Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Implementation of the Space Time Finite Element Method within the *AdhoC*++ Framework

Martin Frank

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Implementation of the Space Time Finite Element Method within the *AdhoC*++ Framework

Author:	Martin Frank
Examiner:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Assistant advisor:	M.Sc.(hons) Hayden Liu Weng
Submission Date:	July 12, 2021

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Frank

July 12, 2021 Martin Frank

Acknowledgments

First, I would like to thank Univ.-Prof. Dr. Hans-Joachim Bungartz for giving me the opportunity to write my thesis at the Chair of Scientific Computing and for providing me with the necessary access to computing resources.

Next, I would like to thank M.Sc.(hons) Hayden Liu Weng for supervising the thesis. With his patience and motivation, he created a wonderful working environment that encouraged me to do my best. Whenever challenging times occurred, he supported me with good advice and helpful tips.

I would like to give a very special thanks to Dr.-Ing. habil. Stefan Kollmannsberger on behalf of the group Simulation in Applied Mechanics for providing access to *AdhoC*++. I have benefited a lot from their previous work and expertise and without *AdhoC*++ the thesis would not have been the same.

I would also like to thank my family for their unconditionally support and love. Without your encouragement and care, I would have never had the same opportunities.

A special thanks to all my friends who encouraged and supported me during the thesis, but also provided the necessary distraction from the thesis itself or the pandemic.

Last but not least, I would like to thank my girlfriend Sonja, who not only lovingly supported me during the master thesis, but also during the whole study. Thank you for always being by my side.

Abstract

The aim of this work is to apply the Space Time Finite Element Method on problems of additive manufacturing, such as Selective Laser Melting, within the parallel framework *AdhoC++*. Therefore, the nonsymmetric matrix discretizations for the heat and latent heat equation are derived and solved with direct solvers and GMRES. The results from the linear and nonlinear heat equation are compared to analytic solutions, and convergence studies of the applications are presented. Throughout the thesis, the differences between the Space Time Finite Element Method and Finite Element Method are highlighted, resulting in a final benchmark of the two methods. The solutions of the latent heat equation, which results in a phase change, are compared against the solutions of a Stefan problem and a commercial Finite Element Method software package. Additional insights regarding stability, computational implementation and parallel aspects of the Space Time Finite Element Method are given.

Contents

1 Introduction and outline

1.1 Introduction

From an industrial perspective, manufacturing is an expensive and limiting part of developing new machines and products. The ability to create objects from the developers dreams can make the difference between an average and a state of the art product. However, the initial shape of the parts is not the key to success: adaption steps within the various stages of the overall component development must be done efficiently with respect to costs, but still provide the flexibility to adapt to other components. Previous manufacturing techniques as the injection molding process or CNC milling made some changes infeasible, not only in terms of costs, but also with respect to the geometric shapes [33]. As a result, efficient ways of creating and adapting new products were the focus of many research groups.

The outcomes are additive manufacturing solutions, such as Selective Laser Melting (SLM) or Laser Powder Bed Fusion (LPBF). Additive manufacturing is rapidly growing due to its benefits regarding fast prototyping, green technology and production-on-demand. The manufacturing relies on distributing several layers to obtain the geometry. Each individual layer is obtained by locally melting material powder with a laser beam to create a melting pool. The melting pool solidifies with the previous layer. By solidification and melting of the material in combination with support layers, arbitrary geometries can be manufactured. These flexible shapes with varying densities are often used for applications in aerospace [11] or health medicine [56].

As the method is used for rapid prototyping, the production defects should be as low as possible. To decrease them, simulations help to better understand the behaviour and adapt the machines to higher precision. The simulations must tackle two physical phenomena to obtain a reasonable geometry. Firstly, the layers are added through melted metal which solidifies during the process. This process requires multidimensional phase change modeling to obtain a reasonable temperature within the added layer. In particular, the moving boundary of the molten material and the complex heat flux across the phase change require nonlinear modeling to obtain valid simulation results. Secondly, the size of the products change due to thermal compression and expansion. Hence, the challenge in terms of simulating the SLM process relies not only on phase change modeling, but also on thermo-structural analysis. During compression and expansion, internal stresses may lead to deformation or failure of the prototype [10]. As a result, only coupled simulations may predict the final geometry and the respective stresses in order to decrease the production defects.

Over the past decades, the Finite Element Method (FEM) has become a well established technique for solving elliptic and parabolic problems. This includes thermo-structural coupling problems with respect to deformation. For the structural equations, the FEM is the method of choice and by now various descriptions exist. The temperature for coupling within solid materials can be obtained by modeling and solving the heat equation. To model the SLM, a liquid part needs to be approximated as well. The final temperature from the phase change can be obtained efficiently by solving the latent heat equation. The latent heat equation is capable of providing a solution for the liquid and solid phase. Previous applications of the latent heat equation within coupled Finite Element simulations have led to promising results for the SLM [40, 42, 43, 44].

In addition, the simulations are time-dependent problems, as the distribution of the layers is done within several minutes or hours. Time stepping schemes are still an open topic of research, because every method has advantages and disadvantages. Previous simulations used various explicit, implicit or multi-timestepping schemes for simulating the time frame [15].

Another approach to obtain a solution for the time domain is to discretize the domain with Finite Elements not only in space, but also in time. This so called Space Time Finite Element Method (STFEM) showed promising results in structural analysis [6] and electrodynamics [14]. For the STFEM, the order of the error within time can be adapted by using low or high order elements within the time direction. Moreover, the resulting large, nonsymmetric matrices can now be solved very efficiently on parallel clusters. In contrast to the previous time stepping methods, the solution is obtained continuously over time. This continuous-time representation can be advantageous to gain new insights into the physical phenomena and simplify time synchronisation within coupled simulations. Previous work on the STFEM with focus on additive manufacturing has led to interesting results [58], but did not provide the necessary coupling algorithms, as they would be needed to model the coupled SLM process. As a result, a framework that is capable of modeling thermostructural analyses and providing coupling interfaces for different meshes is needed.

The *AdhoC*++ framework already focuses on additive manufacturing [36, 37] and provides some coupling algorithms. It is capable of solving the respective problems in parallel [34] and provides advanced mesh refinement strategies to increase the local solution quality [59]. Therefore, the goal of this thesis is to apply the STFEM to additive manufacturing problems within the *AdhoC*++ framework. This includes implementing the STFEM within *AdhoC*++ for the linear and nonlinear heat equation to simulate the temperature distribution and revise the important theoretical aspects of the STFEM. Additionally, the STFEM should be applied to the nonlinear latent heat equation in order to verify if it is capable of simulating the phase change within the manufacturing process of SLM.

1.2 Outline

First, the FEM for the heat equation is derived in chapter 2. This includes the problem formulation, the derivation of the weak equation and a recapitulation of the fundamentals to obtain a discrete solution. In chapter 3, the problem formulation is adapted to apply the STFEM to the heat equation. Additionally, the space time slab is introduced and its advantages are highlighted. The nonlinear problem formulation for both methods is found in chapter 4, where additionally the solution procedures are explained. Chapter 5 shortly revises the solver and preconditioner to solve the symmetric and nonsymmetric matrix system from the FEM and STFEM. The implementation concepts are explained in Chapter 6. First, the *AdhoC*++ framework is introduced and afterwards, the implementation concepts of the STFEM for physics, mesh and space time slab are presented. Chapter 7 compares the obtained results for STFEM to analytic solutions and connects theoretical aspects of the time order with the results from STFEM. Finally, the FEM is compared against the STFEM. The STFEM is applied to the nonlinear latent heat equation in chapter 8. Two examples modeling the phase change are presented. The thesis is concluded in chapter 9 with a summary, a conclusion and an outlook on open topics.

2 Finite Element Method for the heat equation

This chapter introduces the fundamental concepts of the FEM by applying them to the heat equation. First, the partial differential heat equation in the strong form is explained in section 2.1 with the according problem formulation. The weak formulation of the heat equation for the FEM is derived in section 2.2. In order to discretize the weak equation, section 2.3 states the basis functions and numerical integration, which are used to obtain the discrete solution. This includes an overview of different explicit and implicit methods for solving the initial problem. The fundamentals of FEM are closed by revising the refinement strategies.

2.1 Introduction to the heat equation

The heat equation is a partial differential equation (PDE), which models the evolving temperature field T during the time t due to diffusion in a domain Ω. It can be derived from the conservation of energy within the domain, since for every point of the domain the heat flux must be balanced according to the sources and sinks. The equation can be stated as:

$$\rho c \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = f(\mathbf{x}) \quad \text{in} \quad \Omega. \tag{2.1}$$

The material parameters ρ, c, k describe the density, heat capacity and thermal conductivity respectively. The sinks and sources are defined within $f(\mathbf{x})$.

In order to find a valid solution of the PDE, the boundary conditions must be prescribed, resulting in a boundary value problem. The boundary conditions Γ can be split into Dirichlet Γ_D, Neumann Γ_N and Robin Γ_R boundaries, which need to be well defined over the whole boundary $\partial\Omega$:

$$\partial\Omega = \Gamma_N \quad \dot{\cup} \quad \Gamma_D \quad \dot{\cup} \quad \Gamma_R. \tag{2.2}$$

Dirichlet boundary conditions from equation 2.3 prescribe a specific Temperature T_0 at the boundary, whereas the Neumann boundary Γ_N in equation 2.4 specifies the gradient of T into a direction at the boundary. The Neumann boundaries can be used to establish physical boundary conditions resulting from convection or radiation.

$$T = T_0 \quad \text{in} \quad \Gamma_D \tag{2.3}$$
$$-k_n \frac{\partial T}{\partial n}\bigg|_{\Gamma_N} = q_0 \quad \text{in} \quad \Gamma_N \tag{2.4}$$

To describe a boundary with conduction and convection, Robin or mixed boundary conditions with an appropriate convection coefficient h and temperature of the boundary material T_{BCM}, may be applied:

$$-k_n \frac{\partial T}{\partial n}\bigg|_{\Gamma_N} + h \cdot (T - T_{BCM}) = 0 \quad \text{in} \quad \Gamma_R. \tag{2.5}$$

Since mainly Dirichlet boundary conditions are used in this thesis, additional information can be found in [32, 52] and the Robin boundary is not considered further, i.e. $\Gamma_R = 0$.

The material parameters additionally may depend on space, time and temperature. The time interval is specified within τ. Thus, equation 2.1 with the boundary conditions 2.2-2.4 is adapted to obtain the final initial boundary value problem:

$$\rho(\mathbf{x}, T)c(\mathbf{x}, T)\frac{\partial T}{\partial t} - \nabla \cdot (k(\mathbf{x}, T)\nabla T) = f(\mathbf{x}, t) \qquad \text{in} \quad \Omega \times \tau \tag{2.6}$$
$$T(\mathbf{x}, 0) = T_{init}(\mathbf{x}, t) \qquad \text{in} \quad \Gamma_D \times \tau \tag{2.7}$$
$$-k_n \frac{\partial T}{\partial n}\bigg|_{\Gamma_N} = q_{init}(\mathbf{x}, t) \qquad \text{in} \quad \Gamma_N \times \tau. \tag{2.8}$$

The solution $T(\mathbf{x}, t)$ of the equation is a scalar field varying within space and time. As the material parameters also depend on the temperature field T, complex material descriptions may change the PDE 2.6 into a nonlinear PDE.

2.2 The weak solution of the heat equation

Finding a solution which fullfills equation 2.6 for every point in the solution domain is not always feasible due to the arising arbitrary complex domains or solutions [8]. Therefore, the FEM provides a remedy, since it satisfies the equation only in a weak or integral sense. The weak form can be derived in two steps. First, multiply the equation with a test function ϕ and integrate over the domain:

$$\int_\Omega \rho(\mathbf{x}, T)c(\mathbf{x}, T)\frac{\partial T}{\partial t}\phi \, d\Omega - \int_\Omega \nabla \cdot (k(\mathbf{x}, T)\nabla T)\phi \, d\Omega = \int_\Omega f(\mathbf{x}, t)\phi \, d\Omega. \tag{2.9}$$

The divergence theorem is applied to the second term in order to eliminate the second order derivatives:

$$-\int_\Omega \nabla \cdot (k(\mathbf{x}, T)\nabla T)\phi \, d\Omega = \int_\Omega \nabla\phi(k(\mathbf{x}, T)\nabla T) \, d\Omega - \int_{\partial\Omega} k(\mathbf{x}, T)\nabla T\phi \, d\partial\Omega. \tag{2.10}$$

This results in the equation weak form:

$$\begin{aligned}\int_\Omega \rho(\mathbf{x},T)c(\mathbf{x},T)\frac{\partial T}{\partial t}\phi \, d\Omega + \int_\Omega \nabla\phi(k(\mathbf{x},T)\nabla T)\, d\Omega = \int_\Omega f(\mathbf{x},t)\phi\, d\Omega + \\ \int_{\partial\Omega} k(\mathbf{x},T)\nabla T\phi \, d\partial\Omega.\end{aligned} \tag{2.11}$$

Typically, the solution space $\boldsymbol{X}$ and the test function space $\boldsymbol{\Phi}$ are both chosen from the Hilbert space $H^1(\Omega)$:

$$\boldsymbol{X} = \{T|T \in H^1(\Omega), T = T_0 \; \forall \, \mathbf{x} \in \Gamma_D\} \tag{2.12}$$

$$\boldsymbol{\Phi} = \{\phi|\phi \in H^1(\Omega), \phi = 0 \; \forall \, \mathbf{x} \in \Gamma_D\}. \tag{2.13}$$

This is the so called Bubnov-Galerkin approach, since the same space is used for the test and solution functions. The surface integral reduces to:

$$\int_{\partial\Omega} k(\mathbf{x},T)\nabla T\phi \, d\partial\Omega = \cancelto{0}{\int_{\partial\Gamma_D} k(\mathbf{x},T)\nabla T\phi \, d\Gamma_D} + \int_{\partial\Gamma_N} k(\mathbf{x},T)\nabla T\phi \, d\Gamma_N, \tag{2.14}$$

as the test space is 0 per definition on the Dirichlet boundary.

2.3 Fundamentals of FEM

This chapter revises the basics of the standard FEM approximation. It includes a short introduction regarding basis functions and integration techniques in order to derive the semi-discrete equation. An outlook is given to discretize the solution within the time domain. This includes a recap of the explicit or implicit time stepping methods. Finally, different refinement strategies are presented.

2.3.1 Basis functions

The basis functions need to satisfy certain properties as consistency and partition of unity, which are defined as:

$$\sum_{i=0}^{n} N_i(x) = 1 \quad \text{and} \quad \bar{g}(x) = \sum_{i=0}^{n} N_i(x) \cdot g_i, \tag{2.15}$$

in order to provide a valid basis for the elements [2]. The basis functions can either be defined locally on a reference element and mapped to the global problem or globally be defined with respect to the whole domain. This depends on the implementation of the mesh and its refinement strategy [62]. The standard hat functions are usually defined with Lagrange polynomials on the reference interval. The coefficients g_i for the polynomial can

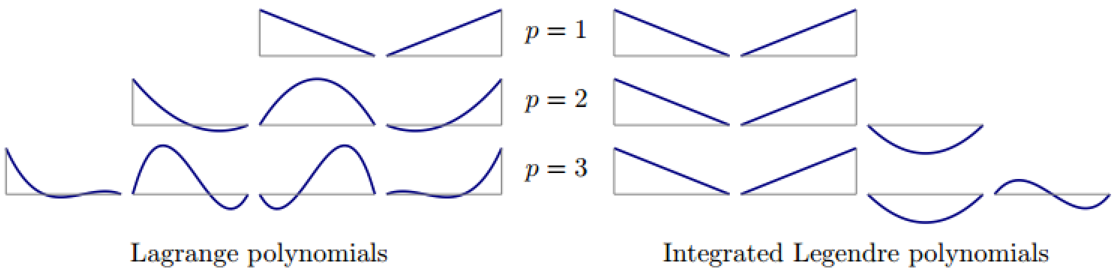


Figure 2.1: Comparison of the basis functions, adapted version from [59]

easily be computed by using the fact that the Lagrange polynomial at the point x_k equals to the function:

$$l_i(x_k) = \delta_{ik} = \begin{cases} 1, & \text{if} \quad i = k \\ 0, & \text{if} \quad i \neq k \end{cases}, \tag{2.16}$$

where $l_i(x)$ is defined as:

$$l_i(x) = \prod_{j=0,j\neq i}^{n} \frac{x - x_j}{x_i - xj}. \tag{2.17}$$

The final polynomial approximation $\bar{g}(x)$ of the function $g(x)$ can be stated as:

$$\bar{g}(x) = \sum_{i=0}^{n} l_i(x) \cdot g(x_i). \tag{2.18}$$

A drawback of elements with Lagrange basis is that during p-refinement methods the basis functions and the coefficients of the functions inside the refinement area need to be recomputed. A recomputation of the basis functions is needed as the points x_k change with higher order. The integrated Legendre polynomials provide a solution, since the polynomial of higher order is orthogonal to all polynomials of lower order. The integrated Legendre polynomials can be defined as:

$$P_n = \frac{1}{2^n \cdot n!} \cdot \frac{d^n}{dx^n}((x^2 - 1)^n) \tag{2.19}$$

$$\int_a^b w(x) P_n(x) P_m(x) dx = 0. \tag{2.20}$$

Hence, during p-refinement, the basis functions are not recomputed and an additional basis can simply be inserted. Figure 2.1 visualizes the differences between the Lagrange and Legendre basis functions. Only the coefficients of the function g_i need to be approximated with the corresponding L^2-Projection. For the L^2-Projection, the residual between the error and the polynomial or function space ϕ should be minimized as stated in equation:

$$\int_\Omega (g - P_{L^2} g)\phi d\Omega \quad \forall \phi \in \Phi^h, \tag{2.21}$$

where P_L^2 is the interpolant and g is the function to project. The resulting system:

$$M\hat{g} = G \tag{2.22}$$

must be solved for the coefficients of the degrees of freedoms (DOFs) $\hat{g}$, where M and G are specified as:

$$M = \int_\Omega N^T N d\Omega \quad \text{and} \quad G = \int_\Omega N^T g d\Omega. \tag{2.23}$$

In addition, the condition number of the global matrix system highly depends on the choice of basis, therefore a suited basis, such as the integrated Legendre, may lead to less computational effort.

