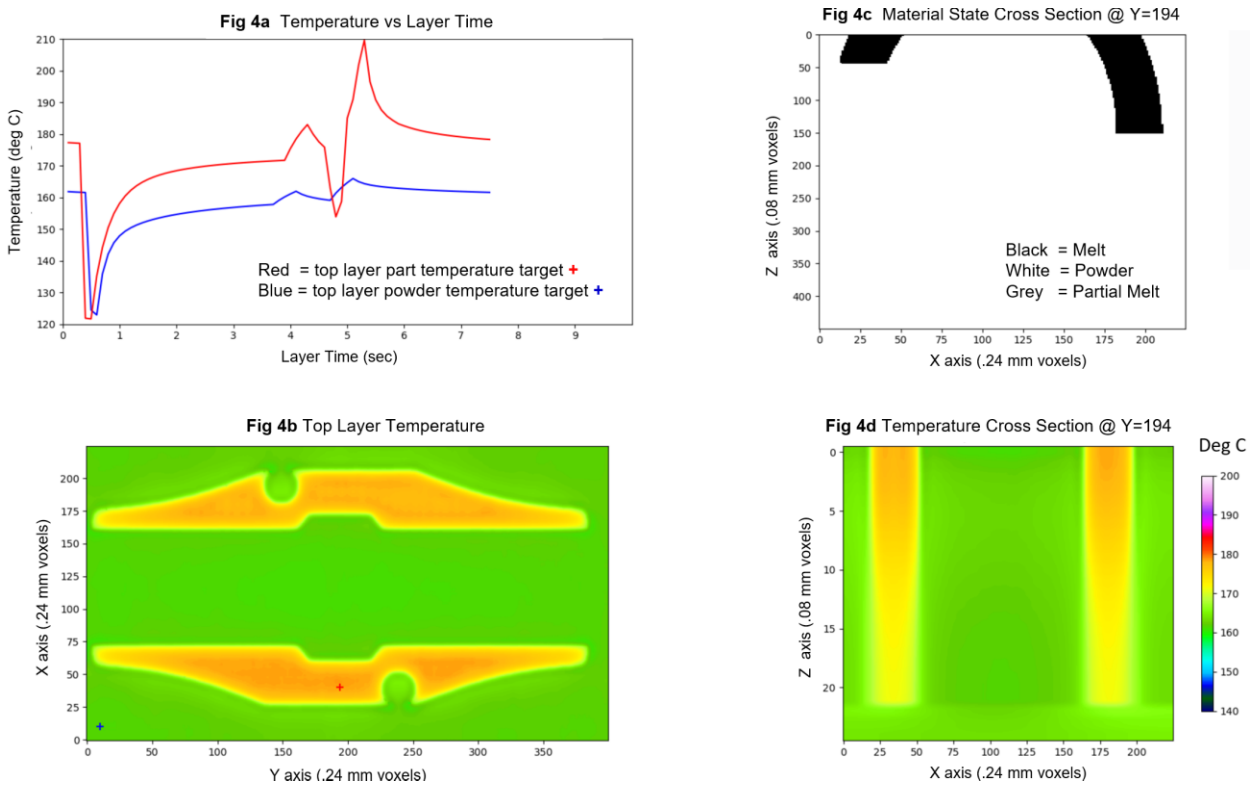

Figure 4 *graphical output at layer time = 7.5 sec*

2.3 Temperature Solver. The solver method used is an explicit finite difference solution to the heat equation. The current temperature is found by element wise multiplication of the transition matrix with the previous time step temperature. This is added to the Boundary condition matrix that is element wise multiplied to the boundary temperatures and finally the Forcing Function (FF) matrix is added to get the next timestep temperature.

$$\boldsymbol{T = A \odot To + h \odot L \odot Tb + FF}$$

A is a whole bed array transition matrix. It can change at every time step.

To is a whole bed Temperature array from the previous time step.

h is the convection matrix

L is the boundary condition matrix

Tb is the whole bed boundary Temperature. It is constant in time.

FF is a whole bed Forcing Function array for the current time step.

