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 simulations of metal additive manufacturing processes on the scale of real parts

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

[a] *Institute for Computational Mechanics, Technical University of Munich, 85748 Garching b. München, Germany*
[b] *Institute of Mathematics, University of Augsburg, 86159 Augsburg, Germany*

ARTICLE INFO

Keywords:
Powder bed fusion additive manufacturing
Part-scale
Explicit time integration
Finite-element computations
Fast operator evaluation

ABSTRACT

This article proposes a novel high-performance computing approach for the prediction of the temperature field in powder bed fusion (PBF) additive manufacturing (AM) processes. In contrast to many existing approaches to part-scale simulations, the underlying computational model consistently resolves physical scan tracks without additional heat source scaling, agglomeration strategies or any other heuristic modeling assumptions. A growing, adaptively refined mesh accurately captures all details of the laser beam motion. Critically, the fine spatial resolution required for resolved scan tracks in combination with the high scan velocities underlying these processes mandates the use of comparatively small time steps to resolve the underlying physics. Explicit time integration schemes are well-suited for this setting, while unconditionally stable implicit time integration schemes are employed for the interlayer cool down phase governed by significantly larger time scales. These two schemes are combined and implemented in an efficient fast operator evaluation framework providing significant performance gains and optimization opportunities. The capabilities of the novel framework are demonstrated through realistic AM examples on the centimeter scale including the first scan-resolved simulation of the entire NIST AM Benchmark cantilever specimen, with a computation time of less than one day. Apart from physical insights gained through these simulation examples, also numerical aspects are thoroughly studied on basis of weak and strong parallel scaling tests. As potential applications, the proposed thermal PBF simulation approach can serve as a basis for microstructure and thermo-mechanical predictions on the part-scale, but also to assess the influence of scan pattern and part geometry on melt pool shape and temperature, which are important indicators for well-known process instabilities.

1. Introduction

Metal additive manufacturing (AM) offers a variety of advantages over conventional manufacturing techniques [1,2]. This contribution focuses on powder bed fusion AM (PBFAM) where the desired part geometry is molten into a powder bed by means of a laser (or electron) beam. However, the approach presented in this article is also transferable to other processes such as directed energy deposition (DED).

One of the most commonly cited advantages of AM is the ability to produce complex geometries in a near net shape manner. As exciting as this promise may be for the industry as a whole, it also poses new challenges for part design: due to the high geometrical complexity a part may not be manufacturable with the desired quality or adequate process parameters are hard to find. Various defects such as porosity, dimensional warping and delamination are known in the literature [3], and it remains difficult to predict where and when any of these will appear during the build process of a given part.

Instead of experimentally tuning the process parameters or part geometry, predictive simulation tries to offer an alternative. The different kinds of modeling approaches for PBFAM can be characterized by the length scales they operate on [4,5]. Mesoscale models are used to analyze the melt pool on length scales from a few powder particles up to one laser scan track [6–13]. They can also be used to study the powder recoating process [14–17]. Microscale models are concerned with the formation of anisotropic metallurgical microstructures during solidification [18–23]. In this contribution, we investigate the problem on the macroscale. Since practically relevant geometries are complex, in general, the build process of whole parts needs to be simulated in order to answer questions about the build quality. For this, the term *part-scale* simulation or model is often used in the literature. Virtually all existing part-scale models employ the finite element method (FEM) due to its excellent suitability for thermo(-mechanical) simulations. In this work, we develop an efficient simulation approach for part-scale simulations of the thermal problem.

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

https://doi.org/10.1016/j.addma.2023.103921
Received 11 July 2023; Received in revised form 11 November 2023; Accepted 11 December 2023
Available online 13 December 2023

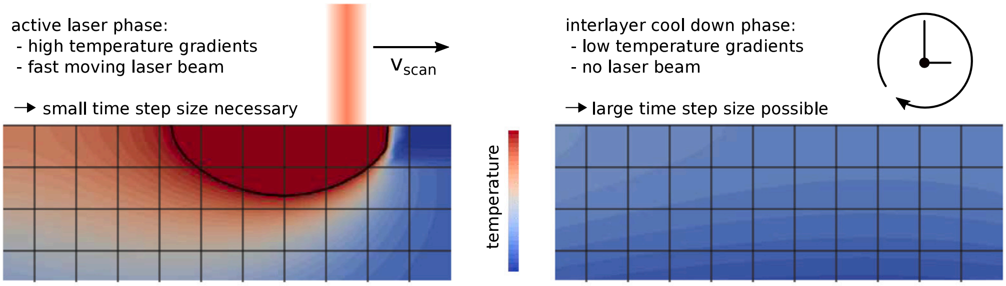

active laser phase:
- high temperature gradients
- fast moving laser beam
→ small time step size necessary
V_{scan}
temperature
interlayer cool down phase:
- low temperature gradients
- no laser beam
→ large time step size possible

Fig. 1. The different phases of the PBFAM build process come with different requirements for the time step size.

The fundamental computational challenge in part-scale simulation lies not so much in the spatial approximation. Although millions of unknowns are necessary to resolve the geometry, state-of-the-art codes and libraries are well-suited to handle this task with mesh adaptivity and parallel processing. Rather, the challenge lies in the temporal domain. Taking the recent “AM Bench 2022” [24] build setup as an example, one finds that in order to simulate one of its cantilever specimens with a total scan track of approximately 853 m a total of around 44 million time steps (of step size 20 μs) are necessary. Put differently, to obtain a solution to this problem within 10 days, one time step may not take longer than 20 ms of wall time. Most classical implementations of FEM models of PBFAM [25–27], including some of the authors' work [28,29], are suitable for the simulation of a few tracks or layers but do not achieve the level of performance necessary for part-scale simulations. Instead, existing part-scale models use one or more of the following techniques.

A straight-forward approach to part-scale simulations uses a layer-based approach, where whole layers (or parts thereof) are heated at once and the scanning pattern is neglected [30–33]. To speed up the simulation further, multiple physical powder layers can be lumped into larger *process layers* [34,35]. Typically, these agglomerated models are calibrated with experimental data or resolved single-track or single-layer simulations. Despite the strong simplifications, these models are able to predict, e.g., thermal hot spots or dimensional warping — but only when calibrated correctly, which can act as a bottleneck or limitation of such approaches.

In contrast to the literature cited so far, the aim of this contribution is an efficient implementation of PBFAM process simulation with *resolved* scan tracks on hundreds of realistically-sized layers. One prerequisite to enable efficient simulations on that scale is adaptive mesh refinement (AMR). This technique has been employed in various contributions and in different forms [36–40]. Generally speaking, in AM applications AMR means that the mesh is not static but adapted dynamically over the course of the simulation to be as fine as necessary in the vicinity of the heat source and coarse in regions further away. In addition, the geometry needs to grow to represent the layer deposition in the manufacturing process [41]. Building on top of the `deal.II` library [42], its parallel data structures [43,44], and the `p4est` [45] library, we develop our own methodology for AMR and growing domains in the targeted PBFAM application. Our approach is to some extent inspired by a similar strategy, also based on parallel distributed octree meshes, previously presented in [39]. While not discussed here, the presented method complements the dual-mortar approach shown in [29], which is still relevant for meshing complex geometries.

A rarely discussed aspect of efficiency in PBFAM simulations is the choice of time discretization scheme. The PBFAM process can be split into a highly dynamic, active laser phase and a subsequent interlayer cool down phase governed by significantly larger time scales, see Fig. 1. Traditionally, the heat equation is discretized with an unconditionally stable, implicit scheme such as the backward Euler method or generalized trapezoidal scheme. In many applications an implicit scheme seems appropriate as it enables large time steps. This is the case in our application for the cool down phase. Explicit schemes have considerably cheaper evaluation costs per time step and offer better parallel scalability since they can circumvent the assembly of global matrices and the solution of (non)linear systems. However, they are restricted to smaller time steps by a stability limit. It turns out that in the specific scenario of the active laser phase of a scan-resolved PBFAM simulation, the stability limit is not restrictive compared to the time step limitation mandated by the moving heat source. Importantly, the time step limitation due to the moving heat source is required for accuracy and not stability: it holds for explicit and implicit schemes, and thus can be considered an inherent characteristic of the physical problem when modeled in a scan-resolved manner. This consideration has also been stated independently of the authors in the very recent contribution [46]. Explicit time stepping has been used for the simulation of PBFAM in [47], directed energy deposition in [48] and wire arc AM in [49]. In this work, we employ an explicit scheme for the active laser phase and an implicit scheme for the cool down phase. Potential future extensions include a local time stepping scheme or multi-rate time integration [50,51] and techniques for temporal decoupling [47,52].

For the active laser phase, the evaluation time of FEM integrals becomes the main focus of performance engineering, since a linear system solve can be avoided by using explicit schemes. Some recent contributions in the AM community use GPUs to accelerate the evaluation [40,48,49]. In [40] the authors presented a matrix-free implicit solver for scan-resolved PBFAM simulations.

In this contribution, we will focus on an implementation for CPUs.

Notable other CPU-based implementations of PBFAM models that investigate computational performance make use of AMR and load balancing [39]. In addition to AMR, different techniques to scale up the heat source either by elongating it [53], layer-averaging [54] or layer-agglomeration [37] are applied in the literature. The present contribution does not use such techniques to stay as close to the physical process as feasible but the methods presented are general enough to incorporate any of these in the future.

To the best of our knowledge, our implementation, facilitated by the `deal.II` library and fast application of FEM operators [55,56], outperforms competing implementations for the thermal PBFAM problem in terms of time to solution. The performance is a result of the parallel distributed, high-performance implementation of a single time step. The implementation integrates modern hardware features such as vectorized CPU instructions and tries to alleviate the memory-bound nature (i.e., overall performance is mainly limited by the memory bandwidth rather than the necessary CPU cycles) by the efficient utilization of caches. Another reason for considering CPU-based implementations in this work is that strong scaling is highly relevant for the present application, where CPU-based systems with appropriate tuning often have an edge over GPU systems [57].

The capabilities of the proposed approach are demonstrated on the basis of some challenging examples, the first one being a bridge geometry. Various performance studies show the scalability of the approach on large distributed machines. Finally, and, to the best of the authors' knowledge, for the first time, we present a full scan-resolved simulation of all 312 layers of the NIST AM Bench 2022 cantilever specimen [24]. The proposed thermal PBF simulation approach already allows to assess the influence of scan pattern and part geometry on melt pool shape, overheated zones, zones with residual porosity, which are important indicators for process instabilities. Furthermore, it can serve as a future basis for a thermo-mechanical model to predict thermal distortion and residual stresses or the microstructure in terms of homogenized phase fractions [20] on the scale of real parts.

The remainder of this article is structured as follows: first, we present the mathematical model of the physical process and subsequently derive the spatially and temporally discrete numerical model from it. Next, we discuss aspects of the high-performance implementation with a focus on mesh adaptivity and fast operator evaluation. We present two exemplary numerical simulations of PBFAM on representative geometries and study the performance of the proposed model before we conclude with a short summary and an outlook on future research.

2. Mathematical model

The present model seeks the solution for the temperature field T in the domain $\varOmega$, which is governed by the heat equation:

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

with the following parameters: ρ is the density and c is the specific heat capacity of the material. The heat capacity could be used to model the effects of latent heat through an apparent capacity model [28]. However, the contribution of latent heat to the overall energy balance is rather small and often neglected in the literature on part-scale AM simulations. For the modeling of a phase-dependent heat conductivity k we briefly summarize the approach from our previous work [28]. The liquid fraction $g(T)$ is defined as

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

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

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

captures the irreversible powder-to-melt phase transition and allows to set 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 single phase parameters. Within each state of the material, all material parameters are fixed, i.e., the single phase problems are linear. This choice is made for the sake of simplicity since the focus of this work lies on a HPC implementation of the model rather than on calibration of material data although the implementation presented in this work also supports temperature-dependent parameters. Note that the history variable r_c necessitates a proper handling of history data when using mesh adaptivity, e.g., a consistent interpolation of tensor-valued history data [58].

The volumetric heat source q_{vol} models the incident energy from a laser (or electron) beam. In this work, it is given by the following cylindrical model:

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

which is formulated in a local coordinate system $(\hat{x}, \hat{y}, \hat{z})$ moving along the scan track. The shape in the xy-plane is described by a normal distribution with mean $(\hat{x}, \hat{y}) = (0, 0)$ and standard deviation $\sigma = R/2$. Thus, R can be interpreted as an effective beam radius of the incident energy beam. Furthermore, W_{eff} is the effective power, which is reduced compared to the nominal power due to various losses and the material's absorptivity, and h_{powder} is the powder layer thickness. The chosen heat source (6) is deliberately kept simple. Other often employed models such as a Gusarov [59] or Goldak [60] heat source could be easily used instead. In the authors' experience the exact choice does not notably influence the simulation results on the part-scale.

