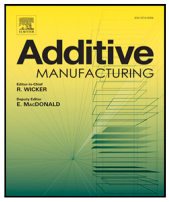
Additive Manufacturing 92 (2024) 104380

Contents lists available at ScienceDirect

Additive Manufacturing

journal homepage: www.elsevier.com/locate/addma



Research paper

A highly efficient computational approach for fast scan-resolved microstructure predictions in metal additive manufacturing on the scale of real parts

Sebastian D. Proell [a,*], Julian Brotz [a], Martin Kronbichler [b], Wolfgang A. Wall [a], Christoph Meier [a]

[a] Institute for Computational Mechanics, Technical University of Munich, Garching b. München, 85748, Germany
[b] Faculty of Mathematics, Ruhr University Bochum, Bochum, 44780, Germany

ARTICLE INFO

Keywords:
Powder bed fusion additive manufacturing
Part-scale
Microstructure
Multiscale modeling
Performance modeling

ABSTRACT

In metal additive manufacturing (AM), fast and efficient simulation approaches are essential to explore the full potential of these promising processes, particularly in generating components with tailored microstructures via laser powder bed fusion (LPBF). Due to the inherent multiscale nature of LPBF, existing approaches often need to resort to strong simplifications, such as layer-wise heating models, to make part-scale simulations feasible. In contrast, the present article proposes a scan-resolved approach, which consistently resolves the laser scan path in a coupled thermo-microstructural model of LPBF. Building on a high-performance computing model for the thermal problem, we propose a highly efficient implementation of a recently developed microstructure model for Ti-6Al-4V with three main constituents: stable α_s-phase, martensitic α_m-phase and β-phase. The implementation is tailored to modern hardware features using vectorization and fast approximations of transcendental functions. A performance model and selected numerical examples of LPBF manufacturing of parts on the centimeter scale are studied to verify the high degree of optimization. Depending on the specific example, results were obtained with moderate computational resources in a few hours to days. We demonstrate how the proposed scan-resolved model allows us to predict the correlation between scan strategy and resulting microstructure composition, an aspect that layer-wise heating models cannot capture. The numerical examples include scan-resolved thermo-microstructure simulations of the full NIST AM Benchmark cantilever specimen. It is shown that varying the build plate temperature by only 100 K can significantly change the microstructure composition from α_m- to α_s-dominated.

1. Introduction

Laser powder bed fusion (LPBF) is a prominent additive manufacturing (AM) technique that allows the design and production of parts with complex geometry in a near-net-shape manner. However, a successful build can require expensive trial-and-error runs beforehand or the adoption of overly conservative parameter choices. One crucial aspect for the quality of produced parts is its material behavior and a critical insight into this is provided by the microstructure [1]. Understanding its evolution through numerical simulations offers the opportunity to significantly reduce the costs measured in time and money and enhance the physical understanding of the LPBF process. While subsequent heat treatment often changes the as-built microstructure, the information is highly relevant during processing, as significant differences in material behavior can lead to defects and even failure of the part (e.g., due to cracking) during the process. Ultimately, a great promise lies in the local control of the microstructural phase composition as the specific application desires.

From a numerical modeling point of view, LPBF is a multi-scale and multi-physics problem [2,3]. In the present contribution, we focus on the effects on the microscale, specifically the composition and evolution of microstructure phases for the commonly used alloy Ti-6Al-4V. The considered microstructure model was proposed in our previous contribution [4]. Based on the classification in [5], it falls in the category of *phenomenological* [6–10] microstructure models, which rely on equations describing the relation between phase fractions and process variables. An alternative approach is provided by *statistical* models [11,12], sometimes called *probabilistic* models [13]; this category includes the widely used cellular automaton approach [14–17]. Finally, there are *mechanistic* models which are based directly on fundamental physics, most prominently *phase-field* [18,19] models. A

* Corresponding author.
E-mail address: sebastian.proell@tum.de (S.D. Proell).

https://doi.org/10.1016/j.addma.2024.104380
Received 8 May 2024; Received in revised form 12 July 2024; Accepted 22 August 2024
Available online 26 August 2024

phenomenological microstructure model provides a reasonable trade-off between the costly evaluation of a phase-field model and the sometimes limited physical motivation of purely statistical models. A significant difference between [4] and other phenomenological models is that the evolution of the microstructural phases by means of diffusive and instantaneous transformations is governed by temperature-dependent forces driving the phase composition towards an equilibrium state. From a mathematical perspective, the model consists of coupled ordinary differential equations (ODEs) for the three phases, which are the β-phase, stable α_s-phase, and metastable martensitic α_m-phase. This model was recently validated with experimental data in [20].

To the best of the authors' knowledge, no coupled thermo-microstructure simulation with scan-resolved tracks on the part-scale has been published before. Nevertheless, some groups have undertaken efforts in this direction. In [21,22], the authors determined correlations between experimentally measured microstructural phase compositions and numerical results for the temperature field of larger parts without an explicit microstructure model. In [23], the authors inform a high-fidelity phase-field model for microstructure evolution with a single-track melt pool simulation. Their model includes detailed effects such as microsegregation and solidification front morphology. These details are neglected in the cellular automata microstructure model used in [15], which aims at part-scale predictions by replicating the thermal information from a few representative layers and tracks over multiple layers. A similar strategy is used in [24] and combined with a phase-field model for the sub-grain scale. Many authors integrate analytical Johnson–Mehl–Avrami–Kolmogorov (JMAK) equations into thermal or thermo-mechanical models [6,7,9,10], which can be substantially cheaper to evaluate. In our publication [4], we determined the microstructure for application-motivated temperature profiles at selected points and for a quenching example of a large block. In contrast to all the cited references, the present article presents a fully coupled thermo-microstructure model that considers the resolved laser scan track and is not restricted to regular geometries. The microstructure is determined in the entire domain for all points in time.

An essential aspect of macroscale simulations is the question of the performance of the implementation in terms of time to solution. For the coupled thermo-microstructure problem, the thermal history drives the evolution of the microstructure. Thus, a fast and accurate solution to the thermal problem is necessary. In our previous work [25], we presented a highly efficient solution to the thermal problem with a resolved laser scan track over hundreds of layers. In contrast to many existing approaches, we can perform scan-resolved simulations of parts on the centimeter scale in a time frame on the order of hours to days. The present contribution builds on that work and allows us to predict the composition of microstructure phases in the same setting with only marginally increased time to solution. We carefully analyzed the many conditional branches in the governing equations to achieve high performance for the microstructure model. The implementation approach is tailored to modern hardware capabilities as it utilizes vectorization efficiently and considers the need to reduce memory transfer as much as possible. Efficient approximations of transcendental functions [26–28], e.g., the exponential function, which can be vectorized efficiently, are discussed. To demonstrate the degree of optimization, we present a detailed performance analysis with the help of a roofline performance model. The proposed efficient vectorized implementation is responsible for more than a three-fold increase in computational throughput compared to an unvectorized implementation. In total, the microstructure model only requires about 10% of the time of the well-optimized thermal model, making large-scale simulations of the coupled problem possible. Based on selected numerical examples, we demonstrate how the proposed scan-resolved model allows us to predict the correlation between scan strategy and resulting microstructure composition. Frequently employed simplified approaches, such as layer-wise heating models, cannot capture this aspect. The numerical examples include scan-resolved thermo-microstructure simulations of the full NIST AM Benchmark cantilever specimen. As one immediately interesting result from the application of our proposed novel approach, it is shown that varying the build plate temperature by only 100 K can significantly change the microstructure composition from α_m- to α_s-dominated.

The article is structured as follows: after briefly reviewing the thermal and microstructure model, we focus on the numerical discretization and implementation details of the latter. Specifically, we discuss the implementation tailored to modern hardware and propose efficient approximations for expensive transcendental functions. We study the implementation performance on benchmarks and application examples. The investigated, practically relevant examples demonstrate a wide range of applicability of the approach and fast solution times.

2. Coupled thermo-microstructure model

This section summarizes the model equations underlying the proposed computational approach. An emphasis is placed on details especially relevant to the efficient solution strategy presented later in this article. We refer to our respective publications for the full details of the thermal [25] and the microstructure [4] model.

2.1. Thermal model

First, we briefly summarize the thermal part of the problem following [25] and our previous works [29,30]. The temperature field T is determined in the domain Ω by solving the heat equation:

$$\rho c \frac{\partial T}{\partial t} = -\nabla \cdot \boldsymbol{q} + q_{\text{vol}}, \quad \boldsymbol{q} = -k(T)\nabla T \quad \text{in } \Omega. \tag{1}$$

Here, ρ is the density and c is the specific heat capacity of the material. The temperature and state-dependent heat conductivity k can be computed from the liquid fraction $g(T)$, defined as

$$g(T) = \begin{cases} 0, & T < T_s, \\ \frac{T - T_s}{T_l - T_s}, & T_s \le T \le T_l, \\ 1, & T > T_l, \end{cases} \tag{2}$$

where T_s and T_l are the solidus and liquidus temperature. The time-dependent consolidated fraction

$$r_c(t) = \begin{cases} 1, & \text{if } r_c(0) = 1 \text{ (initially consolidated)}, \\ \max\limits_{\tilde{t}<t} g(T(\tilde{t})), & \text{if } r_c(0) = 0 \text{ (initially powder)}, \end{cases} \tag{3}$$

captures the irreversible powder-to-melt transition and allows setting the initial material state. From (2) and (3), the actual fractions of powder (p), melt (m) and solid (s) material are computed as

$$r_p(r_c) = 1 - r_c, \quad r_m(T) = g(T), \quad r_s(T, r_c) = r_c - g(T), \tag{4}$$

and finally, the temperature- and history-dependent heat conductivity $k(T, r_c)$ is found:

$$k(T, r_c) = r_p(r_c)k_p + r_m(T)k_m + r_s(T, r_c)k_s, \tag{5}$$

where k_p, k_s and k_m are the parameters for a single state. The actual implementation can also handle temperature-dependent parameters. To keep the discussion concise and in light of the uncertainty associated with the measurement of material parameters, especially at high temperatures, this dependency is neglected. A volumetric heat source q_{vol} formulated in a local coordinate system $(\hat{x}, \hat{y}, \hat{z})$ models the incident energy from a moving laser beam:

$$q_{\text{vol}} = \begin{cases} \frac{2W_{\text{eff}}}{\pi R^2 h_{\text{powder}}} \exp\left(\frac{-2(\hat{x}^2+\hat{y}^2)}{R^2}\right), & \text{if } 0 < \hat{z} < -h_{\text{powder}} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Here, R is the effective beam radius of the incident energy beam, W_{eff} is the effective power and h_{powder} is the powder layer thickness. The presented mathematical model for the thermal problem neglects

Table 1
Thermal model parameters for Ti-6Al-4V.

Symbol	Property	Value	Unit
k_{ms}	Thermal conductivity in melt and solid state	28.6	$\mathrm{W\,m^{-1}\,K^{-1}}$
k_p	Thermal conductivity in powder state	0.286	$\mathrm{W\,m^{-1}\,K^{-1}}$
ρ	Density	4090	$\mathrm{kg\,m^{-3}}$
c	Specific heat capacity	1130	$\mathrm{J\,kg^{-1}\,K^{-1}}$
T_s	Solidus temperature	1878	K
T_l	Liquidus temperature	1928	K
T_∞	Ambient temperature	293	K
ϵ	Emissivity	0.7	–
T_v	Boiling temperature	3130	K
C_P	Recoil pressure factor	54	kPa
C_T	Recoil pressure temperature factor	5.07×10^{4}	K
C_M	Heat loss temperature factor	9.15×10^{-4}	$\mathrm{K\,s^{2}\,m^{-2}}$
M	Molar mass	0.0478	$\mathrm{kg\,mol^{-1}}$
h_v	Latent heat of evaporation	8.84	$\mathrm{MJ\,kg^{-1}}$
$T_{h,0}$	Enthalpy reference temperature	538	K

melt pool hydrodynamics, which would play an important role, e.g., in determining the orientation of microstructure grains [1,16]. This seems acceptable as the presented approach aims at a coarser representation of microstructural phase fractions, which does not resolve such features.

The necessary initial and boundary conditions for the heat equation (1) are given as:

$$T = T_0 \quad \text{in } \Omega \text{ for } t = 0, \tag{7}$$

$$T = T_0 \quad \text{on } \Gamma_D, \tag{8}$$

$$\boldsymbol{q}\cdot\boldsymbol{n} = 0 \quad \text{on } \Gamma_N, \tag{9}$$

$$\boldsymbol{q}\cdot\boldsymbol{n} = q_{\mathrm{rad}} + q_{\mathrm{evap}} \quad \text{on } \Gamma_{RE}, \tag{10}$$

$$q_{\mathrm{rad}} = \epsilon\sigma_S(T^4 - T_\infty^4), \tag{11}$$

$$q_{\mathrm{evap}} = 0.82 C_P \exp\left[-C_T\left(\frac{1}{[T]} - \frac{1}{T_v}\right)\right] \times \sqrt{\frac{C_M}{[T]}}\,(h_v + c([T] - T_{h,0})), \text{ if } [T] > T_v. \tag{12}$$

Here, T_0 is the initial temperature. The material parameters required for the initial boundary value problem are listed in Table 1. In addition, σ_S in (11) is the Stefan–Boltzmann constant governing radiative heat loss. To avoid numerical challenges arising from the strong nonlinearity in the evaporation term (12), the temperature $[T]$ is limited to a maximum value $T_{\max} > T_v$. In this study, we opt for $T_{\max} = T_v + 1000\,\mathrm{K}$, which ensures numerical stability without affecting the overall results.

2.2. Microstructure model

A phenomenological model for the microstructure evolution of Ti-6Al-4V was presented in our previous work [4]. For the details of this model, the interested reader is referred to this work. The main model equations are briefly summarized in the following for better comprehensibility.

The microstructure model focuses on the three most important phases,[1] β, α_s and α_m, of the solid state. The equations presented in the following are only valid for the solid state and, thus, for temperatures $T < T_{\mathrm{sol}}$. In a first step, we define phase fractions $X_i \in [0;1]$ for the microstructure phases along with elementary continuity constraints:

$$X_\alpha + X_\beta = 1, \tag{13}$$

$$X_{\alpha_s} + X_{\alpha_m} = X_\alpha. \tag{14}$$

[1] Note that the word 'phase' refers to the microstructure phases α_s, α_m, and β. In contrast, when we distinguish material into powder, melt, and solid, we speak of the 'state' of the material.