2.3.2 Numerical integration

The Gauss-Quadrature is preferred for numerical integration, since it is able to approximate a polynomial of degree $p = 2n + 1$ exactly. The number of integration points can be stated as n, where the polynomial degree should be previously defined within the basis functions. The approximation is done with the precalculated integration weights w_i, where the functions need to be evaluated at the integration points x_i:

$$\int_a^b g(x)dx \approx \sum_{i=0}^{n} w_i \cdot g(x_i). \tag{2.24}$$

For computational efficiency, the integration points x_i are precomputed on the unit interval $\xi \in [-1, 1]$ and mapped to the global domain $[a, b]$ or point x_{glob} with the mapping Q and the Jacobian matrix J.

$$\int_a^b g(x)dx = \int_{-1}^{1} g(x_{glob})|J|d\xi \approx \sum_{i=0}^{n} w_i \cdot g(x_{glob})|J| \tag{2.25}$$

$$x_{glob} = Q(\xi_i) \quad \text{and} \quad J = \frac{\partial x}{\partial \xi} = \begin{bmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial y}{\partial \xi_1} & \frac{\partial z}{\partial \xi_1} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \tag{2.26}$$

Higher dimensional integrals can be computed by using the Cartesian product of the stated one-dimensional integral.

2.3.3 Discrete solution

In the next step, the discrete basis is selected to approximate our solution field T. This is achieved by defining local elements in the parameter space and transforming them to the global space. In addition, the respective Temperatures T_i are summed over all degrees of freedom(DOFs):

$$T \approx \hat{T} = \sum_{i=1}^{n_{dofs}} N_i^T \cdot T_i \tag{2.27}$$

$$\hat{\phi} = \sum_{j=1}^{n_{dofs}} N_j^T \cdot \phi_j. \tag{2.28}$$

Combining the global, discrete solution from equation 2.27, the basis functions from equation 2.28, the weak form of the heat equation from equation 2.11 and the boundary conditions from equations 2.14, 2.4 leads to the discrete problem:

$$\begin{aligned} \sum_{j=1}^{n_{dofs}} \phi_j \sum_{i=1}^{n_{dofs}} \frac{T_i}{\partial t} \int_\Omega \rho(\mathbf{x}, \hat{T}) c(\mathbf{x}, \hat{T}) N_i^T N_j^T \, d\Omega \, + \\ \sum_{j=1}^{n_{dofs}} \phi_j \sum_{i=1}^{n_{dofs}} T_i \int_\Omega k(\mathbf{x}, \hat{T}) \nabla N_j^T \nabla N_i^T \, d\Omega \, = \\ \sum_{j=1}^{n_{dofs}} \phi_j \int_\Omega f(\mathbf{x}, t) N_j^T \, d\Omega \, + \sum_{j=1}^{n_{dofs}} \phi_j \int_{\partial\Gamma_N} q(\mathbf{x}, t) N_j^T \, d\Gamma_N \end{aligned} \tag{2.29}$$

for which the ϕ_j values cancel out. As a result, the semi-discrete system consisting of the mass matrix M, stiffness matrix K and the force vector F is obtained:

$$M(T)\frac{T}{\partial t} + K(T)T = F(t) \tag{2.30}$$

$$M(T) = \int_\Omega N^T c(\mathbf{x}, NT) \rho(\mathbf{x}, NT) N \, d\Omega \tag{2.31}$$

$$K(T) = \int_\Omega B^T k(\mathbf{x}, NT) B \, d\Omega \tag{2.32}$$

$$F(t) = \int_\Omega N^T f(\mathbf{x}, t) \, d\Omega. \tag{2.33}$$

For the standard FEM, the B Operator is defined as $B = \nabla N$, which equals to the derivative with respect to the domain, hence $B = \left[\frac{\partial N(\mathbf{x})}{\partial x}, \frac{\partial N(\mathbf{x})}{\partial y}\right]$.

2.3.4 Initial value problem

From the previous formulations within equations 2.6-2.8 it is evident that the problem is time-dependent, but in the current matrix equation 2.30 the time-dependent term $\frac{T}{\partial t}$ is not yet discretized. For this subchapter, only linear problems are considered, hence the mass matrix, stiffness matrix and source term do not depend on the temperature T. In order to obtain a solution for every time point, a time stepping scheme needs to be selected. The classic solution strategies for time discretization can be split up into two classes: explicit and implicit methods. For the explicit methods, there exist simple methods as Explicit Euler 2.35 or Runge Kutta [20] schemes of different order. As an example, the equation 2.30 is discretized by the Explicit Euler:

$$M \cdot \frac{T^{n+1} - T^n}{\Delta t} + K \cdot T^n = F(t^n) \tag{2.34}$$

$$T^{n+1} = T^n + \Delta t \cdot M^{-1} \cdot \Big(F(t^n) - K \cdot T^n \Big). \tag{2.35}$$

The mass matrix can be adapted with mass lumping [22] to prevent from inverting it. For the implicit methods, Implicit Euler or Crank-Nicholson are famous solution strategies. Thus, the Implicit Euler is applied to equation 2.30, which leads to the following derivation:

$$\frac{T}{\partial t} = M^{-1} \cdot (F(t) - K \cdot T) \tag{2.36}$$

$$T^{n+1} - T^n = M^{-1} \cdot (\Delta t \cdot F(t^{n+1}) - \Delta t \cdot K \cdot T^{n+1}) \tag{2.37}$$

$$(M + \Delta t \cdot K) \cdot T^{n+1} = \Delta t \cdot F(t^{n+1}) + M \cdot T^n. \tag{2.38}$$

The main difference between explicit an implicit schemes can be seen in equations 2.35 and 2.38, respectively. The explicit form can be reformulated for implementation into an update scheme. This allows to update the same values, which can be done very efficiently in terms of computational effort. A major drawback of explicit methods is the time step size Δt, which must be chosen small enough or automatically adapted in order to ensure stability.

In contrast to the explicit schemes, the implicit schemes need to solve a matrix system in order to get the solution at the following time step T^{n+1}. In principle, the implicit methods are unconditionally stable, but for the equation , the parameters as Δt must be chosen right to obtain a reasonable solution [29].

Both time schemes need an initial condition in order to be well defined. For the Legendre basis, the initial conditions are projected to the DOFs by an L^2-Projection. Depending on the problem formulation, the initial conditions may also be obtained by an initial solve of the system.

Since all ingredients are present, the system can be solved or updated. For solving the system the conjugate gradient method is used, since the arising matrix system is symmetric and positive definite. More on solvers can be found in chapter 5.

2.3.5 Refinement strategies

Refinements of the mesh can be done in various ways, and every strategy has advantages and disadvantages. This chapter gives a brief overview of the basic refinement strategies such as *h*, *p* and *hp*-refinement.

h-refinement

Typically, *h*-refinement is the simplest way to obtain more accurate solutions, as the number of evenly distributed elements within the same domain is increased. Unfortunately, it may not be feasible to refine large global domains evenly, as the computational effort rises. Especially for arbitrary complex geometries, the solution quality may be advanced by refining the complex geometries locally.

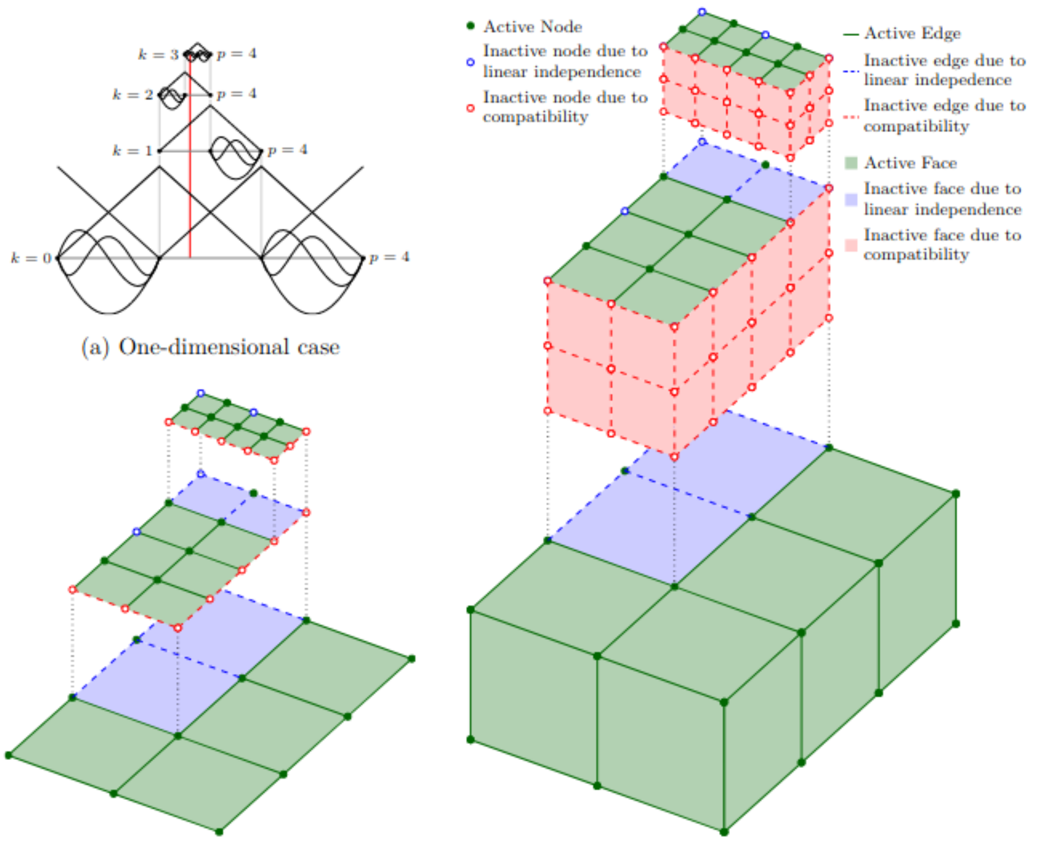
p-refinement

The *p*-refinement increases the number of nodes in an element. Hence the global number of elements is not modified. Typically, the added polynomial modes approximate the solution better. In terms of computational effort, the shape functions can be implemented efficiently by using hierarchical basis functions.

hp-refinement

In general, there may exist parts of the solution domain where lower order refinements are capable of approximating the solution perfectly, and subdomains where higher order *hp*-refinements are needed. Therefore, efficient local refinement strategies as the multi-level *hp*-refinement are applied to expose parallelism through it's algorithmic behaviour and data structures.

The idea of the multi-level *hp*-approach in case of a partitioned base-overlay refinement is to define local overlay layers, which approximate the global solution $T = T_b + T_o$ by summation of a base T_b and a fine solution T_o. Applying this superposition recursively to components of the mesh may result in an simple, but efficient refinement strategy. To eliminate inactive nodes, basis functions like the integrated Legendre polynomial are applicable, as individual shape functions can be identified easily. By eliminating the inactive nodes on a finer mesh, the mesh compatibility between adjacent elements can be ensured. In Figure 2.2, a mesh is refined according to the *hp*-refinement and the active and inactive nodes per layer can be identified accordingly. The continuity between elements are ensured by Dirichlet boundary conditions. To obtain reasonable results, the integration must be projected on the leaf element. More information can be found within [34, 60].


$k = 3$
$p = 4$
$k = 2$
$p = 4$
$k = 1$
$p = 4$
$k = 0$
$p = 4$
● Active Node
Inactive node due to linear independence
Inactive node due to compatibility
Active Edge
Inactive edge due to linear indepedence
Inactive edge due to compatibility
Active Face
Inactive face due to linear independence
Inactive face due to compatibility

(a) One-dimensional case

(b) Two-dimensional case

(c) Three-dimensional case

Figure 2.2: Visualization of the *hp*-mulitlevel method from [60]

3 Space Time Finite Element Method for the heat equation

From the first pioneering work of [30] within solid mechanics the STFEM has been continuously developed as for the Navier-Stokes equation with Discontinuous Galerkin(DG) [21] or parabolic evolution problems [49]. Not only applications have been improved, but also the overall solving process [41].

This chapter solves the heat equation problem stated within equations 2.6-2.8 by using STFEM. First, the derivation from the previously defined standard FEM is adapted and the key differences are explained in section 3.1. In section 3.2 the space time slab is introduced, which subdivides the interval into smaller time steps.

3.1 Derivation of the Space Time Finite Element Method

The STFEM relies on the discretization of the time space τ. Therefore, the time t will simply be treated as an existing space dimension. To get started, the new space Π is introduced:

$$\Pi = \Omega \times \tau, \tag{3.1}$$

consisting of the previous spatial space Ω and the time space τ, where τ is defined as:

$$\tau \in [t_0, t_1]. \tag{3.2}$$

The boundary conditions $\delta\Pi$ can be split up into three parts:

$$\delta\Pi = \Gamma \times \tau \cup \Omega \times [t_0] \cup \Omega \times [t_1]. \tag{3.3}$$

The term at the time t_0 can be interpreted as the inital condition compared to the standard FEM. The other two terms consist of the classic time-dependent boundary parts. It is worthwhile to mention, that by using STFEM, the dimension of the problem rises accordingly to space and time, since the discrete domain is in equation 3.1 extended by the time.

Before deriving the equation, the solution space needs to be stated. Since the solution should statisfy the equation:

$$\frac{\partial T}{\partial t} \in L^2(\tau) \tag{3.4}$$

to fullfill the energy decreasing property, equation 3.4 implies that the Hilbert space $H^1(\tau)$ provides a suitable space. Therefore, the Bubnov-Galerkin approach is replaced by the Petrov-Galerkin approach. Petrov-Galerkin makes use of different function spaces for the solution space T and test space ϕ such that:

$$\boldsymbol{X} = \{T|T \in H^1(\Omega) \times H^1(\tau)\} \tag{3.5}$$

$$\boldsymbol{\Phi} = \{\phi|\phi \in H^1(\Omega) \times L^2(\tau)\}. \tag{3.6}$$

Further information regarding energy decreasing properties and test spaces can be found in [48].

The weak solution derived from equation 2.11 is still valid, but the previously defined space Ω must be adapted to the new space Π. This can be done by adapting the gradient of the problem, since the original problem should be kept. The previous defined derivative ∇ is now separated into $\nabla_{\mathbf{x}}$ and ∇_t, leading to the new problem formulation:

$$\begin{aligned}\int_\Pi \rho(\mathbf{x},T)c(\mathbf{x},T)\nabla_t T\phi \, d\Pi + \int_\Pi \nabla_{\mathbf{x}}\phi(k(\mathbf{x},T)\nabla_{\mathbf{x}}T) \, d\Pi = \\ \int_\Pi f(\mathbf{x},t)\phi \, d\Pi + \int_{\partial\bar{\Gamma}_N} k(\mathbf{x},T)\nabla_{\mathbf{x}}T\phi \, d\bar{\Gamma}_N.\end{aligned} \tag{3.7}$$

As already indicated, the new basis must be time-dependent and can be constructed with a cartesian tensor product of the standard one-dimensional elements:

$$N(\mathbf{x},t) = N(\mathbf{x}) \cdot N(t). \tag{3.8}$$

The properties for the elements are the same as defined in subsection 2.3.1. Combining the new basis from equation 3.8 with the adapted equation 3.7, where the global solution field is defined as in equation 2.27, leads to the discrete problem:

$$\begin{aligned}\sum_{j=1}^{n_{dofs}} \phi_j \sum_{i=1}^{n_{dofs}} T_i \int_\Pi \rho(\mathbf{x},\hat{T})c(\mathbf{x},\hat{T})N_i^T \cdot \nabla_t N_j^T \, d\Pi + \\ \sum_{j=1}^{n_{dofs}} \phi_j \sum_{i=1}^{n_{dofs}} T_i \int_\Pi k(\mathbf{x},\hat{T})\nabla_{\mathbf{x}}N_j^T \nabla_{\mathbf{x}}N_i^T \, d\Pi \; = \\ \sum_{j=1}^{n_{dofs}} \phi_j \int_\Pi f(\mathbf{x},t)N_j^T \, d\Pi \; + \sum_{j=1}^{n_{dofs}} \phi_j \int_{\partial\Gamma_N} q(\mathbf{x},t)N_j^T \, d\Gamma_N,\end{aligned} \tag{3.9}$$

where the B-Operator is split with respect to space and time for $2D$ space :

$$B_x = \left[\frac{N(\mathbf{x},t)}{\partial x}, \frac{N(\mathbf{x},t)}{\partial y}\right] \tag{3.10}$$

$$B_t = \frac{N(\mathbf{x},t)}{\partial t}. \tag{3.11}$$

The final discrete matrix equation is obtained with:

$$\Big(M(T) + K(T)\Big) \cdot T = F(t) \tag{3.12}$$

$$M(T) = \int_{\Pi} N^T c(\mathbf{x}, NT)\rho(\mathbf{x}, NT) B_t \, d\Pi \tag{3.13}$$

$$K(T) = \int_{\Pi} B_x^T k(\mathbf{x}, NT) B_x \, d\Pi \tag{3.14}$$

$$F(t) = \int_{\Pi} N^T f(\mathbf{x}, t) \, d\Pi. \tag{3.15}$$

Due to the space time approach, the mass matrix and the stiffness matrix are now combined to a matrix, which enables to solve one equation for the whole problem. Unfortunately, the resulting matrix is not symmetric positive definite anymore, due to the time discretization and the respective physical behaviour. Thus, using the conjugate gradient method would not lead to a valid solution. The GMRES is the solver of choice and is revised in the section 5.3.

3.2 Space time slab

In terms of computational effort, it may be advantageous to decompose large problems into smaller subproblems. This can be done by using the DG approach for detaching the global solution space Π into multiple smaller time parts $\Pi_n = \Omega \times [t_{n-1}, t_n]$. These smaller space time parts Π_n are so called space time slabs, where each time slab can then be solved sequentially.

The coupling between the elements arise from the interelement compatibility condition, that the solution of an element boundary must be the same with the neighbouring element. The Figure 3.2 shows the continuous approach along the time domain for two elements, where this constrain would enforce equation:

$$T_2^1 = T_1^2. \tag{3.16}$$

By removing the compatibility constraint of the elements, the solution is decoupled and the solution space Φ is not continuous anymore, as there exist two values for $\phi^h(t_n)$, $\phi^h(t_n^-)$ from element E_1 and $\phi^h(t_n^+)$ form element E_2. In Figure 3.1 this is indicated with the two points at the time t_n. As the elements or cells are disconnected from each other, a jump operator must be established in order to connect the elements via the flux F again. Figure 3.1 displays the DG approach with the flux. For the slab, the jump operator is specified as:

$$[[\phi^h(t_n)]] = \phi^h(t_n^+) - \phi^h(t_n^-), \tag{3.17}$$

which couples the elements by setting $F = 0$. The result can be seen in Figure 3.2, where an implicit connection is again established between the DOFs T_2^1 and T_1^2. This results again in equation 3.16. Finally, the blue domain representing the first slab Π_n can be solved independently and the solution at the boundary t_n is then used as initial condition to solve the following time slab Π_{n+1}. This approach is only valid for the time direction, as for other dimensions the flux may not cancel out. The derivation can be found in [31]. Additional information regarding general DG can be found in [16, 19] and advanced DG discretizations in [57, 61].