2.4 Transition Matrix (A) Since the solver method is explicit, and fully vectorized, the voxel temperature matrix (T) can be solved in parallel with a GPU. Therefore, temperature for every voxel is solved in parallel for each timestep based on the previous time step temperature (To). The transition matrix (A) can change for each time step dependent on the material state of each voxel, its direct neighboring voxels, as well as its location in the bed. For instance, the configuration of the voxel nodes can be interior, face, edge, or corner. (See *Fundamentals of Heat and Mass Transfer*, Frank P. Incropera, 2007, pg. 306) In addition, the z axis height of each layer voxel represented by the nodal equations is larger if they are located towards the bottom of the bed as described previously. This solver implementation is described by 36 different forms of the explicit finite difference equation to account for the different 3D node configurations. In addition, each configuration nodal equation is modified locally, dependent on its voxel material state, and the adjacent voxel material states. The configurations are also locally modified for a tapered densification behavior in the z axis when starting up a melt over powder. Assembling the transition matrix (A) would involve considerably additional memory, so instead the transition matrix values are computed and used to calculate each nodal temperature result "along the way" without storing the whole A matrix.

2.5 Convection Matrix (h) Top layer temperature feedback error is provided by comparing temperature measurements from an IR camera to the predicted measurement output from the model. If there is no IR camera feedback then the convection coefficient is constant for all top surface nodes. If the model is implemented with IR camera temperature feedback, the convection coefficients are modified with a loss estimator to correct any drift between the predicted temperature and the measured temperature. This correction would be made on a once per layer basis. The top surface convection coefficients are spatially smoothed to minimize noise. In this way smooth continuous zonal convection behavior for the model is maintained. The resulting convection matrix is input to the L block in figure 2.

2.6 Boundary Condition Matrix (L) The output of the L block in figure 2 multiplied by the boundary temperatures is a 3d matrix of boundary conditions for the sides, bottom, and top surfaces. The values of the internal nodes are set to zero since they have no boundaries. The values for the bottom and side surface nodes are based on the calculated Fourier number for their particular material states, and geometry. The values for the top surface nodes are set based on Fourier and Biot numbers computed with the top surface convection matrix coefficients. The Biot number changes based on the convection coefficient. The L block filters out input that is close to part and powder boundaries since the IR camera resolution measurements will be unable to pick up finer temperature detail at the part boundaries. This is done by using appropriate part layer dilation and erosion methods to create a filter mask. If the part layer consists of only small part features, then the temperature readings used will be primarily of powder and the part areas will rely on the model prediction. The final output of the L block will be a 3d matrix of boundary conditions. This matrix is then multiplied by the boundary temperature matrix Tb in the difference equation to provide the influence of boundaries on the system. Just as with the A matrix, assembling the whole 3d L matrix would require considerable additional memory. So instead, the boundary condition matrix values are computed and used to calculate each nodal temperature result "along the way" without storing the whole L matrix. In this implementation the output of the L block is a 2d array of convection coefficients fed to the difference equations allowing the influence of boundary values to be computed "along the way".