Before the evolution equations for the phase fractions can be defined, we introduce (pseudo-) equilibrium phase fractions for the different phases. Thereto, we consider material cooling down from a molten state to ambient temperature T_∞. All newly solidified solid material consists entirely of β-phase. On further cooling, the β-phase can transform into α-phase: either into stable α_s-phase via a diffusion-based transformation or into metastable α_m-phase via an instantaneous transformation, depending on the cooling rate. It is assumed that the equilibrium phase fraction for the total α-phase, the sum of α_s and α_m, can be described by the following exponential Koistinen–Marburger law:

$$X_\alpha^{\mathrm{eq}}(T) = \begin{cases} 0.9 & \text{for } T < T_{\alpha_s,\mathrm{end}}, \\ 1 - \exp[-k_\alpha^{\mathrm{eq}}(T_{\alpha_s,\mathrm{start}} - T)] & \text{for } T_{\alpha_s,\mathrm{end}} \le T \le T_{\alpha_s,\mathrm{start}}, \\ 0 & \text{for } T > T_{\alpha_s,\mathrm{start}}. \end{cases} \tag{15}$$

In our previous work [4], the parameters k_α^{eq}, $T_{\alpha_s,\mathrm{start}}$ and $T_{\alpha_s,\mathrm{end}}$ have been inversely identified on basis of experimental data. Their values are listed in Table 2. Notably, for general cooling conditions, the actual fraction of the total α-phase X_α at a given temperature is not identical to the equilibrium value in (15). Instead, the equilibrium value can be interpreted as the long-term solution for $t \to \infty$. In the limiting case of very slow cooling, (15) represents the actual α-phase fraction. More precisely, no α_m-phase forms under such thermodynamic equilibrium conditions, and the temperature-dependent α_s-phase fraction is given by (15). In the second limiting case of very fast cooling, diffusion-based transformations are inhibited, and no stable α_s-phase forms. Instead, only metastable martensite α_m-phase arises according to the following, pseudo-equilibrium phase fraction $X_{\alpha_m,0}^{\mathrm{eq}}$:

$$X_{\alpha_m,0}^{\mathrm{eq}}(T) = \begin{cases} 0.9 & \text{for } T < T_\infty \\ 1 - \exp[-k_{\alpha_m}^{\mathrm{eq}}(T_{\alpha_m,\mathrm{start}} - T)] & \text{for } T_\infty \le T \le T_{\alpha_m,\mathrm{start}} \\ 0 & \text{for } T > T_{\alpha_m,\mathrm{start}}. \end{cases} \tag{16}$$

Again, the parameters $k_{\alpha_m}^{\mathrm{eq}}$ and $T_{\alpha_m,\mathrm{start}}$ are listed in Table 2. Fig. 1 shows a schematic representation of the phase composition resulting from the two limiting cases of very slow and very fast cooling. It is emphasized that the formation of α_m in the limit of very fast cooling begins at temperatures $T < T_{\alpha_m,\mathrm{start}}$, which lie significantly below the temperature interval $T_{\alpha_s,\mathrm{end}} \le T \le T_{\alpha_s,\mathrm{start}}$ where α_s would form in the limiting case of very slow cooling. Finally, general cooling conditions shall be considered between the two aforementioned limiting cases. In this scenario, the cooling rates are too fast to complete the diffusion-driven formation of α_s but too slow to completely inhibit α_s-formation before reaching the start temperature $T_{\alpha_m,\mathrm{start}}$ of martensite formation. Assuming that a certain amount X_{α_s} of α_s-phase is already present, (16) is corrected by postulating an *effective* pseudo-equilibrium martensite phase fraction $X_{\alpha_m}^{\mathrm{eq}}(T)$:

$$X_{\alpha_m}^{\mathrm{eq}}(T) = X_{\alpha_m,0}^{\mathrm{eq}}(T) \cdot \frac{0.9 - X_{\alpha_s}}{0.9}. \tag{17}$$

Thus, for general cooling conditions, (17) represents the target value for the α_m-phase fraction X_{α_m}, while (15) represents the target value for the total α-phase fraction $X_\alpha = X_{\alpha_s} + X_{\alpha_m}$.

Based on these temperature-dependent target values, the time-dependent formation and dissolution of the three phases α_s, α_m, and β will be described below. The balance of the different transformation contributions is given by the following ordinary differential equations:

$$\dot{X}_{\alpha_s} = \dot{X}_{\beta\to\alpha_s} + \dot{X}_{\alpha_m\to\alpha_s} - \dot{X}_{\alpha_s\to\beta}, \tag{18}$$

$$\dot{X}_{\alpha_m} = \dot{X}_{\beta\to\alpha_m} - \dot{X}_{\alpha_m\to\alpha_s} - \dot{X}_{\alpha_m\to\beta}, \tag{19}$$

$$\dot{X}_{\beta} = \dot{X}_{\alpha_m\to\beta} + \dot{X}_{\alpha_s\to\beta} - \dot{X}_{\beta\to\alpha_m} - \dot{X}_{\beta\to\alpha_s}, \tag{20}$$

where $\dot{X}_{i\to j} > 0$ is the formation of phase j from phase i or, equivalently, the dissolution from phase i into phase j. In general, $\dot{X}_{i\to j} \neq \dot{X}_{j\to i}$ holds since the reverse transformation may be governed by different kinetics or might not even exist (e.g., for the transformation from

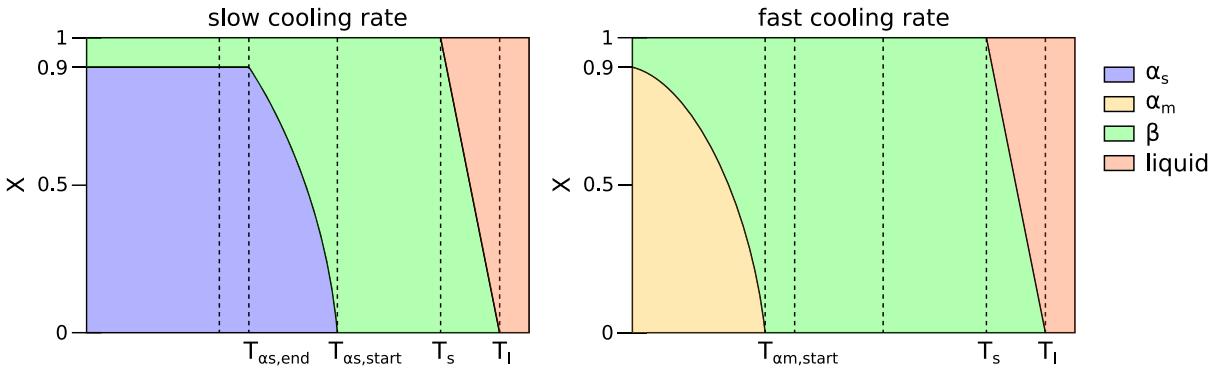


Fig. 1. Phase composition for slow and fast cooling rates. Left: Equilibrium composition of X_{α_s} and X_β for slow cooling rates, implying $X_{\alpha_m} = 0$. Right: Metastable (pseudo-) equilibrium composition of X_{α_m} and X_β for fast cooling rates, implying $X_{\alpha_s} = 0$. Simplified schematic reproduced from [4].

Table 2
Parameters of the microstructure evolution model. A detailed analysis and literature review of all parameters is given in [4].

Parameter	Description	Value	Unit
T_l	Liquidus temperature	1928	K
T_s	Solidus temperature	1878	K
$T_{\alpha_s,\text{start}}$	Upper end of temperature range for $\beta \to \alpha_s$ transformation	1273	K
$T_{\alpha_s,\text{end}}$	Lower end of temperature range for $\beta \to \alpha_s$ transformation	935	K
$T_{\alpha_m,\text{start}}$	Upper end of temperature range for $\beta \to \alpha_m$ transformation	848	K
T_∞	Ambient temperature	293	K
k_α^{eq}	α_s-phase equilibrium concentration constant	0.0068	K^{-1}
$k_{\alpha_m}^{\text{eq}}$	α_m-phase equilibrium concentration constant	0.00415	K^{-1}

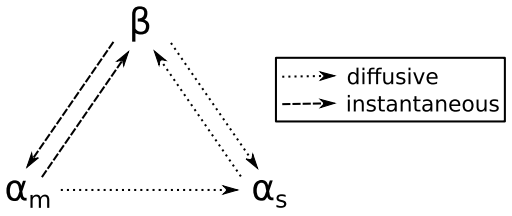


Fig. 2. Possible transformation paths between different microstructure phases.

martensite to stable α_s phase). Note also that the sum of (18)–(20) yields $\dot{X}_{\alpha_s} + \dot{X}_{\alpha_m} + \dot{X}_\beta = 0$, which is consistent with the continuity constraints (13) and (14). From these continuity constraints, the β-phase fraction can directly be determined as

$$X_\beta = X_{\text{sol}} - X_{\alpha_s} - X_{\alpha_m}, \tag{21}$$

making time integration of (20) unnecessary. A graphical overview of the overall transformation processes is shown in Fig. 2. In the following, diffusion-based and instantaneous transformations are distinguished. In particular, the transformations between the β- and the α_s-phase as well as the dissolution of α_m- into α_s-phase are considered to be diffusion-controlled. These diffusion-based transformations are modeled by logistic differential equations:

$$\dot{X}_{\beta\to\alpha_s} = \begin{cases} k_{\alpha_s}(T)\left(X_{\alpha_s}\right)^{\frac{c_{\alpha_s}-1}{c_{\alpha_s}}}\left(X_\alpha^{\text{eq}} - X_\alpha\right)^{\frac{c_{\alpha_s}+1}{c_{\alpha_s}}} & \text{if } X_\alpha < X_\alpha^{\text{eq}}, \\ 0 & \text{otherwise,} \end{cases} \tag{22}$$

$$\dot{X}_{\alpha_m\to\alpha_s} = \begin{cases} k_{\alpha_s}(T)\left(X_{\alpha_s}\right)^{\frac{c_{\alpha_s}-1}{c_{\alpha_s}}}\left(X_{\alpha_m}\right)^{\frac{c_{\alpha_s}+1}{c_{\alpha_s}}} & \text{if } X_{\alpha_m} > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{23}$$

$$\dot{X}_{\alpha_s\to\beta} = \begin{cases} k_\beta(T)\left(0.9 - X_\alpha\right)^{\frac{c_\beta-1}{c_\beta}}\left(X_\alpha - X_\alpha^{\text{eq}}\right)^{\frac{c_\beta+1}{c_\beta}} & \text{if } X_\alpha > X_\alpha^{\text{eq}}, \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

These are completed by the following temperature-dependent diffusion rates,

$$k_{\alpha_s}(T) = \frac{k_1}{1 + \exp[-k_3(T - k_2)]} \quad \text{and} \quad k_\beta(T) = f \cdot k_{\alpha_s}(T). \tag{25}$$

In total, this diffusion model contains six parameters, c_{α_s}, c_β, k_1, k_2, k_3 and f. Their values have been inversely identified via experimental data (see [4]) and are summarized in Table 3. The formation of α_m out of β and the dissolution of α_m into β are modeled as instantaneous transformations using Karush–Kuhn–Tucker (KKT) conditions:

$$\begin{aligned} & X_{\alpha_m} - X_{\alpha_m}^{\text{eq}} \geq 0 \quad \wedge \\ & \dot{X}_{\beta\to\alpha_m} \geq 0 \quad \wedge \quad (X_{\alpha_m} - X_{\alpha_m}^{\text{eq}}) \cdot \dot{X}_{\beta\to\alpha_m} = 0 \end{aligned} \tag{26}$$

$$\begin{aligned} \text{If} \quad X_{\alpha_m} > 0: \quad & X_\alpha - X_\alpha^{\text{eq}} \leq 0 \quad \wedge \\ & \dot{X}_{\alpha_m\to\beta} \geq 0 \quad \wedge \quad (X_\alpha - X_\alpha^{\text{eq}}) \cdot \dot{X}_{\alpha_m\to\beta} = 0 \end{aligned} \tag{27}$$

When rapidly cooling the material such that $X_\alpha < X_\alpha^{\text{eq}}$ (i.e., increasing pseudo-equilibrium phase fraction $X_{\alpha_m}^{\text{eq}}$ for $\dot{T} < 0 \wedge T < T_{\alpha_m,\text{start}}$), conditions (26) ensure that X_{α_m} exactly follows the effective pseudo-equilibrium phase fraction $X_{\alpha_m}^{\text{eq}}$ by means of instantaneous martensite formation, i.e., X_{α_m} will never fall below $X_{\alpha_m}^{\text{eq}}$ (but can exceed $X_{\alpha_m}^{\text{eq}}$, e.g., when heating the material, since $\alpha_m \to \alpha_s$ dissolution is modeled as a time-delayed diffusion process). On the other hand, when heating the material at high temperatures (i.e., decreasing equilibrium phase fraction X_α^{eq} for $\dot{T} > 0 \wedge T > T_{\alpha_s,\text{end}}$), conditions (27) ensure that X_α exactly follows the pseudo-equilibrium phase fraction X_α^{eq} through instantaneous $\alpha_m \to \beta$ dissolution as long as remaining martensite is available (i.e., $X_{\alpha_m} > 0$). Thus, X_α will not exceed X_α^{eq} as long as $X_{\alpha_m} > 0$ (but can fall below X_α^{eq}, e.g., when cooling the material, since the formation of α_s out of β is modeled as a time-delayed diffusion process).

In our previous work [4], the microstructure model has been validated for general cooling scenarios from very low to very high cooling rates. However, the scenario of very rapid *heating* rates, a specific characteristic of LPBF, was not in the focus of this original work. For very rapid heating at high temperatures $T > T_{\alpha_s,\text{end}}$, the α_s-phase fraction cannot follow the decreasing value of X_α^{eq} sufficiently

Table 3
Inversely identified parameters of the microstructure evolution model. A detailed explanation of the calibration process is given in [4].

Parameter	Value	Unit
c_{α_s}	2.51	–
c_β	11.0	–
k_1	0.294	s^{-1}
k_2	850	K
k_3	0.0337	K^{-1}
f	3.8	–

fast due to the time-delayed nature of the diffusion-based $\alpha_s \rightarrow \beta$ transformation, potentially leading to a remaining α_s-phase fraction when reaching T_s (where melting starts). To avoid the complexity of modeling the melting of a mixture of two phases and also due to a lack of experimental data in this high-temperature regime, the original model of [4] is extended by the following regularization condition, ensuring a complete $\alpha_s \rightarrow \beta$ transformation before melting is initiated:

$$X_{\alpha_s} \leq 0.9 \, \frac{T_{\alpha_s,\mathrm{reg},2} - T}{T_{\alpha_s,\mathrm{reg},2} - T_{\alpha_s,\mathrm{reg},1}}, \quad \text{if} \quad T_{\alpha_s,\mathrm{reg},1} < T < T_{\alpha_s,\mathrm{reg},2}. \tag{28}$$

According to (28), this high temperature $\alpha_s \rightarrow \beta$ dissolution is modeled as an instantaneous transformation such that X_{α_s} is limited by a maximal value, decreasing linearly to zero within the temperature interval $[T_{\alpha_s,\mathrm{reg},1}; T_{\alpha_s,\mathrm{reg},2}]$ with $T_{\alpha_s,\mathrm{start}} \leq T_{\alpha_s,\mathrm{reg},1} \leq T_{\alpha_s,\mathrm{reg},2} \leq T_s$. This regularization is justified for the considered application since we are not interested in exactly resolving these intermediate microstructure compositions before melting. In this contribution, we choose $T_{\alpha_s,\mathrm{reg},1} = T_{\alpha_s,\mathrm{start}}$ and $T_{\alpha_s,\mathrm{reg},2} = T_{\alpha_s,\mathrm{start}} + 100\,\mathrm{K}$.

For demonstration purposes, an exemplary temperature history and the resulting phase composition at a single material point are shown in more detail in Appendix A. All model parameters were identified from experimental data in [4], and the model was subsequently validated. Another experimental validation showing good agreement between predictions and measurements was recently presented in [20]. Thus, we can confidently ascribe validity to this model and focus on an efficient implementation for part-scale simulations in the rest of this article.