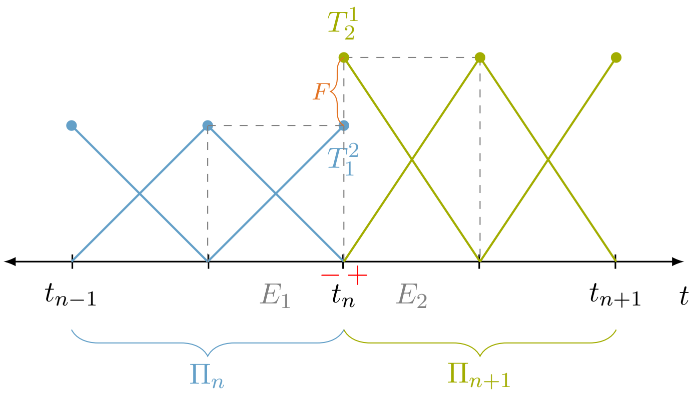


Figure 3.1: General DG in time direction

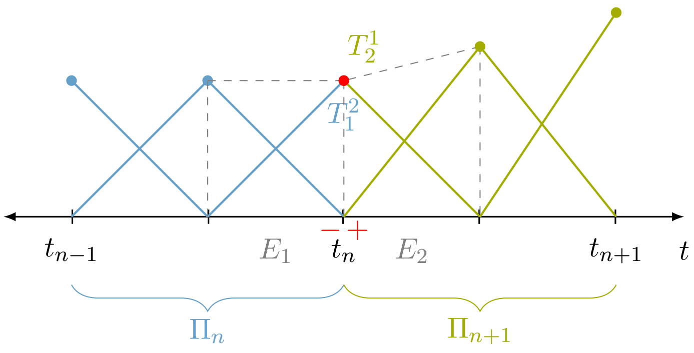


Figure 3.2: Slab visualization in time direction

Another interpretation can be given with domain decomposition. The initial domain is cut into two parts Π_n and Π_{n+1} across a time part t_n, where a solution is obtained within the first domain Π_n. The result of the first domain across the boundary is used as initial boundary condition for the second time domain. In Figure 3.2, the initial boundary conditions of the second domain corresponds to the red dot. This approach works as the current time slab Π_n is independent of the following time slab Π_{n+1}.

4 Nonlinear Finite Element Method

As research continues, more challenging problems and complex materials need to be modeled mathematically precisely to obtain a reasonable solution. Since these problems can no longer be modeled using linear methods, the simulations must be adapted for nonlinear physical problems. The nonlinearities can arise from material descriptions, geometry problems or contact problems [35]. But before the nonlinear FEM is given, a short revision of the Newton solver is presented in section 4.1. As for this chapter, mainly nonlinear materials are considered, the solution strategy for the nonlinear FEM with nonlinear materials is specified in section 4.2. Finally, the chapter is closed with the derivation of the nonlinear STFEM in section 4.3. In addition, for every method the algorithmic changes are presented.

4.1 Newton solver

The Newton solver is one of the most well established methods when it comes to finding the roots of nonlinear equations, especially on higher dimensional problems [24]. The idea is based on Taylor expansion and assumes that the function $f(x)$ can be approximated by its derivative, as the difference between the starting point x_n and the solution point x approaches 0:

$$0 = \frac{f(x)}{\partial x}(x - x_n) + f(x_n). \tag{4.1}$$

For a one-dimensional problem this can easily be written in the update scheme:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{4.2}$$

To obtain a solution, $f(x)$ needs to be in an implicit or residual form and the derivative of $f(x)$ must be well defined within the solution interval:

$$\frac{f(x)}{\partial x} \neq 0 \quad \forall x \in [x_s, x_e]. \tag{4.3}$$

In every Newton iteration, a linear system of equations must be solved. In addition, the starting point must be chosen right, otherwise the solution might not converge.

4.2 Nonlinear Finite Element Method

The derivation of the weak equation does not change due to the nonlinearties, but as it can be seen from equation 2.30, the mass matrix $M(T)$ and stiffness matrix $K(T)$ are generally temperature dependent due to nonlinear heat capacity, density or conductivity. In this case, the system cannot be simply solved, as it is not possible to derive the explicit form for the temperature. The implicit residual form is satisfied for an equilibrium point, consisting of the external, internal and inertia forces:

$$R(T) = \underbrace{F}_{\text{external}} - \underbrace{M(T)\frac{T}{\partial t}}_{\text{inertia force}} \underbrace{-K(T)T}_{\text{internal}}. \tag{4.4}$$

The inertia force depends on the time derivative and needs to be discretized with an appropriate time stepping scheme from subsection 2.3.4. As the system must be solved with the Newton solver due to material nonlinearities, this offers the possibility to solve the Implicit Euler time step in combination with the material solve. As a result the time stepping and the material solve is done within in the same solve. Therefore, a time and temperature dependent residual can be stated as:

$$R(T, t_n, t_{n+1}) = F(t_{n+1}) - K(T_{n+1}) \cdot T(t_{n+1}) - M(T_{n+1}) \cdot \frac{T(t_{n+1}) - T(t_n)}{\Delta t}. \tag{4.5}$$

The scheme uses the Newton from section 4.1 implicitly and is summarized within Algorithm 1.

More sophisticated solution algorithms [9] may include the explicit use of the residual's derivative in combination with line-search methods and load stepping. Previously, the residual's derivative was only specified as $K(T)$. This can simply be enhanced by deriving the residual of equation 4.4:

$$\frac{R(T)}{\partial T} = \cancelto{0}{\frac{F}{\partial T}} - \frac{M(T)}{\partial T}\cdot\frac{T}{\partial t} - \cancelto{0}{M(T)\frac{T}{\partial t \cdot \partial T}} - \frac{K(T)}{\partial T} \cdot T - K(T) \cdot \cancelto{1}{\frac{T}{\partial T}} \tag{4.6}$$

$$R_T = \frac{R(T)}{\partial T} = - M_T(T) \cdot \frac{T}{\partial t} - K_T(T) \cdot T - K(T), \tag{4.7}$$

where the derivative matrices are specified as:

$$K_T(T) = \int_\Omega B^T \frac{k(\mathbf{x}, NT)}{\partial T} B \, d\Omega = \int_\Omega B^T k_T B \, d\Omega \tag{4.8}$$

$$M_T(T) = \int_\Omega N^T \rho(\mathbf{x}, NT) \frac{c(\mathbf{x}, NT)}{\partial T} N \, d\Omega + \int_\Omega N^T \frac{\rho(\mathbf{x}, NT)}{\partial T} c(\mathbf{x}, NT) N \, d\Omega. \tag{4.9}$$

The derivations according to the temperature T of the material parameters k_T, c_T and ρ_T can be derived from the problem formulation as well:

$$k_T = \frac{k(\mathbf{x}, T)}{\partial T} \tag{4.10}$$

$$c_T = \frac{c(\mathbf{x}, T)}{\partial T} \tag{4.11}$$

$$\rho_T = \frac{\rho(\mathbf{x}, T)}{\partial T}. \tag{4.12}$$

Algorithm 1: Nonlinear FEM with Newton and Implicit Euler

```
Result: $T(t_{final})$
initialization $\Delta t$, $T_0$, $NewtonIterations$, $t_0$, $norm$
apply refinement strategy
while $t_n < t_{final}$ do
    update time step $t_n = t_n + \Delta t$
    while $i < NewtonIterations$ and $norm > tolerance$ do
        update derivative of residual $R_T(T_i^{t_n}, t_n) = K(T_i^{t_n})$
        update residual $R(T_i^{t_n}, t_n)$
        solve $R_T(T_i^{t_n}, t_n) \cdot \Delta T_i = R(T_i^{t_n}, t_n)$
        update solution $T_{i+1}^{t_n} = T_i^{t_n} + \Delta T_i$
        update norm $norm = R(T_i^{t_n}, t_n)^T \cdot \Delta T_i$
        update iteration $i + 1$
    end
    if $norm < tolerance$ then
        update solution $T_0^{n+1} = T_i^n$
    else
        algorithm failed
    end
end
```

4.3 Nonlinear Space Time Finite Element Method

In this case, the solution strategy of a nonlinear material $k(\mathbf{x}, t, T)$, density $\rho(\mathbf{x}, t, T)$ and heat capacity $c(\mathbf{x}, t, T)$ term is presented. All material properties depend on space, time and the temperature. To solve this nonlinear equation, the previous defined Newton method is used. Hence, the basic weak equation from 3.12 is transformed into the residual equation, which consists of the external and internal energy part.

$$R(T) = \underbrace{F(t)}_{\text{external}} - \underbrace{\big(M(T) + K(T)\big)T}_{\text{internal}} \tag{4.13}$$

In the case of STFEM, the residual equation must be derived with respect to the temperature T. For the multidimensional case, the update in the solution direction ΔT can be computed with the Jacobian matrix J:

$$R(\Delta T) = R(T_n) + J(T) \cdot \Delta T = 0 \tag{4.14}$$

$$\Delta T = -\Big(J(T_n)^{-1} R(T_n)\Big) \tag{4.15}$$

$$T_{n+1} = T_n - \Big(J(T_n)^{-1} R(T_n)\Big) \tag{4.16}$$

$$J(T_n)T_{n+1} = J(T_n)T_n - R(T_n). \tag{4.17}$$

Applying this now to the residual heat equation leads to:

$$-J(T) = \frac{\partial R(T)}{\partial T} = \frac{-F(T) + \big(M(T) + K(T)\big)T}{\partial T} \tag{4.18}$$

$$\frac{F(T)}{\partial T} = \cancelto{0}{\frac{-\int_\Omega N^T f(\mathbf{x}, t)\, d\Omega}{\partial T}} \tag{4.19}$$

$$\frac{K(T)T}{\partial T} = K_T(T)T + K(T) \tag{4.20}$$

$$\frac{M(T)T}{\partial T} = M_T(T)T + M(T). \tag{4.21}$$

Since in the scope of this thesis mainly manufactured solutions are considered, the source term F is independent of the solution T and is not contributing. It is worth to mention, that this term may not be simply neglected on nonlinear geometry problems or complex source models. The derivative parts can be specified with:

$$K_T(T) = \int_\Omega B_x^T k_T(\mathbf{x}, NT) B_x \, d\Omega \tag{4.22}$$

$$M_T(T) = \int_\Omega N^T \rho(\mathbf{x}, NT) c_T(\mathbf{x}, NT) B_t \, d\Omega + \int_\Omega N^T \rho_T(\mathbf{x}, NT) c(\mathbf{x}, NT) B_t \, d\Omega. \tag{4.23}$$

Even if the derivatives of the density ρ_T, heat capacity c_T and thermal conductivity k_T are unknown, the solution can still be obtained with lower convergence rate. For every update, the equation 4.17 has to be solved. If only one of the heat capacity, density or conductivity is nonlinear, the other derivative parts in equation 4.22 and 4.23 simply cancel out as for the source term 4.19.

The algorithm to obtain the solution for the STFEM is the simple, stationary Newton solver and is stated in Algorithm 2.

Algorithm 2: Nonlinear STFEM with Newton

```
Result: $T$
initialization $NewtonIterations, norm, T_0$
apply refinement strategy
while $i < NewtonIterations$ and $norm > tolerance$ do
    update derivative of residual $R_T(T_i)$
    update residual $R(T_i)$
    solve $R_T(T_i) \cdot \Delta T_i = R(T_i)$
    update solution $T_{i+1} = T_i + \Delta T_i$
    update norm $norm = R(T_i)^T \cdot \Delta T_i$
    update iteration $i + 1$
end
```

5 Solver

In the previous chapters, all basics of FEM and STFEM and their algorithmic solving strategies were presented, but so far the solvers were not specified. This chapter revises the basics of the solvers and selected preconditioning methods to obtain a solution for the linear system. In general, three solvers are presented, beginning with a universal direct solver in section 5.1 for symmetric and non-symmetric problems. Then, in section 5.2, the conjugate gradient method for symmetric positive definite problems is presented. As last solver, the GMRES solver is explained for non-symmetric problems in section 5.3. In addition, the Jacobi and Neumann preconditioners are introduced in section 5.4.

The problem formulation of this chapter can be stated as:

$$Ax = b \qquad A \in \mathbb{R}^{n\times n} \qquad x, b \in \mathbb{R}^n, \tag{5.1}$$

where the matrix A is sparse. For sparse matrices, it is advantageous to store only all non-zero matrix entries, which can be realised with a compressed row or column matrix storage. More information on sparse linear algebra and iterative methods can be found in [7].

5.1 Direct solvers

Direct solvers are efficient for solving small matrices, such as those from system tests. The Gaussian elimination procedure, implemented as LU-Factorization is quite often used. The matrix A is decomposed into an upper U and a lower L matrix and applied to system 5.1:

$$PA = LU \quad PA = LU \tag{5.2}$$

$$PAx = Pb \quad LUx = Pb. \tag{5.3}$$

Depending on the implementation, the permutation matrix P can be specified. The solution of the matrix system can be computed efficiently with the respective forward and backward substitution of the upper U and lower L matrices:

$$y = Ux \quad \tilde{b} = Pb \tag{5.4}$$

$$Ly = \tilde{b} \quad Ux = y. \tag{5.5}$$

It is important to notice that this direct solver is capable of obtaining a solution independently of the definiteness of the matrix. Hence, symmetric or non-symmetric definite problems may be solved. This provides a good starting point for solving equations. Nevertheless, there are several drawbacks of the direct solvers. One of the major drawbacks is the "Fill-in". The "Fill-in" occurs during pivoting and fills up entries of sparse matrices, which were initially 0. As a result, the efficient storage structure of the sparse matrices is lost. The accuracy depends on the implementation, since rounding errors with respect to pivoting or "Fill-ins" may lead to inaccurate results. More information can be found within [25].

5.2 Conjugate gradient method

The conjugate gradient method (CG) is the most famous iterative method for solving symmetric positive definite matrices as they rise from the discretization of an elliptic PDE. The idea behind CG is that search directions of the problem are conjugate with respect to the residual. This can be stated as:

$$r_i^T r_j = 0. \tag{5.6}$$

This special property guarantees that the basis of the new solution increment is not within the previous solution basis. The solution space is therefore stated as a so called Krylov subspace, which is spanned as:

$$D_i = \text{span}\{r_0, Ar_0, A^2 r_0, ..., A^{i-1} r_0\}. \tag{5.7}$$

CG is theoretically able to provide a solution with n iterations, where n is the number of eigenvectors of the matrix A. The convergence speed depends on the condition number of the matrix. For well-conditioned systems the solution can be obtained in less than n iterations, whereas for ill-conditioned matrices, preconditioning is crucial to obtain the solution within a reasonable time. Simple Jacobi preconditioning or enhanced multigrid methods may therefore speed up the computation. A detailed derivation and explanation can be found in [50] and an unoptimized algorithm can be found in Algorithm 3.

Algorithm 3: Conjugate Gradient algorithm

```
input  : $A, b$
output: $x$
initialization $x_0$,$n$
$d_0 = r_0 = b - Ax_0$
while $i \leq n$ do
    $\alpha_i = r_i^T r_i$
    $x_{i+1} = x_i + \alpha_i d_i$
    $r_{i+1} = r_i - \alpha_i A d_i$
    $\beta_{i+1} = \frac{r_{i+1}^T r_{i+1}}{r_i r_i}$
    $d_{i+1} = r_{i+1} + \beta_{i+1} d_i$
end
```

5.3 GMRES

The Generalized Minimal Residual Method or short GMRES is a robust, iterative method for solving general nonsymmetric systems. The idea is to minimize the norm of the residual vector within a Krylov subspace [45]. The n-basis vector within the solution Krylov subspace is defined to be the orthonormal vectors q_n. Hence, the solution of the system can be stated as equation 5.8. The Arnoldi iteration normally provides the largest eigenvalue of a matrix. Refactoring the Arnoldi process leads to equation 5.9, which decomposes the problem into a Hessenberg matrix H. An appropriate start value reduces the equation to the final minimization formulation in equation 5.10 as Q_n is orthonormal due to its definition. The minimization problem can now be solved efficiently with Givens rotation.

$$x_n = x_0 + Q_n y_n \tag{5.8}$$

$$AQ_n = Q_{n+1}\tilde{H}_n \tag{5.9}$$

$$||r_n|| = ||Ax_n - b|| = ||r_0 - AQ_n y_n|| = ||\beta q_1 - Q_{n+1}\tilde{H} y_n|| = ||\tilde{H}_n y_n - \beta e_1|| \tag{5.10}$$

This results in the final Algorithm 4.

Algorithm 4: GMRES algorithm

```
input  : $A, b, max_{iter}$
output: $x$
initialization $Q$,$r$,$e$, $x_0$, $H$
while $i \leq max_{iter}$ do
    update $H, Q$ with Arnoldi($A, Q, k$)
    update $b, H$ with Givens rotation($H, k, i$)
    update residual $\beta$
    update error $e$
end
update solution $x = x_0 + Hb$
```

5.4 Preconditioning

Even though the solution for some iterative methods or direct solvers can be obtained in a finite sequence of steps, the system may still be ill-conditioned. These ill-conditioned matrix systems may lead to rounding errors or low convergence rates. The condition number κ of the system depends on the maximal and minimal eigenvalues λ_{max} and λ_{min}. Systems can be described as ill-conditioned if:

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}} \gg 1. \tag{5.11}$$

For the STFEM, the systems are often ill-conditioned. Thus, preconditioning is helpful to speed up the solving process. Various preconditioner such as incomplete LU-decomposition, Gauss-Seidel, domain decomposition or algebraic multigrid methods exist and some of them were already applied to STFEM [4, 54]. To start, the left preconditioned system:

$$P^{-1}(Ax - b) = 0 \tag{5.12}$$

is considered. For $P = A$, the preconditioner would correspond to the inverted matrix and solve the system. For $P = I$, the previous system has to be solved and nothing has changed. A good preconditioner has therefore two contradicting criteria to satisfy. On the one hand, it should correspond to the matrix A, whereas on the other hand it should be as cheap as possible in terms of computational effort. In the following, the Jacobi iteration and the Neumann series are presented as they are used within the thesis.

5.4.1 Jacobi preconditioner

The stationary Jacobi preconditoning uses the diagonal matrix of A:

$$P = \text{diag}(A). \tag{5.13}$$

By using the Richardson iteration, the solution x^{n+1} gets updated according to:

$$x_i^{n+1} = (I + \frac{1}{A_{ii}} A_{ij}) x_i^n - \frac{b_i}{A_{ii}} \tag{5.14}$$

for every entry of the vector x_i. In terms of computational effort and parallelism the Jacobi iteration is quite good. Unfortunately, it lacks not only in terms of convergence speed, but also only converges for diagonal dominant matrices.