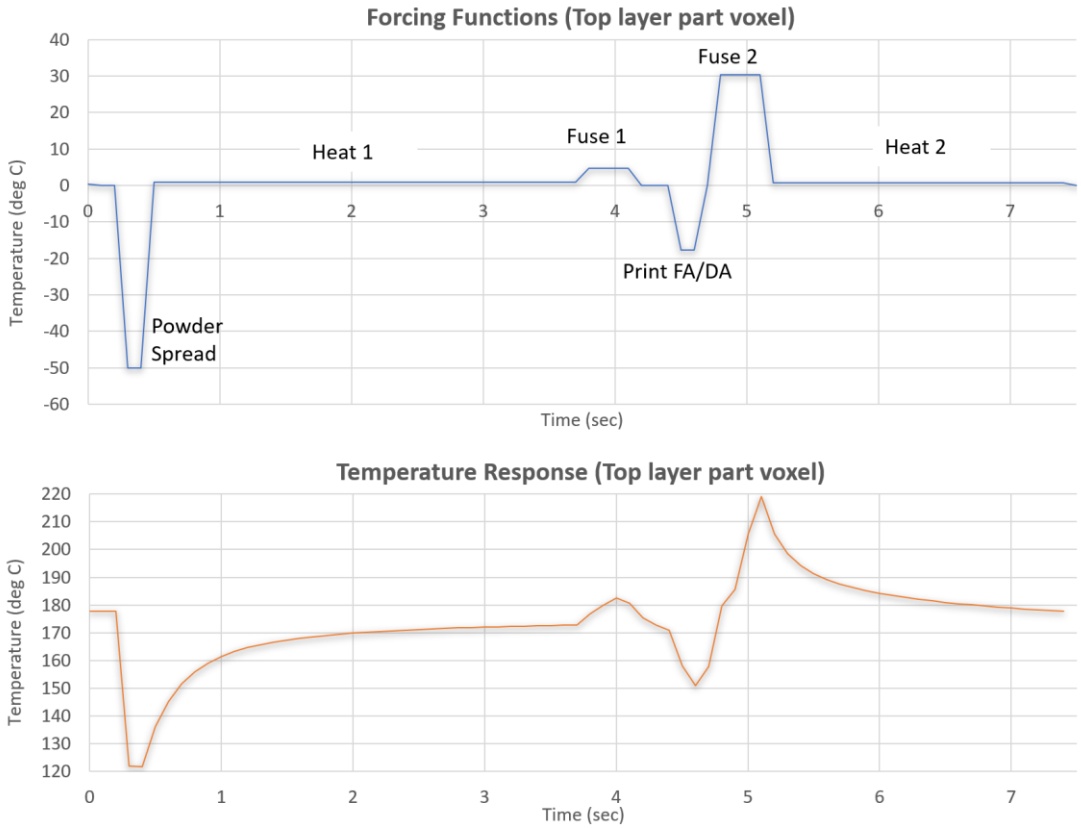

Figure 5 *Model Part Temperature response driven by Forcing Functions*

2.7 Forcing Functions (FF) The Forcing Functions are comprised of 6 parts. See figure 5. First there is a powder spread function where the application of a temperature drop to each top layer voxel for 0.2 seconds occurs to simulate the spread of powder. Second there is constant heat energy (Heat 1) applied to simulate the effect of overhead heating lamps when the powder spread carriage and the print/ fuse carriage are not blocking the radiance. The first heat segment (Heat1) duration is the period of time between when the powder spread carriage has just completed passing over a voxel and the fuse carriage starts to pass over the voxel. Third, there is a pulse of energy (Fuse1) applied to simulate the fusing lamp 1 passing over the bed. Fourth, there is a simulation of the printing of Fusing agent and Detailing agent that occurs just after the Fuse 1 pulse. Both the fusing agent and detail agent forcing functions are calculated as evaporative cooling. Fifth, there is a pulse of energy (Fuse2) just after FA/DA print. The Fuse1, FA/DA print, and Fuse2 functions occur in rapid succession since they are being simulated as three elements on the same carriage. Last, Heat 2 occurs during the period of time that the fuse-print-fuse carriage has just passed over a voxel and the spread carriage starts to pass over a voxel. For a prescribed timestep, the temperature impact of the applied energy sequence is calculated for each particular voxel position and state. These temperature deltas become the FF values for each voxel. The result is that the Forcing Function (FF) is comprised of temperature deltas for each voxel and for each time step of the current layer.

2.7.1 Radiation Extinction The radiant energy of both the heat lamps and fusing lamps penetrates into powder so the FF function applies not just to the surface voxels but also to

underlying voxels. Two radiant extinction models are implemented to distribute the applied energy of the heating and fusing lamps. If a top layer voxel has been printed with FA, the amount of energy absorbed vs reflected increases. Likewise, if the voxel is in a melt state its absorbance increases to a point that blocks lower voxels from radiation. For each top layer voxel, the number of powder layers covering any melted voxel below is determined up to 19 layers to enable the radiant extinction models.

2.7.2 Re-Reflection The fusing lamp applied radiation has an additional complication. The fuse lamp energy flux applied to the top layer voxels varies dependent on the reflective/absorptive content of the top layer. For instance, an area of white powder will reflect energy back to the fuse lamp that then re-reflects back to areas that are absorptive such as FA printed part content. A multi bounce re-reflection model is implemented to account for this behavior. This model utilizes the current printed part slice layer information by convolving it with an appropriate convolution kernel to simulate the re-reflection behavior. Additional corrections are made to account for power bed boundary reflection interaction.