While the microstructure model is specific to Ti-6Al-4V, analogous ODEs may be formulated for other classes of microstructure phases present in different alloys. For instance, transformations between five phases of Steel 5140 are described via JMAK equations in [10], and transformation laws for Inconel 718 are used in [31]. Since these transformations are classifiable as diffusive or instantaneous, the calibration steps for inverse parameter identification discussed in [4] could be performed on experimental data to obtain similar equations and equality constraints. Importantly, all insights into the efficient solution of the equations for Ti-6Al-4V – the focus of the present article – will transfer seamlessly to these equivalent equations for other alloys.

3. Numerical methods and efficient implementation

The thermal problem is solved with an efficient FEM implementation [25] based on fast operator evaluation [32]. Implementation is performed with the `deal.II` library [33]. Notably, we use an explicit scheme for the active laser stage where small time step sizes are necessary to obtain a continuous melt track. In the interlayer cool down stage after every layer, we use an implicit scheme allowing larger time step sizes. This approach is extended to the microstructure model where we introduce a fast explicit and a more accurate and robust implicit scheme.

The microstructure model is integrated into the existing approach as a one-way coupled problem that is solved after the thermal model in every time step. Since the microstructure model does not explicitly depend on the spatial coordinate or spatial derivatives, the problem is fully decoupled in space. Thus, the ODEs can be solved independently at every point in space. It is convenient to place the degrees of freedom (DoFs) of the microstructure model (phases X_β, X_{α_s} and X_{α_m}) at the same spatial positions as the thermal DoFs (temperature T). This choice has the advantage that no communication and interpolation is necessary to obtain the temperature to evaluate the microstructure model.

The spatial discretization is realized as an adaptive mesh. Working on a powder-filled box geometry that encloses the desired final part geometry is sufficient, as the final part geometry emerges via the consolidated fraction due to our three-state material model. When activating a new layer, more refined cells are placed in the highest active layer and a few layers beneath. The mesh can be coarsened at later stages when the material history allows for it. In [25], the only history variable that needed to be considered for the thermal model was the consolidated fraction r_c. In this contribution, the three microstructure phases also represent a material history that should not be coarsened inadvertently. Therefore, an octant consisting of eight sibling cells of equal refinement level will only be coarsened when, for every microstructure phase, all its values are sufficiently close to each other. In particular, to be a candidate for coarsening, the average values of every microstructure phase fraction may only differ by at most 0.02 across the cells in an octant. For more general details of the adaptive and growing mesh, the reader is referred to [25].

3.1. Time integration of microstructure model

To streamline the notation, the unknown fractions of stable and martensitic α-phase are collected in a state vector $\boldsymbol{m} = [X_{\alpha_s}, X_{\alpha_m}]$. The β-phase fractions can be processed for a given state and temperature as,

$$X_\beta(\boldsymbol{m}, T) = 1 - X_{\alpha_s} - X_{\alpha_m}. \tag{29}$$

The right-hand side of the differential equations (18)–(19) is split into a diffusion-based and instantaneous contribution:

$$\dot{\boldsymbol{m}} = \begin{bmatrix} \dot{X}_{\alpha_s} \\ \dot{X}_{\alpha_m} \end{bmatrix} = \underbrace{\begin{bmatrix} \dot{X}_{\beta\to\alpha_s} + \dot{X}_{\alpha_m\to\alpha_s} - \dot{X}_{\alpha_s\to\beta} \\ -\dot{X}_{\alpha_m\to\alpha_s} \end{bmatrix}}_{=:\dot{\boldsymbol{m}}_{\mathrm{diff}}} + \underbrace{\begin{bmatrix} 0 \\ \dot{X}_{\beta\to\alpha_m} - \dot{X}_{\alpha_m\to\beta} \end{bmatrix}}_{=:\dot{\boldsymbol{m}}_{\mathrm{inst}}} \tag{30}$$

Integrating (30) from a point in time t^n to a point in time $t^{n+1} = t^n + \Delta t$ yields,

$$\boldsymbol{m}^{n+1} = \boldsymbol{m}^n + \int_{t^n}^{t^{n+1}} \dot{\boldsymbol{m}}_{\mathrm{diff}} \, \mathrm{d}t + \int_{t^n}^{t^{n+1}} \dot{\boldsymbol{m}}_{\mathrm{inst}} \, \mathrm{d}t \tag{31}$$

where the superscript indicates at which point in time a quantity is evaluated. The phase fractions $\boldsymbol{m}^n$ and the temperature T^{n+1} are known. A numerical time integration scheme will approximate the first integral over the diffusion-driven contribution. Note that the second integral over the (non-differentiable) instantaneous change rate yields a finite value for the absolute change. These instantaneous changes are defined in (26) and (27).

Explicit time integration. We split (31) into a two-stage process, where integration of the diffusion-based term is performed with a forward Euler scheme towards an intermediate state $\tilde{\boldsymbol{m}}^{n+1}$, followed by a correction step:

$$\tilde{\boldsymbol{m}}^{n+1} = \boldsymbol{m}^n + \Delta t \, \dot{\boldsymbol{m}}_{\mathrm{diff}}(T^n, \boldsymbol{m}^n), \tag{32}$$

$$\boldsymbol{m}^{n+1} = \boldsymbol{h}(T^{n+1}, \tilde{\boldsymbol{m}}^{n+1}). \tag{33}$$

Here, we define a correction function $\boldsymbol{h}(T, \boldsymbol{m})$, which contains the instantaneous changes for the martensite phase as well as corrections that are necessary to satisfy the continuity constraints (13)–(14). Furthermore, if either X_{α_s} or X_{α_m} falls below zero, it is instead set to zero. If X_α exceeds the maximum equilibrium fraction of 0.9, both, X_{α_s} or X_{α_m} are reduced while maintaining the ratio $X_{\alpha_s}/X_{\alpha_m}$.

Two exceptional cases become apparent when examining (22) and (24). Both equations pose a problem for the explicit time integration

scheme presented so far. Evaluating (22) for $X_{\alpha_s} = 0, X_{\alpha_m} = 0, X_\beta = 1.0$ gives a rate of zero, which implies that a solution computed with the explicit scheme can never evolve out of the initial state. Thus, we initiate the diffusion process with the help of an approximate analytical solution described in more detail in [4]. The same strategy can be applied to (24) which suffers from the same problem for $X_{\alpha_s} = 0.9, X_{\alpha_m} = 0, X_\beta = 0.1$. Due to the numeric values of the physical constants, the second case is prone to cancellation of significant digits. An appropriate reformulation can be found in Appendix B.

The presented explicit time integration scheme does not come with a stability limit due to the correction function $\boldsymbol{h}(T, \boldsymbol{m})$ being applied after every step. Therefore, the solution cannot grow arbitrarily large, and classical stability considerations do not apply to the specific scheme used here. Still, the time step size should be chosen within limits to obtain a sufficiently accurate solution.

Implicit time integration. We use a Crank–Nicolson time integration scheme for the diffusion-based term and reuse the same correction function $\boldsymbol{h}$ as in the explicit case:

$$\tilde{\boldsymbol{m}}^{n+1} = \boldsymbol{m}^n + \frac{1}{2}\Delta t \left(\dot{\boldsymbol{m}}_{\text{diff}}(T^{n+1}, \boldsymbol{m}^{n+1}) + \dot{\boldsymbol{m}}_{\text{diff}}(T^n, \boldsymbol{m}^n)\right), \tag{34}$$

$$\boldsymbol{m}^{n+1} = \boldsymbol{h}(T^{n+1}, \tilde{\boldsymbol{m}}^{n+1}). \tag{35}$$

The nonlinear equation (35) is solved employing a fixed-point iteration:

$$\boldsymbol{m}_{i+1}^{n+1} = \boldsymbol{h}\left(T^{n+1}, \underbrace{\boldsymbol{m}^n + \frac{1}{2}\Delta t \left(\dot{\boldsymbol{m}}_{\text{diff}}(T^{n+1}, \boldsymbol{m}_i^{n+1}) + \dot{\boldsymbol{m}}_{\text{diff}}(T^n, \boldsymbol{m}^n)\right)}_{\tilde{\boldsymbol{m}}_{i+1}^{n+1}}\right), \tag{36}$$

$$\boldsymbol{m}_0^{n+1} = \boldsymbol{m}^n, \tag{37}$$

until the weighted root-mean-square (WRMS) norm of the residual

$$\boldsymbol{r}_{i+1}^{n+1} = \tilde{\boldsymbol{m}}_{i+1}^{n+1} - \left(\boldsymbol{m}^n + \frac{1}{2}\Delta t \left(\dot{\boldsymbol{m}}_{\text{diff}}(T^{n+1}, \boldsymbol{m}_{i+1}^{n+1}) + \dot{\boldsymbol{m}}_{\text{diff}}(T^n, \boldsymbol{m}^n)\right)\right) \tag{38}$$

$$= \frac{\Delta t}{2}\left(\dot{\boldsymbol{m}}_{\text{diff}}(T^{n+1}, \boldsymbol{m}_i^{n+1}) - \dot{\boldsymbol{m}}_{\text{diff}}(T^{n+1}, \boldsymbol{m}_{i+1}^{n+1})\right) \tag{39}$$

falls below a threshold of 1. The WRMS norm is defined as

$$||\boldsymbol{r}_{i+1}^{n+1}||_{\text{WRMS}} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(r_{i+1,j}^{n+1} w_j)^2}, \quad \text{where } w_j = (\varepsilon_{\text{abs}} + m_j^n \varepsilon_{\text{rel}})^{-1} \tag{40}$$

with an absolute tolerance $\varepsilon_{\text{abs}} = 1 \times 10^{-10}$ and a relative tolerance $\varepsilon_{\text{rel}} = 1 \times 10^{-3}$. This norm ensures that convergence considers the order of magnitude of different solution components via a weight w_j scaled with the solution component m_j^n of the last time step.

It can easily be verified that the iteration scheme prescribed in (36) is a contraction on $X_i \in (0, 0.9)$, and thus converges to a unique solution if the time step size Δt is sufficiently small. This proof holds for time step sizes up to around 0.001 s. However, we experimentally observe fast and robust convergence for time step sizes up to 1 s. We use a subcycling scheme to achieve sufficient accuracy, where a large time step performed in the thermal model is subdivided into substeps not exceeding a maximum size $\Delta t_{\text{sub}} = 0.01$ s. The temperature values are linearly interpolated from the solutions at two thermal time steps enclosing a subcycling step.

3.2. Vectorized computation

Modern CPUs support vector operations on specialized execution units that perform the same operation on a (small) array of different data, an idiom commonly called *Single Instruction Multiple Data* (SIMD). This small array has n_{lanes} entries and will be referred to as a *vectorized array*. A useful C++ type `VectorizedArray` overloading basic arithmetic operations is provided by the `deal.II` library [32]. Similar data types are available in the experimental C++ standard implementation [34] or as stand-alone libraries [35]. The overloaded operations simultaneously perform the arithmetic operation on all lanes by calling the respective, hardware-specific intrinsic functions. For instance, the latest Intel AVX512 instruction set architecture supports eight concurrent double-precision operations. This capability promises to significantly speed up computation-heavy code paths by a factor of n_{lanes} when fully utilized. Using SIMD operations demands contiguous data storage in memory for maximum performance benefits. Furthermore, the concurrently processed data should be independent, i.e., the computation in one vector lane may not depend on a computation in another lane of the same vector.

The classical and automatic approach to vectorization is to look at inner loops or plain code and let the compiler identify nearby operations of the same kind that could go to different lanes. The results of auto-vectorization are poor if the loop kernel contains code beyond basic arithmetic, e.g., conditional branches and transcendental function calls. Therefore, we choose to vectorize the outer loop over all points in the mesh in batches of size n_{lanes} and then solve the microstructure ODEs on a batch of points stored in the different lanes. The different quantities on a batch of points are loaded into the `VectorizedArray` data structure. This approach results in an array-of-struct-of-array layout, with the inner array being a `VectorizedArray`, which allows a concise implementation without explicitly writing the inner-most vectorized loop. However, the numerous conditional branches in the microstructure model do not always allow the same operation to be performed on all lanes. From an implementation standpoint, three distinct scenarios for the conditional branches of equations can be distinguished:

1. *All* vector lanes require evaluation of an expression. In this case, the expression is evaluated for all vector lanes using intrinsic functions.
2. *None* of the vector lanes require evaluation of an expression. In this case, the expression is not evaluated.
3. *Some* but not all of the vector lanes require the evaluation of an expression. In this case, the expression is evaluated for *all* vector lanes, but the result is only stored in *some* vector lanes that require it. Note that the unused additional computations do often not impact the evaluation time compared to an unvectorized implementation.[2]

A condition mask is computed for every branch to see which of the three scenarios is active. In the third case, the results of different branches are combined by blending the results with the condition mask. The general strategy is also visualized in Fig. 3.

The strategy outlined above only makes sense when an efficient vectorized implementation of every required (mathematical) function is available. Although it is possible to fall back to compute expressions on the vector lanes sequentially, doing this in the scenario of mixed operations can, in the worst case, lead to an *increase* in computation time on the order of $\mathcal{O}(n_{\text{lanes}})$ compared to an unvectorized implementation.[3]

Algorithm 1 outlines the overall vectorized solution procedure for the microstructure model. Starting from the solution variables, $X_{\alpha_s}^n$, $X_{\alpha_m}^n$, X_β^n and T^n, at the previous time t_n, the solution after a time increment Δt with temperature T^{n+1} is sought. We load a contiguous data slice from these five global vectors into vectorized arrays. Time integration is then performed on the vectorized arrays. Afterward, the results at time t_{n+1} are written back from the vectorized arrays into

[2] Note that wider vectors might lead to slightly lower clock frequencies on some hardware and complicated functions, like divisions or square roots, might have lower throughput when executed on wider vectors; nonetheless, the overwhelming share of operations has the same throughput for 1 or n_{lanes} results.

[3] If every lane requires a different kind of operation, $\mathcal{O}(n_{\text{lanes}})$ operations are necessary in the unvectorized case. However, when the computation is vectorized and every kind of operation is unnecessarily performed on all lanes, $\mathcal{O}(n_{\text{lanes}}^2)$ operations are necessary.

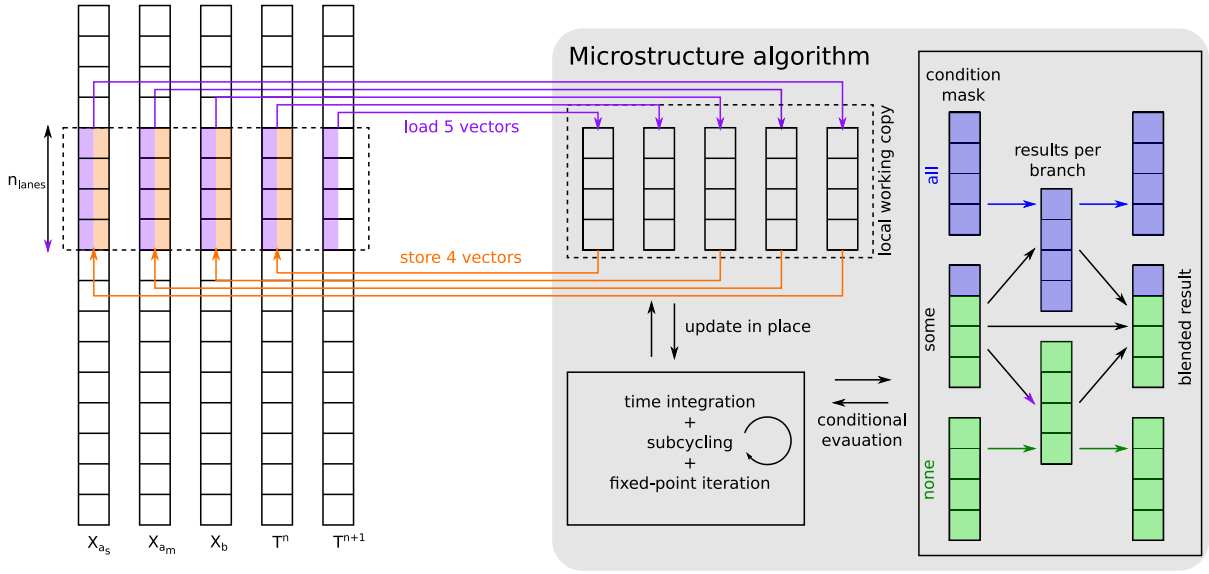


Fig. 3. Illustration of vectorized microstructure algorithm. The algorithm works simultaneously on a local working copy of n_{lanes} entries extracted from the five global data vectors. Time integration and/or fixed-point iteration and (optional) subcycling are all performed locally. All conditional computations are performed by blending the results of different conditional branches based on the condition mask. When storing data back, the additional temperature vector T^n is updated from the current temperature values T^{n+1}.