The heat equation (1) is completed by the following initial and boundary conditions:

$$T = T_0 \qquad \text{in } \varOmega \text{ for } t = 0, \tag{7}$$

$$T = T_\infty \qquad \text{on } \varGamma_D, \tag{8}$$

$$\boldsymbol{q} \cdot \boldsymbol{n} = 0 \qquad \text{on } \varGamma_N, \tag{9}$$

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

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

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

Initially, the whole domain is set to a fixed temperature T_0 as stated in (7). This is also true for parts of the domain which only become activated at a later stage. The temperature is kept fixed at the ambient temperature T_∞ on the Dirichlet part of the boundary $\varGamma_D$ at the bottom of the baseplate (8). Both a radiation and evaporation condition (10) are applied on the free surface $\varGamma_{RE}$ at the top of the built part. Inclusion of a convection boundary condition would be straight-forward but not done in this work as the influence is considered small as compared to radiation and evaporation. The remaining part of the boundary $\varGamma_N$ is modeled as thermally insulating (9). These conditions include the following constants and parameters: For the radiation term (11), ϵ is the emissivity and σ_S the Stefan–Boltzmann constant. For the evaporation condition based on [10,61], $C_P = 0.54p_a$ is a factor with the dimension of pressure computed from the atmospheric pressure p_a and $C_T \approx \bar{h}_v/R$ a factor with the dimension of temperature computed from the molar latent heat of evaporation $\bar{h}_v$ and the molar gas constant R. Moreover, T_v is the boiling temperature, h_v the specific latent heat of evaporation and $T_{h,0}$ is a reference temperature for the enthalpy calculation. The constant $C_M = M/(2\pi R)$ is computed from the molar mass M and the molar gas constant R. Overall this leads to an expression for the heat flux q_{evap} from evaporation that consists of an evaporative mass flux $\dot{m}$ multiplied by a specific enthalpy. To avoid numerical issues with the strong nonlinearity in the evaporation term (12), the temperature $[T]$ used for its evaluation is limited to a maximum value $T_{\max} > T_v$ by setting $[T] = \min(T, T_{\max})$. In this work, we choose $T_{\max} = T_v + 1000\,\mathrm{K}$. This choice does not influence the overall results and leads to a robust numerical scheme.

3. Numerical discretization and solution schemes

3.1. Weak form and spatial discretization

In order to solve the heat equation (1) numerically we employ a finite element (FE) discretization for the spatial dimension. First, the heat equation is multiplied with a test function v and the diffusive term is integrated by parts, yielding

$$\left(v, \rho c \frac{\partial T}{\partial t}\right)_{\Omega} = (\nabla v, \boldsymbol{q})_{\Omega} - (v, \boldsymbol{q}\cdot\boldsymbol{n})_{\Gamma_{RE}} + \left(v, q_{\text{vol}}\right)_{\Omega}, \tag{13}$$

where $(a,b)_{\square} := \int_{\square} ab$. The weak form (13) is equivalent to the strong form (1) if the test function is chosen from the weighting space $\mathcal{W} = \{v \in H_1(\Omega) : v = 0 \text{ on } \Gamma_D\}$ and the solution function is chosen from the trial space $\mathcal{V} = \{T \in H_1(\Omega) : T = T_\infty \text{ on } \Gamma_D\}$, where $H_1(\Omega)$ is the Sobolev space containing functions with square-integrable first derivatives. The solution and test functions are discretized in space based on a (continuous) Bubnov–Galerkin ansatz:

$$T_h(\boldsymbol{x},t) = \sum \varphi_j(\boldsymbol{x}) T_j(t), \quad v_h(\boldsymbol{x},t) = \sum \varphi_j(\boldsymbol{x}) v_j(t), \tag{14}$$

where $\boldsymbol{x}$ is the spatial coordinate, $\varphi_j(\boldsymbol{x})$ are the space-dependent shape functions used for solution and test functions. The discrete degrees of freedom (DoFs) $T_j(t)$ and $v_j(t)$ only depend on time. In this work, we exclusively use first-order Lagrange polynomials but the implementation supports higher order functions as well. After inserting the spatial discretization (14) into the weak form (13) we obtain the following semi-discrete problem:

$$\boldsymbol{C}\dot{\boldsymbol{T}} = \boldsymbol{f}(\boldsymbol{T}) = \boldsymbol{f}_{\text{diff}}(\boldsymbol{T}) + \boldsymbol{f}_{\text{RE}}(\boldsymbol{T}) + \boldsymbol{f}_{\text{vol}}, \tag{15}$$

where $\boldsymbol{C}$ is a capacity matrix, $\boldsymbol{T}$ and $\dot{\boldsymbol{T}}$ are the global vectors of nodal temperatures and their time derivatives and $\boldsymbol{f}(\boldsymbol{T})$ is composed of the nonlinear diffusive term $\boldsymbol{f}_{\text{diff}}(\boldsymbol{T})$ as well as the boundary term $\boldsymbol{f}_{\text{RE}}(\boldsymbol{T})$ and source term $\boldsymbol{f}_{\text{vol}}$. These terms are given in the same order as their equivalent weak form contributions in (13).

3.2. Time integration and solution procedure

In this contribution, an implicit and an explicit time integration scheme are combined. When a layer is scanned, the explicit time integration scheme is used, while the interlayer cool down phase is simulated with the implicit time integration scheme. Since the scanning phase requires most of the computational time, we mainly tune the performance of the explicit scheme, as discussed in the next section.

Explicit scheme. For the active laser phase we apply the forward Euler scheme to (15):

$$\boldsymbol{T}_{n+1} = \boldsymbol{T}_n + \Delta t \tilde{\boldsymbol{C}}^{-1} \boldsymbol{f}(\boldsymbol{T}_n), \tag{16}$$

where the consistent capacity matrix $\boldsymbol{C}$ is replaced with a lumped, diagonal variant [62],

$$\tilde{C}_{ii} = \sum_j C_{ij}, \tag{17}$$

which is trivially invertible. In the implementation, the diagonal matrix can be precomputed and stored as a vector such that its application becomes a simple scaling operation. The computationally most challenging task in (16) is the efficient evaluation of $\boldsymbol{f}(\boldsymbol{T}_n)$. Note that any other explicit time stepping scheme could be used as well. We found the explicit Euler scheme to provide sufficient accuracy for the small time steps sizes (demanded by the moving heat source).

Explicit time integration schemes are not unconditionally stable. In order to find the largest stable time step for the explicit Euler scheme, we replace the nonlinear function in (16) with a linearized version which only considers the critical diffusive term $\boldsymbol{f}_{\text{diff}} \approx \boldsymbol{K}_{\text{diff}}\boldsymbol{T}$:

$$\boldsymbol{T}_{n+1} \approx \boldsymbol{T}_n + \Delta t \tilde{\boldsymbol{C}}^{-1} \boldsymbol{K}_{\text{diff}} \boldsymbol{T}_n = \underbrace{\left(\boldsymbol{I} + \Delta t \tilde{\boldsymbol{C}}^{-1} \boldsymbol{K}_{\text{diff}}\right)}_{=:\boldsymbol{A}} \boldsymbol{T}_n = \boldsymbol{A}^n \boldsymbol{T}_0, \tag{18}$$

where $\boldsymbol{I}$ is the identity matrix. The approximation performed in (18) only neglects the nonlinearity in the heat conductivity which is limited to the small phase change interval $[T_s, T_l]$. Since the matrix $\boldsymbol{A}$ is repeatedly applied to the temperature vector, its spectral radius must be $\rho(\boldsymbol{A}) \le 1$, i.e., its largest absolute eigenvalue needs to be smaller than 1. After some rearrangement one finds for the critical time step:

$$\Delta t \le \frac{2}{\rho\left(\tilde{\boldsymbol{C}}^{-1} \boldsymbol{K}_{\text{diff}}\right)}. \tag{19}$$

A more detailed discussion of stability limits in the context of explicit time integration for PBFAM problems can be found in [46]. In addition to (19), the admissible time step size is also limited by the velocity v_{scan} of the moving heat source: we do not want the heat source to travel further than the radius R of the laser beam in one time step and therefore require:

$$\Delta t \le \frac{R}{v_{\text{scan}}}. \tag{20}$$

It is crucial to realize that (20) is required to achieve a continuous melt track in a scan-resolved simulation. Thus, this restriction also holds for an implicit scheme which might allow much larger time steps from a pure stability perspective. With our choice of heat source model (6), if one were to use a larger time step than (20) allows, the melt track would break up into disjoint segments. Note that this restriction could be weakened by the use of elongated line heat sources of equivalent energy [63,64] but since such an approach requires calibration, we do not follow it here. Together (19) and (20) form a combined criterion for the maximum time step size:

$$\Delta t \le \min\left\{\frac{R}{v_{\text{scan}}}, \frac{2}{\rho\left(\tilde{\boldsymbol{C}}^{-1}\boldsymbol{K}_{\text{diff}}\right)}\right\}. \tag{21}$$

Estimation of the spectral radius is rather expensive; consequently, the stability criterion should only be evaluated in the setup phase of a simulation. In the numerical examples we found it sufficient to only evaluate (21) once for the critical values in a given set of parameters since the characteristics involved in the criterion do not change over layers. For the parameters used in the numerical examples, we found the accuracy criterion (20) (which is independent of the time integration scheme) to be around 5–10 times smaller than the stability criterion for the explicit scheme (19). Evaluation of an explicit time step will be faster than the (iterative) solution of a linear system arising from an implicit scheme. Therefore, we conclude that an explicit scheme is superior for the simulation of the active laser phase.

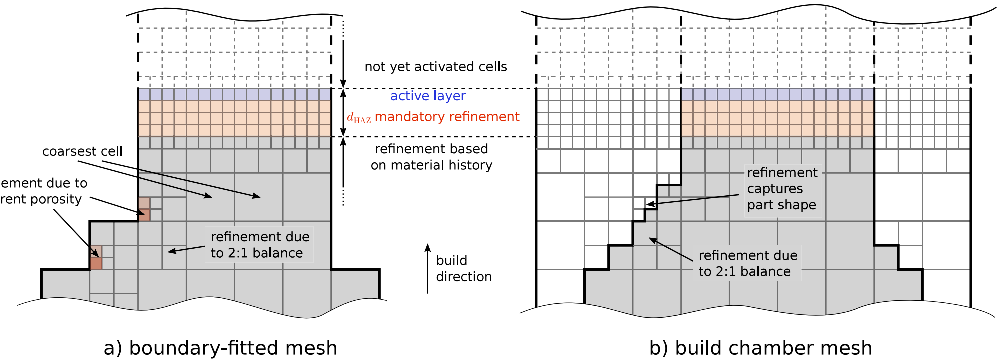
Implicit scheme. The following implicit scheme can be applied to (15):

$$\boldsymbol{r} := \frac{1}{\Delta t}\boldsymbol{C}(\boldsymbol{T}_{n+1} - \boldsymbol{T}_n) - \boldsymbol{f}_{\text{diff}}(\boldsymbol{T}_{n+1}) - \boldsymbol{f}_{\text{RE}}(\boldsymbol{T}_n) - \boldsymbol{f}_{\text{vol}} = \boldsymbol{0}. \tag{22}$$

Note that the radiation and evaporation boundary terms are evaluated for the previous time step. Since the implicit scheme is only used for the interlayer cool down phase (which does not exhibit high temperature gradients and rates compared to the active laser phase), this choice has no influence on the robustness and accuracy of the solution as verified by our investigations. To solve the nonlinear system of equations in residual form (22) for the unknown temperatures $\boldsymbol{T}_{n+1}$ a Newton–Raphson scheme is used:

$$\begin{aligned} &\underbrace{\left(\frac{1}{\Delta t}\boldsymbol{C} - \frac{\partial \boldsymbol{f}_{\text{diff}}}{\partial \boldsymbol{T}_{n+1}}(\boldsymbol{T}^i_{n+1})\right)}_{\boldsymbol{J}^i_{r,n}} \Delta \boldsymbol{T}^{i+1}_{n+1} \\ &= -\underbrace{\left(\frac{1}{\Delta t}\boldsymbol{C}(\boldsymbol{T}_{n+1} - \boldsymbol{T}_n) - \boldsymbol{f}_{\text{diff}}(\boldsymbol{T}_{n+1}) - \boldsymbol{f}_{\text{RE}}(\boldsymbol{T}_n) - \boldsymbol{f}_{\text{vol}}\right)}_{\boldsymbol{r}^i_n}, \end{aligned} \tag{23}$$

$$\boldsymbol{T}^{i+1}_{n+1} = \boldsymbol{T}^i_{n+1} + \Delta \boldsymbol{T}^{i+1}_{n+1}, \tag{24}$$


not yet activated cells
active layer
d_{HAZ} mandatory refinement
refinement based on material history
coarsest cell
refinement due to different porosity
refinement due to 2:1 balance
build direction
refinement captures part shape
refinement due to 2:1 balance

a) boundary-fitted mesh

b) build chamber mesh

Fig. 2. Adaptive mesh refinement concept applied in AM context. The proposed framework can work with (a) boundary-fitted meshes and (b) build chamber meshes.

where $\boldsymbol{J}^i_{\boldsymbol{r},n} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{T}_{n+1}}(\boldsymbol{T}^i_{n+1})$ is the Jacobian of the residual evaluated at the current temperature iterate $\boldsymbol{T}^i_{n+1}$. This iterative scheme (23)–(24) is applied until convergence of the residual (22) is achieved up to a prescribed tolerance. In this contribution, we use the implicit scheme with matrix-free evaluation in combination with an infrequently updated preconditioner (incomplete LU-factorization) for the linear solver only to solve the large time steps in the interlayer cool down phase. While technically, this scheme could be used to obtain a stable result for large time step sizes during the active laser phase, this is not done in this work for the accuracy reasons stated in (20). In Appendix A, we demonstrate temporal convergence and robustness of the combined time integration scheme.

