NERS 570 Project Report
Laser Scan Thermal Field Prediction

Jun Fan*[1] and Weihao Liu†[2]

[1] *Department of Mechanical Engineering, University of Michigan, MI, USA*
[2] *Department of Climate and Space Sciences and Engineering, University of Michigan, MI, USA*

December 8, 2023

Abstract

Laser Additive Metal Manufacturing (LAMM) is a popular metal manufacturing technique with high geometry flexibility and rapid prototyping capability. Studying of thermal field resulting from different laser scan strategies is important to the understanding of the grain growth during the process and the prediction of the product grain structure of a particular scan pattern. However, the current grain growth model relies on external thermal data, which lacks flexibility in case setup and coupling between thermal diffusion and the material property variation induced by grain growth. In this project, we make use of the topics covered in the NERS 570 course and other modeling techniques to develop a thermal model for grain growth prediction, which is given the name of Laser scan Thermal Field Prediction (LTFP). This model supports various boundary conditions, temperature-dependent material properties, and domain increments during the simulation. The highly modularized solver structure reduces development and maintenance difficulty while allowing LTFP to freely reassemble to integrate into the grain prediction model or to meet various simulation needs.

Keywords: Laser additive manufacturing, Laser scanning pattern, Thermal modeling

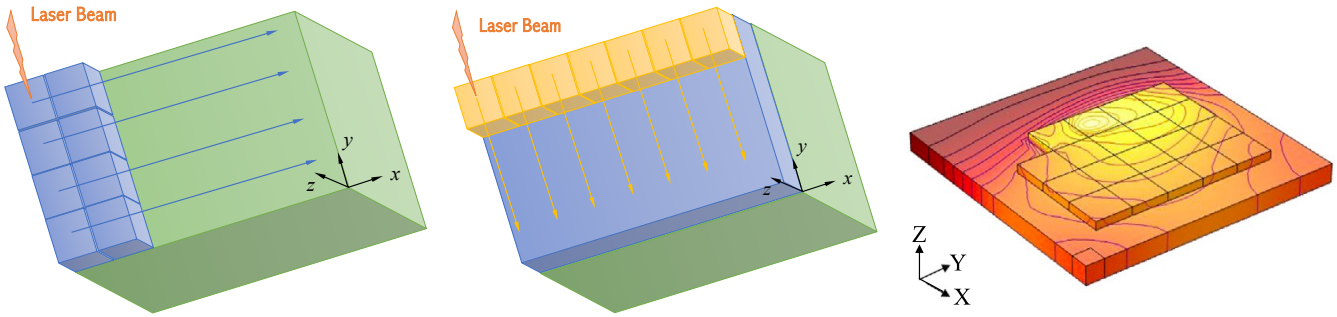
*Email: junfan@umich.edu
†Email: whliu@umich.edu

Contents

1 Introduction

1.1 Background

Laser Additive Metal Manufacturing (LAMM) is at the forefront of technology in the field of additive manufacturing (Gu et al. 2012; Gu 2015). It involves a set of metal additive manufacturing processes that harness the high power density of lasers to melt and fuse the metal powder and build the three-dimensional structure layer by layer. Benefiting from its unprecedented geometry flexibility and rapid prototyping capability, LAMM has gained significant traction across various industries including aerospace, automotive, and healthcare. Despite its widespread adoption, many aspects of LAMM are still not fully understood due to the lack of accurate in-situ monitoring techniques. Consequently, numerical modeling stands as an indispensable tool in the ongoing research and development of LAMM processes.



(a) Laser scans along x-axis. (b) Laser scans along y-axis. (c) Thermal field illustration.

Figure 1: Schematic graph for the laser scan thermal field problem: Panels (a) and (b) illustrate how the laser scans over the top surface of the part. The laser scans along x-axis and then y-axis at the next iteration. A thin layer will be piled up along z-axis after each iteration of scanning. At each scan iteration, the laser beam will also heat the surface through radiation and change the thermal field in this material. Panel (c) gives an illustration of the thermal distribution, taken from Ren et al. 2020.

1.2 Motivation

In most LAMM processes, the laser beam scans a predefined pattern layer by layer to construct the desired 3D geometry of the product, as illustrated in Figure 1(a) and 1(b). This multi-layer deposition has a profound impact on the temperature field of the part, leading to the reheating of the printed layers. The reheating cycles in the thermal history can lead to residual stress concentration, micro-defects, and deformation of the printed part. Furthermore, the thermal history is critical to the grain formation during cooling, thus determining the mechanical and electrical behavior of the printed metal material. Therefore, understanding the thermal history during LAMM is essential for predicting product behavior and can assist in improving the manufacturing process.

The Laser scan Thermal Field Prediction (LTFP) project aims to develop a numerical solver to estimate the thermal field resulting from multi-layer laser scan patterns during the LAMM process, similar to Figure 1(c). The solver should be able to handle dynamic mesh, temperature-dependent material properties, and complex boundary conditions. Later, the solver will be integrated with an established LAMM model as an upstream module that provides thermal history for micro-structure prediction.

2 Methodology

In this section, the governing thermal equation will be first presented in Section 2.1. Then the flowchart of the solver, the algorithm and modules of the code, as well as the documentation part will be covered in Sections 2.2–2.4. These sections act as the base of our subsequent simulations and analysis and are the major achievements of the project in math and programming. Topics to the NERS 570 course are then summarized in Section 2.5.

2.1 Governing Thermal Equations

The thermal field generated by laser scan is governed by the energy equation:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\kappa}{\rho}\nabla^2 T + q, \\ \text{Initial Condition: } T(x,y,z,0) = T_0, \\ \text{Boundary Condition for } T(x,y,z,t). \end{cases} \tag{1}$$

where u is the specific internal energy dependent on the temperature T. Specifically, $u = \int c_p \, \mathrm{d}T$ with c_p being the specific heat capacity. κ denotes the thermal conductivity of the material as a function of temperature, ρ represents the density, and q is the source term of the laser.

In this project, the Finite Volume Method (FVM, Eymard et al. 2000) is used to develop the thermal solver. In the FVM, the computational domain is divided into non-overlapping control volumes with the state stored in each cell (Thompson 1985). In this case, the mean temperature and internal energy values of each control volume are stored at the cell center. At cell interfaces, we define fluxes that provide information on how much of the state flow in and out of the adjacent cells, which can be applied with specific limiters to avoid numerical oscillations (e.g., Sweby 1984; Leonard 1988).