5.4.2 Neumann preconditioner

Polynomial preconditioners use the polynomial $P^{-1} = s(A)$ for approximating the system. Thus, the equation 5.12 can be adapted to:

$$s(A)Ax = s(A)b. \tag{5.15}$$

The general Neumann series expansion:

$$A^{-1} = (I - N)^{-1} = \sum_{k=0}^{\infty} N^k \tag{5.16}$$

is adapted to approximate the polynomial $s(A)$, by defining the matrix N to be:

$$N = I - \omega D^{-1} A, \tag{5.17}$$

where ω denotes a scaling parameter and D the diagonal of matrix A. As a result, the preconditioner and the left side can be stated as:

$$P^{-1} = (1 + N + \ldots N^s)D^{-1} \quad \text{and} \tag{5.18}$$

$$P^{-1}A = \frac{1}{\omega}\left(I - N^{s+1}\right) \tag{5.19}$$

according to [46]. The preconditioner can be well parallelised and the computational effort can be regulated, as s and w can be chosen accordingly. For $s \to \infty$, the preconditioner would correspond to the inverse of A. Unfortunately, for higher order polynomials, numerical instabilities may occur.

6 Implementation

The number of frameworks for solving PDEs, especially with a focus on time-dependant problems, is quite large. Various libraries as DUNE [12], `deal.II` [5] or FEniCS [3] offer a wide range for simulating PDEs. Nevertheless, the object-oriented library *AdhoC*++ was chosen to be extended, as it provides interfaces for coupling algorithms, parallel implementations and refinement strategies. As *AdhoC*++ is not open source, some of the basic concepts are explained in order to give a small insight. The *AdhoC*++ library makes use of other libraries such as Boost [47] for system tests, HDF5 [38] for large data sets, MPI [23] for parallelisation and Trilinos [55] or PARDISO [1] for the solvers.

The implementation of the STFEM in terms of equations and physics is given in the section 6.1. Next, the concepts of boundary conditions and mesh implementation are given in section 6.2 and section 6.3. Section 6.4 describes the parallel implementation within *AdhoC*++ and finally, section 6.5 introduces the implementation of the space time slab.

6.1 Implementation of the STFEM

From the previous derivations in chapter 3 it can be seen that the STFEM approach needs to create a mesh not only in the space dimension, but also in the time dimension. The implemented idea is to simply extend the framework by using the last dimension of the discrete mesh as the time dimension. By using this approach, most of the key features as mesh assembly, computing of ansatz spaces, setting up the global system equation and applying boundary conditions can be reused without further changes within the framework. Only semantic changes need to be considered. The physical operations, as gradients, kinematic equations and energy equations, need to be adapted.

Unfortunately, the framework is capable of meshing and postprocessing only up to maximum of three dimensions. As adding a dimension more to a framework can be quite challenging, this thesis restricts to one and two-dimensional space problems, as all advantages and drawbacks may also be covered within two dimensions.

6.1.1 Gradient

As indicated in the previous section, the gradient or B-Operator is split into a space part B_x and a time part B_t for the implementation. The time derivative is always specified as last direction. As an example, the two-dimensional B-Operator is presented:

$$B = \begin{bmatrix} \frac{N(\mathbf{x},t)}{\partial x} \\ \frac{N(\mathbf{x},t)}{\partial y} \\ \frac{N(\mathbf{x},t)}{\partial t} \end{bmatrix} \rightarrow B_x = \begin{bmatrix} \frac{N(\mathbf{x},t)}{\partial x} \\ \frac{N(\mathbf{x},t)}{\partial y} \\ 0 \end{bmatrix}, B_t = \begin{bmatrix} 0 \\ 0 \\ \frac{N(\mathbf{x},t)}{\partial t} \end{bmatrix}. \tag{6.1}$$

6.1.2 Physics

For the implementation of the STFEM, the formulation derived from chapter 3 must be implemented accordingly. This is done with local elements and the previous defined B-Operators. For the evaluation of the material and density properties, the material tensors are adapted to the higher dimension, as the material should not contribute into the time dimension. As an example the previous material tensor for *2D* is extended for STFEM:

$$\begin{bmatrix} k(x,y) & 0 \\ 0 & k(x,y) \end{bmatrix} \rightarrow \begin{bmatrix} k(x,y,t) & 0 & 0 \\ 0 & k(x,y,t) & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{6.2}$$

The element matrices for the global stiffness part K and global mass M are set up and summed up accordingly.

6.1.3 Implemented system equations

The linear system equation implemented within *AdhoC++* is done with a system object, which can access the system matrices with respect to different matrix traits of the form:

$$M_s \cdot \ddot{x} + D_s \cdot \dot{x} + K_s \cdot x = RHS. \tag{6.3}$$

From the standard FEM point, this implementation makes sense, as the terms can be identified easily and registered on the slots of the equation, respectively. For the STFEM, in contrast to the FEM, the mass term depends on x. As a result, the mass matrix is summed with the previous stiffness matrix and registered at the position of the system stiffness matrix K_s. To avoid confusion, the STFEM mass term is referred as masskinematic term, since it represents the physical behaviour of the mass, but is implemented in the kinetic part of the system equation.

In addition, the system class may also hold the engergy equation, which is needed for the nonlinear problems:

$$Res = ExternalEnergy - InternalEnergy. \tag{6.4}$$

In equation 4.13, the external and internal energy parts were already highlighted and can therefore be registered on the external and internal slots. In addition, the derivative of the residual must be registered as required by the solving process, which is done on the stiffness or system matrix slot.

6.2 Boundary conditions

AdhoC++ offers various methods to enforce boundary conditions on the mesh. In general, Dirichlet and Neumann conditions can be enforced in a strong or weak sense. The boundary conditions may be applied on geometric components of the mesh as nodes, edges and faces. From the geometric components the boundary function is projected with an L^2-Projection to the DOFs. The strong boundary conditions are enforced by using direct or penalty constraints. For the direct constraint, the matrix entry is set explicitly to the value of the boundary function, whereas for the penalty constraint a penalty value is added within the matrix. In the scope of this thesis, the boundary conditions were mainly applied strongly with the direct constraint, as this does not worsen the condition number of the matrix system.

6.3 Implementation of the mesh

The implementation uses the multilevel hp-refinement strategy from subsection 2.3.5 to refine the mesh. Options as refining towards boundaries or varying the polynomial degree per layer are available. To define a refinement section, the geometric domain, refinement strategy and recursive layers must be specified. The geometric domain of the refinement section can be specified with geometric objects such as boxes, circles or other topological components. The implemented refinement strategy decouples the hp-refinement and enables an efficient change between h-refinement and p-refinement in every layer. The number of recursive layers defines how often the refinement strategy is applied to the geometric domain.

6.4 Parallel implementation

Even relatively small problems may need long simulation time, if the available resources provide too little computational power or are not used efficiently. Hence, over the past decades, parallelization for shared and distributed memory models have become the standard within scientific simulations. The *AdhoC*++ framework is already parallelised by [34] and provides interfaces to the libraries Trilinos [55], Paradiso [1] and HDF5 [38]. As indicated in subsection 2.3.5, the main parallelization aspect within the framework is the multilevel refinement. The mesh and refinement strategy is set up in serial. The distribution of the elements to the different processors is done with the geometric load distributor

Zoltan [13], which ensures evenly distributed domains with respect to the computational effort of the elements. The integration of the separate domains are then carried out by each processor individually. The global system is efficiently assembled and solved with the Trilinos extension, since it provides not only the vector and matrix handle for distributed vectors and matrices, but also a wide range of solvers. The AZTECO [27] solver library within Trilinos is used to compute the solution, since it provides all solvers and preconditioners mentioned in chapter 5. The post processing is done in parallel with the HDF5 library, which enables efficient distributed input and output for the files and provides support for the visualisation.

6.5 Space time slab

As the required amount of memory increases for larger systems, it can be advantageous to decouple the matrices into smaller ones. This is done with the space time slab from section 3.2. The slab was implemented by using only one mesh. The summarized procedure can be found in Algorithm 5. After initialization and refinement, the constraint points for the slab are defined. This is done by projecting the integration points of the refined elements on the surface, as for the most quadrature rules, the points do not contribute to the surface. In the following, the first or initial slab problem with the lateral boundary conditions is solved and postprocessed. The corresponding time loop is then started with an update of the time step and boundary conditions. This includes projecting the solution from the last time step at the end of the timeslab to the initial boundary conditions of the next. Finally, the system can be solved and postprocessed.

Algorithm 5: Space time slab algorithm

```
set up parameters, mesh and refinement strategy
set up boundaryvalueproblem
initialize postprocessor
define solver
apply refinement strategy
setup constraint points for slab
set up global equation
apply boundary conditions
solve initial system
apply postprocessing
while $t \leq t_{final}$ do
    update time step
    constrain system with previous solution
    solve system
    apply postprocessing
end
```

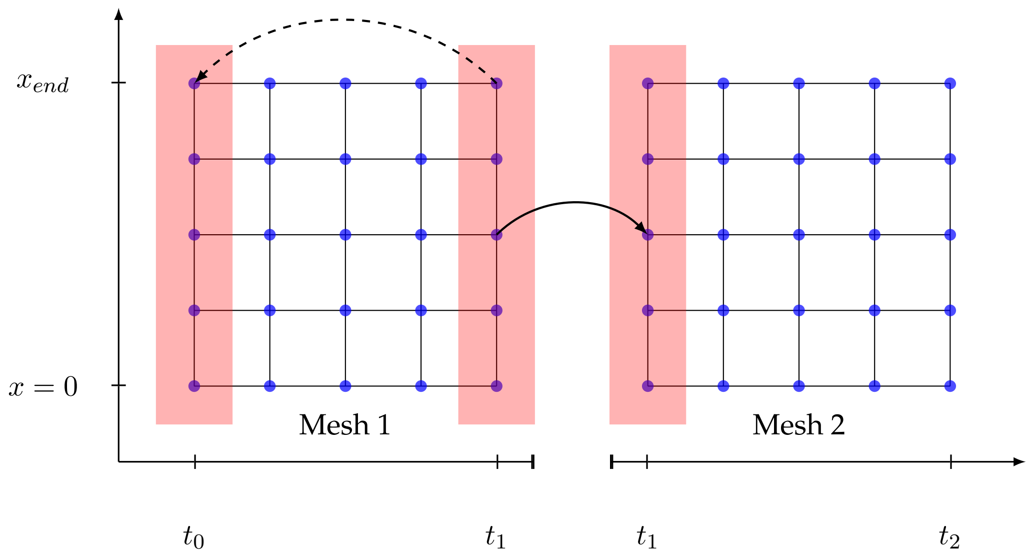


Figure 6.1: Graphic representation of the implemented slab problem

Figure 6.1 visualizes a one-dimensional space and time example, which explains the procedure. The compatibility constraint decouples the meshes, but keeps the DOFs for the same time steps. As a result, the DOFs within the red box at t_1 are the same and are copied for the next mesh. Since the implementation is done with only one mesh, the DOFs are written on the same mesh again. This is indicated with the dashed arrow.

Additionally, the source term functions and time stepping scheme must be adapted as well to ensure the synchronisation between mesh and time. The advantages in terms of computational effort are obvious, as only one mesh needs to be stored and the constraints need to be updated.

It is worth to mention that the approach is not working for nonlinear load stepping problems, as the load stepping is working on the same mesh and therefore overwrites the initial boundary condition of the current time step with the interim solution of the load stepping.

7 Verification

In order to prove the correctness of the implemented code, a verification must be done. This includes a comparison of the error between an appropriate analytic and the obtained solution. As analytic solution the Gaussian function is first introduced in section 7.1, as it is often used to represent the laser beam within SLM simulations. The domain specifications are given in section 7.2. The following sections 7.3, 7.4 and 7.5 contain the verification of the one-dimensional, two-dimensional linear and nonlinear versions. For the linear heat equation, convergence studies are presented to verify the theoretical results of the STFEM within *AdhoC++*. For the verification of the slab in section 7.6, the norms of the surfaces are considered. Section 7.7 evaluates the convergence within time for different orders. Finally, a comparison of the FEM and STFEM is given in section 7.8 to determine the differences between the two methods. Since this chapter deals with manufactured solutions, the units are chosen consistently and not considered further.

7.1 Travelling Gaussian function

First, an analytic solution must be selected. The Gaussian function is perfectly suited, as it satisfies the heat equation 2.1 and in terms of SLM modeling, the external energy added during the process can be modeled as a Gaussian function. In general, the Gaussian function g can be expressed as:

$$g(x) = a \cdot e^{-\frac{(x-b)^2}{2 \cdot d^2}}, \tag{7.1}$$

where the coefficient a stands for the amplitude, b for the center and d for the width of the peak. In Figure 7.1, various peaks are visualized with different parameters.

As the different layers are added by a moving heat source, the Gaussian function must be adapted in order to move within time. For verification, the travelling Gaussian peak in the multidimensional case is selected as a reference solution and can be specified as:

$$T_1(\mathbf{x}, t) = e^{-\frac{(x-t-b)^2}{2 \cdot d^2}} \tag{7.2}$$

with x corresponding to the position vector in Euclidean space for the *1D* space and as:

$$T_2(\mathbf{x}, t) = T_2(x, y, t) = e^{-\frac{(x-t-b_x)^2+(y-b_y)^2}{2 \cdot d^2}} \tag{7.3}$$

for the *2D* space. For the *2D* equation 7.3, the parameter b is split within the space dimensions accordingly.

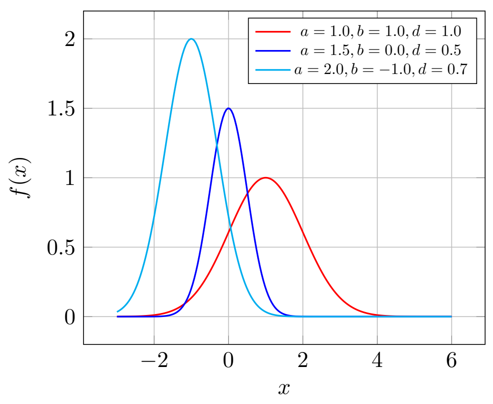
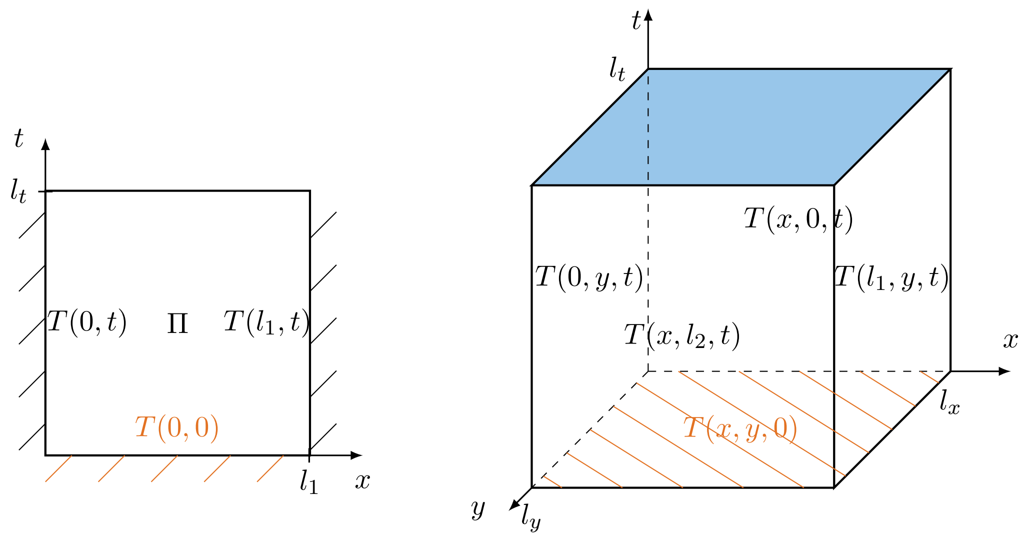


Figure 7.1: Example of Gaussian functions with different parameters a, b and d

7.2 Domain and boundary specification

The verification problem is defined within a square domain with Dirichlet boundary conditions. For the *1D* space problem formulation, this results in a *2D* space time problem, which can be seen in Figure 7.2a. The initial boundary is marked as orange in the Figure 7.2a.



(a) *1D* space time domain Π

(b) *2D* space time domain Π

Figure 7.2: Different domains Π for the space time verification problem

For the *2D* space problem, the *1D* space problem is simply extended to a *3D* problem.

The according edges are changed to faces, resulting in Figure 7.2b. The orange surface within Figure 7.2b determines the initial boundary conditions. The blue surface is the free surface for which the solution should be obtained. The remaining four faces of the cube are the lateral boundary conditions and are constrained with Dirichlet boundary conditions.

In general, it depends on the formulation of the problem, if the initial boundary conditions are specified or not. If the initial boundary conditions are not specified, the solution within STFEM is also obtained for the initial problem at $t = 0$. For the Gaussian function, the initial condition corresponds to the solution of the initial problem. Hence, specifying the initial boundary conditions for the Gaussian function may only result in a more constrained problem. For other problems it may be fatal not to specify the initial conditions.

7.3 Linear transient *1D*

The density, heat capacity and thermal conductivity must now be specified with smooth functions in order to derive the source term $f(\mathbf{x}, t)$ from the manufactured solution of equation 7.2. For the linear case, the three parameters are chosen as:

$$\rho(\mathbf{x}) = (5.0 + x) \cdot cos(x) + 5.0 \tag{7.4}$$

$$k(\mathbf{x}) = (5.0 + x) \cdot sin(x) + 5.0 \tag{7.5}$$

$$c(\mathbf{x}) = 5.0. \tag{7.6}$$

The manufactured solution $T_1(\mathbf{x}, t)$ from 7.2 with parameters $a = 1$, $b = 0.5$ and $d = 0.1$ is used. The grid is defined on the intervall $l_1 = [0, 1]$ and $l_t = [0, 0.5]$ and is discretized with 10 elements per direction. The polynomial degree was defined $p = 3$ for space and time. For the first verification, the solution was obtained serially by solving with the Gaussian elimination from section 5.1. The solution is visualized in Figure 7.3.