4. High-performance implementation

The numerical model summarized in the previous section is implemented in an in-house research code based on the `deal.II` finite element library [42]. General-purpose functionality developed in the context of this work has been contributed to the main `deal.II` repository. In this section, we discuss implementation details. Note, that in this section the term *process* refers to either an *evaluation process* (the act of computing a result, an algorithm) or a *computer process* (a program instance executed by the CPU). It does not refer to the simulated AM process.

4.1. Mesh adaptivity and layer activation

Adaptive meshes can enable large savings in CPU time and memory usage and ultimately speed up the solution time considerably. For the present AM application, we can predict *a priori* when and where a fine mesh is needed without the need for an *a posteriori* error estimator. Therefore, we suggest the following procedure for mesh adaptivity and activation of new layers as illustrated in Fig. 2.

The complete part geometry is created in the beginning and meshed with a coarse mesh consisting of hexahedral cells of uniform edge length. In the current implementation, the edge length h_{coarse} of the coarse mesh should be related to the desired powder layer height h_{powder} via

$$h_{\text{coarse}} = 2^{n_{\text{refine}}} \cdot h_{\text{powder}}, \tag{25}$$

such that n_{refine} is the number of necessary isotropic refinements (by subdivision) of a coarse cell to obtain cells of the same height as the powder layer h_{powder}. This procedure allows for a straight-forward application of new powder layers as the boundaries of the powder layer always coincide with cell boundaries. The coarse mesh size h_{coarse} (or, equivalently, the number of refinements n_{refine}) should be chosen as large as possible to realize the largest computational savings. This approach works well for the geometries investigated so far and allows for a simple transfer of data across matching mesh hierarchies. Should more complex meshes be necessary, one could relax the constraint (25) and use more general transfer operations (e.g., based on mortar meshtying schemes) between potentially non-matching meshes as demonstrated in our previous work [29].

For mesh generation, one option is a coarse mesh which directly represents the final part geometry as a boundary-fitted voxel mesh. In this case, no surrounding powder is modeled, i.e., the boundaries of the coarse mesh are the boundaries of the part, see Fig. 2(a). This approach is justified by the very low thermal conductivity of the powder, which can be approximated by a thermally isolating boundary condition. Alternatively, the coarse mesh can represent a powder-filled build chamber and the boundaries of the coarse mesh can be interpreted as the boundaries of the build chamber. In this case, the final part geometry is defined implicitly from the consolidation status of every material point, see Fig. 2(b). Both geometry descriptions are possible within our framework and they both have specific advantages: the boundary-fitted coarse mesh allows to coarsen most cells that are far away from the currently scanned layer but generation of such a coarse mesh that still represents the part shape accurately can be cumbersome. On the other hand, the non-fitted build chamber mesh is trivial to generate as the domain will be a cuboid. However, there is a certain overhead in areas that are meshed, although they are not necessary for the representation of the final part shape and, in addition, cells close to the implicit part boundary need to stay refined throughout the simulation to capture the final part shape. We present examples utilizing both meshing approaches. Note that the approximation of the part geometry can also be realized via other methods, e.g., the finite cell method [30] or CutFEM [65,66].

Whenever a new powder layer is added, refinement, coarsening and activation of cells takes place according to the following rules: cells are refined such that the currently scanned, top-most layer is represented with a desired number of cells over the layer height. Note that all cells in the current layer are refined upon activation, regardless of when or if the laser reaches them. This avoids the computational effort for frequent remeshing within a layer at the cost of slightly more DoFs. Cells in the heat affected zone (HAZ) – a few layers below the current layer – also stay refined. Cells which have a distance greater than d_{HAZ} (which we choose as $d_{\text{HAZ}} = 4h_{\text{powder}}$) from the current layer *may* be coarsened with the following restriction: for any set of eight cells, which are octants of a previously subdivided parent cell, coarsening only takes place if the material history state – the consolidation state r_c defined in (3) – across this set lies above a threshold of $r_{\text{coarsen}} = 0.9$. This restriction ensures that potential porosity defects are not smoothed out over neighboring cells of full density and that the part boundaries stay refined up to the necessary level when using a build chamber mesh. It should be mentioned that the employed `p4est` library enforces a 2:1 balance between refinement levels of neighboring cells, i.e. for any two neighboring cells the refinement level may differ by at most one.

To save computational resources, all cells that lie above the currently active layer are inactive and coarsened as much as possible.

No DoFs are assigned to them and they need *not* be evaluated; thus, they implicitly represent void. When discussing the parallel distribution of the cells, two aspects should be distinguished: first, how many processes should be used in total, and second, how to distribute the cells among the processes. In this work, the maximum number of unknowns determines the allocated number of processes. No dynamic resource allocation takes place in the current implementation and the same number of processes is used throughout the entire simulation. Due to the growing geometry, it seems reasonable to allocate more and more processes as the simulation progresses. However, such a dynamic resource allocation scheme is expected to save resources but not necessarily speed up the overall simulation, since we are not typically limited by the spatially distributed scale of the problem but rather by the temporal scale. In practical AM simulation setups, the number of CPUs will be increased until the scaling limit is reached, i.e., until no further speedup can be achieved due to an increasing communication overhead and the parallel efficiency drops below an acceptable threshold. Next, regarding the distribution of cells among all processes, the inactive cells are not weighted differently compared to active cells. Instead, all parallel processes receive roughly the same number of cells regardless of the computational effort within the cell. This implies that for large process counts some processes will not have any work in the initial layers. A first attempt at a weighted redistribution of the active cells, that tries to utilize more processes for actual work, did not result in a noticeable speedup, possibly due to non-negligible communication latency for such configurations introduced by the specific Z-curve ordering of cells [45]. Thus, detailed investigations of these aspects are left for future work. The interested reader is referred to [39], where the performance of a similar AMR strategy was also rather insensitive to a weighted partitioning.

In the context of AM problems, the new `deal.II` class `FieldTransfer` was developed which allows to transfer global state vectors between meshes after coarsening and refinement in the presence of inactive cells (that do not have any DoFs). This class was initially developed by the authors of [67] and improved by us; it is thus applicable to various types of growing domain problems encountered when modeling different AM processes.

4.2. Fast operator evaluation

In the discrete problem statement in (16) and (23) we need to evaluate volume and boundary face integrals. Their efficient implementation uses the same techniques and, for the sake of brevity, we demonstrate the fast operator evaluation on the diffusive term in (13), which is a crucial term both in the explicit and implicit formulation of our problem. This global integral over the heat flux $\boldsymbol{q}$ can be transformed into a sum of element-level integrals, which are assembled into a global vector $\boldsymbol{f}_{\text{diff}}$:

$$\begin{aligned}(\nabla_{\boldsymbol{x}} v, \boldsymbol{q})_{\Omega} &= \sum_e (\nabla_{\boldsymbol{x}} v_e, -k(T_e)\nabla_{\boldsymbol{x}} T_e)_{\Omega_e} \\ &= \sum_e v_{e,i}\, \underbrace{(\nabla_{\boldsymbol{x}} N_i, -k(T_e)\nabla_{\boldsymbol{x}} T_e)_{\Omega_e}}_{f_{e,i,\text{diff}}} = \sum_e \boldsymbol{v}_e^T \boldsymbol{f}_{e,\text{diff}} =: \boldsymbol{v}^T \boldsymbol{f}_{\text{diff}},\end{aligned} \tag{26}$$

where the index e indicates a quantity restricted to a single element and the operator $\nabla_{\boldsymbol{x}}$ is used to represent the gradient in physical space. Following [55], we now break down the computation further:

$$\begin{aligned} f_{e,i,\text{diff}} &= \int_{\Omega_e} (\nabla_{\boldsymbol{x}} N_i)^T (-k(T_e)\nabla_{\boldsymbol{x}} T_e)\, \mathrm{d}\boldsymbol{x} \\ &= \int_{\Omega_e} (\boldsymbol{J}_e^{-T} \nabla_{\xi} N_i)^T (-k(N_a T_{e,a})(\boldsymbol{J}_e^{-T} \nabla_{\xi} N_b) T_{e,b}) \det \boldsymbol{J}_e \, \mathrm{d}\boldsymbol{\xi} \\ &\approx \sum_q \underbrace{(\nabla_{\xi} N_i)^T}_{(\boldsymbol{S}_{\text{grad}}^T)_{ic}} \underbrace{\boldsymbol{J}_e^{-1} w (\det \boldsymbol{J}_e) \boldsymbol{J}_e^{-T} (-k \underbrace{(N_a T_{e,a})}_{\boldsymbol{S}_{\text{val}} \boldsymbol{T}_e})}_{:=(\boldsymbol{D}_e)_{cd}} \underbrace{(\nabla_{\xi} N_b) T_{e,b}}_{(\boldsymbol{S}_{\text{grad}} \boldsymbol{T}_e)_d} . \end{aligned} \tag{27}$$

In the second line, the element integral is transformed from the physical space (parametrized by coordinates $\boldsymbol{x}$) into reference space (parametrized by coordinate ξ), thereby introducing the Jacobian mapping $\boldsymbol{J}_e$ between these spaces. Also, the temperature T_e is discretized in space with the element-wise shape functions N_a which correspond to the global spatial discretization introduced in (14). To shorten the notation, we employ the Einstein summation convention. In the third line, the integration is replaced by a weighted sum according to a numerical quadrature rule. Here, the shape functions N_i, their derivatives $\nabla_{\xi} N_i$ and the Jacobian mapping $\boldsymbol{J}_e$ are all evaluated at those quadrature points and w is the quadrature weight. Although the final Eq. (27) is lengthy, it illustrates the sequential nature of the element-level evaluation: first, we obtain the values and gradients of the temperature at the quadrature points through the interpolation $\boldsymbol{S}_{\text{val}}$ and $\boldsymbol{S}_{\text{grad}}$. In the implementation, we use a technique known as sum-factorization, which has been established in the spectral element community [68–70] and is available via `deal.II` [55,56]. Sum-factorization is especially beneficial for tensor-product shape functions of polynomial degree two and higher. However, in the present case of linear shape functions it provides more opportunities for the compiler to optimize code, e.g., through register blocking. Given the values and gradients of T, all physics-related operations happen on quadrature point level inside $\boldsymbol{D}_e$. In this example, we compute the heat flux from the nonlinear conductivity and thus need to evaluate the value of the temperature via $\boldsymbol{S}_{\text{val}}$. Finally, these quadrature point contributions are multiplied with $\boldsymbol{S}_{\text{grad}}^T$ (the shape gradients resulting from the test function) and summed up.