2.7.3 Spatial Voxel timing In this model, a print mode is described as a series of events that occur on a layer basis. This description is loaded from an excel file as FF values and timing parameters that describe what occurs to the voxel at the center of the bed. Powder spread, Heat 1, Fuse1, Print FA/DA, Fuse2, Heat2. See figure 5. The magnitude of these events are set in the print mode except that Heat1 and DA can change layer to layer since they are used as actuators to control powder and part temperature response. Although the timing of the center voxel is set, the timing of all other voxels in the X-Y plane are offset from the center voxel based on their position relative to the center and machine parameters such as carriage speeds and carriage direction of travel. In this way the Forcing Function matrix for each time step incorporates a unique timing for each xy voxel.

2.8 Material State Matrix (TPS) Along with Temperature, the Solver takes the material state for all voxels as input, and outputs an updated material state for the whole xyz bed. The material state matrix is represented as a whole bed array of either 0 powder, 1 melt, or .5 partial melt. See figure 6. If the voxel temperature is less than the melt onset temperature then the state is powder. If it is greater than the melt end set temperature then it is a melt. If it is in between then it is a partial melt. The material state matrix (TPS) drives modifiers that change Fourier numbers in the transition matrix (A) which can result in the transition matrix changing at every time step. Although the thermal analysis acts on only 25 nodes in the z axis, the material state matrix TPS can be much deeper such that the whole part layer history is recorded but only the top 25 voxel material states is used for the solver.

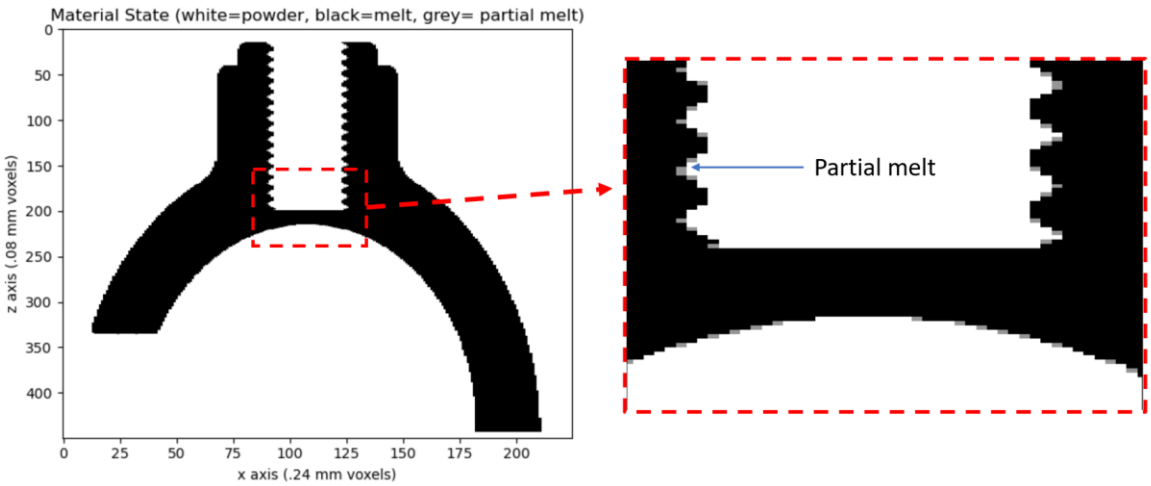

Figure 6 *Material State slice showing powder, melt, and partial melt*

2.9 Melt Level Matrix (meltlevel) The melt level is also an input and output of the solver. It is an accounting of heat energy (J/kg) put into the voxel after the material arrives at the melt onset temperature. Further voxel temperature rise is delayed until the cumulative heat energy input is equal to the heat of fusion (enthalpy) of the material. Once the cumulative heat energy of the voxel is equal to the enthalpy, the temperature rises with continued energy input. A better way to represent the melt state would be to use Differential Scanning Calorimeter (DSC) measurement data for the material when melted close to the process rate to provide a more accurate temperature response. However, this would require significantly smaller time steps and slow down the model. I think there is an opportunity to improve the model in this area to enable a more accurate prediction of less crystalline materials.

2.10 Model Structure The 3d Fuse Simulation model is coded in Python. The main function blocks are shown in figure 7. There are three groupings of functions. The first grouping is executed once at the start of the program. The second contains a set of functions that execute for each 3d print layer, and the third is executed once per time step.