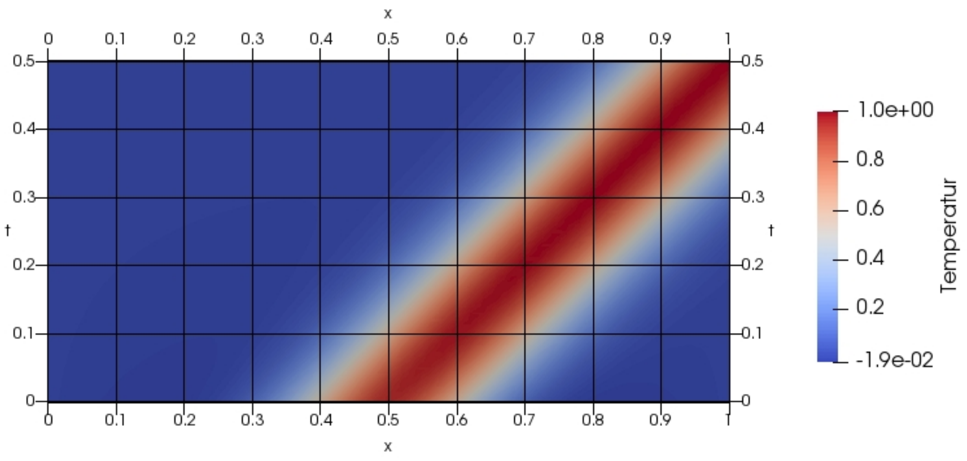


Figure 7.3: Solution of the travelling Gaussian function with 10 elements per direction

According to [53] the L_2 error for piecewise linear basis functions with triangular elements can be stated as:

$$||\hat{T} - \hat{T}_h||_{L_2} \leq ch^{s-1}|\hat{T}|_{H^s} \quad s \in [1, p+1]. \tag{7.7}$$

Even though the error equation 7.7 is not explicitly applicable to the problem previously defined, it still provides an estimate of convergence, since an analogous measure can be derived. Next, a convergence study is made in order compare it with the theory. The L_2 error e_{L_2} within the space time domain is considered as:

$$e_{L_2} = \int_{\Pi} \Big(T_{STFEM} - T_1\Big)^2 d\Pi. \tag{7.8}$$

For the polynomial degree $p = 2$ the convergence can be seen pretty well. The error within Figure 7.4 reduces, according to equation 7.7, by 4 if element size is reduced by half. For the other two polynomials the error reduction takes place within the first steps. As the convergence is finished about 100 DOFs, the convergence can not be seen for further steps.

Even though the error from equation 7.7 is not intended for different material properties, it still is applicable for the formulated problem [53]. The same procedure was applied to the parallel implementation, leading to similar results without further justification.

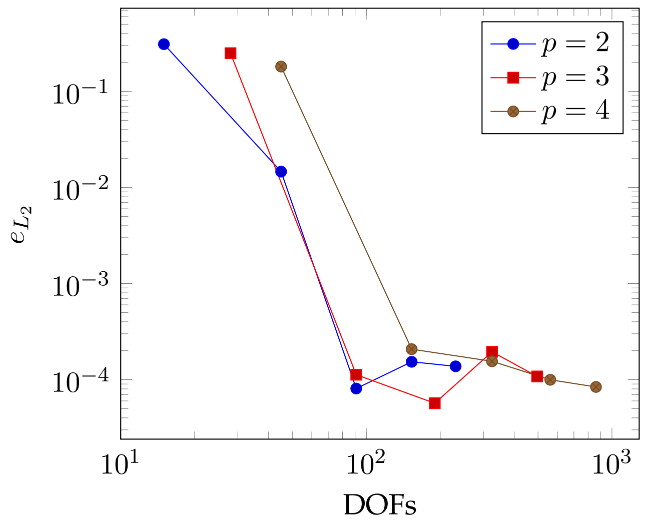


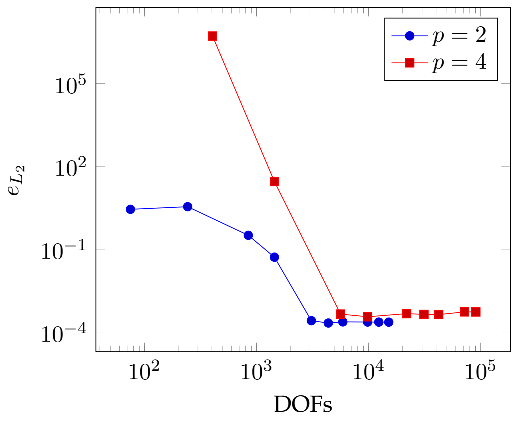
Figure 7.4: *1D* serial convergence study for *h*-refinement with different polynomial degrees

7.4 Linear transient *2D*

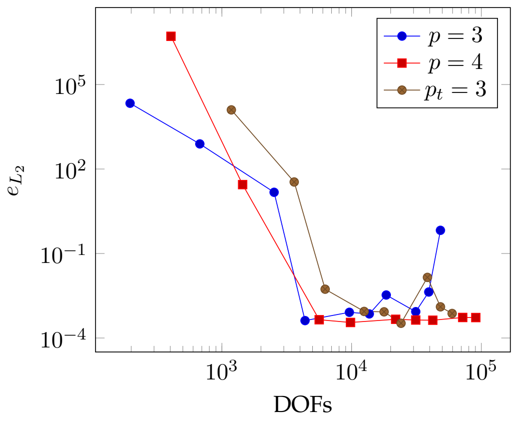
The problem is now extended to *2D*, as it can be seen in Figure 7.2b. The manufactured solution $T_2(\mathbf{x}, t)$ from equation 7.3 is adapted to the verification problem with the material parameters:

$$T_2(\mathbf{x}, t) = e^{\frac{(x-t-0.5)^2+(y-0.5)^2}{2\cdot 0.05^2}} \tag{7.9}$$
$$k(\mathbf{x}) = (5.0 + x) \cdot sin(y) + 5.0 \tag{7.10}$$
$$\rho(\mathbf{x}) = (5.0 + y) \cdot cos(x) + 5.0 \tag{7.11}$$
$$c(\mathbf{x}) = 5.0. \tag{7.12}$$

The domain is specified as $l_x = l_y = 1$ and $l_t = 0.2$ with a uniform discretization of $n_x = n_y = 30$ and $n_t = 3$ elements. The polynomial degree of the elements is specified with $p = 3$. The initial boundary conditions are not specified within this example to demonstrate the initial solve. In this example, the parallel implementation was used with the GMRES solver from section 5.3. The tolerance of the solver was specified as 10^{-5}. The same results can be obtained with the serial version, but the complexity of the problem includes about 2211 elements, which takes too long for the serial implementation.



(a) Same element size for $p = 2$ and $p = 4$



(b) Different element sizes for $p = 3$

Figure 7.5: Convergence study for h-refinement with different polynomial degrees

The convergence study with h-refinement of space and time for the *2D* problem is presented in Figure 7.5. The theory from [53] does not cover *2D* quadrilateral elements, but the convergence in terms of h-refinement for elements with approximately the same size in space and time can still be seen in Figure 7.5a for $p = 2$ and $p = 4$ as the error decreases accordingly. For polynomial $p = 3$ with the equal element length in space and time, the convergence is not optimal, since a larger outlier occurs around 38400 DOFs. For the solution $p_t = 3$, an additional time element is added to estimate whether the outlier can still be seen. Hence, the convergence rate of the solution can be seen better for the first points of $p_t = 3$ in Figure 7.5b, but again a smaller outlier appears at the converged state compared to the reference $p = 4$. Therefore, for a polynomial degree of 3, discretizations with higher error are often found, especially in already converged regions.

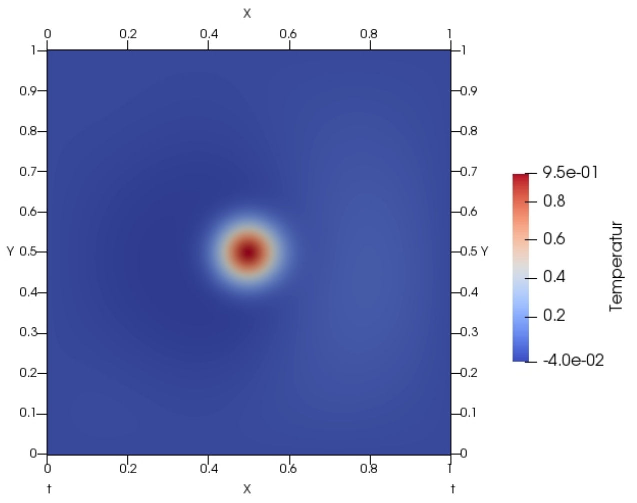


Figure 7.6: Linear Transient $2D$ verification example at $t = 0.0$

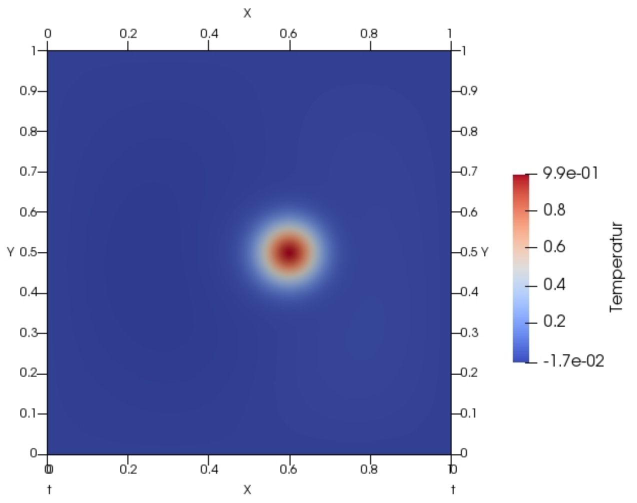


Figure 7.7: Linear Transient $2D$ verification example at $t = 0.1$

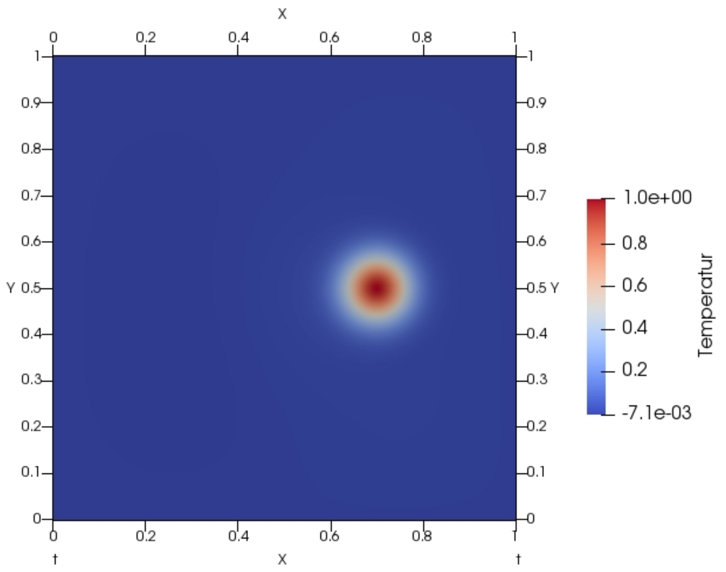


Figure 7.8: Linear Transient *2D* verification example at $t = 0.2$

7.5 Nonlinear problem *1D*

This section will verify the process of nonlinear material problems. First, the material must be adapted according to the derivation in section 4.3. This implies specifying the derivatives of the material. A simple approach is to define the density, heat capacity and thermal conductivity with their respective derivatives as:

$$\rho(\mathbf{x}, T) = (2 \cdot x) \cdot cos(T) + 1.0 \qquad \rho_T(\mathbf{x}, T) = -(2 \cdot x) \cdot sin(T) \tag{7.13}$$
$$k(\mathbf{x}, T) = (2 \cdot x) \cdot sin(T) + 1.0 \qquad k_T(\mathbf{x}, T) = (2 \cdot x) \cdot cos(T) \tag{7.14}$$
$$c(\mathbf{x}, T) = 1.0 \qquad c_T(\mathbf{x}, T) = 0. \tag{7.15}$$

The manufactured solution from equation 7.3 with $a = 1$, $b = 0.5$ and $d = 0.1$ is taken as the reference solution. The domain is specified with $l_x = 1$ and $l_t = 0.2$. The mesh uses 790 DOFs, which arise from $n_x = 20$ space, $n_t = 3$ time elements and a polynomial degree of $p = 2$. The tolerance of the Newton solver is set to $5 \cdot 10^{-3}$, where the direct solver from section 5.1 is used. The L_2 error e_{L_2} reduced to $3.179 \cdot 10^{-6}$ and the result can be viewed within Figure 7.9.

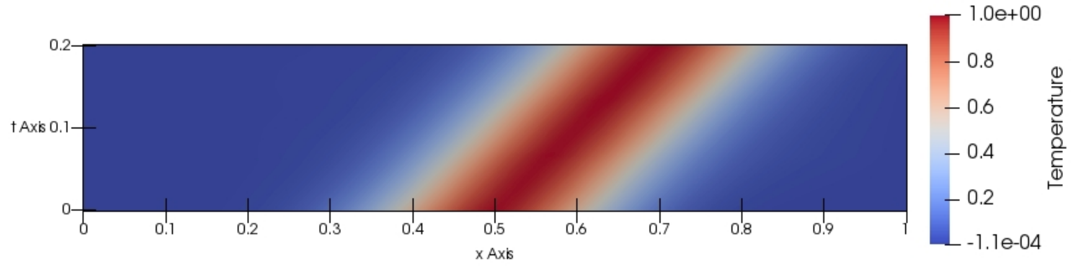


Figure 7.9: Nonlinear *1D* solution of the verification problem

7.6 Verification of the slab

For the verification of the slab, the properties are specified with minor changes compared to section 7.4:

$$T_2(\mathbf{x},t) = e^{\frac{(x-t-0.5)^2+(y-0.5)^2}{2\cdot 0.05^2}} \tag{7.16}$$

$$k(\mathbf{x}) = (5.0+x)\cdot sin(y) + 5.0 \tag{7.17}$$

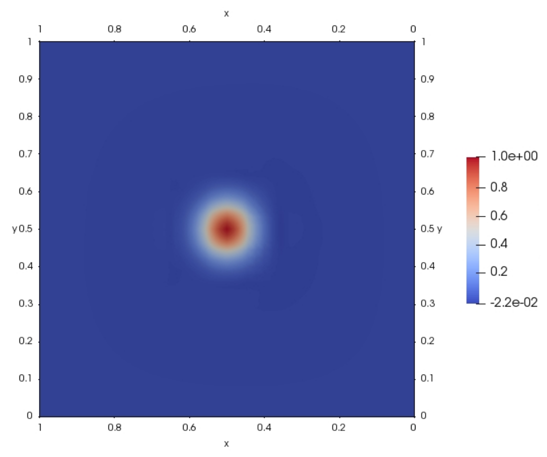
$$\rho(\mathbf{x}) = c(\mathbf{x}) = 1.0. \tag{7.18}$$

The problem size in time is defined as $t_d = 0.3$ with 3 slabs, which results in a slab size of $l_t = 0.1$. The domain size of the space is defined on the unit interval $l_x = l_y = 1$. The mesh is specified with $n_x = n_y = 9$ and $n_t = 1$ elements and a polynomial degree of $p = 2$. In addition, h-refinement is applied twice within the volume $[0.45 \times 0.85, 0.4 \times 0.6, 0 \times 1]$ of the mesh. The results of the slab solution are visualized in the Figures 7.10a-7.10d.

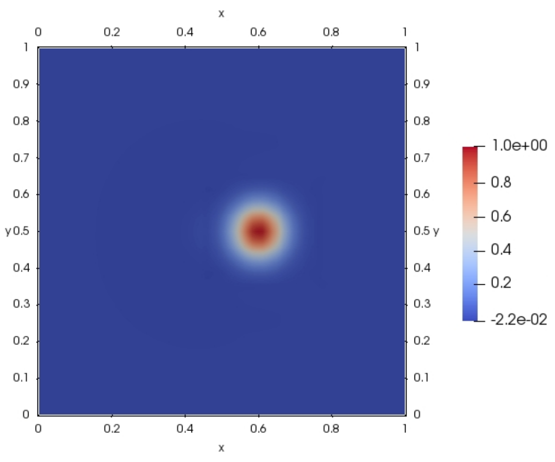
In order to verify the slab, the L_2 norm of the individual surfaces are checked. This must be done, since the boundary conditions are not enforced directly, but via a projection of the topological components. For verification, the integrals of the projected end surface and the new surface should be equal at the same time. Table 7.1 displays the values of the integrals from previous end and new start surface, respectively. As the integrals do not differ much, it can be concluded that the projection error within the slab surfaces is below the tolerance and the slab is working properly.

time	end surface	start surface
0.1	$7.102 \cdot 10^{-3}$	$7.195 \cdot 10^{-3}$
0.2	$7.382 \cdot 10^{-3}$	$7.378 \cdot 10^{-3}$

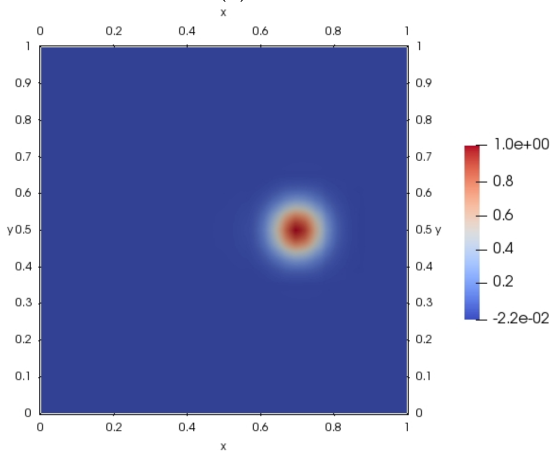
Table 7.1: L_2 integral of the previous and the new slab surface



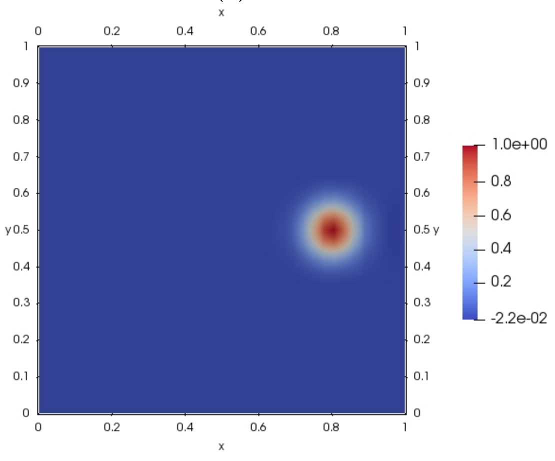
(a) $t = 0.0$



(b) $t = 0.1$



(c) $t = 0.2$



(d) $t = 0.3$

Figure 7.10: Solution of the temperature distribution at different time steps

7.7 Verification of the time order

This section will investigate the order of the time discretization. As example, the verification problem from 7.2b is taken, where the lengths are within the unit cube $l_x = l_y = l_t = 1$ and the surfaces are specified as Dirichlet boundaries. The parameters for the respective material coefficients are taken from section 7.4 and can be found in equations 7.10, 7.11 and 7.12. In addition the solution is adapted to:

$$T_C(\mathbf{x}, t) = e^{-\frac{(x - \frac{sin(\pi t)}{4} - 0.5)^2 + (y - \frac{cos(\pi t)}{4} - 0.5)^2}{2 \cdot 0.05^2}}. \tag{7.19}$$

The resulting solution is a travelling Gaussian function on a circle with diameter 0.5, where the center is located in the center of the unit domain $[0.5, 0.5]$. From the Figures 7.12a-7.12d, it can be seen that the circle starts at the top and moves to the bottom with a circular motion. The error of the solution can be split up into a spatial and time part. For the STFEM the error is expected to decrease as:

$$e_{STFEM} = O(\Delta h^p) + O(\Delta t^p). \tag{7.20}$$

For the following example the spatial error is fixed with $n_x = 20$ space elements and time elements are refined between $n_t = [1, 20]$. The result can be seen within the Figure 7.11. The L_2 error e_{L_2} from equation 7.8 decreases with respect to p, as expected from equation 7.20. For a polynomial degree of $p = 3$, small oscillations for the first odd element numbers occur in the solution. The oscillations are reproduced in the convergence of Figure 7.11, since the error is significantly higher for the odd discretizations with $p = 3$.