There is one missing link for the complete picture, namely the relation between element-level quantities and global quantities. For this purpose, we introduce a gather operation $\boldsymbol{G}_e$ which extracts local DoFs from a global vector via

$$\boldsymbol{T}_e = \boldsymbol{G}_e \boldsymbol{T}. \tag{28}$$

The transpose of the gather operations $\boldsymbol{G}_e^T$ scatters an element contribution back into a global vector such that we can write for the whole evaluation process:

$$\boldsymbol{f}_{\text{diff}} = \sum_e \boldsymbol{G}_e^T \boldsymbol{S}_{\text{grad}}^T \boldsymbol{D}_e \boldsymbol{S}_{\text{grad}} \boldsymbol{G}_e \boldsymbol{T}. \tag{29}$$

Note that internally $\boldsymbol{D}_e$ also requires the values of the temperature at quadrature points (computed via $\boldsymbol{S}_{\text{val}} \boldsymbol{G}_e \boldsymbol{T}$), since the conductivity k depends on it, see (27). Eq. (29) demonstrates how to assemble a global vector from cell-wise contributions. Interestingly, an equivalent strategy can be applied to compute a matrix–vector product without assembling the matrix first. As an example, we look at the matrix–vector product on the left-hand side in (23). Following the same steps as before and taking into account the definition of $\boldsymbol{C}$ and $\boldsymbol{f}_{\text{diff}}$ according to (13) and (15), we arrive at the following evaluation process:

$$\begin{aligned} \boldsymbol{J}_{\boldsymbol{r},n}^{i} \Delta\boldsymbol{T}_{n+1}^{i+1} &= \sum_e \boldsymbol{G}_e^T \boldsymbol{S}_{\text{val}}^T \left[\frac{\rho c}{\Delta t} \boldsymbol{I}\right] \boldsymbol{S}_{\text{val}} \boldsymbol{G}_e \Delta\boldsymbol{T}_{n+1}^{i+1} \\ &\quad - \sum_e \boldsymbol{G}_e^T \boldsymbol{S}_{\text{grad}}^T \left[\boldsymbol{D}_e \boldsymbol{S}_{\text{grad}} + \boldsymbol{L}_e \boldsymbol{S}_{\text{val}}\right] \boldsymbol{G}_e \Delta\boldsymbol{T}_{n+1}^{i+1}, \end{aligned} \tag{30}$$

$$\text{where } (\boldsymbol{L}_e)_c = \boldsymbol{J}_e^{-1} w \det \boldsymbol{J}_e \boldsymbol{J}_e^{-T} \left[-\frac{\partial k}{\partial T} \underbrace{(N_a T_{e,a})}_{\boldsymbol{S}_{\text{val}} \boldsymbol{T}_e} \underbrace{\nabla_{\xi} N_b T_{e,b}}_{(\boldsymbol{S}_{\text{grad}} \boldsymbol{T}_e)_c} \right] \tag{31}$$

Eq. (30) illustrates another feature of the evaluation process: in contrast to classical FEM implementations no global matrix is assembled. Instead, we directly compute the effect of the Jacobian $\boldsymbol{J}_{\boldsymbol{r},n}^{i}$ on the vector $\Delta\boldsymbol{T}_{n+1}^{i+1}$ and obtain the result as another global vector. Thus, the global Jacobian needed in the implicit time integration scheme (23) does not need to be assembled explicitly. Due to the aforementioned property the whole evaluation process is often termed *matrix-free* evaluation. Essentially, this process moves the bottleneck from the memory

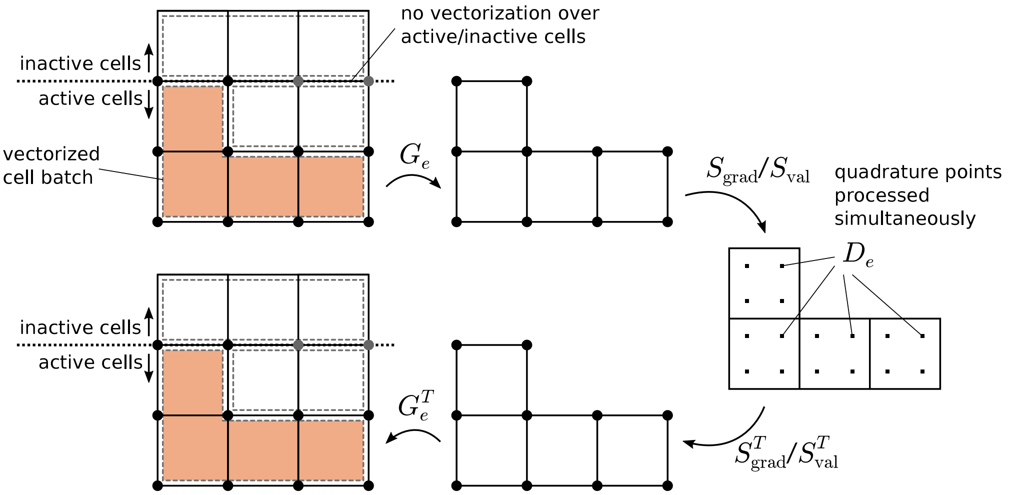


Fig. 3. Illustration of fast operator evaluation on a vectorized cell batch. The operation $\boldsymbol{D}_e$ is simultaneously performed on the same relative quadrature points in a vectorized cell batch with vectorized CPU instructions.

transfer of matrix entries (matrix-based implementation) to a more intense calculation (repeated calculation of (30) for every matrix–vector product). This trade-off is often preferable in modern hardware and especially pronounced for higher-order shape functions [71]. Similar to [72] we use the presented algorithm for explicit time integration in (29) (which is inherently matrix-free) due to its mature and optimized implementation in `deal.II` [55,56].

The process illustrated so far is very general and does not require any assumptions on the integrated term. In fact, the diffusive term includes nonlinear and history-dependent behavior which can be evaluated at each quadrature point. Furthermore, the procedure transfers seamlessly to (boundary) faces and integrals over them. The different steps of the evaluation still allow for a multitude of optimizations that can be chosen under specific circumstances: such optimization strategies and their trade-offs are discussed in detail in [56]. Some algorithmic aspects of special relevance to this article are presented in the following subsections.

Remark. The various operations that make up the whole evaluation process described in this section have been written in a matrix–vector product notation to better illustrate the process. This has only been done for readability, as they are not necessarily implemented with matrix–vector products as this does not lead to the most efficient implementation as discussed in [55,56].

Remark. Note that we can still assemble a global matrix via the matrix-free evaluation in (30) (e.g. to construct a reusable matrix-based preconditioner) by multiplying with each unit vector and assembling the result vectors as the columns of a global matrix.

4.2.1. Vectorization over cell batches

Modern CPUs come with single-instruction-multiple-data (SIMD) capabilities for basic arithmetic operations and loading and storing of memory. For instance, the AVX2 instruction set architecture allows to perform a single operation on 4 different values of type `double` at the same time. The more recent AVX512 instruction set even allows for 8 values of type `double`. Although optimizing compilers will try to identify loops that benefit from a vectorization of instructions on their own, the effectiveness of compiler optimizations depends on the specific implementation of the FEM model. If one were to rely only on the compiler for auto-vectorization and if the quadrature point operation is simple enough, the compiler might vectorize the operations done at a single quadrature point or reorder operations to process operations at different quadrature points of the same cell simultaneously. However, data dependencies and non-unit-stride access often prevent this automatic approach from utilizing SIMD effectively, and much better performance is possible by processing the operations of different cells in each vector lane, as demonstrated in [56]. This outer-loop vectorization ensures that all SIMD lanes can be filled with useful work, avoiding remainder loops or mask operations that a compiler would generate for auto-vectorization. In this contribution, intrinsics-based explicit vectorization is facilitated by abstractions of the `deal.II` library, without leaking the details of the loop for vectorization to the user code.

Fig. 3 illustrates how the operator evaluation described in the last section works on a batch of cells simultaneously. The number of cells in a batch is determined by the hardware which supports a number of lanes n_{lanes}. In the illustration the vectorization width is 4. In our PBFAM application, vectorization is performed separately over the activated cells in/below the current layer and the inactive void cells above (where we do not evaluate the weak form but perform post-processing operations for visualization). The vectorization concept also extends to the evaluation of integrals on (boundary) faces of cells.

For the full performance of the vectorized instructions, the nonlinear history behavior of the material is fully vectorized. For linear shape functions, the quadrature point operation constitutes more than half of the arithmetic work. The original formulation [29] contains branching conditions (e.g. in the liquid fraction evaluation (2)) which is often implemented with `if`-statements in unvectorized codes. These statements can be reformulated via masking operations. For a value a let us denote its vectorized version as $\check{a}$ and access to entry i by $\check{a}[i]$. For instance, we can define various masks that filter for temperatures in a given interval:

$$\check{M}_{T<T_s} = \check{\text{mask}}_<(\check{T},\check{T}_s,\check{1},\check{0}), \tag{32}$$

$$\check{M}_{T_s<T<T_l} = \check{\text{mask}}_>(\check{T},\check{T}_s,\check{1},\check{0}) \cdot \check{\text{mask}}_<(\check{T},\check{T}_l,\check{1},\check{0}), \tag{33}$$

$$\check{M}_{T>T_l} = \check{\text{mask}}_>(\check{T},\check{T}_l,\check{1},\check{0}), \tag{34}$$

$$\text{where } \check{\text{mask}}_\square(\check{a},\check{b},\check{t},\check{f})[i] = \begin{cases} \check{t}[i], & \text{if } \check{a}[i] \square \check{b}[i], \\ \check{f}[i], & \text{otherwise} \end{cases} \quad \text{for } 0 \le i < n_{\text{lanes}}. \tag{35}$$

Each of these masks $\check{M}_C$ contains a one in every lane where the condition C in the subscript is true, and a zero otherwise. To get a filtering effect, the masks can be multiplied with any quantity that should only be considered when the condition is true. The C++ language supports operator overloading such that the vectorized result of the $\check{\text{mask}}$-function and the final masks $\check{M}_C$ support the usual arithmetic operations. Note that the $\check{\text{mask}}$-function is implemented with intrinsic

SIMD-calls specific to a given architecture and (35) only documents its behavior. In our framework, the complete material behavior and weak form are consistently implemented on vectorized data types. For instance, a vectorized version of the liquid fraction (2) can be written with the help of the masks (32)–(34) as

$$\check{g}(\check{T}) = \check{M}_{T<T_s} \cdot \check{0} + \check{M}_{T_s<T<T_l} \cdot \frac{\check{T} - \check{T}_s}{\check{T}_l - \check{T}_s} + \check{M}_{T>T_l} \cdot \check{1}. \tag{36}$$

This illustrates that we can compute the liquid fraction for n_{lanes} quadrature points simultaneously, while the computational cost is comparable to a single quadrature point evaluation in the unvectorized implementation. The liquid fraction $\check{g}$ is used to compute the consolidated fraction history variable $\check{r}_c$ according to (3) which needs to be stored for every quadrature point and is transferred to a new mesh upon coarsening.

4.2.2. Further aspects

So far, the discussion of performance has been about the cell-local operation of a single operator evaluation only. However, we would like to note that we do not evaluate the terms of the weak form (13) separately as the notation in (30) might suggest at first glance. Instead, the gather and scatter operations as well as the interpolation and integration operations are performed only once for every cell, and they are combined with all the different quadrature point contributions (e.g. $\frac{\rho c}{\Delta t}\boldsymbol{I}$, $\boldsymbol{D}_e$ and $\boldsymbol{L}_e$ in (30)). Thus, only one loop over all cells is required, increasing data locality.

Additionally, we have adopted a concept from [73] which allows to load $\boldsymbol{T}_n$ only once from main memory during the evaluation of (16). For this purpose, we interleave cell operations and operations run on ranges of indices. Before an index i is first used in the source vector $\boldsymbol{T}_n$, we clear the content of the destination vector for this index, $T_{n+1,i} \leftarrow 0$. Then, as an intermediate result, we assemble the right-hand side $T_{n+1,i} \leftarrow f_i(\boldsymbol{T}_n)$ via the fast operator evaluation. After all contributions to an index in this vector have been added, we run the update of the temperature vector $T_{n+1,i} \leftarrow T_{n,i} + \Delta t \tilde{C}_{ii} T_{n+1,i}$. This update is cache-efficient since $T_{n,i}$ and $T_{n+1,i}$ are likely still in the cache from the operator evaluation.

For the parallel, distributed computation we utilize MPI (Message Passing Interface). To see the highest possible throughput on the global level, the implementation in `deal.II` takes care of MPI communication and overlaps it with local computations.

By basing our work on the high-performance implementation for operator evaluation in the `deal.II` library and actively contributing new features developed for our specific application, we are well-equipped for future extensions of our framework. The main feature that was added to the `deal.II` library in the context of this work is related to growing geometries by activating cells. The fact that some cells are inactive and do not carry any DoFs means that they must be skipped within the matrix-free evaluation framework. However, the interface between the active cells in the top-most layer and the inactive cells above them represents a boundary for the currently active domain and we want to evaluate the boundary conditions (11) and (12) on these internal faces. These challenges are solved by the new class `ElementActivationAndDeactivationMatrixFree`, which allows to ignore non-active cells and perform integrals at faces shared by active and non-active cells.