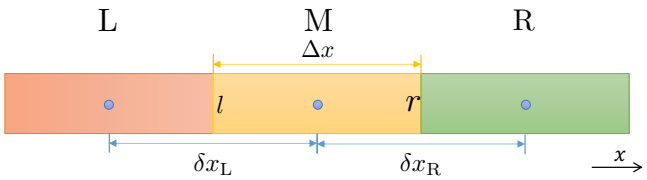


Figure 2: Schmatic graph for the one-dimensional case in the FVM.

In the simplified one-dimensional case, a cell M, its left neighbor L, and its right neighbor R are demonstrated in Figure 2. Let the length of cell M be Δx, and the distances between the center of cell M and L be δx_L, and that between cell M and R be δx_R. In the FVM, with the initial and boundary conditions, Equation 1 can be derived as:

$$\frac{\mathrm{d}\bar{u}}{\mathrm{d}t} = \frac{1}{\rho\Delta x}\left(\kappa\frac{\partial T}{\partial x}\bigg|_r - \kappa\frac{\partial T}{\partial x}\bigg|_l\right) + \bar{q}, \tag{2}$$

$$\Longrightarrow \frac{\mathrm{d}\bar{u}}{\mathrm{d}t} = \frac{1}{\rho\Delta x}\left[\frac{\kappa_r\left(T_\mathrm{R} - T_\mathrm{M}\right)}{\delta x_\mathrm{R}} - \frac{\kappa_l\left(T_\mathrm{M} - T_\mathrm{L}\right)}{\delta x_\mathrm{L}}\right] + \bar{q}, \tag{3}$$

where averaged variables are defined as $\bar{X} \equiv \frac{1}{\Delta x}\int_l^r X \mathrm{d}x$, and the first and second term in the right-hand side of Equation 2 represent the flux caused by the internal flow and external source, respectively. With central discretization, Equation 2 can be discretized in the form of Equation 3.

Furthermore, in the three-dimensional case, the averaged values can be defined as $\bar{X} \equiv \frac{1}{\Delta V}\int_\Omega X \mathrm{d}V$, where Ω is the control volume with the size of ΔV. In this way, Equation 1 can be finally written as

$$\begin{aligned}\frac{\mathrm{d}\bar{u}}{\mathrm{d}t} = \frac{1}{\rho\Delta V}\Bigg[&\frac{k_{x^+}\left(T_{x^+}-T_\mathrm{M}\right)}{\delta x_+}\Delta y\Delta z + \frac{k_{x^-}\left(T_{x^-}-T_\mathrm{M}\right)}{\delta x_-}\Delta y\Delta z \\ &+ \frac{k_{y^+}\left(T_{y^+}-T_\mathrm{M}\right)}{\delta y_+}\Delta z\Delta x + \frac{k_{y^-}\left(T_{y^-}-T_\mathrm{M}\right)}{\delta y_-}\Delta z\Delta x \\ &+ \frac{k_{z^+}\left(T_{z^+}-T_\mathrm{M}\right)}{\delta z_+}\Delta x\Delta y + \frac{k_{z^-}\left(T_{z^-}-T_\mathrm{M}\right)}{\delta z_-}\Delta x\Delta y\Bigg] + \bar{\dot{q}} \equiv f(T),\end{aligned} \tag{4}$$

where the right hand side of Equation 4 is a function of temperature, $f(T)$.

Multiple time-stepping methods can be implemented for the transition model. In the current model, the Forward Euler (FE, Gottlieb and Ketcheson 2016) is used:

$$\frac{\mathrm{d}\bar{u}_{i,j,k}}{\mathrm{d}t} = \frac{\bar{u}^{(n+1)}_{i,j,k} - \bar{u}^{(n)}_{i,j,k}}{\Delta t^{(n)}_{i,j,k}} = f\left(T^{(n)}_{i,j,k}\right) \implies \bar{u}^{(n+1)}_{i,j,k} = \bar{u}^{(n)}_{i,j,k} + \Delta t^{(n)}_{i,j,k}\, f\left(T^{(n)}_{i,j,k}\right). \tag{5}$$

The von Neumann stability condition (Isaacson and Keller 2012) is used to ensure stability of the model:

$$\Delta t^{(n)}_{i,j,k} \leqslant \min\left[\frac{(\Delta x)^2}{6\kappa^{(n)}/\left(\rho c_p\right)}, \frac{(\Delta y)^2}{6\kappa^{(n)}/\left(\rho c_p\right)}, \frac{(\Delta z)^2}{6\kappa^{(n)}/\left(\rho c_p\right)}\right]. \tag{6}$$

After calculating the new internal energy, we can then obtain the new temperature field. Then the cooling rate and temperature gradient are also computed for the grain growth model:

$$-\frac{\partial T^{(n)}}{\partial t} = \frac{T^{(n-1)} - T^{(n)}}{\Delta t}, \quad \nabla T^{(n)} = \left(\frac{T^{(n)}_{i+1,j,k} - T^{(n)}_{i-1,j,k}}{2\Delta x}, \frac{T^{(n)}_{i,j+1} - T^{(n)}_{i,j-1}}{2\Delta y}, \frac{T^{(n)}_{i,j,k+1} - T^{(n)}_{i,j,k-1}}{2\Delta z}\right). \tag{7}$$

2.2 Solver Workflow

A typical LTFP simulation goes through the following steps:

(a) To start with, the inputs are loaded and used to initialize the variables and the data structure, including the mesh size, boundary setup, material property, etc.

(b) Next, the solver moves into the main part of time marching.

1. Check whether domain increment for the new layer should be performed, and pre-compute the laser power distribution.
2. Solve the diffusion equation shown in Equation 1 using the temperature field from the last step, thermal boundaries, and laser power distribution.
3. Compute the cooling rate and temperature gradient for the grain growth model.
4. Output current state at certain time steps during the simulation.
5. Advance in time and print the current status of the simulation to the console.
6. Check if the current time reaches the end time after one complete time step.
   - If yes, exit the loop.
   - Otherwise, continue looping (return to Step 1) till the end time.