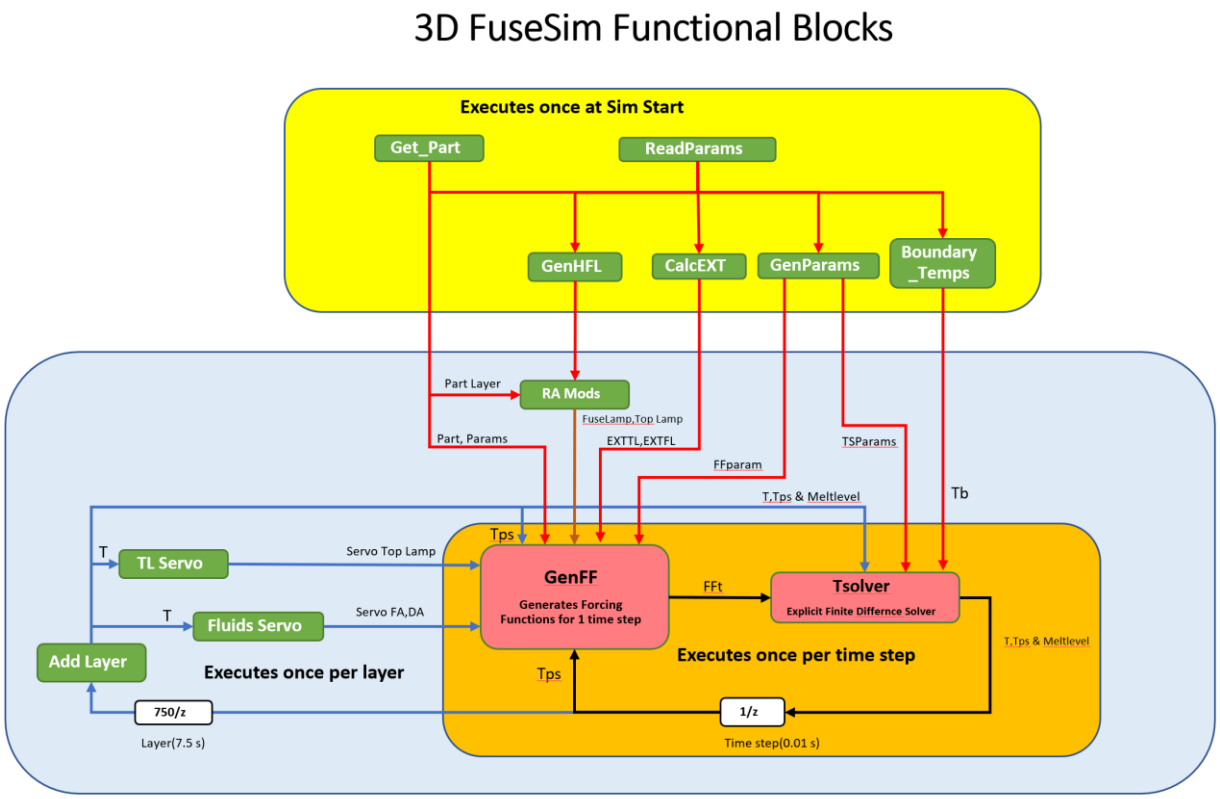

Figure 7

2.10.1 In the first grouping in figure 7 shown in yellow, a part file and a print mode file are loaded when the simulation is started. The part file is read by the Get_Part function. The part file contains a 3d numpy array of a part that has been sliced from an .stl file at the analysis resolution. The print mode file is read by the ReadParams function. The print mode file is an excel spread sheet that contains material properties, machine parameters, and print mode timing values for the center voxel. From the part file and parameter file, the inputs are prepared for the solver (Tsolver) and forcing function (GenFF) blocks with the use of four function blocks that are executed once at the simulation startup. The GenHFL function generates the entire set of layer Forcing Function radiant energy and timing values for each xy location. The CalcEXT generates the extinction coefficients for all voxels that are used by the Forcing Function to distribute radiant energy in the z axis as described in section 2.7.1. GenParams and Boundary-Temps are functions to finalize preparation of the inputs to the Solver and Forcing Function blocks.