5. Results and discussion

All examples are run on our own compute cluster which consists of 52 compute nodes with a dual-socket Intel Xeon E5-2680 v3 CPU with 2 × 12 cores running at 2.5 GHz and 8 DDR4 memory channels running at 2.13 GHz (measured STREAM memory bandwidth of 82 GB/s). For this hardware the code is compiled with the AVX2 instruction-set extension. Importantly, we compared our model implementation on this hardware with a simple Laplace operator with constant coefficient implemented within the matrix-free `deal.II` framework. On one compute node, a full explicit time step of our model implementation reaches 44% of the throughput of the Laplace operator. Note that our explicit operator performs significantly more computations (nonlinear material evaluation) and needs to load more data (material history data on quadrature points) than the Laplace operator. Thus, we can say that our implementation is already well-optimized and only further improvements in `deal.II` might give additional speedup in the future.

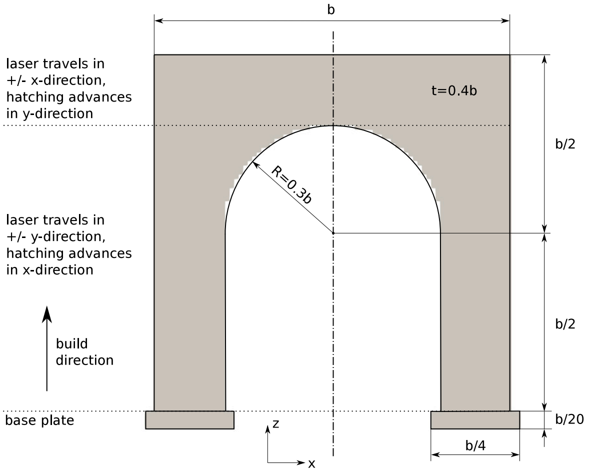

b
laser travels in +/- x-direction, hatching advances in y-direction
t=0.4b
R=0.3b
b/2
laser travels in +/- y-direction, hatching advances in x-direction
build direction
b/2
base plate
z
x
b/4
b/20

Fig. 4. Bridge geometry: outline of ideal geometry design (solid line) and discretized voxel geometry (gray area). This geometry is investigated for different scalings controlled by a single parameter b.

5.1. Bridge example

As a first example we investigate a bridge geometry, schematically depicted in Fig. 4. We use a boundary-fitted voxel mesh that approximates the arc with a stair case profile. The geometry is parametrized by a single parameter b to study different scalings of the problem. The coarsest cell size, which is equal to the voxel discretization size, is computed as $b/80$. The bottom of the base plate is kept at a fixed temperature T_0. The top surface (with positive z normal vector) is subject to radiation (11) and evaporation (12) boundary conditions. All other parts of the boundary are assumed to be thermally insulating, since they would be surrounded by powder (not modeled). Note that the small base plate section is intentionally reduced in size compared to a large, realistic base plate with dimensions in the decimeter scale, since we want its size to also scale with the parameter b. For the studies conducted in this work we found from our previous work [29] that the effect of a large base plate on the global temperature response is negligible. Our framework is capable to include a large base plate, which is adaptively refined in the vicinity of the attached parts, should this become necessary in future validation examples.

The scan pattern consists of serpentine tracks. The laser beam input parameters are given in Table 1. For the material behavior and radiation and evaporation boundary conditions, we choose values representative of the metals used in application, see Table 2. With these material parameters we obtain $\Delta t_{\max} = 2.9 \times 10^{-4}$ s as an estimate for the critical time step from condition (19). The actually used time step is $\Delta t = 2.0 \times 10^{-5}$ s so that the laser beam travels half a cell (less than one laser beam radius of $R = 50$ μm) within one step. As mentioned earlier, for these material parameters and mesh sizes the stability limit (19) turns out to be not restrictive compared to the accuracy requirement of the moving heat source (20). After every layer, we simulate 1 s of interlayer cool down time as follows: the first

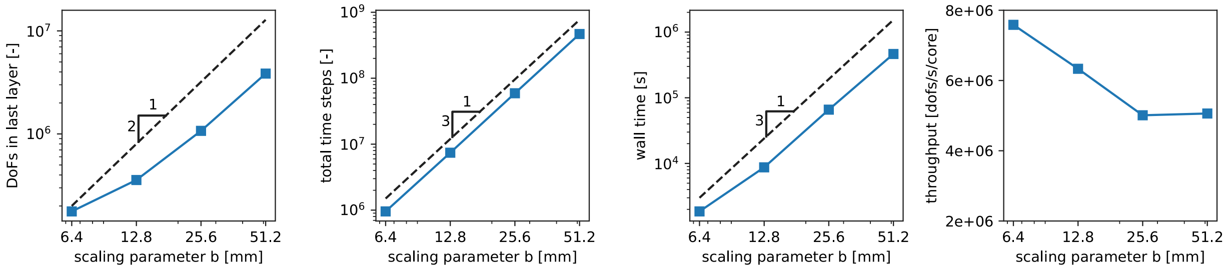


Fig. 5. Scaling study for bridge example. The number of DoFs increases quadratically, while the number of time steps increases cubically. The wall time increases cubically because the number of CPU cores is increased such that the number of DoFs per core stays approximately constant. Therefore, the throughput in DoFs per second per core stays roughly constant as well.

Table 1
Scan parameters for bridge example.

Symbol	Property	Value	Unit
v_{scan}	Scan velocity	1000	$\mathrm{mm\,s^{-1}}$
d_h	Hatch spacing	80	µm
R	Beam radius	50	µm
h_{powder}	Powder layer thickness	40	µm
t_{cool}	Cool down time	1	s

Table 2
Material parameters for bridge example. The parameters are representative for stainless steel and taken from our previous publication [10].

Symbol	Property	Value	Unit
k_{ms}	Thermal conductivity in melt and solid phase	20	$\mathrm{W\,m^{-1}\,K^{-1}}$
k_p	Thermal conductivity in powder phase	0.2	$\mathrm{W\,m^{-1}\,K^{-1}}$
ρ	Density	7430	$\mathrm{kg\,m^{-3}}$
c	Specific heat capacity	965	$\mathrm{J\,kg^{-1}\,K^{-1}}$
T_s	Solidus temperature	1500	K
T_l	Liquidus temperature	1900	K
T_0, T_∞	Initial and ambient temperature	303	K
ϵ	Emissivity	0.7	–
T_v	Boiling temperature	3000	K
C_P	Recoil pressure factor	54	kPa
C_T	Recoil pressure temperature factor	50 000	K
C_M	Heat loss temperature factor	0.001	$\mathrm{K\,s^2\,m^{-2}}$
M	Molar mass	0.052	$\mathrm{kg\,mol^{-1}}$
h_v	Latent heat of evaporation	6.0	$\mathrm{MJ\,kg^{-1}}$
$T_{h,0}$	Enthalpy reference temperature	663	K

1000 time steps of the cool down phase are still simulated with the explicit time integration scheme of the active laser phase to capture the highly dynamic behavior. Afterwards, the time step is increased to $\Delta t = 2.0 \times 10^{-2}$ s and the implicit time integration scheme is used to simulate the remaining 0.98 s of cool down time.

First, we perform a type of weak-scaling study, where we increase the dimension parameter b as indicated in Table 3 and at the same time increase the computational resources. The scaling of the mentioned quantities is also visualized in Fig. 5. The scalability of the implementation in the `deal.II` library has already been demonstrated [71,74]. The goal of this type of scaling study is to illustrate the various scaling effects that occur in a part-scale PBFAM simulation. Since we repeatedly increase the domain size by a factor of 2 in each dimension (via parameter b), the build volume increases cubically, i.e., by a factor of 8. Remember that – compared to many other works – we do not artificially scale up the heat source (nor the layer thickness) in this contribution. Therefore, the length of the complete laser track and consequently the number of time steps scales directly with the build volume, i.e., it increases by factor 8 as well (Fig. 5, panel 2). Notice that, as the domain grows larger and more refinement levels are necessary, the number of cells and total DoFs only increases by a factor of approximately 4 per scaling step (Fig. 5, panel 1). This is a consequence of AMR: the number of top-most layers with the highest refinement is constant. When the geometry is scaled up, a relevant increase of DoFs only happens in these top-most layers in the x- and y-directions

To understand how our framework behaves for growing domain sizes, we scale the computational resources by a factor of 4, i.e., the expected scaling of the number of DoFs in the later layers. The reasoning behind this choice is the good parallel scalability of the spatially distributed single-step evaluation: by scaling the computational resources by the same factor as the number of DoFs, we keep the work per process approximately constant in the later scaling steps. Therefore, assuming weak scalability, we expect the wall time to grow proportional to the number of time steps, i.e., by a factor 8 per scaling step, as observed in Fig. 5, panel 3. Perfect weak scaling is observed for the last scaling step where the throughput, defined as

$$\text{throughput} = \frac{\text{number of DoFs}}{\text{eval time per step} \times \text{number of cores}}, \tag{37}$$

stays constant (Fig. 5, panel 4). Note that we cannot simply scale up the computational resources by an additional factor 8 to counteract the increased number of time steps, since the work is not parallelized in time.

As a second study, we investigate the strong scaling capabilities of the framework. To this end, the largest geometry of the first study (with $b = 51.2$ mm) is simulated with a varying number of CPU cores $n_{\mathrm{core}} \in \{24, 48, 96, 192, 384, 768\}$. Since we are now interested in the scalability over different layers, we only simulate 1000 steps of the active laser phase per layer. The results are transferable to the whole layer because layers are activated in full at the beginning of a new layer and all cells are immediately active (though still in powder state).

The resulting average evaluation time for a single step in different layers is shown in Fig. 6. The strong scaling is close to the ideal behavior in the higher layers and, in the lower to medium layers, we are approaching the scaling limit of 1×10^{-4} s reported in [71]. In the first layer, there is not enough work for the assigned cores such that an increased number of cores does not result in an equivalent speedup. This is further illustrated by the total throughput (measured in DoFs per second per core per time step) and the parallel efficiency η, defined here as

$$\eta = \frac{T_{\mathrm{ref}} N_{\mathrm{ref}}}{T_{\mathrm{scaled}} N_{\mathrm{scaled}}}, \tag{38}$$

where T_{ref} and N_{ref} are the wall time and resources used for a reference run (in this case, a run on one compute node with 24 CPUs). T_{scaled} is the wall time for a run with $N_{\mathrm{scaled}} = s N_{\mathrm{ref}}$ (s times more) resources. The parallel efficiency η is a measure for the efficient use of resources, where a value of 1 means the additional resources manifest in a perfect speedup. In Fig. 6, the parallel efficiency is mostly close to 1. It drops to around 50% for the first layer when run with the largest number of cores. However, in the last layer 1280, the parallel efficiency stays at around 90% even for the highest core count which justifies the use of these computational resources. Note that we sometimes see a parallel efficiency slightly greater than one, e.g. in layer 500. This happens since the work per process and especially the number of ghosted cells varies with the layer number and the number of processes which can lead to a case where communication overhead is slightly worse for smaller core counts.

Table 3
Weak scaling study for bridge examples. Geometry and mesh information as well as performance results (only for the active laser phase).

b [mm]	Build vol. [mm^3]	Layers	Cores	h_{coarse} [mm]	n_{refine}	Max. DoFs (per core)	Time steps	Wall time
6.4	59	160	12	0.08	1	177k (14.8k)	956,504	0.55 h
12.8	469	320	48	0.16	2	357k (7.44k)	7,425,200	2.7 h
25.6	3,749	640	192	0.32	3	1070k (5.57k)	59,187,712	18.3 h
51.2	29,991	1280	768	0.64	4	3850k (5.01k)	468,263,936	128.9 h

Table 4
Single-step solution time and relative speedup for different time stepping schemes and vectorization levels in the active laser phase. To get a direct comparison of evaluation costs, a linear problem is solved such that the implicit* scheme only performs one nonlinear iteration.

	26×10^3 DoFs/core			6.5×10^3 DoFs/core		
	No vectorization		4-wide SIMD	No vectorization		4-wide SIMD
Implicit*	1.07 s	$\xrightarrow{\times 1.79}$	0.598 s	0.323 s	$\xrightarrow{\times 1.77}$	0.182 s
	↓ ×99.1		↓ ×114	↓ ×89.2		↓ ×143
Explicit	0.0108 s	$\xrightarrow{\times 2.07}$	0.00524 s	0.00362 s	$\xrightarrow{\times 2.85}$	0.00127 s

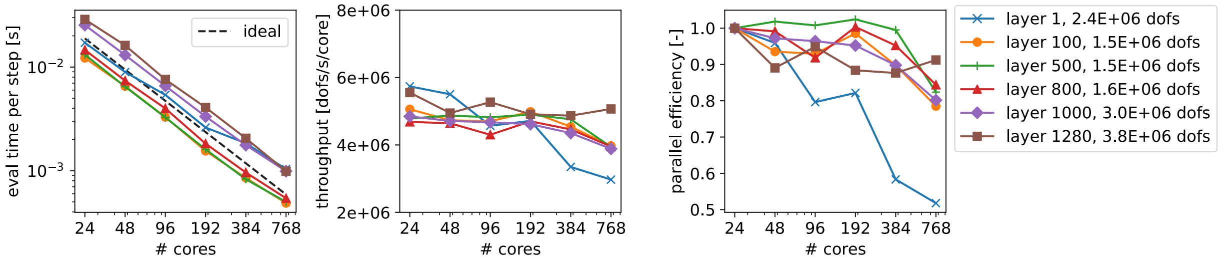


Fig. 6. Strong scaling study for different layers of 1280 layer bridge example: evaluation time for a single time step (left), throughput (middle) and parallel efficiency (right).