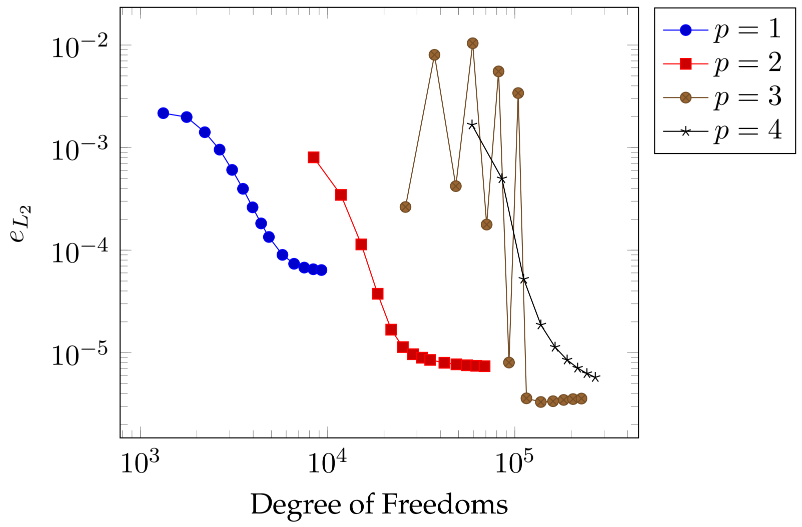
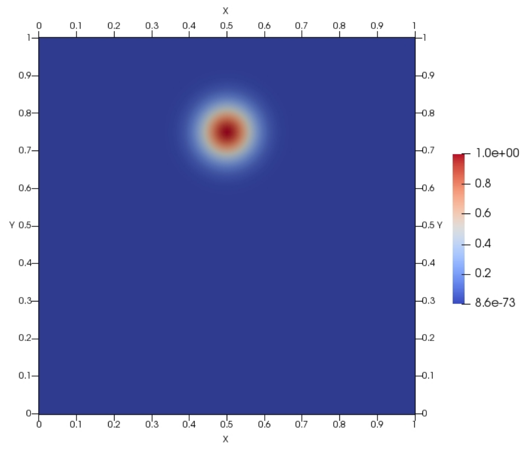
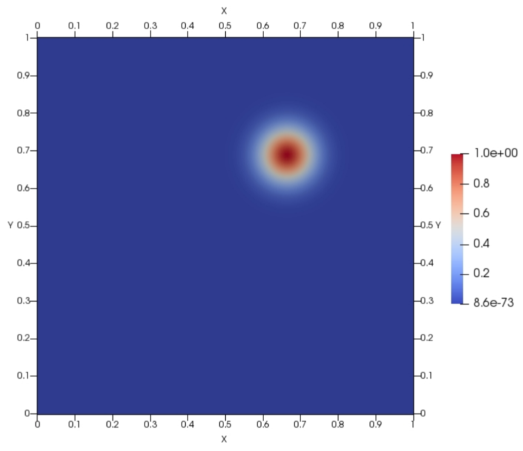


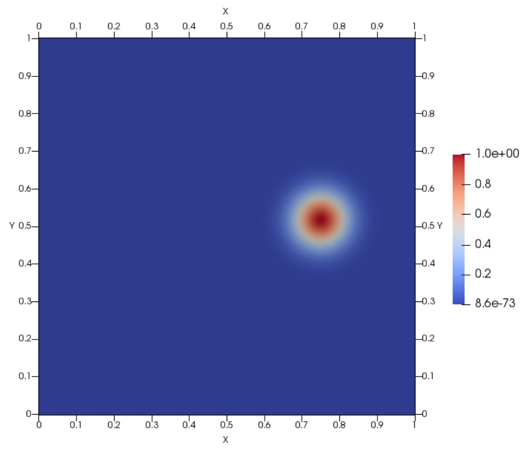
Figure 7.11: Error e_{L_2} during h-refinement within time for different polynomial degrees



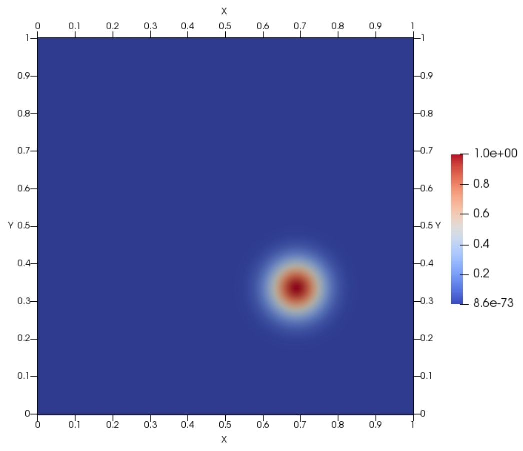
(a) $t = 0.00$



(b) $t = 0.25$



(c) $t = 0.5$



(d) $t = 0.75$

Figure 7.12: Solution T_C at different timesteps

In summary, as expected from equation 7.20, the error decreases within time for $p = 1$ and $p = 2$. For $p = 3$ the expected error for odd element numbers could not be verified. For the remaining error with $p = 4$, the range of the solution is too small to fully verify the fourth order convergence, but it is indicated for some points.

7.8 Comparison of STFEM and FEM

This section will finally compare the implementations of the implicit FEM from chapter 2 to the STFEM from chapter 3. As the FEM was already verified within *AdhoC++*, only the specific example was set up as indicated in section 7.7. Hence, all parameters and domain properties can be found in chapter 7. Since the focus of the framework was on symmetric-definite problems, the CG was further optimized as the framework provides preconditioning methods, such as algebraic multigrid for the CG. In order to enable a comparison between the two methods, the solving process is done with the same direct solver. The following solutions are obtained by a parallel version with 1 MPI thread, as the L_2 error is not obtained parallel.

The reference is defined as the L_2 error norm. Previous considerations of the L_2 error e_{L_2} from equation 7.8 within the whole space-time domain Π can not be made because the FEM has no solution between the different time steps. Thus, the $e_{L_2}^{t_n}$ error at a timestep is defined as:

$$e_{L_2}^{t_n} = \int_{\Omega} \Big(T_{FEM}(t_n) - T_C(t_n) \Big)^2 d\Omega. \tag{7.21}$$

For the following comparison, the error $e_{L_2}^{l_t}$ is evaluated at the end of the time domain, therefore at time $t_n = l_t$. But before, it should be noted that the implicit FEM version has not the same order of error with respect to time. The error for the implicit FEM can be stated as:

$$e_{FEM} = O(\Delta h^p) + O(\Delta t) \tag{7.22}$$

and is of first order within time. For the STFEM, the order depends on the polynomial degree p and is the same for space and time, as only isotropic quadrilateral elements are considered for the comparison. The idea is to evaluate the qualitative solution at the final time with respect to the error within time. Therefore, the space discretization is fixed with a number of elements and the time elements are refined accordingly. The number of time points for the implicit FEM is chosen according to the polynomial degree $p \cdot n_t$. This results in a sufficient number of points for the space time discretization and leads to a linear rise of DOFs in the Figures 7.13 and 7.14 for the FEM. In Figure 7.13, it can be seen that the STFEM is of higher order with respect to time.

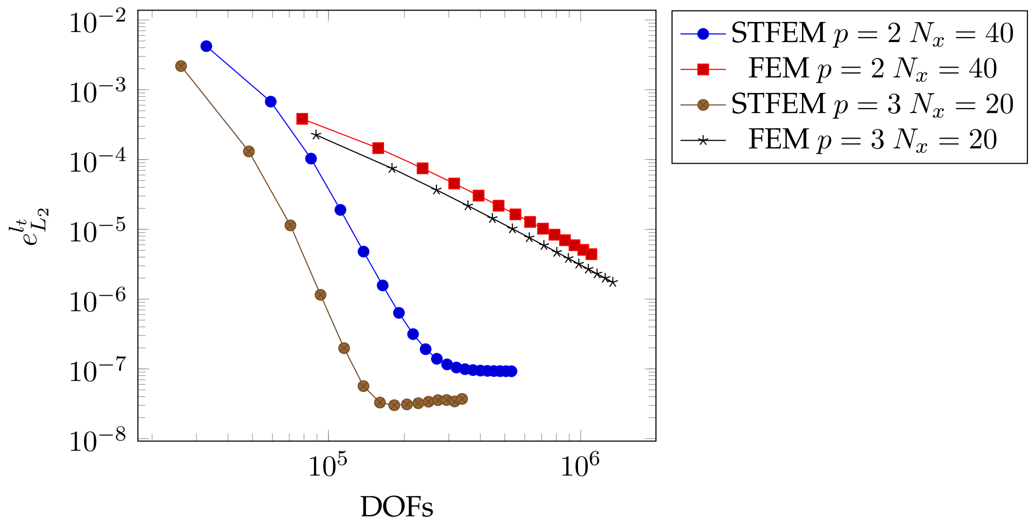


Figure 7.13: Error $e^{l_t}_{L_2}$ during h-refinement within time

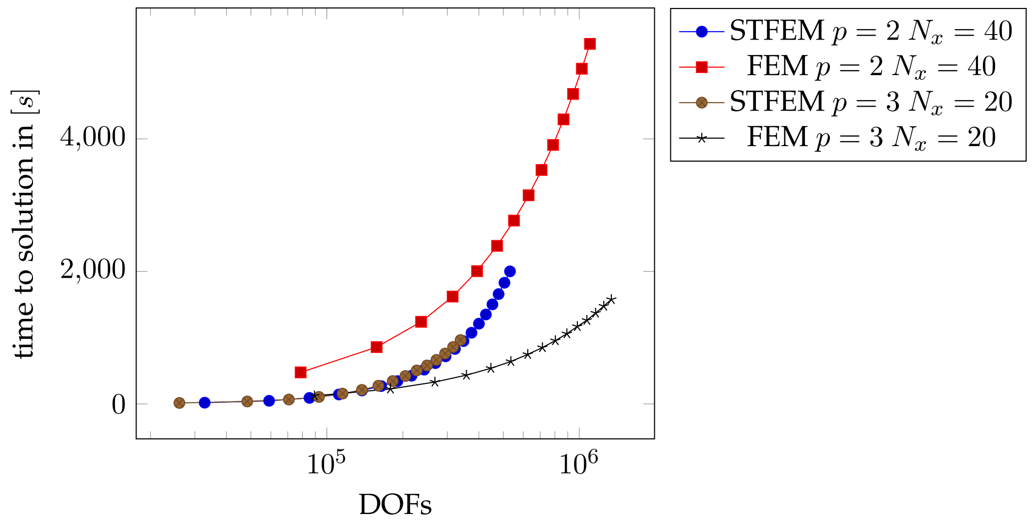


Figure 7.14: Time required to solve the FEM and STFEM

In terms of solution time, the STFEM within the *AdhoC*++ was faster than the implicit FEM, as it can be seen from Figure 7.14. The curves for the STFEM are identical, as the most time spent within the STFEM is for the system solve. Hence, the characteristic of the solver can be found in the Figure 7.14 again.

As the error order of the methods differ according to equations 7.20 and 7.22, the timestep must be adapted to:

$$\bar{t} = \begin{cases} \Delta t = \frac{l_t}{n_t}, & \text{for FEM} \\ \Delta t^p = \frac{l_t}{n_t^p}, & \text{for STFEM} \end{cases}, \tag{7.23}$$

to enable a comparison. For the following Table 7.2 the values $n_x = 40$ and a polynomial degree of $p = 2$ the error and time are compared. From the Table 7.2, it can be concluded that the proportional error is higher at the STFEM, but the solution was obtained faster.

$\bar{t}$	FEM $e_{L_2}^{l_t}$	STFEM $e_{L_2}^{l_t}$	FEM time $[s]$	STFEM time $[s]$
$\frac{1}{2}^2$	$3.830 \cdot 10^{-4}$	$4.214 \cdot 10^{-3}$	475.714	18.262
$\frac{1}{4}^2$	$4.537 \cdot 10^{-5}$	$6.757 \cdot 10^{-4}$	1620.195	46.731
$\frac{1}{6}^2$	$1.023 \cdot 10^{-5}$	$1.031 \cdot 10^{-4}$	3530.632	90.406

Table 7.2: Error and time for selective points

In terms of memory consumption, the most storage is needed for the allocation of the global matrix system. For the FEM, the symmetric matrix can be stored efficiently, whereas for STFEM the whole non-symmetric matrix needs to be stored. During the solving process, the efficient symmetric matrix storage of the FEM is however lost due to the "Fill-Ins" of the direct solver. In addition, for the STFEM, the matrix size depends on the spatial and time discretization, whereas for the FEM, the global matrix size depends only on the spatial discretization. As the time discretization is refined, corresponding matrix system rises and for arbitrary long time problems, this results in a large non-symmetric matrix.

To summarize the results, the STFEM was able to obtain the solution faster than the implicit FEM within *AdhoC*++. In addition, the solution is obtained with higher precision after a certain discretization by using the same direct solver within a shared memory context. A drawback of the STFEM is the storage and size of the system matrix.

8 Phase change with Space Time Finite Elements

As already indicated in the introduction, one goal of the thesis is to apply the STFEM to the solidification and melting within the SLM process. The phase change problem can be modeled as tracking or fixed-domain model. Tracking methods include following the boundaries of the moving phase change and working on deformed meshes with the interface boundaries. This is typical for a Stefan Problem [17]. In contrast to this, the phase change is modeled in the same domain but using descriptions, such as the latent heat model, to obtain the temperature. The nonlinearity within the latent heat equation arises from the discontinuous phase change function. Both models result in nonlinear problems due to the physics of the phase change. This chapter first gives an introduction to the latent heat equation in section 8.1 and derives the STFEM for the latent heat model in section 8.2. In the following section 8.3 comments to the implementation are made. The results are verified using an example of a melting bar and a solidification of a square in section 8.4. To close the chapter, the results of the application are summarized in section 8.5.

8.1 Introduction to the latent heat model

The quantity of interest within the phase change problem is the temperature distribution T, as the phase state of the material is dependent on the temperature. According to the energy equation 8.1, the change within temperature, the specific energy ω and the density ρ_0 must be balanced with the specific heat source r and the heat flux q in every part of the domain Ω as:

$$\rho_0 \frac{\partial \omega}{\partial T} \cdot \dot{T} = -\nabla q + \rho_0 \cdot r \quad \Omega \times \tau. \tag{8.1}$$

To start with the derivation, the specific internal energy is defined as:

$$\omega = \int_{T_{ref}}^{T} c \, dT + L f_{pc}. \tag{8.2}$$

The first term consists of the integrated specific heat capacity c between the reference temperature T_{ref} and the final temperature T. The second term consists of the latent heat L, which releases or absorbs energy during the phase change. To determine whether latent heat is present, a discontinuous phase change function f_{pc} is specified:

$$f_{pc}(T) = \begin{cases} 0, & T \leq \bar{T}_m \\ 1, & T > \bar{T}_m. \end{cases} \tag{8.3}$$

The phase change function is the source of the nonlinearity within the problem formulation. It sets the latent heat free, if the temperature T exceeds the melting temperature $\bar{T}_m$ of the material. For the phase change function f_{pc} from equation 8.3 the model assumes the boundary Γ_{PC} of the latent heat to be isothermal. For numerical reasons, it is therefore advantageous to smooth the discontinuity and model the phase change function with the temperatures of the solid T_s and liquid T_l phases:

$$f_{pc}(T) = \frac{1}{2}\left[tanh\left(\frac{2}{T_l - T_s} \cdot \left(T - \frac{T_s + T_l}{2}\right)\right)\right] + \frac{1}{2}. \tag{8.4}$$

Since the only unknown term is the heat flux q, it must be substituted with a constitutive law such as Fourier's law:

$$q = -k\nabla T. \tag{8.5}$$

Combining the energy equation 8.1 with the specific energy equation 8.2 and Fourier's law 8.5 leads to the final equation 8.6. In order to extend equation 8.6 to a problem formulation, the boundary and initial conditions are defined as well:

$$\rho_0 \cdot \left(c + \frac{Lf_{pc}(T)}{\partial T}\right) \cdot \dot{T} = -\nabla(-k\nabla T) + \rho_0 \cdot r \qquad \text{in} \quad \Omega \times \tau \tag{8.6}$$

$$T(\mathbf{x}, t) = T_{init}(\mathbf{x}, t) \qquad \text{in} \quad \Gamma_D \times \tau \tag{8.7}$$

$$-k_n \frac{\partial T}{\partial n}\bigg|_{\Gamma_N, t} = q_0 \qquad \text{in} \quad \Gamma_N \times \tau. \tag{8.8}$$

Figure 8.1 displays the problem. The phase state is defined over the temperature T, which causes to move the boundary Γ_{PC} according to the latent heat L and the phase change function $f_{pc}(T)$. The global domain does not change, only the respective solid Ω_s and liquid Ω_l state of the domain.