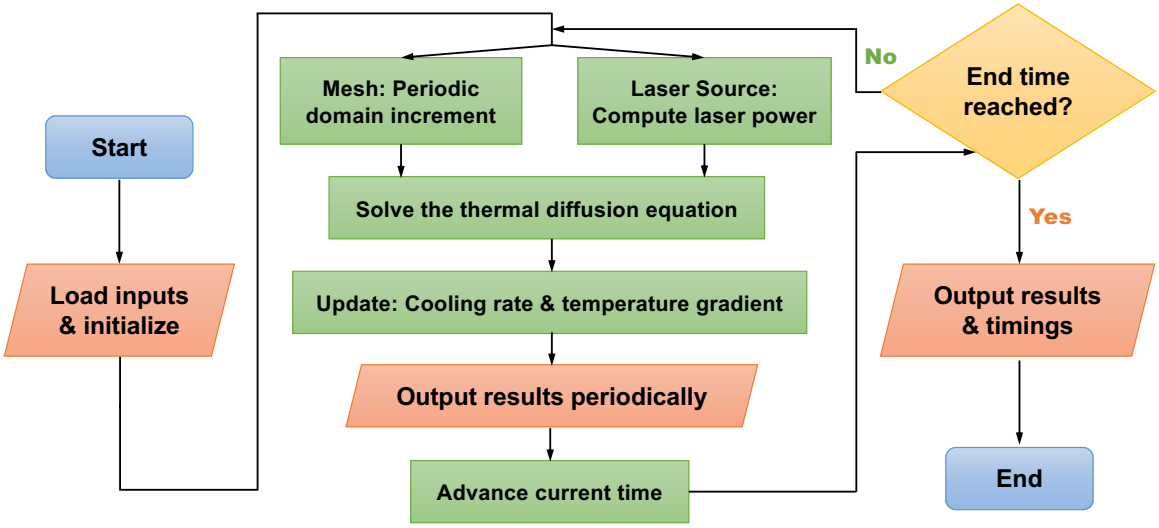


Figure 3: Flowchart of a typical LTFP simulation.

(c) Lastly, output the final state and deallocate all variables and pointers used in the computation.

Figure 3 shows a general flowchart of the solver. It is worth noticing that this section only represents the standard workflow of LTFP. Given the highly modularized design of the solver, the user can easily modify the workflow by switching the sequence of steps, removing unwanted modules, or adding new modules.

2.3 Code Implementation

The code is mainly written in C++ with an object-oriented design. All modules involved in the simulation are written as different class objects. The name and major functionality of the modules are listed in Table 1.

Table 1: Task description and assignment.

Module	Description
Simulator	Manage overall simulation workflow
Scene loader	Load simulation setup and material property from configuration files
Time manager	Keep track of time step size and simulation time
Thermal solver	Solve the thermal equations
Mesh and data	Store mesh information and mesh-based data structure
Material property	Store material properties and compute temperature-dependent values
Thermal boundary	Store various types of boundaries in the form of inherited class objects
Laser source	Keep track of laser positions and compute laser power distribution
Exporter	Export results to csv and vtk format

The module relations and communications are shown in Figure 4. The main function calls a simulator object, which manages the simulation workflow from initialization, time marching, to finalization. The simulator then calls the modules to conduct different simulation steps. The major advantage of modu-

larization is that we can easily reassemble them to meet different simulation needs. Most modules here are made singleton, and thus they can call each other without passing the data through the simulator class. In this way, when the code is later integrated into the grain growth model, communication among different modules will not be affected.

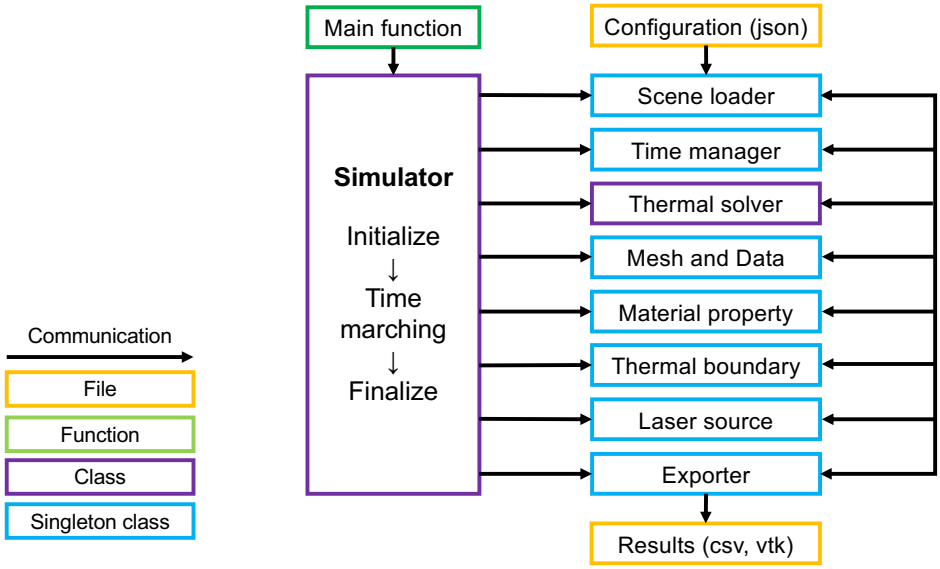
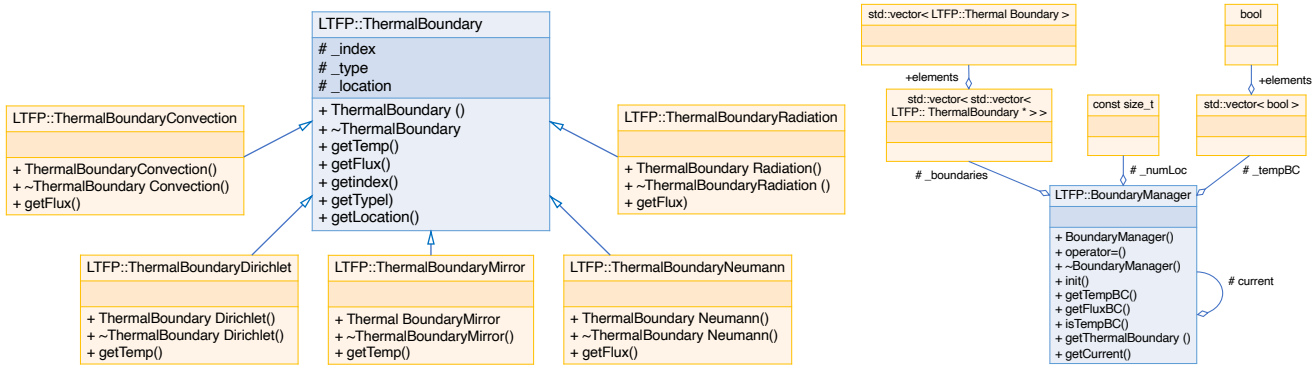


Figure 4: The modules and flow of communication in LTFP.