2.10.2 The function grouping shown in blue executes at the beginning of each new layer. The Add Layer function shifts the voxel Temperature, material state, & melt level down one layer and adds a new powder layer to the top of the stack. The RA_Mods function takes as inputs the top layer fuse lamp radiant energy and the top layer part slice information to redistribute x, y lamp energy due to re-reflection as described earlier in section 2.7.2. The TL servo function takes the top layer voxel temperature just before the first Fuse pass to determine a correction to the heat lamp power (Heat1) needed to maintain powder temperature control to a reference temperature. The fluid servo function takes the previous end of layer temperature for each top

layer voxel and determines the amount of DA evaporative cooling fluid to be printed on the next layer to control the part region of the top layer to a reference temperature. There are six reference temperatures for DA fluids control. The first one is the primary control temperature for the part. Next there are three successive reference temperatures that control the startup of the first few layers of a part. Next there is a reference temperature that controls the powder at the edge of a part to control thermal bleed. And last there is a top layer reference temperature to prevent thermal bleed to successive powder spread layers. All of these temperature references act on each voxel dependent on its location in the build process.

2.10.3 The function grouping shown in orange contains the GenFF function and the Tsolver function. Both functions utilize a GPU to parallel process the voxels for each time step. The Gen FF function is processed first and then its output is input to the Solver function to be processed. The outputs generated by the Solver are temperature, material, & melt states for each voxel which are graphically updated or stored for each time step or layer as the user chooses.

2.10.4 The model requires a maximum time step of about .01 seconds for numerical stability reasons. The primary mode for this model executes the Forcing Function block and the Solver block every .01 seconds. A second (fast) mode has been created to execute the Forcing function data at .1 second intervals once per layer. In this mode the Solver is updated with new Forcing Function data every .1 sec. Inside the solver block, the model runs 10 iterations with Fourier numbers that are divided by 10. This maintains numerical stability at the cost of some spatial noise during fast transitions. This spatial noise is due to the parallel processing of 10 timesteps of all voxels. So, for those ten timesteps, execution is not necessarily sequential but the Fourier numbers are such that numerical stability is maintained. The spatial noise has minimal effect on the control temperatures of interest because the rate of temperature change at those points is low. The advantage of this mode is a 2.5X increase in computation speed while maintaining temperature control capability

3. Results & Discussion

The model was run using a Nvidia 4070 GPU on a HP Z4 Workstation with an Intel I9 10 core processor and 64 gigabytes of ram. Output of a 4x8 inch model with .25 mm xy resolution and .08 mm z resolution is computed at a speed of 4.5 secs per layer. Real time for the Nylon PA-12 print mode being modeled is 7.5 seconds per layer. This model should be scalable to a larger bed by running it with a larger GPU and more memory resources making it fully capable of real time control of temperature on a voxel basis.

Model correlation with measured MJF printer IR temperatures is shown in figure 8. Temperature measurements are taken from the center of a part where edges don't impact the measurement. Likewise, powder temperatures are taken from an area away from part edges. Measured temperatures drop off where the carriages block IR camera vision. The value that the model brings is that it can predict both the blocked areas as well as part edge areas to provide feedback data for automatic control of voxel temperature.