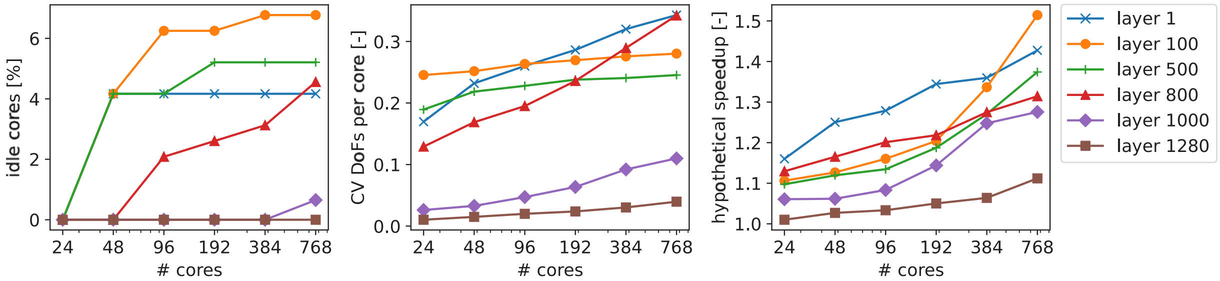


Fig. 7. Imbalance in work across cores in different layers of 1280 layer bridge example: percentage of idle cores without any DoFs (left), coefficient of variation (CV) of the DoFs per core (middle) and hypothetical speedup, if the DoFs were distributed evenly among cores and additional communication overhead is neglected.

More detailed insights into the imbalance of the DoF distribution among cores is shown in Fig. 7. Only a small fraction of cores is idle, i.e, has no DoFs at all. This fraction increases with larger core counts and decreases in the higher layers. As another metric, the coefficient of variation (CV) of the DoFs assigned to a core is defined as the ratio between standard deviation and mean of that same quantity. The CV reveals a strong imbalance in layer 800, which once again shows that the quality of the partitioning is layer-dependent. We can estimate an upper bound for the hypothetical speedup obtained by a better distribution, if we divide the maximum number of DoFs per core by the mean. This yields the speedup factor for an even distribution of DoFs among cores when additional communication overhead is neglected. The hypothetical speedup is at most 1.5 (layer 100 in Fig. 7), although it can never be fully realized and the estimate is very optimistic. Since this hypothetical gain is lower in most layers, we did not yet work on a more optimal parallel distribution in this paper. A simple weighting of active cells by a factor of 10 or 100 (compared to inactive cells) combined with a parallel redistribution when a new layer is activated did not produce a notable speedup which is in line with results reported in [39] for an adaptive mesh similar to the one investigated here.

In order to analyze the impact of different implementation aspects, we present relative speedup data in Table 4. This data was obtained from running the active laser phase in the last layer of the 1280 layer bridge example. For a fair comparison, we deliberately do not include a comparison with less optimized code and all cases use the matrix-free, highly-optimized code infrastructure from `deal.II`. Also, we choose the parameters such that the implicit system is linear and solved within a single nonlinear iteration to get a better comparison of the inherent cost associated with an implicit linear solve step. Since the actual problem is nonlinear, in practice, the evaluation costs for an implicit scheme are even higher when a few nonlinear iterations are required. As Table 4 reveals, an explicit step is around 100 times cheaper than an implicit step. However, it should be emphasized that our implicit scheme together with the preconditioner for the linear solver has not been optimized to the same degree as the explicit scheme since it does not present a bottleneck when only using it for the cool down phase. The speedup obtained from vectorization is more pronounced for the explicit than the implicit scheme since the operator evaluation is fully vectorized while the implicit linear solver also contains unvectorized code. Notably, for the explicit scheme the benefit of vectorization is higher at a lower load per CPU core because the problem is small enough to fit in the cache which in turn makes SIMD parallelism more impactful. The results further illustrate how the implementation approach scales well also for low loads per CPU core.

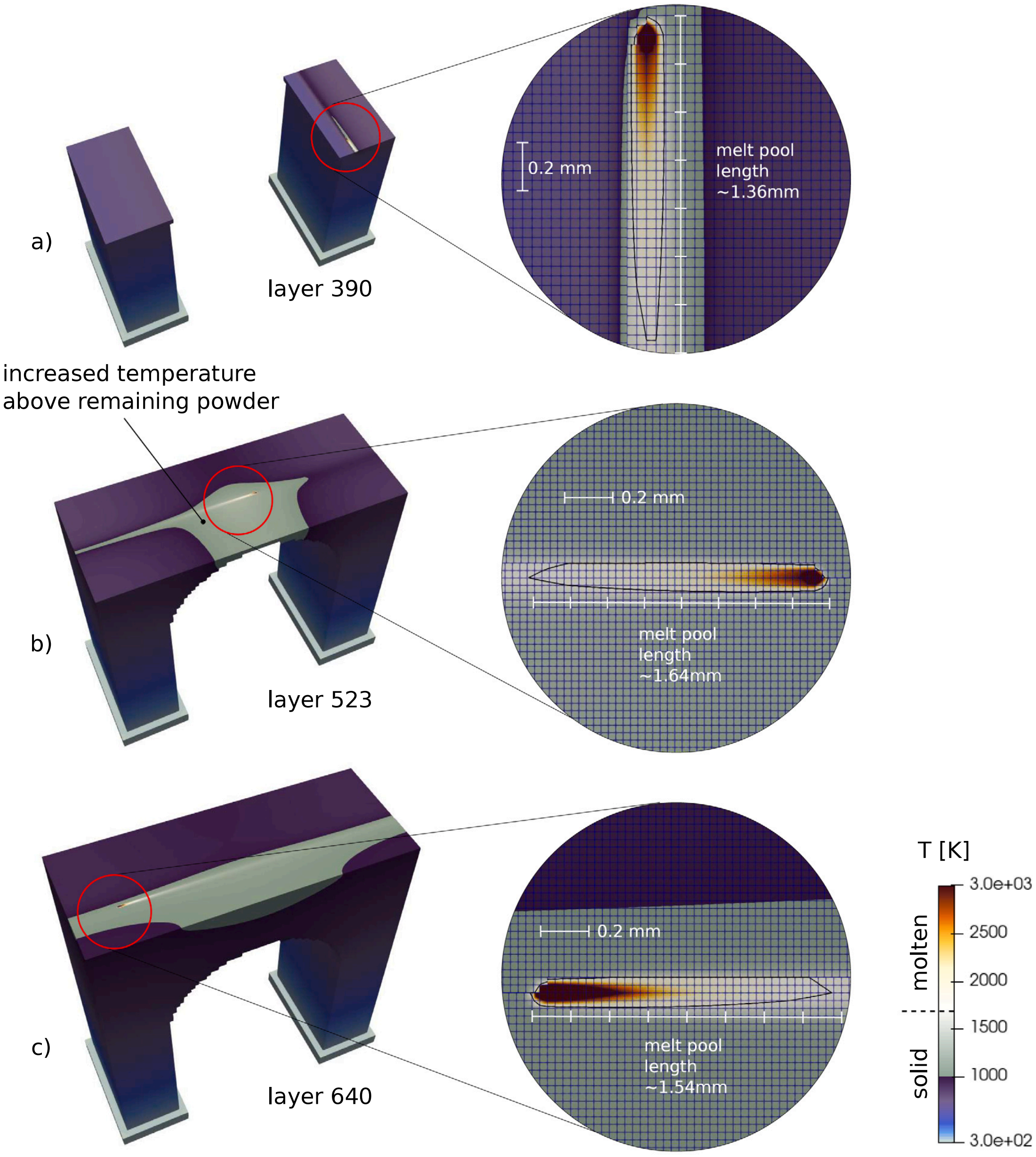


Fig. 8. Temperature distribution in 640-layer variant of the bridge geometry at different stages of the build process. Regions with temperature above T_m are defined as 'molten' (surrounded by a black solid line), otherwise as 'solid'. The heat affected zone (HAZ) with temperatures between 1000 K and melt temperature is visualized in gray color.

To conclude this section, we show the temperature distribution on the bridge geometry with 640 layers in Fig. 8. In the beginning of the AM process, when the two legs are still separated, the region of high temperatures in the solid material (indicated in gray color) is localized around the melt pool (Fig. 8(a)). Once the legs join up into a continuous layer the heat affected zone (HAZ) – with temperatures between 1000 K and melt temperature visualized in gray color – stretches over the strongly overhanging middle region on top of a powder domain, which is thermally insulating in good approximation (Fig. 8(b)). Note the complex and asymmetric shape of this region, which can only be captured by a scan-resolved simulation as performed in this work. In the future, the model can be extended to predict the influence of this overheated region on the microstructure evolution and, ultimately, residual stress formation. In the last layer (see Fig. 8(c)) the enlarged high-temperature region still persists. This is a result of the parameters chosen in this example, especially the cool down time between layers, which is 1 s in this example. As a result, the initial temperature when the scanning of a new layer begins, increases with increasing number of layers.

To further motivate the utility of the model in the present state let us mention a few more highly relevant and AM-specific effects that can be studied with it. As seen in this example, the melt pool length can be easily extracted from the temperature results (or, even be calculated while running the simulation) which allows a prediction of balling due to the Plateau–Rayleigh instability. An analysis of the peak temperatures would allow us to predict zones of excessive evaporation and gas-bubble-induced porosity. Residual porosity due to lack-of-fusion can be directly determined from the consolidation history (as shown in the next example). The presented model is still a part-scale model and all of the mentioned effects are captured in a qualitative manner.

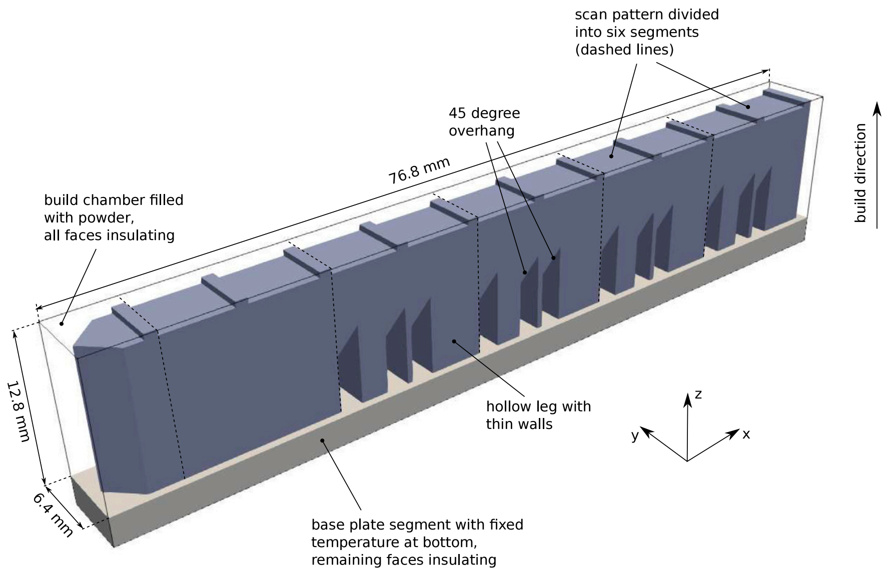


Fig. 9. Overview of NIST AM Benchmark 2022 cantilever geometry and special features. Detailed information about the dimensions, geometry and scan strategy can be found in [24].

Table 5
Scan parameters for cantilever example.

Symbol	Property	Value	Unit
v_{scan}	Scan velocity	960	$\text{mm}\,\text{s}^{-1}$
d_h	Hatch spacing	110	μm
R	Beam radius	60	μm
h_{powder}	Powder layer thickness	40	μm
t_{cool}	Cool down time	1	s

For a detailed quantitative analysis one needs to resort to mesoscale models.