Besides, we use the inherited class and corresponding singleton manager object to keep track of different types of objects in the same category and provide unified interfaces. For example, the thermal boundary module utilizes a singleton manager class to manage various boundary types including Dirichlet, Neumann, convection, radiation, and mirror boundaries. All these boundaries are the inherited classes of a non-singleton base class, which contains the unified interface for access to boundary temperature or heat flow. This inherited-manager structure realizes the separation between the specific boundary types and the thermal solver, which simplifies the coding of new thermal solvers as well as the addition of new boundary types. Figure 5 demonstrates the class object relations within this type of structure.



(a) Boundary class inheritance. (b) Thermal boundary manager class.

Figure 5: UML diagrams of thermal boundary structure generated by Doxygen.

2.4 Documentation

The LTFP source code is managed on a GitHub repository at https://github.com/Lazy-Beee/ners570f23-LTFP, where it serves as a central hub for version control and collaborative coding. Moreover, to ensure clarity and comprehensiveness, we have integrated comprehensive documentation into our code through Doxygen, which allows the automatic generation of a well-organized HTML document.

2.5 Topics Related to the Course

In this project, with the formulas derived in Section 2.1 for the numerical solvers, we construct the framework of the solver using some high-level design techniques as shown in Section 2.3, including the modules in Figure 4. During this process, we will validate the solver outputs by having some benchmark tests as illustrated in Section 3. Related topics of the course materials are demonstrated in Table 2, such as object-oriented programming, parallel computing, software development, testing, etc.

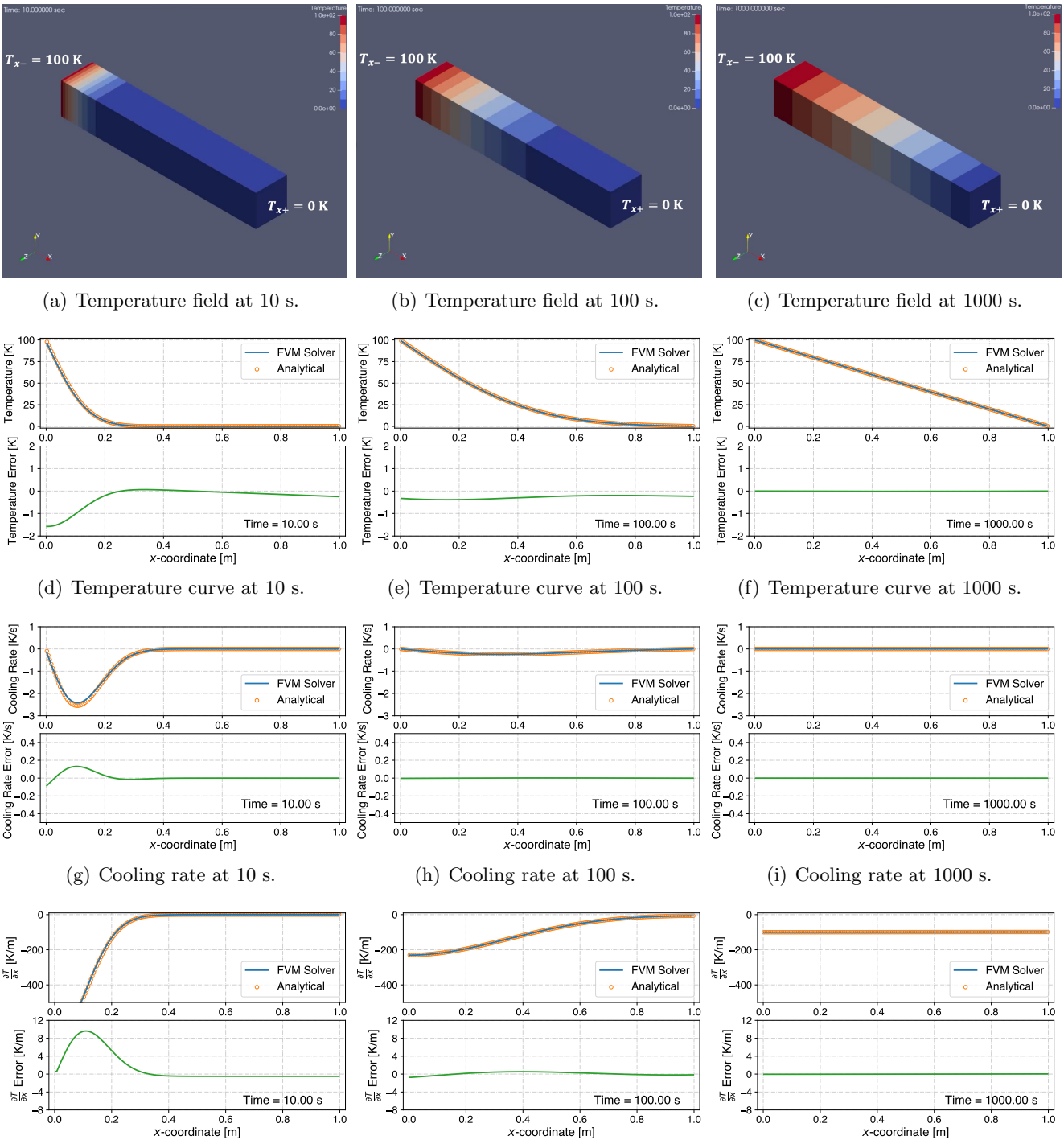
Table 2: Task description and assignment.