Algorithm 1: Time integration of microstructure model with vectorized data.

Data: Δt, Global vectors $X^n_{\alpha_s}$, $X^n_{\alpha_m}$, X^n_β, T^n, T^{n+1}
Result: Global vectors $X^{n+1}_{\alpha_s}$, $X^{n+1}_{\alpha_m}$, X^{n+1}_β, T^{n+1}
$i \leftarrow 0$
while $i <$ `size`(T^n) **do**
  `// Load into vectorized array`
  $\check{X}^n_{\alpha_s} \leftarrow X^n_{\alpha_s}[i : i + n_{\text{lanes}}]$
  $\check{X}^n_{\alpha_m} \leftarrow X^n_{\alpha_m}[i : i + n_{\text{lanes}}]$
  $\check{X}^n_\beta \leftarrow X^n_\beta[i : i + n_{\text{lanes}}]$
  $\check{T}^n \leftarrow T^n[i : i + n_{\text{lanes}}]$
  $\check{T}^{n+1} \leftarrow T^{n+1}[i : i + n_{\text{lanes}}]$
  `// Solve local time integration problem`
  $\check{X}^{n+1}_{\alpha_s}, \check{X}^{n+1}_{\alpha_m}, \check{X}^{n+1}_\beta \leftarrow$ `time_integration`$(\check{X}^n_{\alpha_s}, \check{X}^n_{\alpha_m}, \check{X}^n_\beta, \check{T}^n, \check{T}^{n+1}, \Delta t)$
  `// Store from vectorized array`
  $X^{n+1}_{\alpha_s}[i : i + n_{\text{lanes}}] \leftarrow \check{X}^{n+1}_{\alpha_s}$
  $X^{n+1}_{\alpha_m}[i : i + n_{\text{lanes}}] \leftarrow \check{X}^{n+1}_{\alpha_m}$
  $X^{n+1}_\beta[i : i + n_{\text{lanes}}] \leftarrow \check{X}^{n+1}_\beta$
  $T^n[i : i + n_{\text{lanes}}] \leftarrow \check{T}^{n+1}$
  $i \leftarrow i + n_{\text{lanes}}$
end

Algorithm 2: Local explicit time integration of microstructure model on vectorized data.

Data: $\check{\boldsymbol{m}}^n = [\check{X}^n_{\alpha_s}, \check{X}^n_{\alpha_m}]$, $\check{X}^n_\beta$, $\check{T}^n$, Δt
Result: $\check{X}^{n+1}_{\alpha_s}$, $\check{X}^{n+1}_{\alpha_m}$, $\check{X}^{n+1}_\beta$
$\dot{\check{\boldsymbol{m}}}^n \leftarrow$ `compute_rates`$(\check{\boldsymbol{m}}^n, \check{T}^n)$
$\check{\boldsymbol{m}}^{n+1} \leftarrow \check{\boldsymbol{m}}^n + \Delta t\, \dot{\check{\boldsymbol{m}}}$
$\check{\boldsymbol{m}}^{n+1}, \check{X}^{n+1}_\beta \leftarrow$ `instantaneous_corrections`$(\check{\boldsymbol{m}}^{n+1}, \check{X}^n_\beta, \check{T}^n)$

Algorithm 3: Local implicit time integration of microstructure model on vectorized data.

Data: $\check{\boldsymbol{m}}^n = [\check{X}^n_{\alpha_s}, \check{X}^n_{\alpha_m}]$, $\check{X}^n_\beta$, $\check{T}^n$, $\check{T}^{n+1}$, Δt
Result: $\check{X}^{n+1}_{\alpha_s}$, $\check{X}^{n+1}_{\alpha_m}$, $\check{X}^{n+1}_\beta$
$\dot{\check{\boldsymbol{m}}}^n \leftarrow$ `compute_rates`$(\check{\boldsymbol{m}}^n, \check{T}^n)$
$\check{\boldsymbol{m}}^{n+1}_0, \check{X}^{n+1}_{\beta,0} \leftarrow$ `instantaneous_corrections`$(\check{\boldsymbol{m}}^n, \check{X}^n_\beta, \check{T}^{n+1})$
$\dot{\check{\boldsymbol{m}}}^{n+1}_0 \leftarrow$ `compute_rates`$(\check{\boldsymbol{m}}^{n+1}_0, \check{T}^{n+1})$
$i \leftarrow 0$
repeat
  $\check{\boldsymbol{m}}^{n+1}_{i+1} \leftarrow \check{\boldsymbol{m}}^n + \frac{\Delta t}{2}(\dot{\check{\boldsymbol{m}}}^{n+1}_i + \dot{\check{\boldsymbol{m}}}^n)$
  $\check{\boldsymbol{m}}^{n+1}_{i+1}, \check{X}^{n+1}_{\beta,i+1} \leftarrow$ `instantaneous_corrections`$(\check{\boldsymbol{m}}^{n+1}_{i+1}, \check{X}^{n+1}_{\beta,i}, \check{T}^{n+1})$
  $\dot{\check{\boldsymbol{m}}}^{n+1}_{i+1} \leftarrow$ `compute_rates`$(\check{\boldsymbol{m}}^{n+1}_{i+1}, \check{T}^{n+1})$
  $e \leftarrow$ `weighted_root_mean_square`$(\frac{\Delta t}{2}(\dot{\check{\boldsymbol{m}}}^{n+1}_{i+1} - \dot{\check{\boldsymbol{m}}}^{n+1}_i))$
  $i \leftarrow i + 1$
until $e < 1.0$

the respective global vectors. Note that the update of the temperature vector $T^n \leftarrow T^{n+1}$ is also performed in this loop since the data is already loaded. Five load and four store operations must be performed for every evaluation point, totaling 72 bytes of memory transfer per evaluation point or 24 bytes per DoF (3 DoFs per evaluation point).

The explicit time stepping is outlined in Algorithm 2. The function `compute_rates` evaluates the diffusion-based rates (22)–(24), and the function `instantaneuous_corrections` evaluates instantaneous transformations between α_m- and β-phase. Only a few arithmetic operations are needed for every set of vectorized arrays. In contrast, the implicit solution procedure shown in Algorithm 3 usually requires fixed-point iteration and, thus, at least twice as many arithmetic operations as the explicit step. However, the amount of loaded and stored data is equal. As we will demonstrate in the examples, this typically leads to the explicit time integration scheme being memory-bound and the implicit scheme being compute-bound.

3.3. Efficient approximation of transcendental functions

The evolution equations of the microstructure model contain a few terms that necessitate the computation of a (non-integer) power or exponential function. Note that the power of a positive number can equivalently be written as

$$a^x = \exp(x \ln a), \quad a > 0, \tag{41}$$

which allows to compute the power of a number via a natural logarithm and an exponential function. At the time of writing, the C++ standard did not offer an implementation of these functions that could leverage SIMD hardware support. While copying and adapting the sophisticated implementations from the standard library or wrapping an existing library supporting vectorized data types would, in theory, be possible, we want to follow a different strategy here. The C++ standard implementation and most other libraries [36] must deal with a wide range of applications and consequently are implemented to be accurate up to machine precision. However, in the context of a numerical method, we accept a much less accurate result than machine precision. The chosen strategy embeds this fact by allowing the definition of an approximation that is accurate enough while minimizing the number of arithmetic operations.

The strategy employed in this work assumes the ubiquitous IEEE-754 standard [37] to represent floating point numbers. In this standard, a real number is represented as

$$(-1)^s 2^{p-b}(1+m). \tag{42}$$

For a 64-bit representation, i.e., the `double` data type in C/C++, the bias is set to $b = 1023$, the sign s consumes a single bit, the exponent p (an integer) consumes 11 bit, and the mantissa m (a binary fraction) consumes 52 bit. The idea of directly computing and manipulating the bitwise representation was initially brought up in [28]. Here, we follow the refined implementation discussed in [26,27]. Let us first find an approximation to the exponential function $z_{\exp}$ by rewriting,

$$z_{\exp} := \exp(x) = 2^{x \log_2(e)} = 2^y = 2^{y_i} 2^{y_f}, \tag{43}$$

where $y_i = \lfloor x \log_2(e) \rfloor$ is an integer,[4] and $y_f = x \log_2(e) - y_i \in [0, 1)$ is a rational number. By comparing (43) to (42), we find that

$$s = 0, \tag{44}$$

$$2^{y_i} = 2^{p-b}, \tag{45}$$

$$2^{y_f} = 1 + m. \tag{46}$$

The sign bit is always zero, as expected for exponentiation. The exponent p can directly be computed from (45) as

$$p = y_i + b. \tag{47}$$

By rearranging (46) and introducing a correction function $\mathcal{K}_{\exp}(y_f)$, we write:

$$2^{y_f} = 1 + m = 1 + y_f - \mathcal{K}_{\exp}(y_f), \quad \text{with} \quad \mathcal{K}_{\exp}(y_f) = 1 + y_f - 2^{y_f}, \tag{48}$$

which leads to the mantissa

$$m = y_f - \mathcal{K}_{\exp}(y_f). \tag{49}$$

The correction function $\mathcal{K}_{\exp}(y_f)$ is replaced with a polynomial approximation to circumvent the need to compute a (non-integer) power of 2. Note that this is the only approximation that is performed for the computation of the exponential function. The polynomial approximation can be tailored to the required accuracy by polynomial interpolation or regression. In this contribution, we use a least-squares fit of a polynomial of degree 7 to data sampled on 1000 equidistant points in $[0, 1)$. The resulting coefficients are listed in Table 4. Tests revealed that adding more sample points does not increase the accuracy of the approximation in a relevant manner for our application. In contrast to [27], we do not consider the derivatives of $\mathcal{K}_{\exp}(y_f)$ in the computation of the coefficients as we found their impact negligible.

[4] The floor function $\lfloor x \rfloor$ returns the largest integer not exceeding x.

Table 4
Polynomial coefficients for the approximated correction functions $\mathcal{K}_{\exp}(y_f)$ and $\mathcal{K}_{\ln}(y_f)$.

i	a_i in $\mathcal{K}_{\exp}(y_f) = \sum_{i=0}^{n} a_i y_f^i$	b_i in $\mathcal{K}_{\ln}(y_f) = \sum_{i=0}^{n} b_i y_f^i$
0	$1.213\,071\,811\,889 \times 10^{-10}$	$1.847\,756\,720\,962\,93 \times 10^{-10}$
1	$3.068\,528\,102\,657 \times 10^{-1}$	$1.442\,695\,040\,841\,32$
2	$-2.402\,263\,423\,993\,59 \times 10^{-1}$	$-0.721\,347\,520\,143\,005$
3	$-5.550\,533\,134\,149\,54 \times 10^{-2}$	$0.480\,898\,345\,526\,187$
4	$-9.613\,524\,328\,848\,3 \times 10^{-3}$	$-0.360\,675\,000\,332\,004$
5	$-1.342\,884\,759\,630\,84 \times 10^{-3}$	$0.288\,048\,466\,919\,235$
6	$-1.431\,317\,444\,835\,89 \times 10^{-4}$	$-0.235\,306\,287\,368\,882$
7	$-2.159\,565\,612\,634\,9 \times 10^{-5}$	$0.183\,102\,904\,829\,435$
8		$-0.120\,996\,268\,979\,3$
9		$0.059\,150\,381\,159\,211\,3$
10		$-0.018\,114\,949\,248\,998\,9$
11		$0.002\,544\,886\,756\,057\,43$

Since we determined the sign bit $s = 0$, the exact exponent $p = y_i + b$, and the approximate mantissa $m = y_f - \mathcal{K}_{\exp}(y_f)$ of $z_{\exp}$ in (43), all that remains to be done is filling their bitwise representations into the IEEE754 conforming layout. An elegant way to achieve this can be derived by filling a 64-bit integer (`int64`) with the values of s, p, and m and then interpreting the result as a (64-bit) `double` value. A graphical depiction of the approach is shown in Fig. 4. The exponent p – an integer stored inside a `double` representation – is converted into the equivalent `int64` format and then multiplied with 2^{52}, shifting it 52 bits to the left. Multiplying the mantissa $m < 1$ by 2^{52} shifts the decimal point behind the last (binary) digit, thus making it an integer, which is then converted into the equivalent `int64` format. In C++, the conversion from `double` to `int64` is achieved by a `static_cast<int64>` operation. The two `int64` numbers derived from p and m have no overlapping non-zero bits. They are added together to yield a combined `int64` with the bitwise representation of $z_{\exp}$ when reinterpreted as a `double` number (using `reinterpret_cast<double>` in C++). Programmatically speaking, this algorithm can be written as:

$$z_{\exp} \leftarrow \texttt{reinterpret\_cast}\langle\texttt{double}\rangle(2^{52} \times \texttt{static\_cast}\langle\texttt{int64}\rangle(p) + \texttt{static\_cast}\langle\texttt{int64}\rangle(2^{52} \times m)) \tag{50}$$

For an integer stored inside a `double` representation, we may flip the ordering of multiplication with another integer and a `static_cast<int64>` to integer format. Thus, we can rewrite (50) as,

$$z_{\exp} \leftarrow \texttt{reinterpret\_cast}\langle\texttt{double}\rangle(\texttt{static\_cast}\langle\texttt{int64}\rangle(2^{52} \times (p + m))), \tag{51}$$

and with (47) and (49) after rearranging as

$$y \leftarrow x \log_2(e) \tag{52}$$

$$y_f \leftarrow y - \lfloor y \rfloor \tag{53}$$

$$z_{\exp} \leftarrow \texttt{reinterpret\_cast}\langle\texttt{double}\rangle(\texttt{static\_cast}\langle\texttt{int64}\rangle(A \times (y - \mathcal{K}_{\exp}(y_f)) + B)) \tag{54}$$

The coefficients $A = 2^{52}$, $B = 2^{52} \cdot 1023$ and $\log_2(e)$ can be precomputed. The operations required in (52)–(54) are multiplication, addition, flooring, and type conversions. All of these are typically available in an instruction set for vector extensions. Thus, it is straightforward to implement (52)–(54) using SIMD in a given instruction set architecture.