5.2. Cantilever benchmark

As a second numerical example, we investigate the cantilever structure shown in Fig. 9, which was designed for the NIST AM Benchmark series 2022 [24]. The purpose of this example is not yet to validate the model against experimental measurements. Rather we want to demonstrate the capabilities of the framework on realistic geometries. Since the geometry is more complex than in the previous example, we use a build chamber mesh of dimensions $76.8 \times 6.4 \times 12.8\,\text{mm}^3$ which also discretizes the remaining powder. The path of the laser beam is used as an input for the active laser phase and the tracks are scanned into a box that encloses the desired geometry such that a small buffer of remaining powder lies around the final part. Such a tightly fitting build chamber mesh is deemed acceptable due to the negligible powder conductivity. A base plate section of $2.56\,\text{mm}$ thickness is added below the build chamber. Its bottom face is kept fixed at the initial temperature T_0.

The active scan phase in every layer is followed by a cool down phase of 1s, which uses the same time step sizes as described in the last section. After simulating all 312 layers, the built geometry is implicitly defined by the solid phase fraction at each quadrature point according to Eq. (4). The scan parameters are given in Table 5 while the same material parameters as for the bridge examples are reused, see Table 2.

The temperature distribution in the part at difference points in time of the process is shown in Fig. 10. A video of the complete process may also be found in the supplementary material of this article. In the first layers the temperature quickly drops to the initial temperature $303\,\text{K}$ after the laser passed, see Fig. 10(a–b). This fact can be explained by the small total heat capacity of the consolidated material and the small distance to the base plate with prescribed temperature at its bottom face. As more and more layers are processed, the residual temperature steadily rises but the HAZ with $T > 1000\,\text{K}$ (indicated in white color) is always limited to the direct vicinity of the melt pool (Fig. 10(c–d)). High temperature gradients are thus also limited to this area which justifies the use of a refined mesh only in these areas. A detailed view of the melt pool in layer 120 is shown in Fig. 11. Fig. 10(e) illustrates the different cooling rates resulting from the different geometrical features: the thin legs are at an elevated temperature compared to the thicker leg due to the smaller thermal conductivity (and heat flux concentrations) in these regions. An exception is the thick but internally hollow leg which exhibits an equally poor heat conduction to the base plate as the thin legs.

In layer 174 the initially separate legs join up into a continuous layer (Fig. 10(f)). The melt pool and the HAZ surrounding it are elongated when the beam travels over the overhanging regions and the hollow leg. A detailed view of the melt pool is shown in Fig. 12. For the depicted point in time, the laser beam travels across the hollow leg and previously across an overhang region which leads to an elongated melt pool and an enlarged HAZ due to the decreased conductivity to the base plate. This effect persists in higher layers whenever the laser beam is moving across the hollow leg (Fig. 10(g)). Note that such geometrical influences cannot be predicted with layer-wise part-scale models or melt pool models but only with scan-resolved, true part-scale models. The residual heat after cool down of the final part is shown in Fig. 10(h). In Fig. 13 we compare the final temperature distribution when the (previously used) cool down time of $1\,\text{s}$ or an (alternative) cool down time of $5\,\text{s}$ is used after every layer. The cool down time has a strong influence on the temperature level after cool down. A systematic investigation of the impact of cool down time is thus possible with the approach presented in this work since the real scan and cool down times are used consistently in this model (in contrast to layer-wise simulation approaches, where heating and cooling times are often calibration parameters).

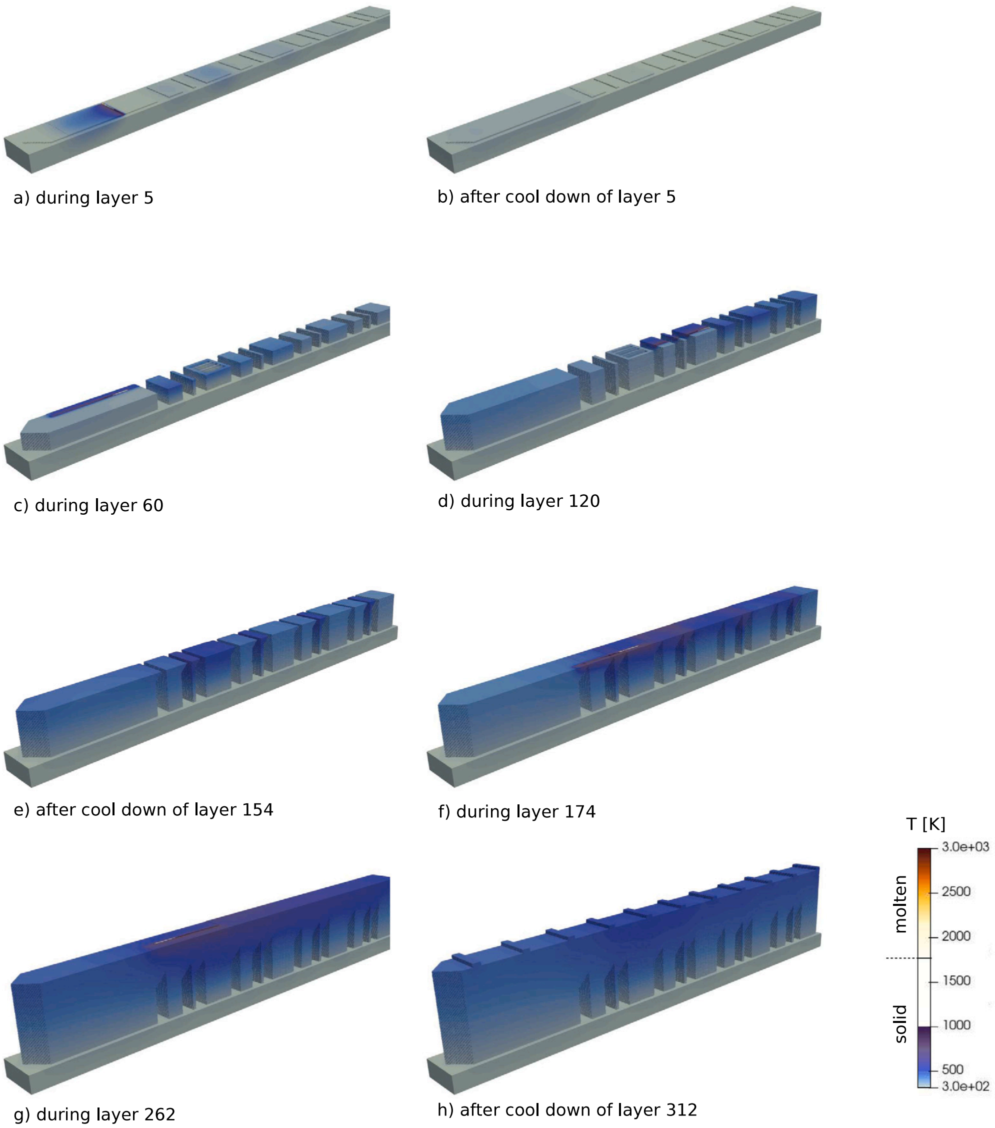


Fig. 10. Temperature distribution in cantilever at various stages in the build process. Regions with temperature above T_m are defined as 'molten', otherwise as 'solid'. A video of the complete process is attached as supplementary material to this article.

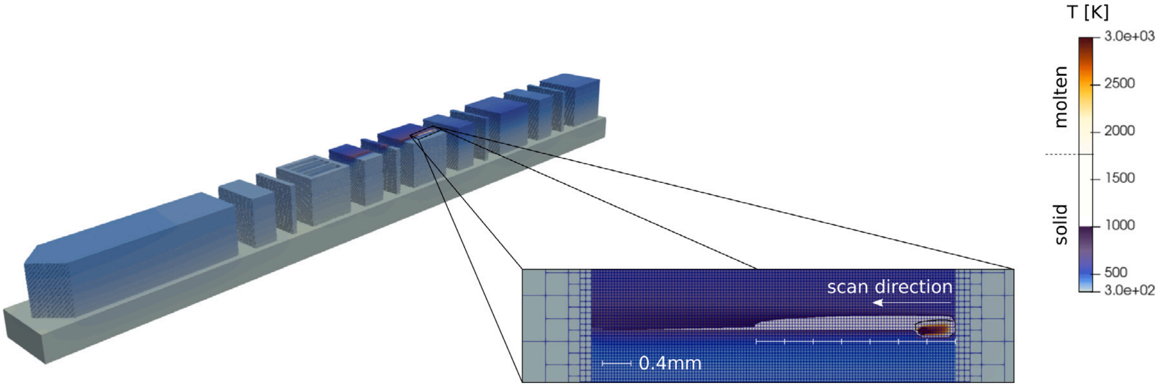


Fig. 11. Detailed view of the melt pool (indicated by solid black line) at a turning point and start of a new track segment during processing of layer 120. Regions with temperature above T_m are defined as 'molten', otherwise as 'solid'.

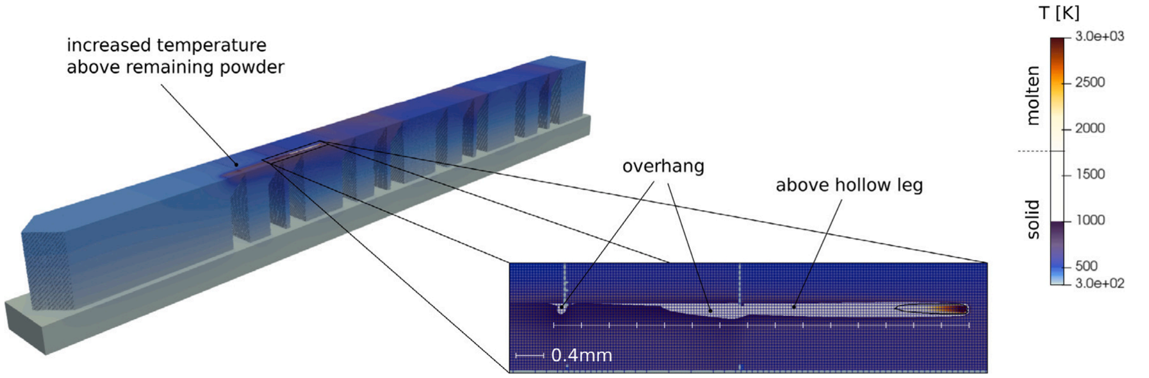


Fig. 12. Detailed view of the melt pool (indicated by solid black line) during processing of layer 174. In this layer the separate legs join up into a continuous layer. Regions with temperature above T_m are defined as ‘molten’, otherwise as ‘solid’.

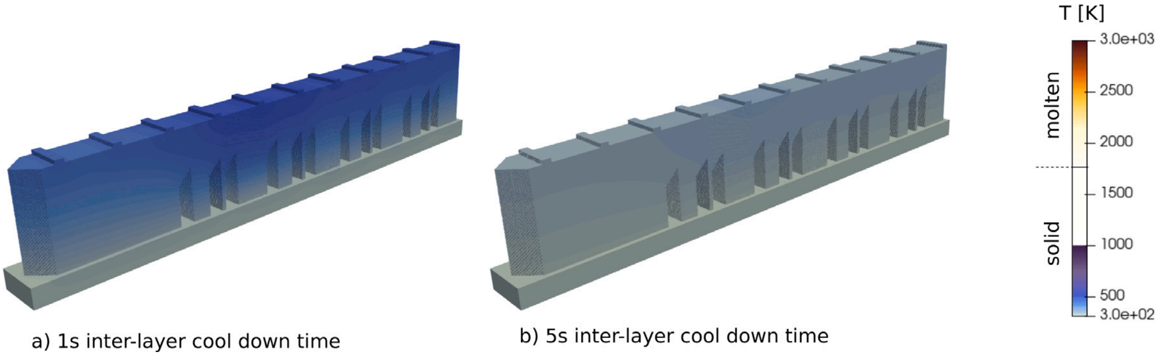

a) 1s inter-layer cool down time
b) 5s inter-layer cool down time
T [K]
3.0e+03
2500
2000
1500
1000
500
3.0e+02
molten
solid

Fig. 13. Residual heat after completed build process (a) for 1s cool down time and (b) for 5 s cool down time.

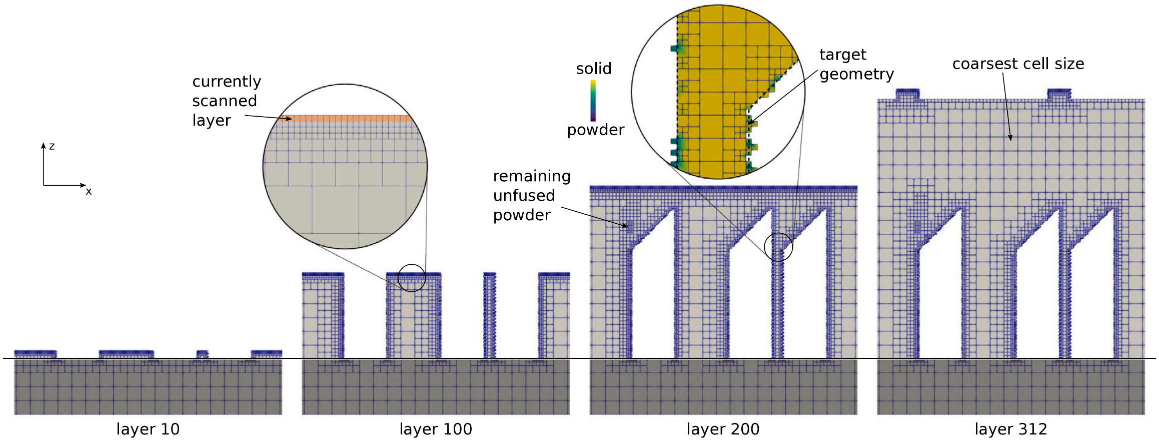


Fig. 14. Detailed view of the part buildup and adaptive mesh in the symmetry (xz-) plane of the cantilever. The finest mesh resolution is only kept when necessary to capture the final part shape. Any cell with more than 50% solid fraction is visualized. The baseplate is overlaid in dark gray.