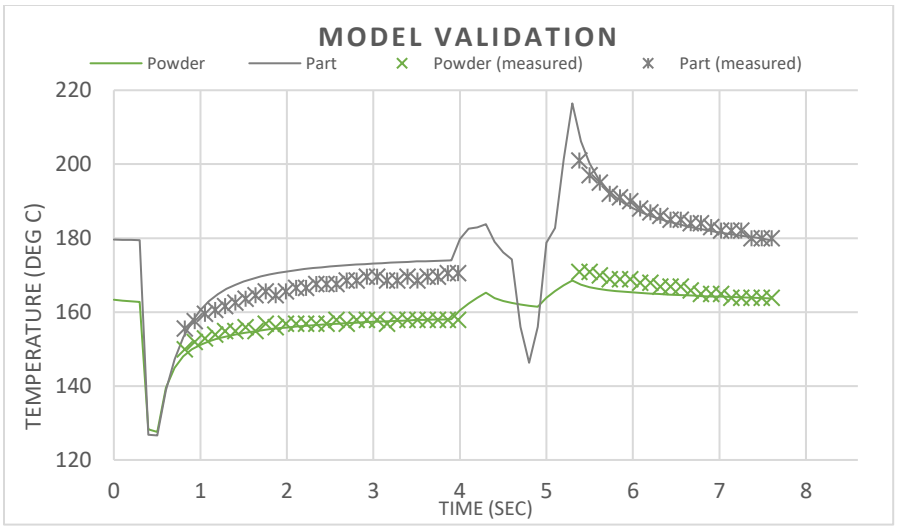

Figure 8 *(Model Validation with MJF IR measurements courtesy of HP)*

As a model based controller, the MJF process can be modified with a limited set of lamp energy, temperature references, and layer time settings. This would allow a customer to quickly optimize a print mode for a particular job. In addition to model-based control of the MJF process, the simulator is useful to study the effects of Forcing Function and Temperature control setpoints before testing in a real machine. As a simulation tool, there are two measures for evaluating the performance of candidate setpoint parameters. First there is the graphical visualization of the melt state. Input parameters can be iterated until voxels with partial melts can be minimized. On the edges of a part layer, this shows up as thermal bleed causing an increase in melt beyond the part slice boundaries. In Figure 9, a small block with 4 square holes of increasing size demonstrates the effect of tuning Fluids Servo parameters. The top right image in figure 9 shows melt growth outside the part slice boundaries that causes holes to be smaller and small holes to be occluded. It also causes the part width to change from the bottom layer as the part builds. However, with proper tuning of the parameters, very little thermal bleed occurs as shown in the lower right image of figure 9.

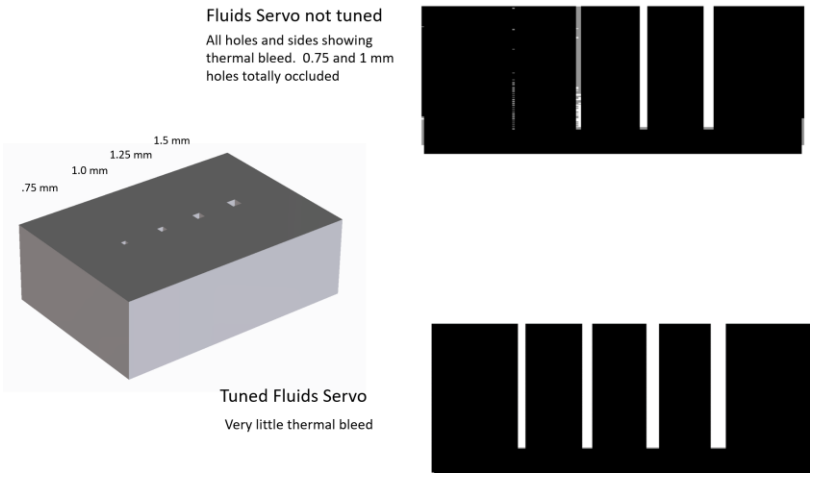

Figure 9 *(Fluids Servo tuning impact on part quality, simulator results)*

The second performance measure is the degree of Coalescence that is reached during a layer process. With the ability to predict voxel temperature history, the degree of coalescence that occurs during the top layer processing can be computed for each voxel and time step. This is accomplished by using an empirically fit finite difference form of the Frenkel coalescence equation for PA-12 powder. The results of this process are mapped out for the top layer of the bed are shown in the lower two images in figure 10 and for a particular voxel in the upper graphs.