The same ideas can be applied to derive a fast, vectorized approximation of the natural logarithm $\ln(x)$. We perform a change of basis and insert the IEEE754 double representation (42) to obtain:

$$z_{\ln} = \ln(x) = \ln(2) \log_2(x) = \ln(2) \log_2 \left[2^{p-b}(1+m)\right]$$

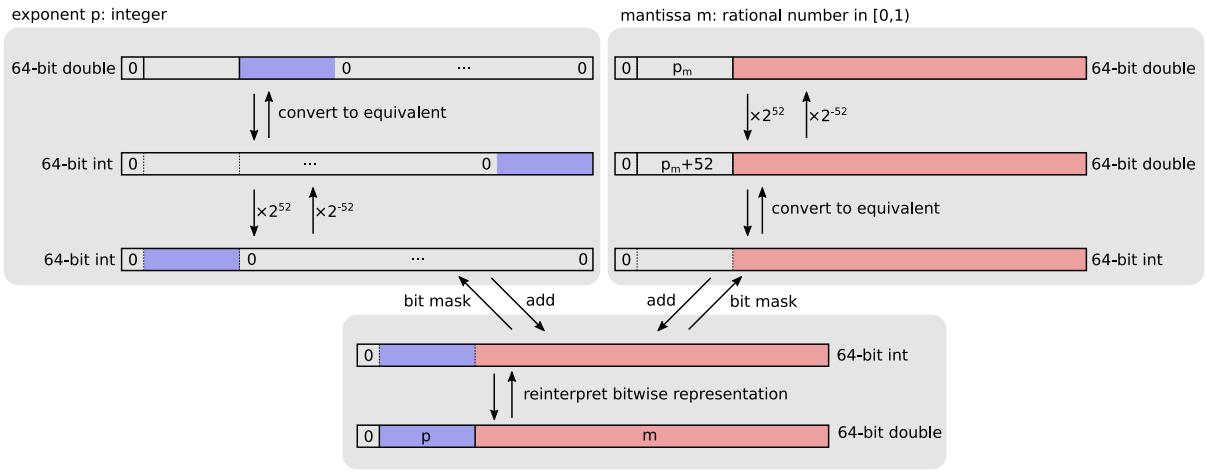


Fig. 4. Following the arrows from top to bottom illustrates how to synthesize an IEEE754 `double` representation from a separate exponent p (blue) and mantissa m (red). Following the arrows from bottom to top shows the inverse operation, namely a decomposition of a `double` representation into exponent and mantissa. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$= \ln(2)[\underbrace{(p-b)}_{l_i} + \underbrace{\log_2(1+m)}_{l_f}]. \tag{55}$$

Again, we identify an integer part l_i and a fractional part l_f. The integer part l_i is the exponent in the `double` representation of x. The fractional part l_f is once more replaced by a correction function $\mathcal{K}_{\ln}(m) \approx \log_2(1+m)$ which takes the mantissa m of x as an argument. The exact form is replaced with an approximated polynomial of degree 11 determined via a least-squares fit of 1000 sample points in the relevant interval $m \in [0, 1)$. The coefficients are given in Table 4. The exponent $p - b$ and the mantissa m are extracted from x by bit manipulations. Note that these operations are the inverse of the operations performed to synthesize a `double`, as shown in Fig. 4 when following the arrows from the bottom to the top. Again, (55) is straightforward to implement with SIMD instructions.

To evaluate polynomials $\mathcal{K}_{\exp}$ of degree 7 and $\mathcal{K}_{\ln}$ of degree 11, we make use of Estrin's scheme [38]. Although this scheme requires more floating point operations than the classical Horner scheme, it is better suited for the two separate fused multiply-add (FMA) units typically available on modern hardware since it allows independent computation of terms, thus shortening dependency chains. Applying this technique to the polynomial approximation of the correction function $\mathcal{K}_{\exp}$ yields,

$$\mathcal{K}_{\exp}(y_f) = (((a_7 y_f + a_6)y_f^2 + (a_5 y_f + a_4))y_f^4 + ((a_3 y_f + a_2)y_f^2 + (a_1 y_f + a_0))), \tag{56}$$

where pairs of parentheses group expressions that can be computed by an FMA instruction.

4. Numerical examples

The numerical examples are run on compute nodes consisting of two Intel Xeon Gold 6230 CPUs (total of 40 cores per compute node) running at 2.1 GHz with a measured peak performance of 2.5 TFlops/s and a DAXPY memory bandwidth of 162 GByte/s. The DAXPY benchmark updates a vector $\boldsymbol{y}$ in place according to $\boldsymbol{y} \leftarrow \boldsymbol{y} + \alpha \boldsymbol{x}$, which, from a memory access perspective, is close to the microstructure model. A second setup consists of compute nodes with two AMD EPYC 9354 CPUs (total of 64 cores per compute node) running at 3.25 GHz with a measured peak performance of 3.75 TFlops/s and a DAXPY memory bandwidth of 617 GByte/s. All performance measurements are conducted and analyzed with the `LIKWID` suite [39].

4.1. Performance analysis

As a first numerical study, we investigate the microstructural model implementation in isolation. A significant challenge for a vectorized implementation of the microstructure equations lies in the conditional branches. As outlined above, we rely on the fact that spatially close material points (in the physical domain) likely need the same treatment. Such points are often located next to each other in a global vector of unknowns arising from spatial discretization. Three different synthetic sets of test data are used. The first test layout consists of alternating data, so neighboring entries in global vectors require different conditional branches in the microstructure model. The second consists of contiguous blocks of 10 entries, and the third consists of contiguous blocks of 100 entries. The throughput, defined as the number of DoFs divided by the solution time, is shown for these three block sizes and the explicit and implicit time integration scheme in Fig. 5. Most importantly, the throughput increases for all block sizes when increasing the vectorization width n_{lanes}. This holds for explicit and implicit time integration schemes. This behavior is achieved by implementing the expensive exponential and power functions to exploit SIMD instructions. Without this implementation, one would observe increased computation times and lower throughput for high vectorization widths at small block sizes. Due to unavoidable conditional computations in the case of block size 1, the performance improvement when increasing the vectorization width cannot scale perfectly here. For the Intel hardware, the flattening of the curve above 6 GDoF/s is in good agreement with the maximum theoretically possible throughput of $(162\,\text{GByte/s})/(24\,\text{Byte/DoF}) = 6.75\,\text{GDoF/s}$ for a fully memory-bound code.

Fig. 6 shows a roofline plot of the tested data layouts. For increasing vectorization width, data points move upwards and thus closer to the hardware limits. The explicit scheme quickly saturates the maximum memory bandwidth for larger block sizes. This explains why no speedup is visible when increasing the vectorization width from four to eight in Fig. 5: the implementation is memory-bound, and higher vectorization widths cannot speed up the computation if memory transfer is the bottleneck. An improvement can only be expected if data transfer is minimized further or more work is performed for loaded data. One way to achieve this is subcycling, which performs multiple time steps on the same loaded data in place and only stores the result of the last subcycle. However, in a practical setting, the microstructure model already outperforms the thermal model by a significant factor, so further optimization in that direction is not investigated here. For block size 1, we observe a growing arithmetic intensity for increasing vectorization

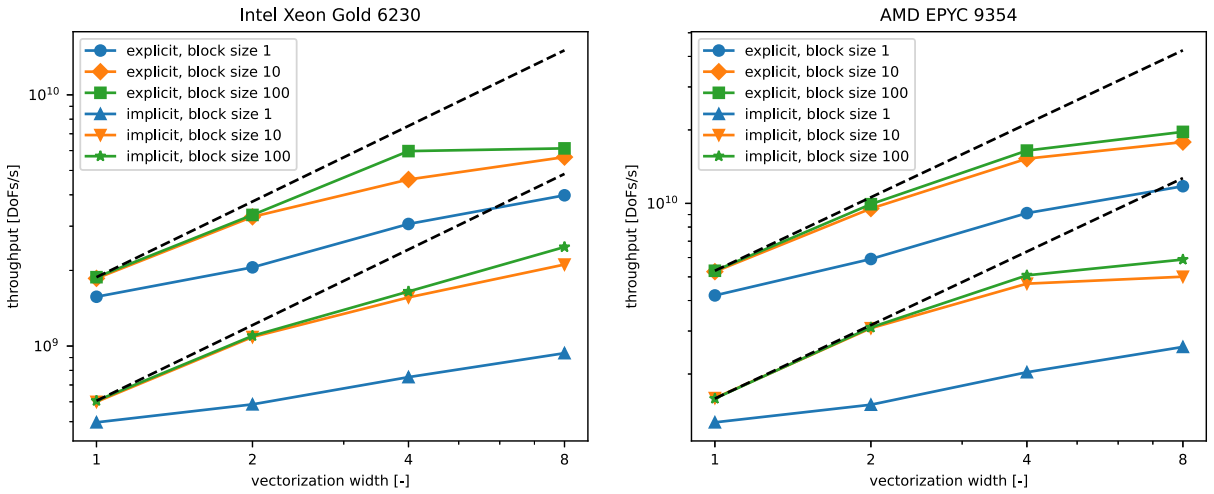


Fig. 5. Impact of vectorization width on throughput of the microstructure model solved with an explicit or implicit scheme on Intel and AMD hardware.

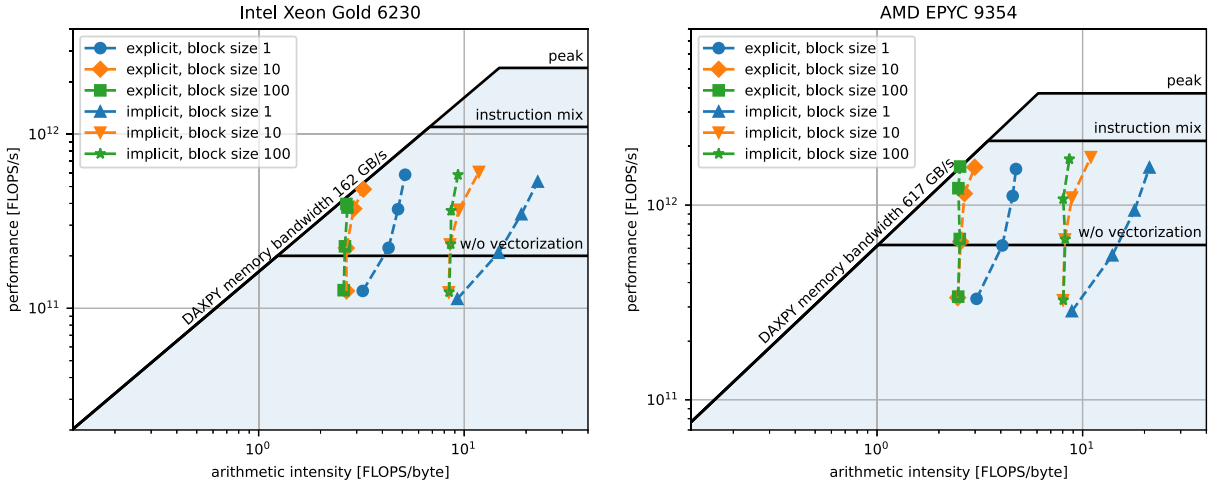


Fig. 6. Roofline plot of a single time step performed in the microstructure model with an explicit or implicit scheme at different data block sizes. Data points move upwards along the dashed lines for increasing vectorization widths.

width. In this case, multiple branches must be computed in all lanes, although the result is only stored in some lanes. For larger block sizes, it becomes less likely to require different branch evaluations on a vectorized array, and thus, the number of (unproductive) arithmetic operations decreases.

Furthermore, we analyzed the generated machine code for the implicit scheme with the machine code analyzer of the LLVM project [40]. Analyzing the instruction mix reveals that the achievable performance is limited by dependency chains and expensive instructions that occupy the same execution ports as the productive floating point operations. In particular, executing the necessary instructions to compute diffusion from β- to α_s-phase without any branching or load/store instructions reveals that only 43% of the work performed on the two relevant execution ports of the Intel hardware contributes to the floating point operation metric. For the AMD hardware, the equivalent factor is 57%. Therefore, we introduce an application-specific roofline into the roofline plots, where the (unrealistic) peak performance is scaled down by the respective factor for the instruction mix. Fig. 6 shows that the actual implementation achieves 58% of the instruction mix performance on Intel Gold hardware and 82% on AMD EPYC.

4.2. Cube geometry

In this example, we simulate manufacturing a $1 \times 1 \times 1\,\mathrm{cm}^3$ cube from 200 powder layers with thickness $0.05\,\mathrm{mm}$, which is placed on a base plate of dimensions $1.2 \times 1.2 \times 1\,\mathrm{cm}^3$ (see also Fig. 9). This is discretized as a cuboid geometry of dimensions $1.2 \times 1.2 \times 2\,\mathrm{cm}^3$, where the lower half constitutes the base plate, and the upper half initially consists of powder into which the cube geometry is scanned. Thus, a thin $0.1\,\mathrm{cm}$ powder band will remain around the final cube geometry. The origin of the coordinate system is located at the corner of the manufactured cube such that the cube lies fully in the positive octant. The processing parameters are summarized in Table 5. A time step size of $\Delta t_{\text{active}} = 2 \times 10^{-5}\,\mathrm{s}$ is used in the active laser stage taking around $1.3\,\mathrm{s}$ of physical time. After every layer, a cool down stage of $1\,\mathrm{s}$ is simulated. The first 2000 steps of this stage are simulated with the same time step size as the active stage. Afterward, the macroscopic time step size is doubled after every ten time steps for the thermal model up to a maximum step size of $0.1\,\mathrm{s}$. The microstructure model uses the identical time step size as the thermal model if the step size is less than or equal

Table 5
Processing parameters of cube example.