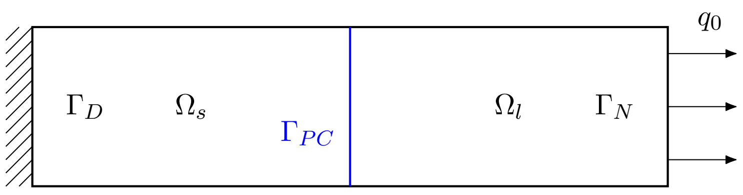


Figure 8.1: A phase change problem

8.2 STFEM for the phase change

As already done for the heat equation in section 2.2, the weak equation for the latent heat equation is again derived by multiplying with the test function ϕ and integrating over the domain $\Pi = \Omega \times \tau$:

$$\int_\Pi \rho_0 \cdot \left(c + \frac{Lf_{pc}(T)}{\partial T}\right) \cdot \dot{T} \cdot \phi \, d\Pi = \int_\Pi -\nabla(-k\nabla T) \cdot \phi \, d\Pi + \int_\Pi \rho_0 \cdot r \cdot \phi \, d\Pi. \tag{8.9}$$

By applying the divergence theorem and cancelling the Dirichlet boundary, the final equation is obtained:

$$\begin{aligned}\int_\Pi \rho_0 \cdot \left(c + \frac{Lf_{pc}(T)}{\partial T}\right) \cdot \dot{T} \cdot \phi \, d\Pi + \int_\Pi (k\nabla T) \cdot \nabla\phi \, d\Pi = \int_\Pi \rho_0 \cdot r \cdot \phi \, d\Pi + \\ \int_{\Gamma_N} (k\nabla T) \cdot \phi \, d\Gamma.\end{aligned} \tag{8.10}$$

The solution space is obtained from the Petrov-Galerkin approach and can be stated as:

$$\boldsymbol{X} = \{T | T \in H^1(\Omega) \times H^1(\tau), T = T_0 \, \forall \, \mathbf{x} \in \Gamma_D\} \tag{8.11}$$

$$\boldsymbol{\Phi} = \{\phi | \phi \in H^1(\Omega) \times L^2(\tau), \phi = 0 \, \forall \, \mathbf{x} \in \Gamma_D\}. \tag{8.12}$$

For the discretization, appropriate basis functions in space and time are selected:

$$T \approx \hat{T} = \sum_{i=1}^{n_{dofs}} N_i(\mathbf{x}, t)^T \cdot T_i \tag{8.13}$$

$$\hat{\phi} = \sum_{j=1}^{n_{dofs}} N_j(\mathbf{x}, t)^T \cdot \phi_j \tag{8.14}$$

and the gradient is separated to derive the equation:

$$\begin{aligned}\sum_{j=1}^{n_{dofs}} \phi_j \sum_{i=1}^{n_{dofs}} T_i \int_\Pi \rho_0 \cdot \left(c + \frac{Lf_{pc}(T)}{\partial T}\right) \cdot \nabla_t N_i^T \cdot N_j^T \, d\Pi + \\ \sum_{j=1}^{n_{dofs}} \phi_j \sum_{i=1}^{n_{dofs}} T_i \int_\Pi k \nabla_{\mathbf{x}} N_i^T \cdot \nabla_{\mathbf{x}} N_j^T \, d\Pi = \\ \sum_{j=1}^{n_{dofs}} \phi_j \int_\Pi \rho_0 \cdot r \cdot N_j^T \, d\Pi + \sum_{j=1}^{n_{dofs}} \phi_j \int_{\Gamma_N} q_0 \cdot N_j^T \, d\Gamma_N,\end{aligned} \tag{8.15}$$

as already done for the heat equation.

Up to this point, there is not much difference to the derivation of the heat equation from the previous section 2.2. If the equation 8.15 is combined with the B-Operator from equations 3.10 and 3.11, this results in the same matrix equation:

$$\big(M(T) + K(T)\big) \cdot T = F(t) \tag{8.16}$$

as the one previously defined in equation 3.12. Thus, a comparison of the respective terms $M(T)$, $K(T)$ and $F(t)$ is made in Table 8.1 to highlight the differences. For the mass term $M(T)$, the density stays the same, but the heat capacity c gets the additional latent heat L with the respective phase change function f_{pc}. The stiffness term $K(T)$ remains the same, as the constitutive law is not changed. For the comparison of the source vector $F(t)$, the Neuman part is neglected and the source term $f(\mathbf{x}, t)$ is split into a specific source part r and the density ρ_0. Hence, mainly the coefficients of the respective terms change.

term	heat equation	latent heat equation
$M(T)$	$\int_\Pi N^T c\rho B_t \, d\Pi$	$\int_\Pi N^T \rho_0 \Big(c + \dfrac{L f_{pc}(T)}{\partial T}\Big) B_t \, d\Pi$
$K(T)$	$\int_\Pi B_x^T k B_x \, d\Pi$	$\int_\Pi B_x^T k B_x \, d\Pi$
$F(t)$	$\int_\Pi N^T f(\mathbf{x}, t) \, d\Pi$	$\int_\Pi N^T \rho_0 \cdot r \, d\Pi$

Table 8.1: Comparison of the heat equation and the latent heat equation

8.3 Implementation of the latent heat equation

In order to simulate the latent heat problem, the physical equations had to be added. This was done by implementing the respective coefficients and source terms within the appropriate factories and application files. From the previously implemented heat equation, the basic routines were slightly adapted. This included a revision of the energy equation, matrix assembly, source terms and material properties. Hence, the boundary conditions, mesh and refinement strategy did not need to be adapted.

8.4 Verification

To proof the correctness of the implemented method, two verification examples are presented. The first example is the melting of a solid bar. Therefore, the important aspects of the analytic solution are explained and the results are compared with the solution of the Stefan problem. Afterwards, the solidification of a *2D* plate is simulated and compared to a reference solution. The different parameters are stated to enable a comparison.

8.4.1 Melting of a bar

As a first application, the melting of a one-dimensional semi-infinite bar is considered. The bar is at an initial temperature T_{init}, which is below the melting temperature $\bar{T}_m$. At the initial state $t = 0\ s$, a stationary temperature T_{BC} is applied on one end of the bar. As the boundary temperature exceeds the melting temperature, the bar starts to melt. Figure 8.2 visualizes the problem.

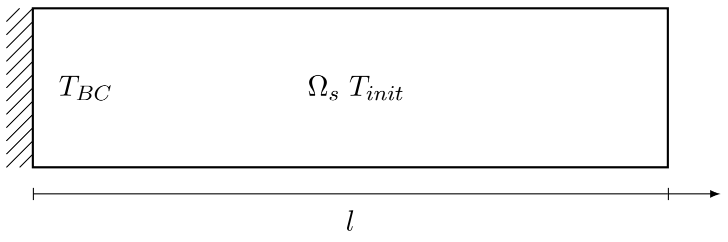


Figure 8.2: One-dimensional bar for the phase change

The material of the bar is specified as Titanium with the material description in Table 8.2. The problem was already solved with the standard FEM by [36] within *AdhoC++*, hence the material parameters and boundary conditions are adapted to enable a comparison.

k	$16\ W/m^3$°C	T_{BC}	2000 °C
ρ	$4510\ kg/m^3$	$\bar{T}_m$	1670 °C
c	$520\ J/(m^3$°C$)$	T_{init}	1500 °C
L	$325000\ J/kg$		

Table 8.2: Thermo-physical material properties of Titanium for liquid and solid

The analytic solution for the liquid domain can be obtained from solving the respective Stefan problem for the heat equation, which can be found in [18, 26]. The solution of the temperature distribution can be stated as:

$$T(\mathbf{x},t) = \begin{cases} T_l - \left(T_l - \bar{T}_M\right) \frac{\operatorname{erf}\left(x/2\sqrt{\alpha_l t}\right)}{\operatorname{erf}(\lambda)} & \text{if} \quad x \leq I(t) \\ T_s + \left(\bar{T}_M - T_s\right) \frac{\operatorname{erfc}\left(x/2\sqrt{\alpha_s t}\right)}{\operatorname{erfc}\left(\lambda\sqrt{\alpha_1/\alpha_s}\right)} & \text{if} \quad x > I(t), \end{cases} \tag{8.17}$$

where the position of the interface between the liquid and solid state $I(t)$ is defined by:

$$I(t) = 2\lambda\sqrt{\alpha_l t}. \tag{8.18}$$

The coefficient λ can be obtained by solving the nonlinear equation:

$$\frac{\mathrm{St}_1}{\exp\left(\lambda^2\right)\operatorname{erf}(\lambda)} - \frac{\mathrm{St}_s\sqrt{\alpha_s}}{\sqrt{\alpha_l}\exp\left(\alpha_l\lambda^2/\alpha_s\right)\operatorname{erfc}\left(\lambda\sqrt{\alpha_l/\alpha_s}\right)} = \lambda\sqrt{\pi} \tag{8.19}$$

with the missing Stefan numbers:

$$\mathrm{St}_1 = \frac{C_l\left(T_l - \bar{T}_m\right)}{L} \quad \text{and} \quad \mathrm{St}_s = \frac{C_s\left(\bar{T}_m - T_s\right)}{L} \tag{8.20}$$

and the diffusivity constants for liquid and solid:

$$\alpha_l = \frac{k_l}{\rho_l c_l} \quad \alpha_s = \frac{k_s}{\rho_s c_s}. \tag{8.21}$$

The results of the STFEM in comparison with the analytic solution are displayed in Figures 8.3 and 8.4. The bar has a length $l = 0.5\ m$ and is discretized with $n_x = 50$ space and $n_t = 10$ time elements of order $p = 4$. As a refinement strategy, 2 h-refinements in space interval $[0.0, 0.1]\ m$ and time interval $[0, 1]\ s$ are applied. The solution is obtained by a direct solver.

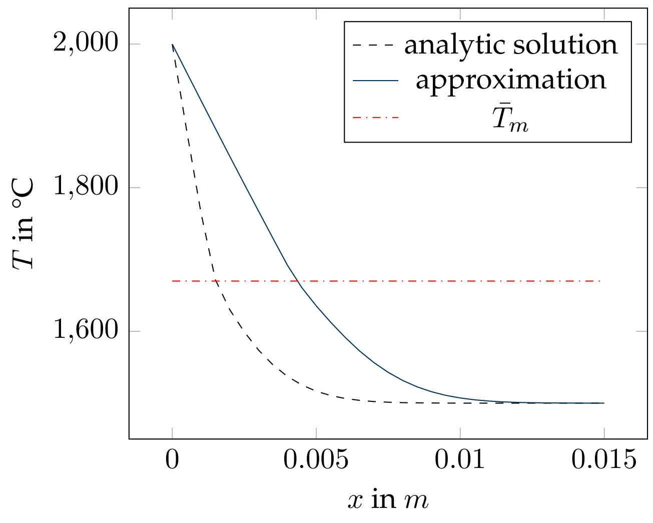


Figure 8.3: Temperature distribution within the bar at $t = 0.5\ s$

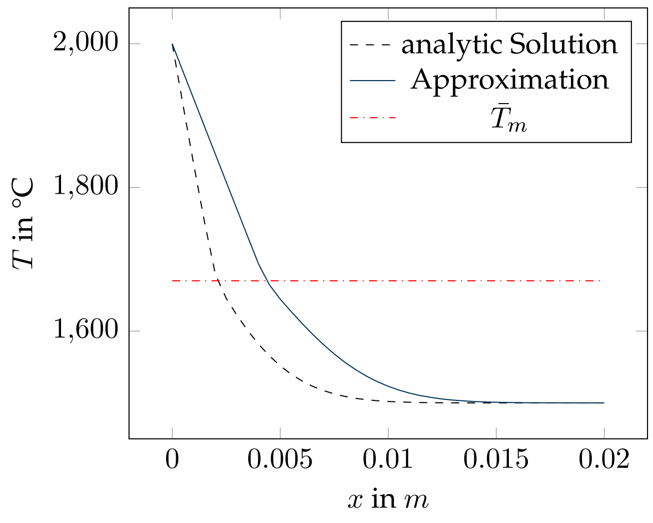


Figure 8.4: Temperature distribution within the bar at $t = 1\ s$

Even though the results look quite promising, the solution is obtained with a relatively fine discretization. For coarser problems, the solution was very diffusive, leading to a faster rise of the temperature. This behaviour can also be seen in Figure 8.3, as the temperature is higher compared to the analytic solution.

8.4.2 Freezing of a liquid square

For the two-dimensional verification, the analytic solution for the Stefan problem could not be derived, but previous considerations by [39] are compared within the commercial software Abaqus[51]. Therefore, the Abaqus example "freezing of a square solid" is taken as reference solution. The problem consists of a *2D* plate filled with liquid, where the temperature of the liquid is just above the freezing temperature. The outer Dirichlet boundaries are set below the freezing point. As a result, the square starts to freeze from outside towards the center. The material parameters are defined same for both problems and can be found in Table 8.3.

k	$1.08\ W/m^3$°C	T_s	-0.25 °C
ρ	$1.0\ kg/m^3$	T_l	-0.15 °C
c	$1.0\ J/(m^3$°C$)$	$\bar{T}_m$	0 °C
L	$70.26\ J/kg$	$\hat{T}_{BC}$	-45 °C

Table 8.3: Thermo-physical material properties for the *2D* freezing square

The problem including the discretization, can be viewed in Figure 8.5. To follow the solidification in two dimensions, the temperature distribution along the blue line in Figure 8.5 is taken at different time steps for the comparison. The temperature distribution at the point R_1 at $[x = 1\ m, y = 1\ m]$ is investigated continuously over the time domain. The

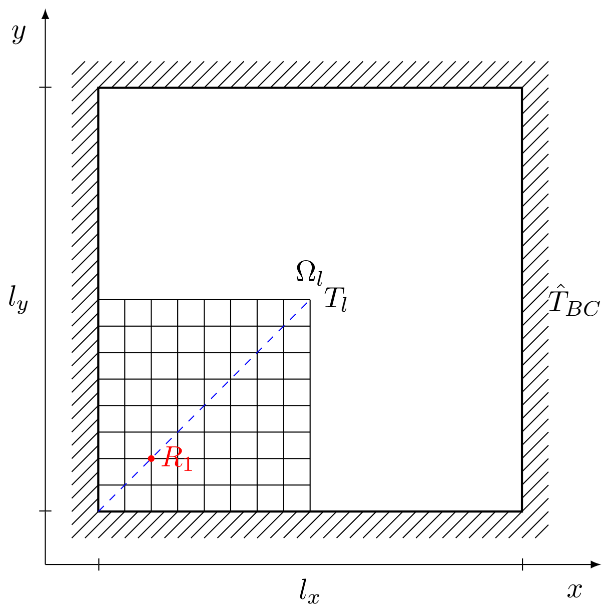


Figure 8.5: *2D* plate with discretization

length of the domain is specified as $l_x = l_y = 8\,m$. The reference solution is obtained by using a mesh consisting of linear, quadrilateral elements with the FEM for the heat equation. The standard, implicit time stepping is used within Abaqus, which is implemented as a half-step residual approach [28]. This approach selects a maximum temperature rise and adapts the time step according to the maximum temperature difference. For the reference solution in Abaqus, the temperature difference is set to 2 °C. The boundary conditions are specified as symmetric. As a result only a quarter must be simulated in comparison to the STFEM.

The solution of the *AdhoC*++ is compared with two different discretizations. The first discretization uses a mesh with space $n_x = 16$ elements, time elements $n_t = 12$ and a polynomial degree of $p = 3$. This results in the same number of elements as for Abaqus. The second solution is obtained by using a lower discretization, as a polynomial degree of $p = 1$ with 32 space and 9 time elements are used. The results can be viewed in the Figures 8.6, 8.7 and 8.8.

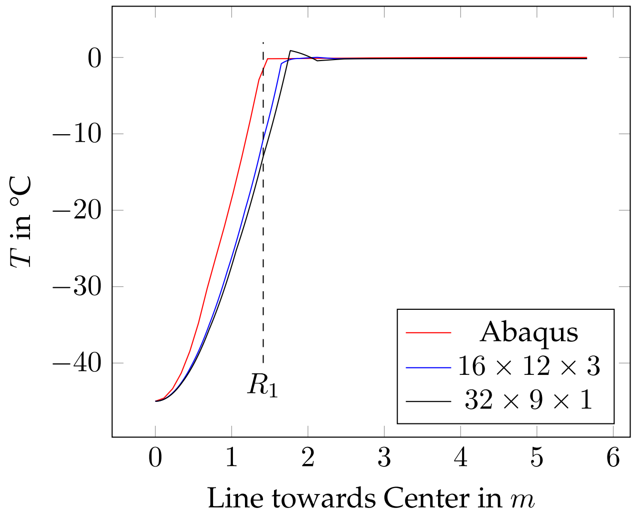


Figure 8.6: Temperature distribution towards the center at $t = 0.5\ s$

Figures 8.6 and 8.7 visualize the solution across the blue line from Figure 8.5 at the time steps $t = 0.5\ s$ and $t = 1.0\ s$. For both figures it can be seen that the cube freezes faster within *AdhoC*++, as the temperature gradient is steeper than for Abaqus. In addition, the polynomial degree with $p = 1$ shows small oscillations around the freezing temperature.

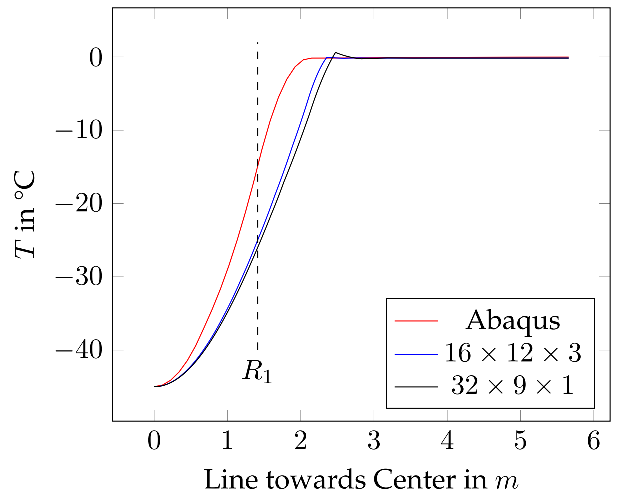


Figure 8.7: Temperature distribution towards the center at $t = 1\ s$

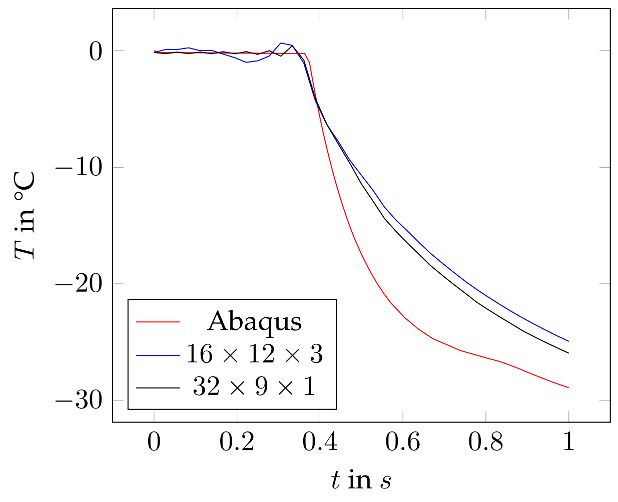


Figure 8.8: Timeline of the solution at the point R_1

Figure 8.8 visualizes the solution for the reference point R_1. The solution does not behave physically correct, because the temperature exceeds the initial temperature before the freezing point. From the different polynomial solutions, it can be concluded that the higher polynomials lead to bigger oscillations. The freezing point at $t = 0.4\ s$ is for both *AdhoC*++ solutions shifted. In addition, the temperature gradient with respect to time is steeper for the reference solution than for the solution from *AdhoC*++. This results in a lower temperature drop over time. The temperature gradient is better approximated for the lower polynomial degree.