Task	Description	Member	Related topics
Derivation	Derive math formulas for numerical solvers	Weihao	
High-level design	Develop high-level design of solver and maintain main function	Jun	High-level design, Object-orientated programming
Pre-processing	Load and save configuration of simulation, initialize class objects and solver environment	Jun	
Mesh and data	Generate mesh and save mesh-based data including temperature field	Weihao	Matrix storage
Boundary models	Store three types of thermal boundary model and manage laser heat source	Jun	
FVM solvers	Solve thermal diffusion using multiple types of finite volume solvers	Weihao	Solving linear systems
FVM solver parallelization	Parallelize FVM solvers to increase solver performance	Jun	Parallel computing using OpenMP
Post-processing	Export data to vtk files and visualize results	Weihao	
Validation	Validate solver using benchmark cases	Both	Software development and testing
Documentation	Generate documentation using Doxygen	Both	Version control

3 Results and Discussions

3.1 One-dimensional Test

The testing begins with a simple one-dimensional case. The boundary conditions for y and z directions are adiabatic, while the Dirichlet boundary conditions at $x^- = 0$ and $x^+ = 1$ have values of 100 K and 0 K, respectively. The temperature field is initialized to be 0 K, and there is no external heat source.



(a) Temperature field at 10 s. (b) Temperature field at 100 s. (c) Temperature field at 1000 s.

(d) Temperature curve at 10 s. (e) Temperature curve at 100 s. (f) Temperature curve at 1000 s.

(g) Cooling rate at 10 s. (h) Cooling rate at 100 s. (i) Cooling rate at 1000 s.

(j) Temperature gradient of x at 10 s. (k) Temperature gradient of x at 100 s. (l) Temperature gradient of x at 1000 s.

Figure 6: Figures for the one-dimensional test: Panels (a)(b)(c): The temperature field visualized in Paraview at 10, 100, and 1000 s, respectively. Panels (d)(e)(f): The temperature curve dependent on x-coordinates at 10, 100, and 1000 s, respectively. Panels (g)(h)(i): The cooling rate curve dependent on x-coordinates at 10, 100, and 1000 s, respectively. Panels (j)(k)(l): The temperature gradient along x-direction curve $\left(\frac{\partial T}{\partial x}\right)$ dependent on x-coordinates at 10, 100, and 1000 s, respectively.

In this case, the analytical solution for the temperature field can be expressed as:

$$T^*(t,x) = \sum_{n=1}^{+\infty} \left\{ \frac{200}{n\pi} \left[\frac{(-1)^n}{n\pi} - 1 \right] \sin(n\pi x) \cdot \mathrm{e}^{-\frac{k}{c_p \rho}(n\pi)^2 t} \right\} + 100 - 100x, \ t \geqslant 0, \ 0 \leqslant x \leqslant 1, \tag{8}$$

for any y and z within the domain. Figure 6 then shows every detail for the numerical solution compared with the analytical one at 10, 100, and 1000 s. It can be observed from Figures 6(a), 6(b), and 6(c) that, as time evolves, the temperature starts to diffuse from the hotter side to the colder side with heat flux transported, and finally the temperature field is close to a linear function of x-coordinate. During this process, there is no variation in the temperature field along the y- and z-axes. Figures 6(d), 6(e), and 6(f) show the temperature compared with the analytical solution, Figures 6(g), 6(h), and 6(i) show the cooling rate compared with the analytical solution, and Figures 6(j), 6(k), and 6(l) show the temperature gradient along x compared with the analytical solution.

With the numerical and analytical solutions, the error norms of these three variables are computed over all cells, normalized by the initial ones, i.e., the error norm of the outputs at the first step. The relative error norm convergence is plotted as a function of time in Figure 7. Though visible differences exist between the solver outputs and the analytical functions in Figures 6(d), 6(g) and 6(j), it is clear in Figure 7 that after 300 s, the error norm of the temperature, cooling rate, and temperature gradient converges with a similar asymptotic order, indicating a first success of the numerical test of the solver.

Similar tests are also conducted with the diffusion direction being the y- and z-coordinates. The same resulting temperature fields are obtained along the $y-$ and z-coordinates, indicating the model is functional in all three dimensions.

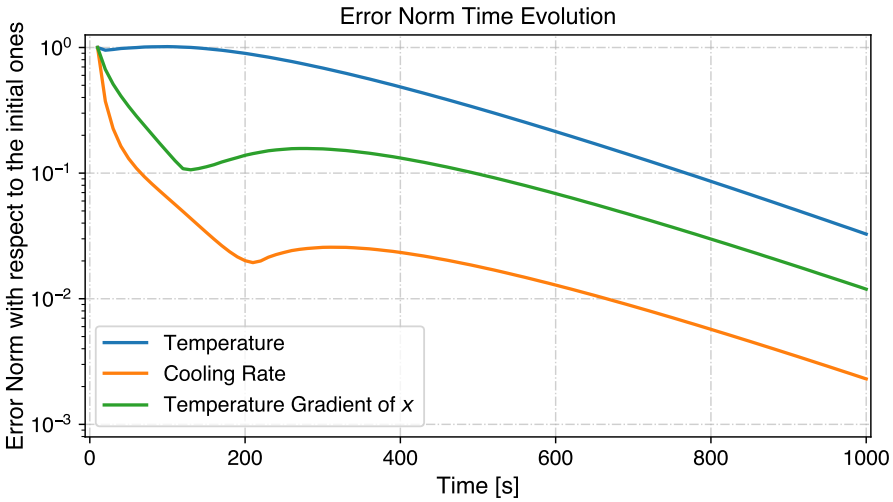
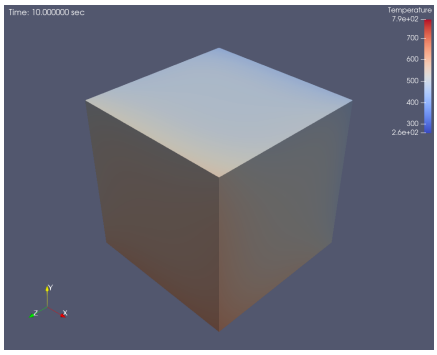


Figure 7: The error norm functions between the numerical outputs (with the FE time-stepping method) and analytical solutions, normalized by the respective initial values.