Parameter	Description	Value	Unit
v_{scan}	Laser scan speed	960	$\mathrm{mm\,s^{-1}}$
d_h	Hatch spacing	0.08	mm
W_{eff}	Effective laser power	180	W
R	Laser beam radius	0.06	mm
h_p	Powder layer thickness	0.05	mm

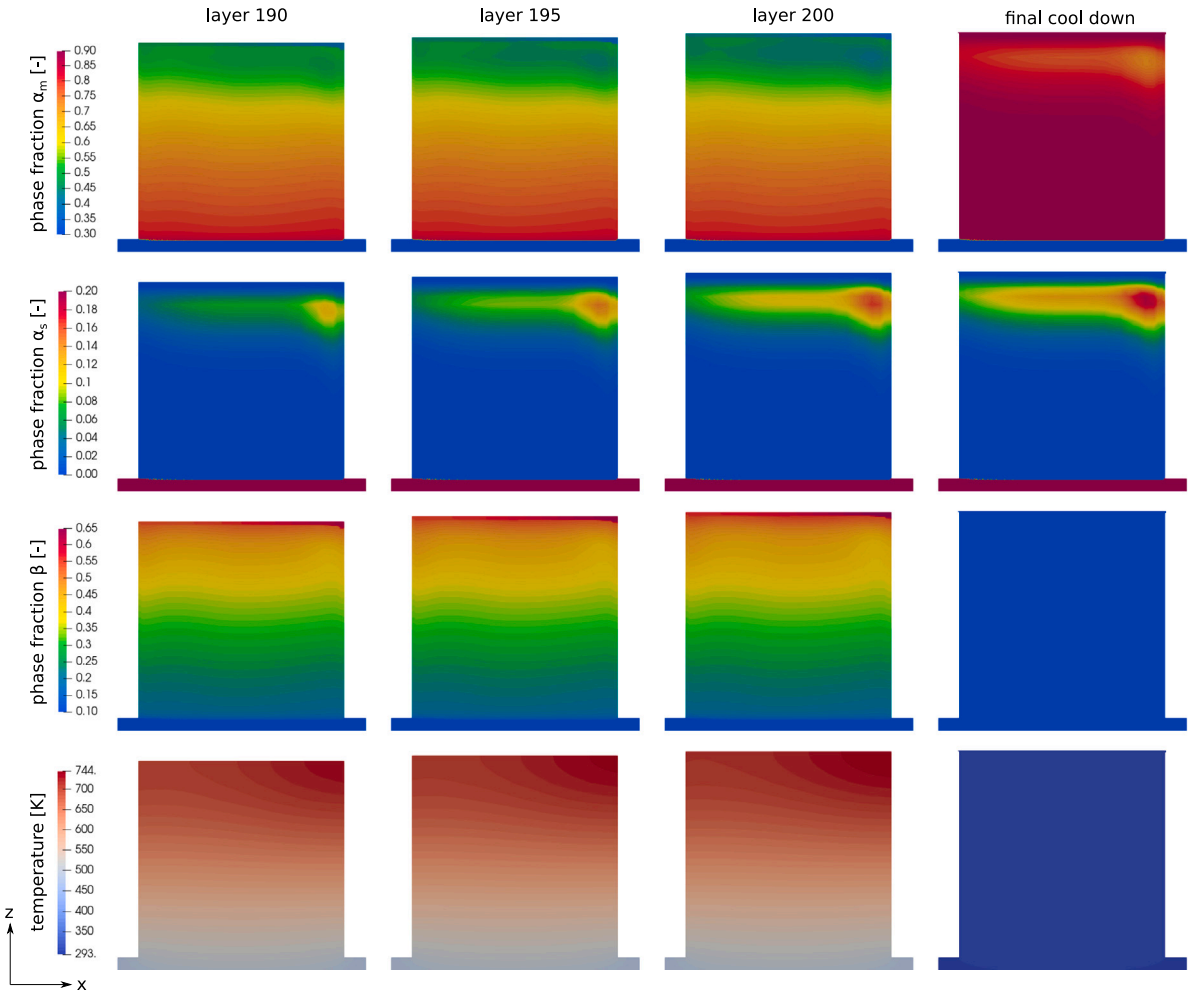


Fig. 7. Phase fractions and residual temperature after processing and cooling of layers 190, 195, 200, and after a final cool down of the cube geometry. Results are depicted in a vertical slice at $y = 5\,\text{mm}$ and the base plate is cropped.

to $1 \times 10^{-2}\,\text{s}$ and, otherwise, uses the subcycling technique described earlier. After processing the last layer, the geometry is cooled to room temperature over 100 s. Overall, this numerical example requires the solution of 13 million time steps. The vectorization width is $n_{\text{lanes}} = 8$.

As a first variant of this example, every layer is processed as a single island consisting of a continuous serpentine track, where the laser beam moves in y-direction and hatching proceeds in x-direction. The three phases and the residual temperature after layers 190, 195, and 200 have been processed are shown in Fig. 7. A noticeable asymmetry can be seen in the higher layers. Due to the long and continuous serpentine track, heat accumulates as the track hatches progress in positive x-direction. This leads to lower cooling rates and decreased martensite formation and, instead, to the formation of a band of stable α_s phase in layers 170 to 190. This happens because the residual temperature lies barely below the martensite start temperature in higher layers. The material is held at elevated temperatures, giving enough time for the diffusion-driven formation of stable α_s-phase. Due to the time required for the underlying diffusion process, this α_s-phase formation happens a few layers below the currently processed layer. The observed band structure of the α_s-phase distribution agrees well with results reported in [41]. Note that the final cool down stage is essential to obtain the actual phase composition close to room temperature. During this stage, the already formed stable α_s-phase remains. In the highest layers, the cooling rate is now large enough because no heat is added above layer 200. Consequently, most of the β-phase remaining after processing of layer 200 transforms into α_m-phase. At room temperature, the β-phase fraction reaches its equilibrium value of 10%.

The cube test geometry is well-suited to study the effects of different scan strategies on the resulting microstructural composition. In addition to the single island scan track consisting of a continuous serpentine track, we investigate a scan track split into four disjoint islands, each consisting of a serpentine track. The tracks are shown in Fig. 8 along with the stable α_s- and martensite α_m-phase fractions after the final cool down in a horizontal slice at $z = 8\,\text{mm}$. A strong asymmetry is visible in the in-plane distribution of the α_s- and α_m-phase fractions for the single-island scan strategy. On the other hand, the four-island scan strategy produces a more homogeneous distribution with a higher average martensite fraction than the single-island scan. The different phase distribution is caused exclusively by the scan strategy, as all other parameters are identical. This observation again shows the need for scan-resolved models, such as the one presented in this work.

To get a better idea of the different scales involved in a resolved part-scale simulation, a view of the adaptive mesh is shown in Fig. 9. The heights of the smallest cells correspond to one powder layer thickness. To capture the geometry, the mesh must stay more refined close to the surface of the cube. Alternatively, one could employ a boundary-fitted mesh instead of the unfitted powder box mesh for this simple geometry to save DoFs, as done in our previous work [25].

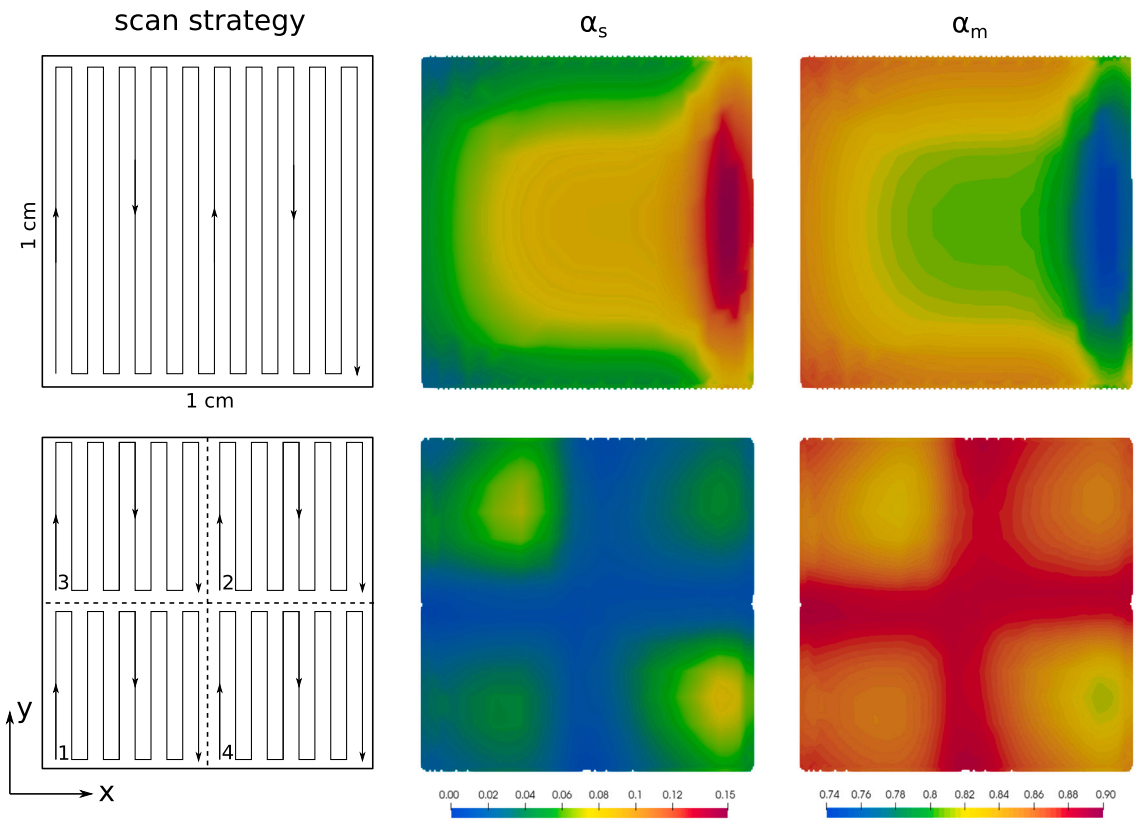


Fig. 8. Final phase fraction of stable α_s and martensite α_m for cube example in a horizontal slice at $z = 8$ mm. Two different scan strategies are used to manufacture the cube: the first row shows the results for a continuous serpentine track extending over the cross-section. The second row shows the results for a scan path split into four islands, each containing a serpentine track.

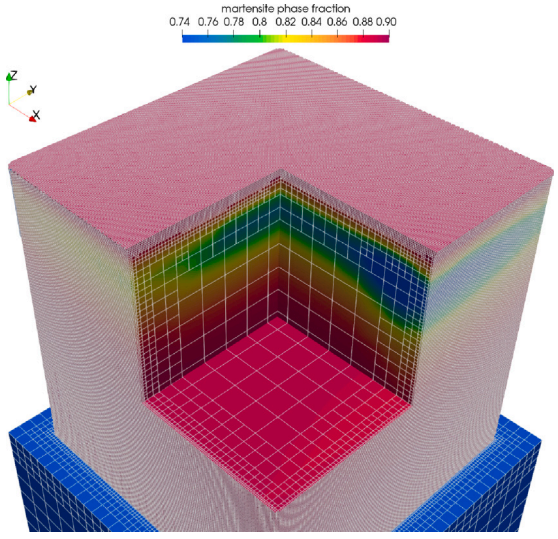


Fig. 9. Final phase fraction of martensite α_m for cube example with single island scan track. An octant is cut out to show the asymmetric distribution of martensite induced by the asymmetric scan track. An adaptively refined mesh, here shown in the last time step, captures the built geometry.

Finally, we also want to judge the performance of our implementation in this more practical example. The scenario with the single-island scan strategy is simulated in a strong scaling study on 1, 2, 4, and 8 nodes of the Intel Gold hardware mentioned in the introduction to this section connected by an Infiniband FDR (56 Gbit/s) interconnect. The resulting average time per time step, throughput, and parallel efficiency are shown in Fig. 10 for the thermal and microstructure models separately. Here, parallel efficiency is defined as:

$$\text{parallel efficiency} = \frac{\text{reference compute time} \times \text{reference number of cores}}{\text{scaled compute time} \times \text{scaled number of cores}} \tag{57}$$

Overall, the explicit and implicit solution of the microstructure equations requires around 10% of the time of the respective thermal model, details depending on the exact number of CPU cores used. Note that the microstructure problem carries three times the number of DoFs as the thermal problem; thus, the throughput is further increased compared to the thermal problem. The microstructure implementation exhibits excellent parallel scalability, even in the first layer. This result is to be expected since no communication between processes is involved. The total wall time when running the example on eight Intel Gold nodes is 2.32 h.

Table 6
Processing parameters of cantilever example.

Parameter	Description	Value	Unit
v_{scan}	Laser scan speed	960	mm s^{-1}
d_h	(Approximate) hatch spacing	0.11	mm
W_{eff}	Effective laser power	180	W
R	Laser beam radius	0.06	mm
h_p	Powder layer thickness	0.04	mm

4.3. Cantilever

As a last example, we investigate the well-known NIST AMBench cantilever geometry [42]. The geometry, shown in Fig. 11, features

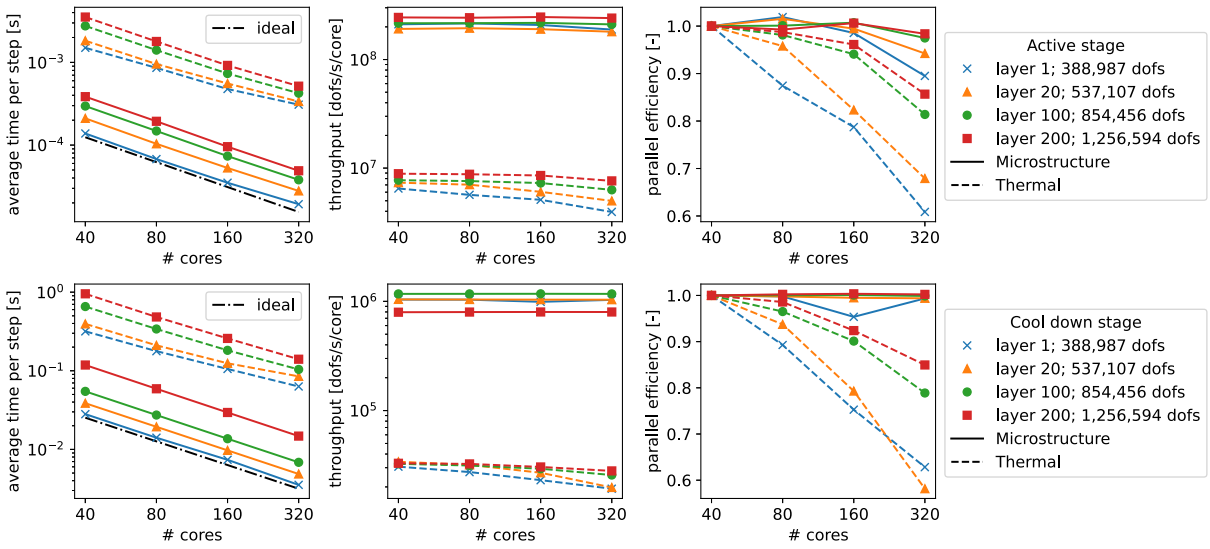


Fig. 10. Strong scaling study of cuboid example with adaptive mesh coarsening. The first row shows the average solution time per step, the throughput, and the parallel efficiency in the active laser stage; the second row shows the same metrics in the cool down stage. Note that the reported time per step includes subcycling for the microstructure model in the later cool down steps. The number of DoFs is given for the thermal problem; the microstructure problem carries three times this number.

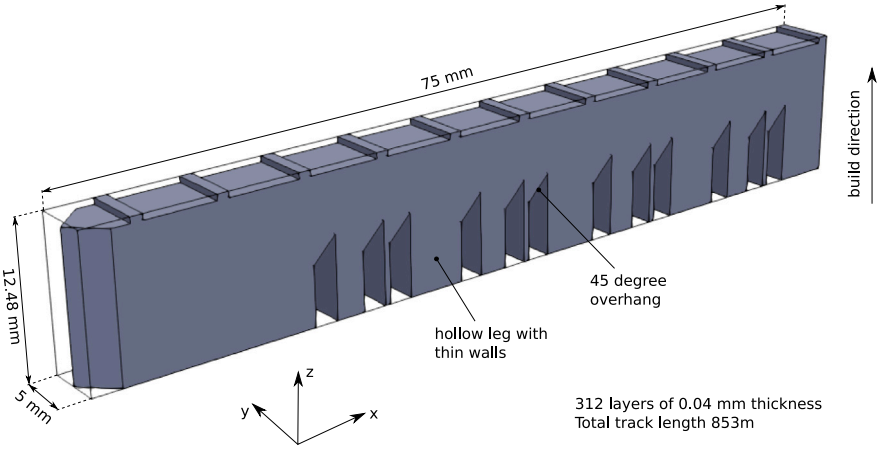


Fig. 11. NIST AMBench 2022 cantilever geometry.

thin-walled legs and overhang regions, leading to different local cooling rates. The geometry is built on a 10.56 mm high base plate section. To investigate the effect of different pre-heating temperatures, the bottom of the base plate is constrained to a fixed temperature $\hat{T} \in \{293\,\mathrm{K}, 500\,\mathrm{K}, 550\,\mathrm{K}, 600\,\mathrm{K}\}$. The scan strategy is directly taken from [42]. On average, processing a layer takes 2.8 s of physical time. The active scan stage in every layer is followed by a cool down stage of 1s, which uses identical time step sizes as described for the cube example. After simulating all 312 layers, a final cool down of 100 s is simulated, during which the temperature on the bottom of the baseplate is set to 293 K room temperature. The scan parameters are given in Table 6. In total, around 45 million time steps need to be solved. The vectorization width is $n_{\text{lanes}} = 8$.