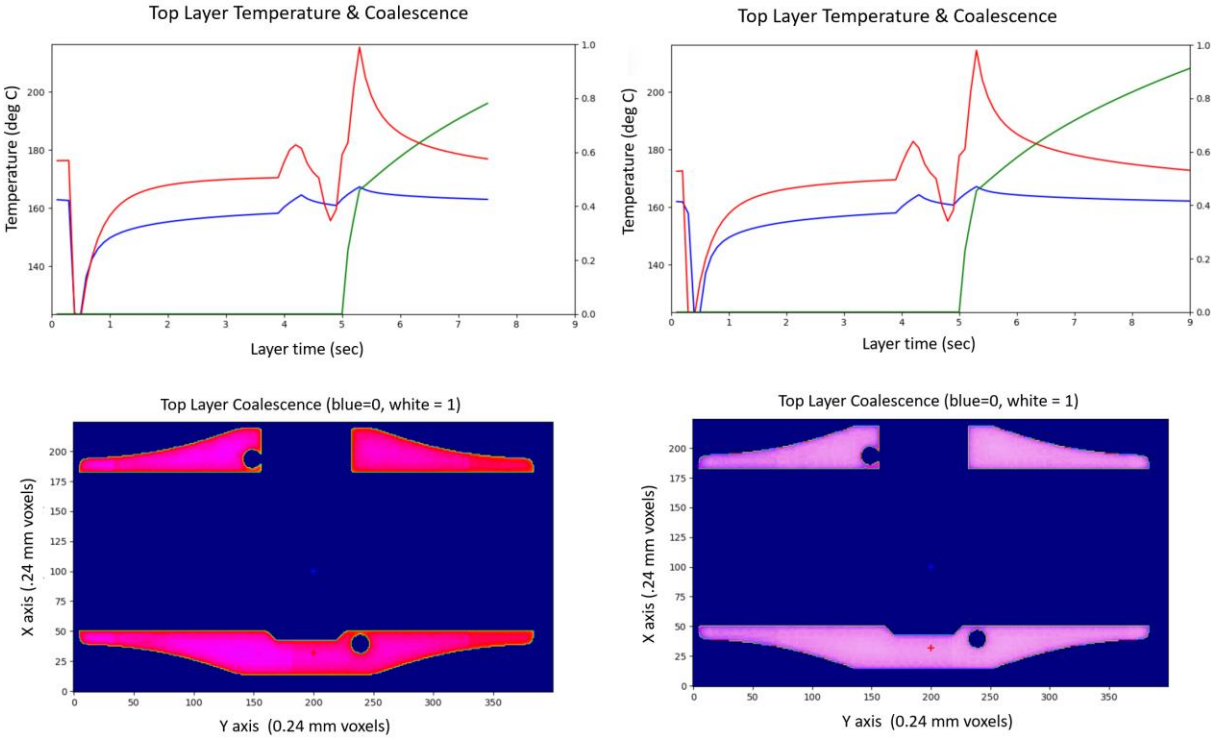

Figure 10 *Coalescence graph mode*

The upper graphs in Figure 10 show the powder temperature in blue, the part temperature in red, and the coalescence index in green. As can be seen on the left upper graph the coalescence process does not fully complete (reach 1) before the next layer of powder is spread. Lack of coalescence completion before the next layer of powder spread can in theory contribute to dimensional variation and reduced material properties. Therefore, it is desired to complete coalescence before the next layer of powder is spread. The model can be used to optimize coalescence by making modifications that impact Temperature history. The upper right graph in figure 10 shows improved coalescence prediction by increasing the layer time by 1.5 seconds to allow for the coalescence to complete. Increasing Temperauture by increasing Fuse Energy could also help but will result in control inefficiencies. Experimenting with the model shows that increasing the coalescence time is the easiest way to improve but at the cost of print time.

In the ways discussed above, the model could be used to speed up the development of print modes for new materials. The material modeled for this study was Nylon PA-12. PA-12 is a fairly crystalline material for which the melt model used for this study is adequate. However, it

may not work so well for less crystalline materials. There is an opportunity to incorporate a more sophisticated melt model to enable the study of a larger material set.

Initially, it was not clear that the simulator could be written to be parallel enough to scale with bed size to achieve real time performance. A 2D ConvLSTM based Machine Learning model was created and trained using the inputs and outputs of the simulator to see if that approach could be faster. The results showed reasonable accuracy and sufficient speed for real time control. However, with further work, the physics-based simulator was coded to be sufficiently parallel and showed similar speed performance but with better accuracy so the Machine Learning approach was dropped.

4. Conclusion

In summary, model-based voxel temperature control can potentially reduce variance of material properties and part dimensions. The MJF simulator can be used as a digital twin to provide automatic control of the MJF process by utilizing recent developments in GPU based parallel computing. In this way the complexity and number of input parameters needed to control the process is reduced. In addition, less time and resources are required to develop print modes for new materials by starting with the MJF simulator.

5. References

Frank P.**Incropera …[et al]** *Fundamentals of Heat & Mass Transfer*, **6th ed. John Wiley & Sons, Inc. , 2007**

2D-Sintering Kinetics of Two Model Fluids as Drops. **J. Muller, Macromolecules 2008, 41, 2096-2103**