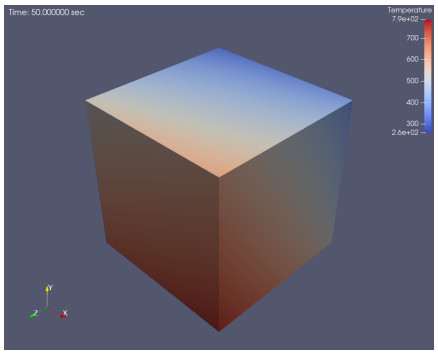
3.2 Energy-Conserved Test

The conversation of the solver is then tested. On the six different surfaces of this cube domain, we apply different flux boundary conditions. All heat flow through the boundary sums up to zero, so the total internal energy in the domain should not be changed over the simulation. In this test, the thermal conductivity and specific heat of the material are temperature-dependent to test the conservation under varying material properties.

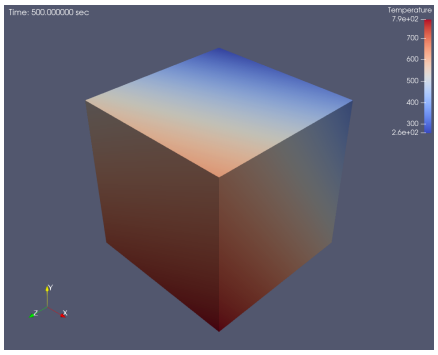
Figures 8(a), 8(b), and 8(c) show the evolved temperature field at 10, 100, and 1000 s, respectively. Figure 8(d) then depicts the total internal energy in the domain and its difference compared to the initial value. As the simulation approaches the steady state, the internal energy does not change and the difference compared with the initial value is zero, suggesting that the solver is energy conserved.



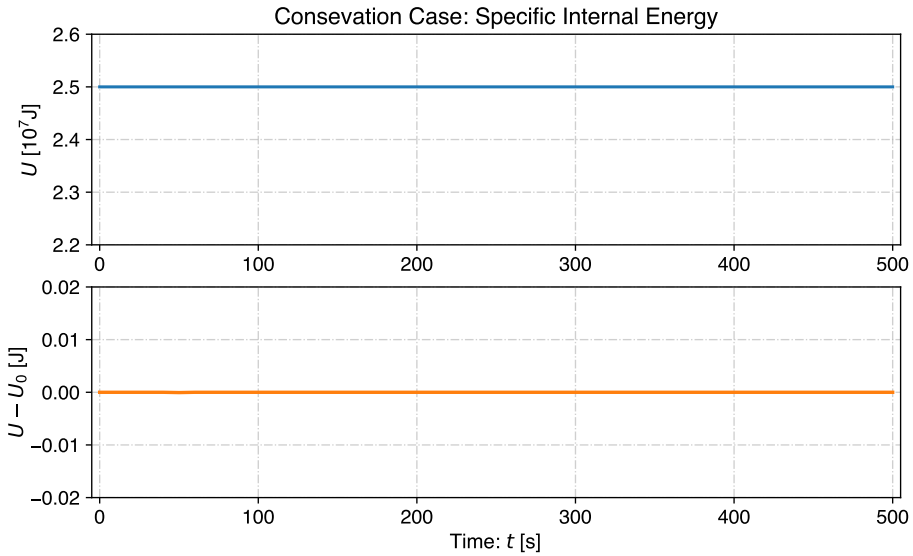
(a) Temperature field at 10 s.



(b) Temperature field at 50 s.



(c) Temperature field at 500 s.



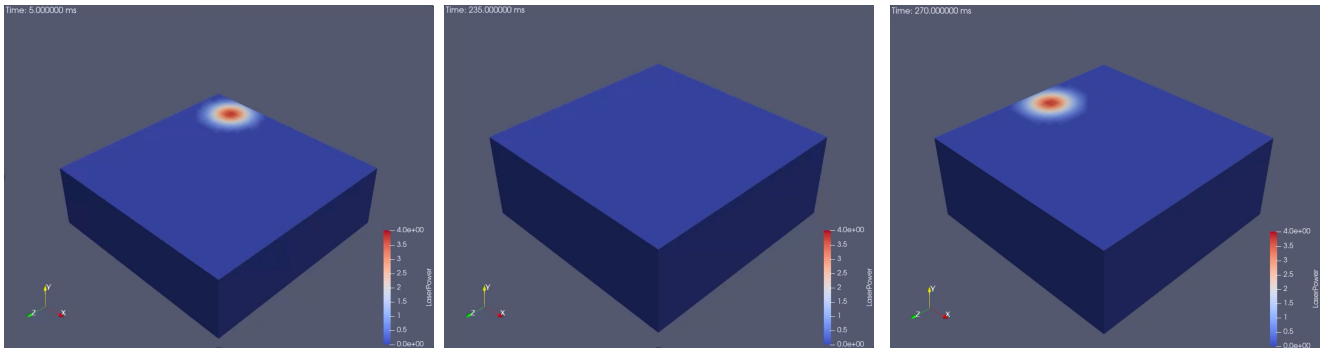
(d) Specific internal energy summed over all the cells and its difference with respect to the initial one as a function of time.

Figure 8: Time evolution of the temperature field, the summed specific internal energy (U) and the difference with respect to the initial value ($U - U_0$).