The case of $\hat{T} = 293\mathrm{K}$ leads to a fully martensitic microstructure and is not shown in more detail. Instead, we focus on the results for higher pre-heating temperatures. Fig. 12 shows the phase fractions and residual temperatures at various points in time for a pre-heating temperature $\hat{T} = 600\,\mathrm{K}$. In the overhang regions and above the legs, a substantial amount of α_s-phase forms over time due to the reduced cooling rates resulting from the powder regions between the thin-walled legs. Again, the phase fraction α_s increases visibly a few layers after initial processing. It keeps growing until reaching 90% stable α_s-phase, the equilibrium value. The highest layers are not held at an elevated temperature for a sufficient time, so almost no α_s-phase forms here, and the microstructure is fully martensitic after the final cooling step.

While we varied the scan pattern in the last example, we now vary the pre-heating temperature and show the evolution of α_s-phase for $\hat{T} \in \{500\,\mathrm{K}, 550\,\mathrm{K}, 600\,\mathrm{K}\}$ in Fig. 13. The results reveal that the final microstructure composition is very sensitive to the pre-heating temperature within this temperature range. A change of only 100 K leads to a drastically different phase composition. The dynamic evolution of the α_s-phase is shown in more detail in the supplementary video 1.

The full simulation of the case with $\hat{T} = 293\,\mathrm{K}$ takes 52.3 h on four AMD Epyc nodes. It is emphasized that further reduction of the time to solution is still possible by employing more computational resources since the computational throughput is not yet limited by parallel communication overhead. Fig. 14 breaks down the total solution time into active and cool down stage and the thermal and microstructure problem. Due to the high degree of optimization of the microstructure implementation, the coupled thermo-microstructure problem is obtained at only marginally increased computation time compared to the thermal problem alone.

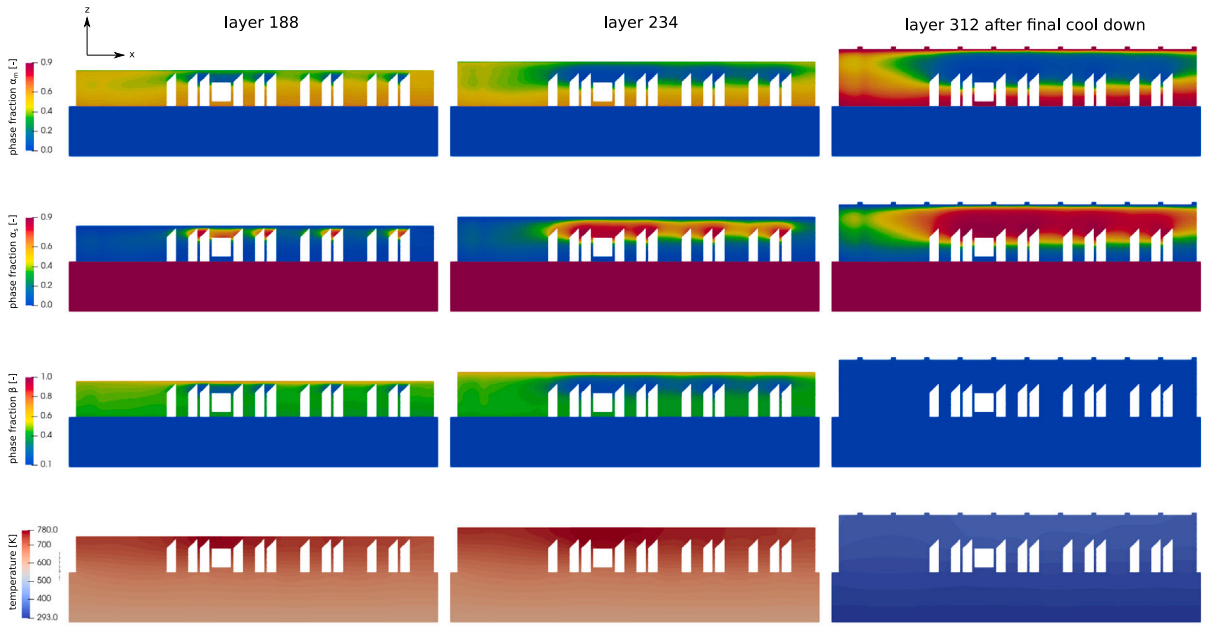


Fig. 12. Phase fractions and residual temperature after processing and cooling of layers 188, 234, and 312 (including final cool down time) for $\hat{T} = 600\,\mathrm{K}$. Results are depicted in the symmetry xz-plane of the cantilever geometry.

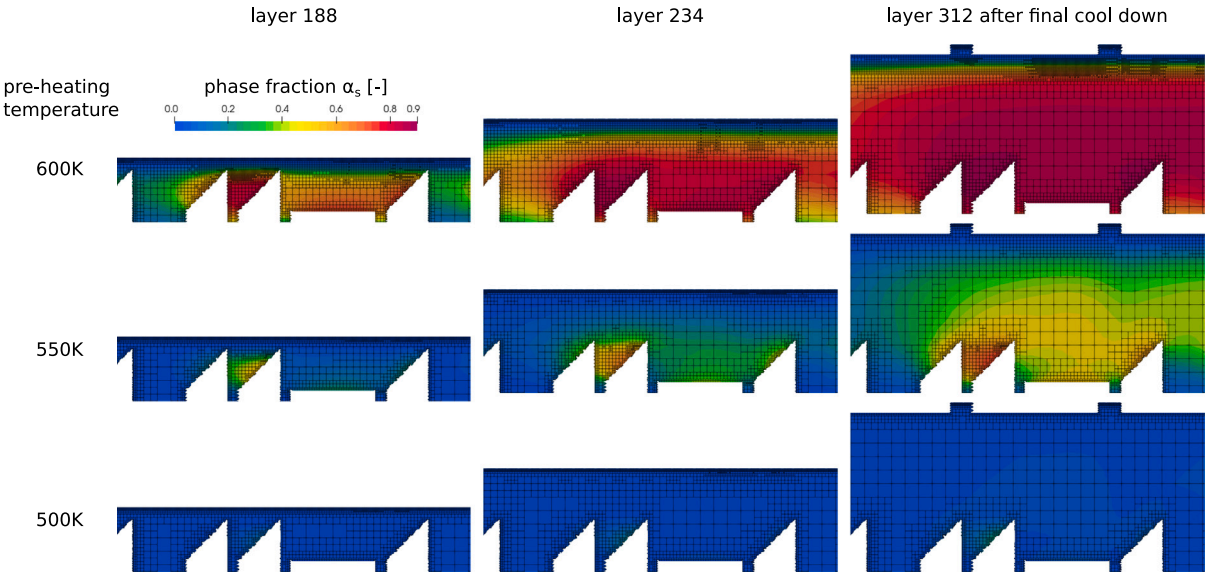


Fig. 13. Detailed view of α_s-phase fractions for different preheating temperatures after processing and cooling of layers 188 and 200. Results are depicted in the symmetry xz-plane of the cantilever geometry near the hollow leg structure. The adaptive mesh is more refined close to the boundaries and in regions of strong gradients in the microstructural composition.

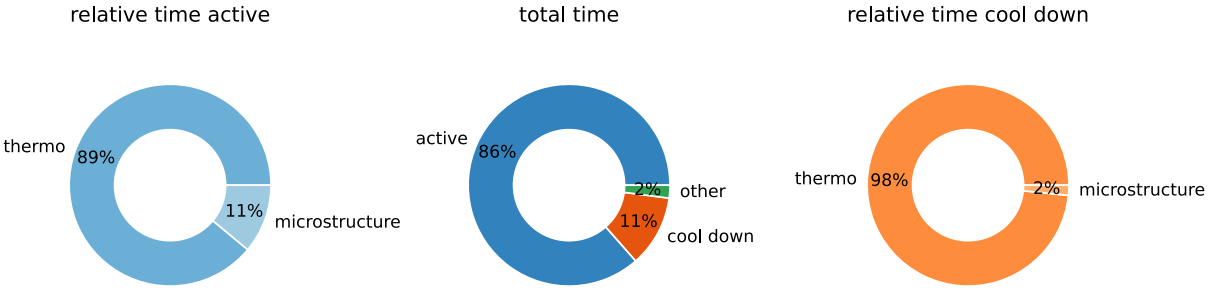


Fig. 14. Distribution of total solution time over different parts of the solution procedure.

5. Conclusion

We presented a highly efficient implementation of a coupled thermal-microstructure model for laser powder bed fusion (LPBF) of Ti-6Al-4V. The proposed approach enables simulations on the scale of realistic part sizes while consistently resolving the laser beam scan track. As demonstrated in the numerical examples, considering the physical scan track, which is not done by frequently employed simplified approaches such as layer-wise heat source models, is vital to capture variations in the microstructural composition caused by the scan strategy rather than the geometry. While the investigated geometries are of a relevant scale, the presented methodology may be combined with layer-wise heating approaches in areas where this simplification is applicable to tackle scales beyond decimeters in future research. Currently, simulations with hundreds of layers are possible in a few hours to days, depending on the build volume. Among others, the numerical examples include scan-resolved thermo-microstructure simulations of the full NIST AM Benchmark cantilever specimen. It is shown that variation of the build plate temperature by only 100 K can result in a significantly changed microstructure composition from α_m- to α_s-dominated.

The microstructure model equations contain conditional branches and computationally expensive mathematical functions. Through special approximations and a careful data layout, the proposed methodology can utilize modern hardware capabilities efficiently. The evaluation of the thermo-microstructure model comes with less than a 10% increase in run time compared to the thermal model. A crucial ingredient is the vectorized evaluation of the microstructure model, which speeds up the calculation by more than a factor of three. The coupled thermo-microstructural model exhibits a high degree of parallel efficiency and shows excellent strong scalability.

In our future research, the current model may be refined to include homogenized information on the anisotropic orientation of microstructural grains as typically induced by the very high thermal gradients in LPBF. The model can then serve as the basis for microstructure-informed material laws utilized in solid mechanics simulations of the LPBF process.

CRediT authorship contribution statement

Sebastian D. Proell: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation. **Julian Brotz:** Writing – review & editing, Software, Methodology, Investigation, Validation. **Martin Kronbichler:** Writing – review & editing, Supervision, Software. **Wolfgang A. Wall:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Christoph Meier:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors thank Maximilian Bergbauer for valuable discussions about performance modeling. Furthermore, the authors thank Neil Hodge, Jonas Nitzler, and Nils Much for their initial work on the microstructure model.

Funding

Christoph Meier gratefully acknowledges the financial support from the European Research Council through the ERC Starting Grant *ExcelAM* (project number: 101117579). Furthermore, work on this article has been supported by funding of the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within project 437616465 and project 414180263.

Appendix A. Evolution of microstructural phases for exemplary temperature history

To illustrate the evolution of the microstructure phases in the presented model, we show two variations of a temperature profile. Material with equilibrium phase composition is heated from room temperature up to 2000K. When reaching a temperature of 1200 K, the temperature is held for either 8 s (short hold) or 40 s (long hold) before heating continues. After reaching the peak temperature, the material is cooled down to room temperature. The absolute heating or cooling rate equals $25\,\mathrm{K\,s^{-1}}$. The short and long hold results are shown in Figs. 15 and 16, respectively. The duration of the short hold is chosen such that the diffusion $\alpha_s \rightarrow \beta$ initially decreases the α_s–phase fraction. However, the continued heating above $T_{\alpha_s,\text{start}}$ triggers the regularization in (28), which reduces the α_s–phase fraction to zero faster than the natural diffusion process. For the long hold, there is enough time for the diffusion process to reduce the α_s–phase to the equilibrium value (15) at the hold temperature of 1200 K and the fractions reach a stationary plateau. Once the material is heated further, diffusion continues, and the remaining α_s–phase transforms into β–phase. The material becomes liquid upon further heating. Only β–phase is present after resolidification. Once the temperature falls below $T_{\alpha_s,\text{start}}$, the diffusion process $\beta \rightarrow \alpha_s$ starts. The cooling rate is so high that the diffusion is too slow to reach the α_s–phase equilibrium (15) before the temperature drops below $T_{\alpha_m,\text{start}}$ and martensite forms according to (17).

Appendix B. Analytical solution for initiating diffusion-based transformations

When initiating the diffusion process $\alpha_s \rightarrow \beta$ from an initial state $X_{\alpha_s} = 0.9$, $X_\beta = 0.1$, the rate form (24) cannot be used in combination with an explicit scheme since it always yields zero for the initial values. Instead, an approximate solution is used for the initial time steps when diffusion starts.

The Crank–Nicolson integration scheme with a fixed point iteration would not face this issue if one were to perturb the initial guess. However, for a unified implementation, the analytical solution is also used to initiate diffusion when using the Crank–Nicolson scheme.

To derive the approximate analytical solution, we rewrite (24), with the help of (13) and the relations $X_\beta^{\text{eq}} = 1 - X_\alpha^{\text{eq}}$, $\tilde{X}_\beta := X_\beta - 0.1$ and $\tilde{X}_\beta^{\text{eq}} := X_\beta^{\text{eq}} - 0.1$ as

$$\begin{aligned}\dot{X}_\beta &= k_\beta(T)(0.9 - X_\alpha)^{\frac{c_\beta - 1}{c_\beta}} (X_\alpha - X_\alpha^{\text{eq}})^{\frac{c_\beta + 1}{c_\beta}} \\ &= k_\beta(T)(\tilde{X}_\beta)^{\frac{c_\beta - 1}{c_\beta}} (\tilde{X}_\beta^{\text{eq}} - \tilde{X}_\beta)^{\frac{c_\beta + 1}{c_\beta}}.\end{aligned} \tag{58}$$

We perform the substitution $\xi := \tilde{X}_\beta / \tilde{X}_\beta^{\text{eq}}$ and obtain

$$\dot{\xi} = \underbrace{\tilde{X}_\beta^{\text{eq}} k_\beta(T)}_{\tilde{k}} (\xi)^{\frac{c_\beta - 1}{c_\beta}} (1 - \xi)^{\frac{c_\beta + 1}{c_\beta}}, \tag{59}$$

which, assuming that $\tilde{k} = \text{const.}$, has a known solution given in [43]. For an initial value ξ_n at t_n, the solution in the next step $t_n + \Delta t$ can be

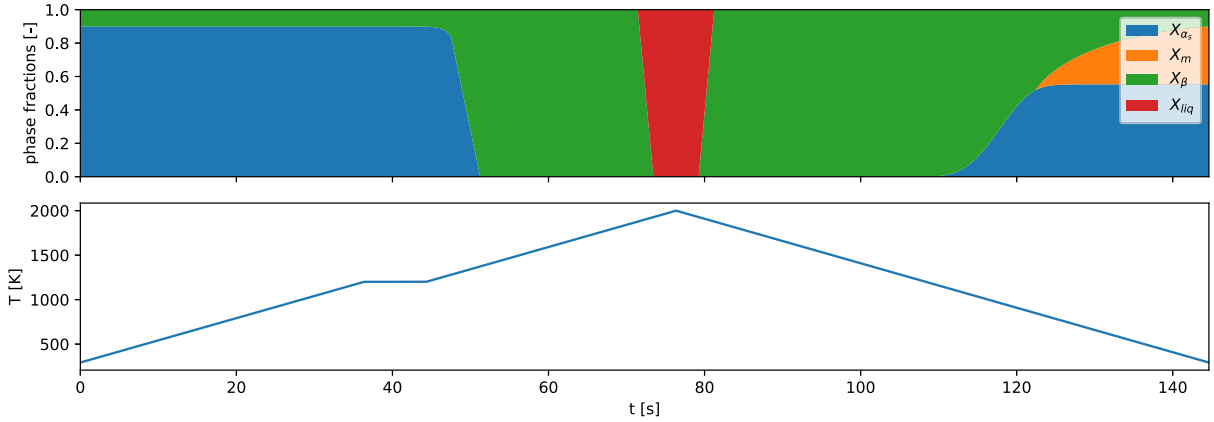


Fig. 15. Evolution of microstructure phases for a temperature history with an absolute heating/cooling rate of $25\,\mathrm{K\,s^{-1}}$ and a short hold of $8\,\mathrm{s}$ at $T = 1200\,\mathrm{K}$.