8.5 Summary of phase change

To summarize the results of the phase change, for both cases the temperature was slightly more diffusive than expected from the solutions. The one-dimensional bar was approximated well, as the melting of the bar is only slightly faster than the analytic solution. For the *2D* plate, the freezing of the the STFEM proceeds faster and leads to nonphysical oscillations around the freezing point. For the higher polynomial degree, the oscillations rise.

The previous examples showed that it is possible to obtain plausible results for the STFEM applied to the phase change. Nevertheless, both examples needed fine discretizations and a high number of Newton iterations. This included various tests of the solvers and the parameters to obtain the results. Therefore, further investigations with respect to stability should be made. For the *2D* plate, the solution was achieved significantly faster with Abaqus that a comparison with AdhoC++ is superfluous.

9 Summary and Conclusion

Finally, the summary revises the chapters and the conclusion states the results of the thesis. Further research directions are presented in the outlook.

9.1 Summary

The STFEM was applied to heat transfer problems to provide solution strategies for adaptive manufacturing. First the heat equation with the mathematical problem formulation was introduced and the FEM for the heat equation was derived. This included a revision of the fundamental concepts of the FEM and time stepping schemes.

The STFEM was derived for the heat equation and the differences to the FEM were stated. In addition, the space time slab was introduced. The general procedure for the nonlinear FEM was revised, beginning with the appropriate Newton solver and the derivations of the nonlinear FEM and STFEM. This included illustration of the implemented concepts with algorithms. The respective solvers and preconditioners were briefly discussed. For the solver, these were direct solvers, CG and GMRES, and for the preconditioner, Jacobi and Neumann.

Additionally, the implementation concepts of *AdhoC++* were revised and the respective changes to implement the STFEM were highlighted. Deeper insights regarding mesh were given with additional information on parallel concepts and external libraries.

For verification, several linear and nonlinear transient test cases were setup and reviewed by different analytic solutions with the Gaussian function. Theoretical results and expectations for the linear STFEM were confirmed within different convergence studies. Finally, the different methods were compared with respect to mathematical aspects as accuracy and convergence. In addition, the required resources as memory and time consumption were evaluated.

The latent heat model for phase change was introduced and the STFEM was derived for the problem. Differences to the heat equation were discussed and the implementation was verified against an analytic solution of a Stefan Problem. As one-dimensional verification, the melting of a bar was simulated. For the two-dimensional version, the solution was compared to a benchmark example of Abaqus.

9.2 Conclusion

The STFEM was successfully implemented and verified within *AdhoC++*, which included an implementation of a serial and parallel version. The idea of using the last dimension for time proved to enable a fast implementation within existing frameworks and the use of present features and optimizations. For the serial version, the space time slab proved to decompose the space time problem into smaller pieces and enable an efficient way of solving.

The verification problems from the Gaussian function showed that the STFEM can be applied for SLM. The convergence studies for *1D* and *2D* proved the expected convergence of order p within the examples. For a polynomial degree $p = 3$, different stability issues could be seen. For some discretizations, the error did not behave as expected from the convergence results. In particular, for the time refinement, a significantly higher error was observed for odd numbers of elements. However, the explicit refinement in the time direction indicated that for $p = 1$ and $p = 2$ the convergence within time is of order p.

A comparison of the linear, implicit FEM and the STFEM showed that the solution for the STFEM was more accurate and achieved faster within a shared memory context. The non-symmetric, large matrix did not have a negative influence on the solution time within the examples.

The STFEM was derived and implemented for the latent heat equation. The verification procedure indicated that the method was implemented successfully, but the obtained results were more diffusive than expected. In addition, stability issues for some discretizations were noted. In the one-dimensional bar, the material melted faster than the analytic solution and in the freezing of the two-dimensional plate example, the solidification of the liquid was faster than the reference solution. In addition, nonphysical behaviour due to oscillations occurred in the *2D* plate. For higher order polynomials, the oscillations were reinforced.

9.3 Outlook

At last, the outlook is presented, as there are still some open topics to discuss. Future research could focus on the element ratio, the time scale and the solving process. In addition, the results should be compared to other high performance computing PDE libraries to obtain a better overview of the performance. This should include more complex problem formulations and more applications. Additional equations, such as the wave equation, could also be investigated. This could include implementing the third space dimension.

9.3.1 Element ratio and anisotropic elements

A mindful reader may have already noticed that for the verification process, even though the respective subdomains have changed, the ratio between the length of space and time for the elements was mostly even. Hence, trying different discretizations may lead to better or worse approximations. As the framework is capable of using different orders of basis functions per direction, an investigation of anisotropic elements within space and time could lead to interesting results. Not only a comparison to low order time schemes could be enabled, but also more stable schemes for nonlinear problems could be derived. Especially, the oscillations within the freezing plate could vanish with lower time orders.

9.3.2 Solver

Even though the AdhoC++ framework has been further developed for symmetric positive definite problems, the choice of non-symmetric solvers and direct solvers is quite large. In the previous sections, it was shown that the GMRES and direct solvers are capable of obtaining solutions in a reasonable amount of time. However, the implemented solvers did not provide preconditioners like the multigrid methods for non-symmetric solvers. Since the resulting problems are generally ill-conditioned, algebraic multigrid methods could improve the solution process. In addition, the direct solvers could be optimized for parallel applications.

9.3.3 Benchmark

With enhanced solving methods and preconditioners, the STFEM could be compared to the FEM using CG and algebraic multigrid methods. This comparison would state interesting results with focus on the current state of the art methods. The obtained results of a benchmark example should be compared to the implementation within other PDE libraries as `deal.II`.

9.3.4 Time scale

The provided examples were restricted to smaller time scales. Many applications need much larger time scales, as in the verification examples. Therefore, the behaviour of the long term evolution of STFEM should be investigated with respect to accuracy. The circular travelling Gaussian function could be used as a starting point to investigate the phase error within STFEM. In terms of parallel aspects, this could include enhancing the slab implementation within the parallel implementation.

9.3.5 Space dimension

As in the current framework only a maximum of two space dimensions can be represented, the number of applications is restricted. Many applications require a three-dimensional space representation to provide the insights. Hence, implementing a fourth dimension within *AdhoC*++ will be needed for benchmarking and applications. This could be done by inventing a method to apply the time elements on an existing space discretization explicitly or including an existing multidimensional mesh generator to *AdhoC*++. In addition, the corresponding algorithmic changes must be maintained within *AdhoC*++.

List of Figures

List of Tables

Bibliography

[1] Christie Alappat, Achim Basermann, Alan R. Bishop, Holger Fehske, Georg Hager, Olaf Schenk, Jonas Thies, and Gerhard Wellein. A recursive algebraic coloring technique for hardware-efficient symmetric sparse matrix-vector multiplication. *ACM Trans. Parallel Comput.*, 7(3), June 2020.

[2] Aliprantis, Charalambos D, and Border Kim C. *Infinite Dimensional Analysis: a Hitchhiker's Guide*. Springer, Berlin; London, 2006.

[3] Martin S. Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. The fenics project version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[4] Matthew Anderson and Jung-Han Kimn. A numerical approach to space-time finite elements for the wave equation. *Journal of Computational Physics*, 226(1):466–476, September 2007.

[5] Daniel Arndt, Wolfgang Bangerth, Bruno Blais, Thomas C. Clevenger, Marc Fehling, Alexander V. Grayver, Timo Heister, Luca Heltai, Martin Kronbichler, Matthias Maier, Peter Munch, Jean-Paul Pelteret, Reza Rastak, Ignacio Thomas, Bruno Turcksin, Zhuoran Wang, and David Wells. The `deal.II` library, version 9.2. *Journal of Numerical Mathematics*, 28(3):131–146, 2020.

[6] Czesław I. Bajer and Bartłomiej Dyniewicz. *Numerical Analysis of Vibrations of Structures under Moving Inertial Load*. Springer Berlin Heidelberg, 2012.

[7] R.M. Barrett, Berry MW, T. Chan, Demmel JW, Donato JM, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Chris Romine, and Henk Van der Vorst. *TEMPLATES for the Solution of Linear Systems: Building Blocks for Iterative Methods*, volume 43. 01 1994.

[8] Klaus J. Bathe. *Finite Element Procedures*. K. J. Bathe, Watertown, MA, second edition, June 2014.

[9] Klaus-Jürgen Bathe and Mohammad R. Khoshgoftaar. Finite element formulation and solution of nonlinear heat transfer. *Nuclear Engineering and Design*, 51(3):389–401, 1979.

[10] Peiying Bian, Xiaodong Shao, and Jingli Du. Finite element analysis of thermal stress and thermal deformation in typical part during SLM. *Applied Sciences*, 9(11):2231, May 2019.

[11] Michele Bici, Salvatore Brischetto, Francesca Campana, Carlo Giovanni Ferro, Carlo Seclì, Sara Varetti, Paolo Maggiore, and Andrea Mazza. Development of a multifunctional panel for aerospace use through slm additive manufacturing. *Procedia CIRP*, 67:215–220, 2018. 11th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 19-21 July 2017, Gulf of Naples, Italy.

[12] M. Blatt, A. Burchardt, A. Dedner, Ch. Engwer, J. Fahlke, B. Flemisch, Ch. Gersbacher, C. Gräser, F. Gruber, Ch. Grüninger, D. Kempf, R. Klöfkorn, T. Malkmus, S. Müthing, M. Nolte, M. Piatkowski, and O. Sander. The Distributed and Unified Numerics Environment, Version 2.4. *Archive of Numerical Software*, 4(100):13–29, 2016.

[13] E. Boman, Ümit V. Catalyürek, C. Chevalier, and K. Devine. The zoltan and isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering and coloring. *Sci. Program.*, 20:129–150, 2012.

[14] M. Buch, A. Idesman, R. Niekamp, and Stein E. Finite elements in space and time for parallel computing of viscoelastic deformation. *Comput. Mech.*, 24:386–395, 1999.

[15] Lin Cheng and Gregory J. Wagner. An optimally-coupled multi-time stepping method for transient heat conduction simulation for additive manufacturing. *Computer Methods in Applied Mechanics and Engineering*, 381:113825, August 2021.

[16] Bernardo Cockburn, Karniadakis, George E., and Chi-Wang Shu. *Discontinuous Galerkin Methods - Theory, Computation and Applications*. Springer, Berlin; London, 2000.

[17] L. Crivelli and S. Idelsohn. A temperature-based finite element solution for phase-change problems. *International Journal for Numerical Methods in Engineering*, 23:99–119, 1986.

[18] Stephen H. Davis. *Theory of Solidification*. Cambridge University Press, October 2001.

[19] V. Dolejší and M. Feistauer. *Discontinuous Galerkin Method - Analysis and Applications to Compressible Flow*. Springer, Berlin; London, 2015.

[20] J.R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[21] Truman Everett Ellis. *Space–time discontinuous Petrov–Galerkin finite elements for transient fluid mechanics*. Phd thesis, TX: University of Texas at Austin, 2016.

[22] Carlos A. Felippa, Qiong Guo, and K. C. Park. Mass matrix templates: General description and 1d examples. *Archives of Computational Methods in Engineering*, 22(1):1–65, May 2014.

[23] Message P Forum. Mpi: A message-passing interface standard. Technical report, USA, 1994.

[24] A. Galántai. The theory of newton's method. *Journal of Computational and Applied Mathematics*, 124(1-2):25–44, December 2000.

[25] Steven Michael Hadfield and Timothy A. Davis. *On the Lu Factorization of Sequences of Identically Structured Sparse Matrices within a Distributed Memory Environment*. PhD thesis, USA, 1994. AAI9606793.

[26] David W Hahn and M Necati Özisik. *Heat conduction*. John Wiley & Sons, 2012.

[27] Michael Heroux. Aztecoo user guide for version 3.6. Sand report, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California.

[28] H. D. Hibbitt and B. I. Karlsson. Analysis of pipe whip. [PWR; BWR]. Technical report, November 1979.

[29] T. J. R. Hughes and W. K. Liu. Implicit-explicit finite elements in transient analysis: Stability theory. *Journal of Applied Mechanics*, 45(2):371–374, June 1978.

[30] Thomas J.R. Hughes and Gregory M. Hulbert. Space-time finite element methods for elastodynamics: Formulations and error estimates. *Computer Methods in Applied Mechanics and Engineering*, 66(3):339–363, 1988.

[31] Gregory M. Hulbert and Thomas J.R. Hughes. Space-time finite element methods for second-order hyperbolic equations. *Computer Methods in Applied Mechanics and Engineering*, 84(3):327–348, 1990.

[32] Frank P. Incropera and David P. DeWitt. *Fundamentals of Heat and Mass Transfer*. John Wiley & Sons, Inc., New York City, New York, 4th edition edition, 1996.

[33] Dilip Sahebrao Ingole, Abhay Madhusudan Kuthe, Shashank B. Thakare, and Amol S. Talankar. Rapid prototyping – a technology transfer approach for development of rapid tooling. *Rapid Prototyping Journal*, 15(4):280–290, July 2009.

[34] John N. Jomo, Nils Zander, Mohamed Elhaddad, Ali Özcan, Stefan Kollmannsberger, Ralf-Peter Mundani, and Ernst Rank. Parallelization of the multi-levelhp-adaptive finite cell method. *Computers & Mathematics with Applications*, 74(1):126–142, July 2017.

[35] Nam-Ho Kim. *Introduction to Nonlinear Finite Element Analysis*. Springer US, 2015.

[36] S. Kollmannsberger, A. Özcan, M. Carraturo, N. Zander, and E. Rank. A hierarchical computational model for moving thermal loads and phase changes with applications to selective laser melting. *Computers & Mathematics with Applications*, 75(5):1483–1497, March 2018.

[37] Stefan Kollmannsberger, Massimo Carraturo, Alessandro Reali, and Ferdinando Auricchio. Accurate prediction of melt pool shapes in laser powder bed fusion by the

non-linear temperature equation including phase changes. *Integrating Materials and Manufacturing Innovation*, 8(2):167–177, April 2019.

[38] Sandeep Koranne. Hierarchical data format 5: Hdf5. In *Handbook of Open Source Tools*, pages 191–200. Springer, 2011.

[39] Anastas Lazaridis. A numerical solution of the multidimensional solidification (or melting) problem. *International Journal of Heat and Mass Transfer*, 13(9):1459–1477, 1970.

[40] Yingli Li, Kun Zhou, Pengfei Tan, Shu Beng Tor, Chee Kai Chua, and Kah Fai Leong. Modeling temperature and residual stress fields in selective laser melting. *International Journal of Mechanical Sciences*, 136:24–35, 2018.

[41] Martin Neumüller. *Space–Time Methods: Fast Solvers and Applications*. Phd thesis, Austria: TU Graz, 2013.

[42] Deepankar Pal, Nachiket Patil, Khalid Haludeen Kutty, Kai Zeng, Alleyce Moreland, Adam Hicks, David Beeler, and Brent Stucker. A generalized feed-forward dynamic adaptive mesh refinement and derefinement finite-element framework for metal laser sintering—part II: Nonlinear thermal simulations and validations. *Journal of Manufacturing Science and Engineering*, 138(6), January 2016.

[43] E. L. Papazoglou, N. E. Karkalos, and A. P. Markopoulos. A comprehensive study on thermal modeling of SLM process under conduction mode using FEM. *The International Journal of Advanced Manufacturing Technology*, 111(9-10):2939–2955, November 2020.

[44] Dario Pitassi, Enrico Savoia, Vigilio Fontanari, Alberto Molinari, Valerio Luchin, Gianluca Zappini, and Matteo Benedetti. Finite element thermal analysis of metal parts additively manufactured via selective laser melting. In *Finite Element Method - Simulation, Numerical Analysis and Solution Techniques*. InTech, February 2018.

[45] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986.

[46] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[47] Boris Schaeling. *The Boost C++ Libraries*. XML Press, 2011.

[48] F. Schieweck. A-stable discontinuous galerkin–petrov time discretization of higher order. 18(1):25–57, 2010.

[49] Christoph Schwab and Rob Stevenson. Space–time adaptive wavelet methods for parabolic evolution problems. *Math. Comput.*, 78:1293–1318, 09 2009.

[50] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. August 1994.

[51] Michael Smith. *ABAQUS/Standard User's Manual, Version 6.9*. Dassault Systèmes Simulia Corp, United States, 2009.

[52] Denis Spiridonov, Maria Vasilyeva, and Wing Tat Leung. A generalized multiscale finite element method (GMsFEM) for perforated domain flows with robin boundary conditions. *Journal of Computational and Applied Mathematics*, 357:319–328, September 2019.

[53] Olaf Steinbach. Space-time finite element methods for parabolic problems. *Computational Methods in Applied Mathematics*, 15(4):551–566, 2015.

[54] Olaf Steinbach and Huidong Yang. Comparison of algebraic multigrid methods for an adaptive space-time finite-element discretization of the heat equation in 3d and 4d. *Numerical Linear Algebra with Applications*, 25(3):e2143, January 2018.

[55] The Trilinos Project Team. *The Trilinos Project Website*.

[56] Ben Vandenbroucke and Jean-Pierre Kruth. Selective laser melting of biocompatible metals for rapid manufacturing of medical parts. *Rapid Prototyping Journal*, 13(4):196–203, August 2007.

[57] Shogo Wada, Rui Zhang, Seetha R. Mannava, Vijay K. Vasudevan, and Dong Qian. Simulation-based prediction of cyclic failure in rubbery materials using nonlinear space-time finite element method coupled with continuum damage mechanics. *Finite Elements in Analysis and Design*, 138:21–30, January 2018.

[58] Hayden Liu Weng. Partitioned hierarchical space-time fem for transient heat problems. Master's thesis, Technical University of Munich, 3 2019.

[59] Nils Zander. *Multi-level hp-FEM: dynamically changing high-order mesh refinement with arbitrary hanging nodes*. Dissertation, Technical University of Munich, 2017.

[60] Nils Zander, Tino Bog, Mohamed Elhaddad, Felix Frischmann, Stefan Kollmannsberger, and Ernst Rank. The multi-level hp-method for three-dimensional problems: Dynamically changing high-order mesh refinement with arbitrary hanging nodes. *Computer Methods in Applied Mechanics and Engineering*, 310:252–277, 2016.

[61] Rui Zhang, Lihua Wen, Jinyou Xiao, and Dong Qian. An efficient solution algorithm for space–time finite element method. *Computational Mechanics*, 63(3):455–470, July 2018.

[62] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. The finite element method: Its basis and fundamentals. In *The Finite Element Method: its Basis and Fundamentals (Seventh Edition)*, page iii. Butterworth-Heinemann, Oxford, seventh edition edition, 2013.