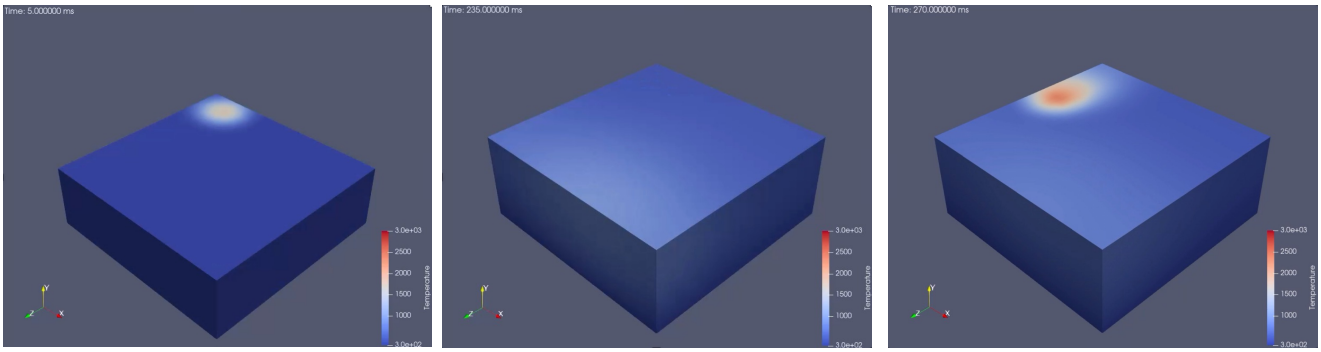
3.3 Functional Test

With the previous two successful tests, we also try a case that mimics the multi-layer scan process. This test case utilizes various solver functionalities including the laser heat source, domain increment, and multiple thermal boundary types to test the overall capability of the modules. A small block of material is used as the simulation domain. Figures 9(a), 9(b), and 9(c) illustrate the laser energy distribution on the top surface. The laser will scan the first layer along the x direction (see Figure 9(a)). On the next layer, the laser will scan along the z direction (see Figure 9(c)). After each layer is scanned, we add a certain thickness on the top surface of the domain (see Figure 9(b) when the laser scan is halted and the domain thickness is increased).

Figures 9(d), 9(e), and 9(f) illustrate the temperature distribution evolved with time. In this process, when the domain gets increment, the temperature field in the new top layer is taken as the same as that of the second top layer. We can observe that the laser-scanned region is heated and the thermal solver is behaving as expected. As the laser continues scanning, the domain gradually heats up, with the remaining thermal field diffusing before the laser scans the next layer. Overall, the simulation result matches our expectations and the functional test is successful.



(a) Laser power at 5 s. (b) Laser power at 235 s. (c) Laser power at 270 s.



(d) Temperature field at 5 s. (e) Temperature field at 235 s. (f) Temperature field at 270 s.

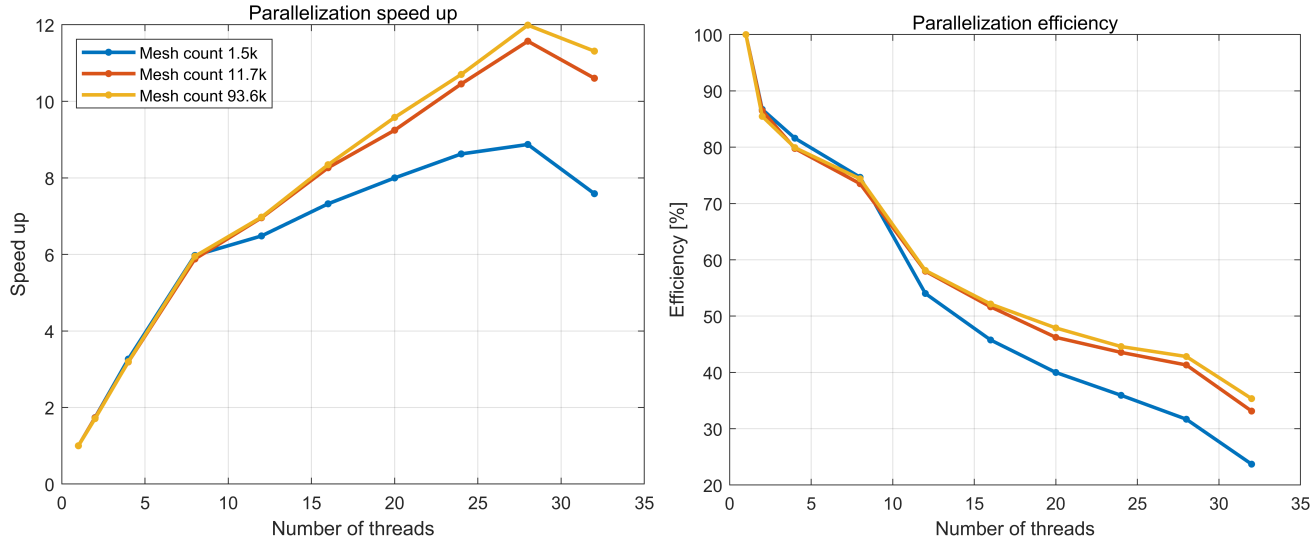
Figure 9: The laser scan power and temperature field in the material at 5, 235, and 270 s, respectively, when the laser scan pattern is along x-axis, halted, and z-axis at these epochs, respectively.

3.4 Parallelization

Based on the functional test in Section 3.3, the parallel performance of the code is analyzed. In the current version of LTFP, parallelization is achieved through OpenMP. The mesh size is changed to generate three cases of different scales, each with 1.5k, 11.7k, and 93.6k cells, respectively. All cases are tested with the thread count from 1 to 32 on a Ryzen 7950X (16C/32T, @ 5.4GHz) CPU.

Figures 10(a) and 10(b) show the parallelization speedup and efficiency, respectively. We can find that all cases have similar speedups for thread counts smaller or equal to 8. Once the thread count is greater than 8, the slope of the speedup curve starts to decrease. This phenomenon is likely caused by the microarchitecture of the CPU. For cases with a thread number smaller than 8, all threads can run on the same die with a shared L3 cache and much lower communication latency. In cases with more threads, the communication cost increases due to the involvement of the slower die-to-die communication. The bend-down curve in Figure 10(a) demonstrates that communication overhead is greatly affecting the

parallel performance in our case. Another equivalently important finding in Figure 10(a) is that, for thread count larger than 8, the cases behave differently. There is an obvious speedup increase from the smallest mesh size to the medium mesh, but not much improvement from the medium mesh to the largest mesh. It is safe to conclude from Figure 10(a) that the parallelization behavior will not change much for even larger cases. For the efficiency, we can observe a dramatic decrease in all cases from Figure 10(b). If the linear speed-up curve in Figure 10(a) is to continue, the efficiency curve will level off around 20% with a large number of threads, which is rather low. In conclusion, though we have applied simple parallelization and made significant improvements compared to the serial performance, there is still room for optimization.



(a) Parallelization Speedup.

(b) Parallelization efficiency.

Figure 10: Parallelization speedup and efficiency as a function of thread numbers.

Furthermore, the computational costs of different parts of the code are recorded. In the serial run, solving the thermal equation takes 99% of the total run time, while this percentage reduces to 76% when parallelized with 32 threads. The ratio change indicates that solving the thermal equations still consumes a large portion of the computational cost, and we still need to focus on improving the parallel performance on this part for solver development in the future.