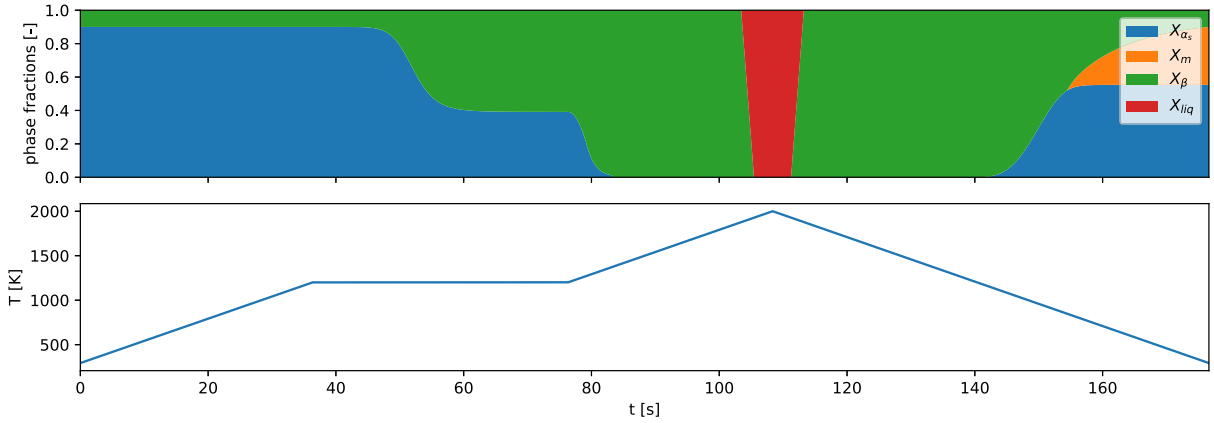


Fig. 16. Evolution of microstructure phases for a temperature history with an absolute heating/cooling rate of $25\,\mathrm{K\,s^{-1}}$ and a long hold of $40\,\mathrm{s}$ at $T = 1200\,\mathrm{K}$.

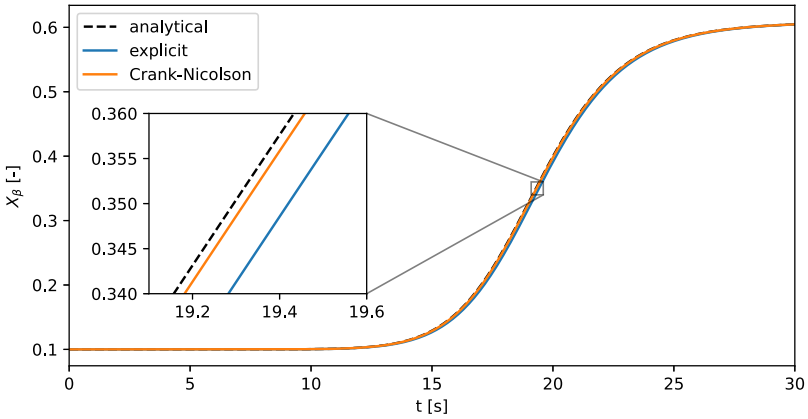


Fig. 17. Comparison of analytical solution for transformation $\beta \rightarrow \alpha_s$ to explicit and Crank–Nicolson time integration. The solution is computed for a constant temperature $T = 1200\,\mathrm{K}$ and initial values $X_{\alpha_s} = 0.9$ and $X_\beta = 0.1$.

found as:

$$\tilde{X}_\beta^{n+1} = \tilde{X}_\beta^{\mathrm{eq}} \left[1 + \left(\frac{c_\beta}{\tilde{X}_\beta^{\mathrm{eq}} k_\beta(T) \Delta t + c_\beta \left(\frac{\xi_n}{1-\xi_n} \right)^{\frac{1}{c_\beta}}} \right)^{c_\beta} \right]^{-1} \tag{60}$$

For numerical reasons, this solution is computed using a shifted fraction $\tilde{X}_\beta$, which has an equilibrium value of zero at room temperature. Due to the careful formulation of (60), it is possible to accurately work on a small number below machine precision $\epsilon(1) \approx 1 \times 10^{-16}$ without losing precision by adding to a numeric value on the order of 1. The analytical solution (60) is evaluated in subsequent time steps until $X_\beta^{n+1} \Delta t > 1 \times 10^{-15}$ which is the decrement in α_s-fraction according to (24). This criterion ensures a sufficient change is noticeable in the α-fraction and allows to continue the evolution with (24). While using the approximate analytical solution, the fraction $\tilde{X}_\beta$ is tracked independently and not computed from the continuity constraint (29). The applicability of this strategy has been verified for a constant temperature $T = 1200\,\mathrm{K}$ as shown in Fig. 17. The analytical approximation (60) is exact in the case of a constant temperature.

An analytical approximation for the initial diffusion step from $\beta \rightarrow \alpha_s$ has already been discussed in [4]. In this case, the values obtained from the analytical expression are large enough so that significant digits are not absorbed.

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.addma.2024.104380.

References

[1] J. Yang, H. Yu, J. Yin, M. Gao, Z. Wang, X. Zeng, Formation and control of martensite in Ti-6Al-4V alloy produced by selective laser melting, Mater. Des. 108 (2016) 308–318.

[2] C. Meier, et al., Physics-based modeling and predictive simulation of powder bed fusion additive manufacturing across length scales, GAMM-Mitt. 44 (3) (2021) e202100014.

[3] C. Meier, R.W. Penny, Y. Zou, J.S. Gibbs, A.J. Hart, Thermophysical phenomena in metal additive manufacturing by selective laser melting: Fundamentals, modeling, simulation, and experimentation, Ann. Rev. Heat Transf. 20 (2017).

[4] J. Nitzler, C. Meier, K.W. Müller, W.A. Wall, N.E. Hodge, A novel physics-based and data-supported microstructure model for part-scale simulation of laser powder bed fusion of Ti-6Al-4V, Adv. Model. Simul. Eng. Sci. 8 (1) (2021) 16.

[5] D. Furrer, S. Semiatin, Introduction to fundamentals of modeling for metals processing, in: Fundamentals of Modeling for Metals Processing, ASM International, 2009.

[6] A. Crespo, Modelling of heat transfer and phase transformations in the rapid manufacturing of titanium components, in: Convection and Conduction Heat Transfer, InTech, 2011.

[7] L. Lindgren, A. Lundbäck, M. Fisk, R. Pederson, J. Andersson, Simulation of additive manufacturing using coupled constitutive and microstructure models, Addit. Manuf. 12 (2016) 144–158.

[8] C.C. Murgau, R. Pederson, L.-E. Lindgren, A model for Ti–6Al–4V microstructure evolution for arbitrary temperature changes, Modelling Simul. Mater. Sci. Eng. 20 (5) (2012) 055006.

[9] E. Salsi, M. Chiumenti, M. Cervera, Modeling of microstructure evolution of Ti6Al4V for additive manufacturing, Metals 8 (8) (2018) 633.

[10] Q. Zhang, J. Xie, Z. Gao, T. London, D. Griffiths, V. Oancea, A metallurgical phase transformation framework applied to SLM additive manufacturing processes, Mater. Des. 166 (2019) 107618.

[11] S. Mishra, T. DebRoy, Measurements and Monte Carlo simulation of grain growth in the heat-affected zone of Ti–6Al–4V welds, Acta Mater. 52 (5) (2004) 1183–1192.

[12] P. Nie, O. Ojo, Z. Li, Numerical modeling of microstructure evolution during laser additive manufacturing of a nickel-based superalloy, Acta Mater. 77 (2014) 85–95.

[13] J.H.K. Tan, S.L. Sing, W.Y. Yeong, Microstructure modelling for metallic additive manufacturing: A review, Virtual Phys. Prototyp. 15 (1) (2020) 87–105.

[14] R. Ding, Z. Guo, Microstructural evolution of a Ti–6Al–4V alloy during β-phase processing: Experimental and simulative investigations, Mater. Sci. Eng. A 365 (1–2) (2004) 172–179.

[15] J. Koepf, D. Soldner, M. Ramsperger, J. Mergheim, M. Markl, C. Körner, Numerical microstructure prediction by a coupled finite element cellular automaton model for selective electron beam melting, Comput. Mater. Sci. 162 (2019) 148–155.

[16] A. Rai, M. Markl, C. Körner, A coupled Cellular Automaton–Lattice Boltzmann model for grain structure simulation during additive manufacturing, Comput. Mater. Sci. 124 (2016) 37–48.

[17] M. Rolchigo, S.T. Reeve, B. Stump, G.L. Knapp, J. Coleman, A. Plotkowski, J. Belak, Exaca: A performance portable exascale cellular automata application for alloy solidification modeling, Comput. Mater. Sci. 214 (2022) 111692.

[18] Q. Chen, N. Ma, K. Wu, Y. Wang, Quantitative phase field modeling of diffusion-controlled precipitate growth and dissolution in Ti–Al–V, Scr. Mater. 50 (4) (2004) 471–476.

[19] X. Gong, K. Chou, Phase-field modeling of microstructure evolution in electron beam additive manufacturing, JOM 67 (2015) 1176–1182.

[20] F. Nahr, et al., Geometrical influence on material properties for Ti6Al4V parts in powder bed fusion, J. Manuf. Mater. Process. 7 (3) (2023) 82.

[21] P. Promoppatum, S.-C. Yao, P.C. Pistorius, A.D. Rollett, P.J. Coutts, F. Lia, R. Martukanitz, Numerical modeling and experimental validation of thermal history and microstructure for additive manufacturing of an Inconel 718 product, Prog. Addit. Manuf. 3 (2018) 15–32.

[22] J. Munk, E. Breitbarth, T. Siemer, N. Pirch, C. Häfner, Geometry effect on microstructure and mechanical properties in laser powder bed fusion of Ti-6Al-4V, Metals 12 (3) (2022) 482.

[23] J. Berry, et al., Toward multiscale simulations of tailored microstructure formation in metal additive manufacturing, Mater. Today 51 (2021) 65–86.

[24] J.A. Turner, et al., ExaAM: Metal additive manufacturing simulation at the fidelity of the microstructure, Int. J. High Perform. Comput. Appl. 36 (1) (2022) 13–39.

[25] S.D. Proell, P. Munch, M. Kronbichler, W.A. Wall, C. Meier, A highly efficient computational approach for fast scan-resolved simulations of metal additive manufacturing processes on the scale of real parts, Addit. Manuf. 79 (2024) 103921, http://dx.doi.org/10.1016/j.addma.2023.103921.

[26] A.C.I. Malossi, Y. Ineichen, C. Bekas, A. Curioni, Fast exponential computation on SIMD architectures, in: Proc. of HIPEAC-WAPCO, Amsterdam NL, 56, 2015.

[27] F. Perini, R.D. Reitz, Fast approximations of exponential and logarithm functions combined with efficient storage/retrieval for combustion kinetics calculations, Combust. Flame 194 (2018) 37–51.

[28] N.N. Schraudolph, A fast, compact approximation of the exponential function, Neural Comput. 11 (4) (1999) 853–862.

[29] S.D. Proell, W.A. Wall, C. Meier, On phase change and latent heat models in metal additive manufacturing process simulation, Adv. Model. Simul. Eng. Sci. 7 (2020) 1–32.

[30] S.D. Proell, W.A. Wall, C. Meier, A simple yet consistent constitutive law and mortar-based layer coupling schemes for thermomechanical macroscale simulations of metal additive manufacturing processes, Adv. Model. Simul. Eng. Sci. 8 (1) (2021) 24.

[31] X. Lu, M.V. Li, H. Yang, Simulation of precipitates evolution driven by non-isothermal cyclic thermal history during wire and arc additive manufacturing of IN718 superalloy, J. Manuf. Process. 65 (2021) 258–270.

[32] M. Kronbichler, K. Kormann, A generic interface for parallel cell-based finite element operator application, Comput. & Fluids 63 (2012) 135–147.

[33] D. Arndt, et al., The deal.II library, version 9.5, J. Numer. Math. 31 (3) (2023) 231–246.

[34] C++ SIMD library, https://en.cppreference.com/w/cpp/experimental/simd. (Accessed 02 June 2024).

[35] A. Fog, VCL – C++ vector class library manual, 2024, https://raw.githubusercontent.com/vectorclass/manual/master/vcl_manual.pdf. (Accessed 02 June 2024).

[36] N. Shibata, F. Petrogalli, SLEEF: A portable vectorized library of c standard mathematical functions, IEEE Trans. Parallel Distrib. Syst. 31 (6) (2019) 1316–1327.

[37] IEEE standard for floating-point arithmetic, in: IEEE Std 754-2019 (Revision of IEEE 754-2008), 2019, pp. 1–84, http://dx.doi.org/10.1109/IEEESTD.2019.8766229.

[38] G. Estrin, Organization of computer systems: The fixed plus variable structure computer, in: Papers Presented At the May 3-5, 1960, Western Joint IRE-AIEE-ACM Computer Conference, 1960, pp. 33–40.

[39] J. Treibig, G. Hager, G. Wellein, Likwid: A lightweight performance-oriented tool suite for x86 multicore environments, in: 2010 39th International Conference on Parallel Processing Workshops, IEEE, 2010, pp. 207–216.

[40] llvm-mca – LLVM Machine Code Analyzer, https://llvm.org/docs/CommandGuide/llvm-mca.html. (Accessed 02 June 2024).

[41] X. Lu, G. Zhang, J. Li, M. Cervera, M. Chiumenti, J. Chen, X. Lin, W. Huang, Simulation-assisted investigation on the formation of layer bands and the microstructural evolution in directed energy deposition of Ti6Al4V blocks, Virtual Phys. Prototyp. 16 (4) (2021) 387–403.

[42] B. Lane, L. Levine, D. Deisenroth, H. Yeung, V. Tondare, S. Mekhontsev, J. Neira, AM Bench 2022 3D Build Modeling Challenge Description Data (AMB2022-01), Technical report, National Institute of Standards and Technology, 2022, http://dx.doi.org/10.18434/mds2-2607.

[43] I. Avramov, J. Šesták, Generalized kinetics of overall phase transition explicit to crystallization, J. Therm. Anal. Calorim. 118 (2014) 1715–1720.