However, the time consumption of the rest part of the code increases from 1% in serial code to 24% when parallelized with 32 threads. If the code is to run on a platform with stronger parallel capability, the percentage of the thermal equations will continue to decrease. Therefore, optimization of the rest part of the code is still critical to further improvements in the performance.

4 Conclusion

4.1 Summary and Project Value

To summarize, in this project, the algorithm design and major components of LTFP are completed. We first derive the equations implemented in our solver in Section 2.1. We also develop the solver workflow (see Section 2.2) and code structure using modules (see Section 2.3) with skills learned in this course

listed in Section 2.5, for example, the high-level design in the object-oriented programming language, parallelization, test cases. We also create the repository on Git and documentation by Doxygen (see Section 2.4).

Specifically, the following three cases have been simulated using our LTFP solver:

1. One-dimensional test (see Section 3.1): The temperature field along y and z directions is homogeneous and dependent on x-coordinates and time. The outputs have been validated against the analytical results with a steady asymptotic converged rate.
2. Energy-conserved test (see Section 3.2): The boundary conditions differ in different faces on the domain, with the total heat inflow balanced by the outflow, so the total internal energy is expected to be conserved, which is given by the outputs.
3. Functional test (see Section 3.3): Starting from a small block of material, the laser continues the multi-layer scan process with the direction changing from $x(z)$ to $z(x)$ in different layers. The pattern indicates that all the functions work well.

Based on the results of the tests, the code is tested with different numbers of threads on three different meshes with 1.5k, 11.7k, and 93.6k cells, respectively. We then calculate the speedup and efficiency in parallelization (see Figure 10) and find that all these cases have similar speedup for thread count smaller or equal to 8. As the thread number increases, the speedup does not increase so much with the thread number, due to the increasing communication cost. Also, as the thread number gets larger, the parallelization efficiency continues to drop in all cases. We then carefully examine the computational cost of the code in serial and parallelized computing, finding that solving the thermal equations is still the dominating part in time cost, indicating that there is still room for improvements of the solver in the future.

Even with a solver for the thermal diffusion equation that needs further enhancements, we have developed a framework and algorithm for the LTFP so the solutions can be used for the subsequent study such as grain growth. Note that our code implementation is built by modules without hard connection, so the entire code would be like a platform providing a way for users with their own solvers to plugin and generate results, which is the main value of this project. In this way, our functional test becomes more important as it suggests that all modules in our code work well.

4.2 Takeaway and Objectives Satisfaction

From our perspectives, we learn how to develop a software-liked code with high-level development and testing from scratch, combined with the matrix storage tricks and parallelization implementation for speeding up. This project covers most of the main topics in this course so it is a great practice for us who are beginners in scientific programming.

As mentioned in Section 1.2, the LTFP project is targeted to develop a numerical solver to estimate the thermal field resulting from multi-layer laser scan patterns during the LAMM process. The solver can handle dynamic meshes, temperature-dependent material properties, and complex boundary conditions, as demonstrated in Sections 3.1, 3.2, and 3.3. This is supported by the fact that each module, or each combination of modules, in the solver has been tested with simple but powerful cases for robustness and effectiveness. Despite this, we still learn from Section 3.4 that there is room for improvements of the solver in parallelization performance, which may not meet our expectations and needs more careful

design in parallelizing the loops.

4.3 Future Work

In the future, the outputs can be used to validate against the experimental data, thus providing a more robust way to predict the thermal field mainly caused by laser scans. As mentioned in Section 1.2, we can also think of integrating the LTFP code into the grain prediction model, and handle the case with uneven domain increment as shown in Figure 1(c), which would be more complicated and needs more careful processes for domain increment and solver development. As mentioned in Section 3.4, the parallelization performance needs to be improved, and maybe different ways of data storage, such as the Z-order (Morton 1966) and Hilbert-order curves (Hilbert and Hilbert 1935).

References

R. Eymard, T. Gallouët, and R. Herbin. Finite Volume Methods. *Handbook of Numerical Analysis*, 7:713–1018, 2000. doi: 10.1016/S1570-8659(00)07005-8.

S. Gottlieb and D. I. Ketcheson. Time Discretization Techniques. In *Handbook of Numerical Analysis*, volume 17, pages 549–583. Elsevier, 2016. doi: 10.1016/bs.hna.2016.08.001.

D. Gu. *Laser Additive Manufacturing of High-Performance Materials*. Springer, 2015. ISBN 978-3-662-46088-7. doi: 10.1007/978-3-662-46089-4.

D. Gu, W. Meiners, K. Wissenbach, and R. Poprawe. Laser Additive Manufacturing of Metallic Components: Materials, Processes and Mechanisms. *International Materials Reviews*, 57(3):133–164, 2012. doi: 10.1179/1743280411Y.0000000014.

D. Hilbert and D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes: Nebst Einer Lebensgeschichte*, pages 1–2, 1935. doi: 10.1007/978-3-662-38452-7_1.

E. Isaacson and H. B. Keller. *Analysis of Numerical Methods*. Courier Corporation, 2012.

B. P. Leonard. Simple High-accuracy Resolution Program for Convective Modelling of Discontinuities. *International Journal for Numerical Methods in Fluids*, 8(10):1291–1318, 1988. doi: 10.1002/fld.1650081013.

G. M. Morton. A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing. Technical report, International Business Machines Company New York, 1966.

K. Ren, Y. Chew, Y. Zhang, J. Fuh, and G. Bi. Thermal Field Prediction for Laser Scanning Paths in Laser Aided Additive Manufacturing by Physics-based Machine Learning. *Computer Methods in Applied Mechanics and Engineering*, 362:112734, 2020. doi: https://doi.org/10.1016/j.cma.2019.112734.

P. K. Sweby. High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws. *SIAM Journal on Numerical Analysis*, 21(5):995–1011, 1984. doi: 10.1137/0721062.

J. F. Thompson. A Survey of Dynamically-adaptive Grids in the Numerical Solution of Partial Differential Equations. *Applied Numerical Mathematics*, 1(1):3–27, 1985. doi: 10.1016/0168-9274(85)90026-1.