To illustrate the adaptive meshing strategy, Fig. 14 shows the growing part geometry on a slice. The coarse mesh uses a cell size $h_{\text{coarse}} = 640\,\mu\text{m}$, i.e., four refinement levels are necessary to reach a cell size of h_{powder}. The number of DoFs grows linearly with the number of layers, see Fig. 16, with a visible kink once the initially separate legs of the cantilever join up into a continuous layer in layer 174. Since the interface surface between solid and powder is larger as long as the legs are separated and consequently more refined cells are necessary to capture this interface, the number of DoFs grows more quickly before layer 174 than afterwards. Fig. 14 also shows how well the build chamber mesh in combination with the adaptive mesh strategy can capture the final part geometry: the detail view in layer 200 overlays the target geometry outline over the built geometry in the overhang region. The solidified part shape agrees very well with this target geometry. Only in some places a small amount of partially molten powder sticks to the surface. Note that we can also capture an area where lack of fusion occurred and a refined mesh remains necessary. This illustrates a further aspect relevant for AM applications that can be predicted on the part-scale with the help of this model.

The example was run on the same hardware as the previous bridge example. The simulation of the process for the total build volume of $3770\,\text{mm}^3$ requires around 44 million time steps and a maximum of 8.7 million DoFs. With 480 CPU cores, the total simulation time was 51.9 h. As an extension to the strong scaling study performed in the last

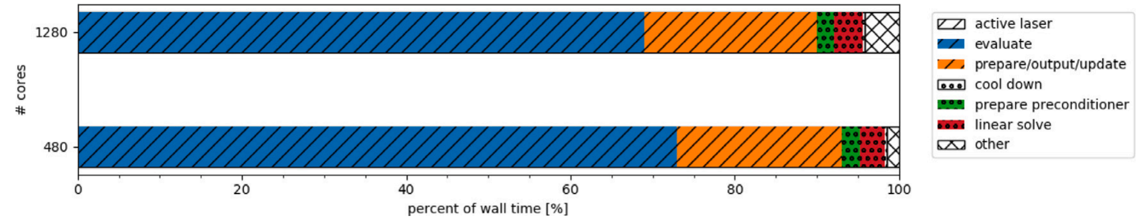


Fig. 15. Percentage of wall time spent in the different parts of the algorithm for the 1248 core simulation compared to the 480 core simulation. Timings averaged over all 312 layers.

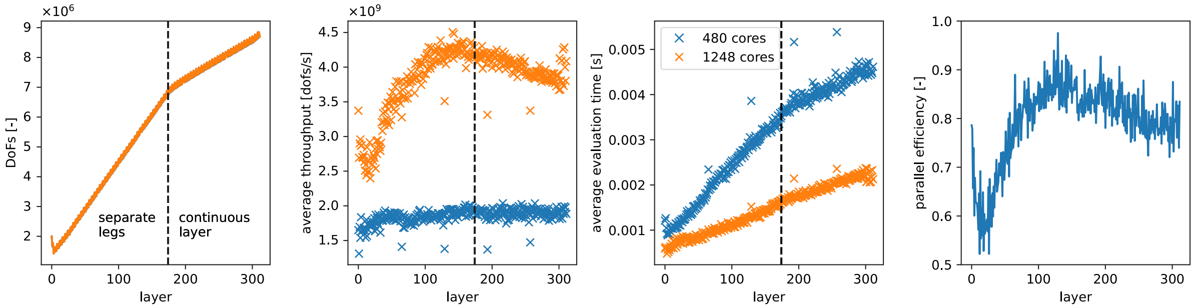


Fig. 16. Number of DoFs, throughput, average evaluation time per step and parallel efficiency over all 312 layers of the cantilever example for the 1248 core simulation compared to the 480 core simulation. Layer 174, where the separate legs join up into a continuous layer, is indicated by a dashed line. The plotted data only includes the active laser phase solved with the explicit time stepping scheme.

section we increased the number of cores to 1248 (all cores available on the test machine), giving a total simulation time of 23.7 h (which is an overall parallel efficiency of 84% compared to the previous run). In both cases, most of the time is spent in the active laser phase, with the cool down phase (1 s of simulated time per layer) only taking around 5% of the total wall time, see Fig. 15. The implementation of the cool down phase has not been optimized as much as the active laser phase in this contribution since we would not see a significant overall speedup in our numerical examples. Note that in another recent and performant model for scan-resolved part-scale analysis [40], the authors simulate only four sets of four successive layers of a similar cantilever geometry. No exact timings are given. To the best of the authors' knowledge, no other scan-resolved model has been published so far which simulates, in practice, the build process on the scale of a real part. For numerical examples with around 2 million DoFs other authors report single step solution times in the range of a few seconds in [39] or a few hundred milliseconds in [31]. By comparison, our presented approach is several orders of magnitude faster.

The average throughput of the evaluation in terms of DoFs per second and the average time for one time step are also shown in Fig. 16. The throughput in the first few layers for the high core count indicates that we initially underutilize the assigned computational resources. Indeed, when examining the parallel efficiency for every layer separately it becomes clear that the core count and distribution of the problem could be improved in the first layers. As already mentioned, in the future dynamic resource allocation could be used so that shared computational resources are only used when necessary. In the later layers we reach a very good parallel efficiency of 80%–90%, which still justifies the use of the increased computational resources.

Note that the throughput and evaluation time in Fig. 16 show outliers which occur exactly every 64 layers. In these layers, processes need a comparatively large number of ghosted information from other processes. This behavior can be linked back to the way `deal.II` distributes the cells and DoFs among processes [45], which might lead to non-contiguous subdomains and a non-uniform distribution of expensive hanging nodes [75]. Again, fine tuning might give a speedup in the problematic configuration but is not further investigated in this work since on a global view it would only give a negligible speedup.

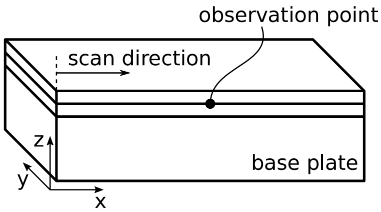


Fig. 17. Example used to judge temporal convergence.

6. Conclusion and outlook

A high-performance approach for the simulation of part-scale laser powder bed fusion additive manufacturing (LPBFAM) with a resolved scan track has been proposed. The physics-based model includes phase-dependent material parameters and consistent boundary conditions. The dynamic heat equation is discretized with an explicit time stepping scheme which has a smaller computational cost per time step and better parallel scalability compared to implicit schemes. The stability limit inherent to explicit schemes is found to be less restrictive than the restriction imposed by the moving heat source (which, in the scan-resolved regime, should not travel further than its own radius within one step).

We studied numerical aspects on basis of weak and strong parallel scaling tests. The implementation shows excellent scalability on a moderately-sized distributed compute cluster. Due to the explicit time stepping scheme and the high-performance implementation the time to solution for application-relevant problems is superior to other implementations in the literature that try to solve this problem. Notably, we achieve wall clock times per time step of a few milliseconds which is several order of magnitudes lower than the timings reported in other implementations in the literature.

Although we were able to reduce the cost of a single time step significantly within this work, scan-resolved simulation of LPBFAM parts on the scale of several decimeters or more most likely remains unrealistic due to the excessive number of time steps to be solved. Therefore, in future work the advances in this contribution could be

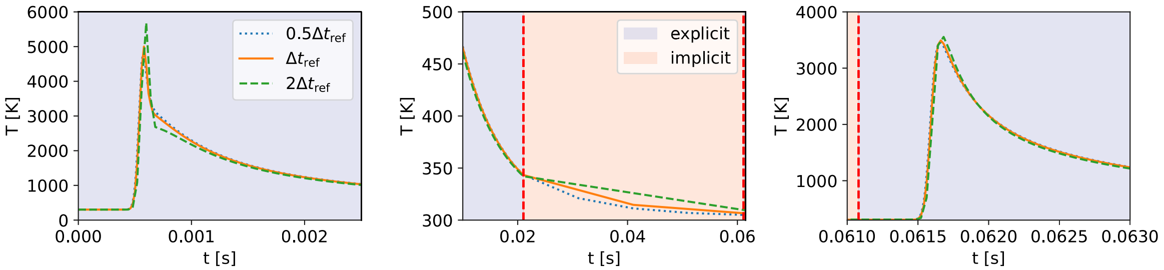


Fig. 18. Temporal convergence of temperature for the combined explicit-implicit time stepping scheme with different time step sizes. Left: detailed view of heat source moving over the observation point. Middle: switch from explicit to implicit scheme during cool down. Right: heat source in the second layer passes above the observation point.

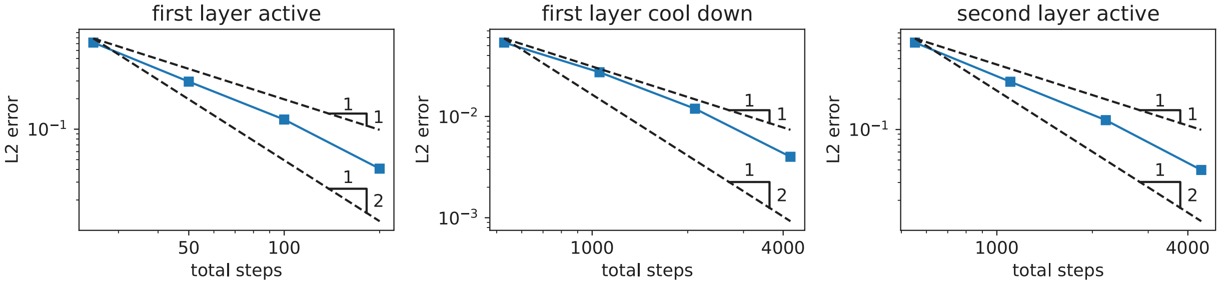


Fig. 19. Rate of convergence for time integration schemes. Left: convergence of the explicit scheme used during the active laser phase in layer 1. Middle: convergence of the implicit scheme used during the cool down phase after layer 1. Right: convergence of the explicit scheme used during the active laser phase in layer 2.

combined with techniques that try to tackle the temporal scale, such as parallel-in-time methods [47,52] or space–time formulations [76].

With the presented adaptive mesh refinement strategy, using either a boundary-fitted or build chamber mesh, we are able to simulate general problem settings of LPBFAM as demonstrated on two realistic AM geometries. Notably, we performed the first full scan-resolved simulation of the NIST AM Benchmark cantilever in just below one day. Since the framework does not make any strong physical assumptions that require detailed calibration (such as layer-wise heat source models would), the obtained results already show interesting physical effects that are relevant for designers. A validation with real material data against measurements is a next step. The proposed thermal simulation model can serve as a basis for microstructure predictions on the part-scale, but also to study the influence of scan pattern and part geometry on melt pool shape and temperature, which are important indicators for process defects. These opportunities have been indicated throughout the discussion of the results.

A natural extension of the current framework will deal with the thermo-mechanical problem. The groundwork has been laid in our contribution [29] and needs to be incorporated into the high-performance framework presented in this work. Matrix-free implementations with efficient solution strategies exist for the solid mechanics problem [77,78]. They will likely require application-specific adaptations to complement the high-performance implementation of the thermal problem presented in this work.

Although the current implementation can be said to be optimized when compared to a benchmark, a few performance-related topics for future investigation remain unresolved. In this work, we only looked at performance on CPUs. Due to the increasing popularity and availability of powerful GPUs, a compliant implementation might make the methodology available to a wider audience. Using deal.II's GPU features, we are planning an extension of the framework in this direction. As we saw in the results sections, the required resources that can be efficiently used vary over the layers: in the first layer, many processes do not receive any work and if they do, the communication overhead is too high to justify their use. Dynamic reallocation of more CPU cores as the problem domain grows would free the claimed but unused resources for other users of a compute cluster during the processing of the earliest layers.

CRediT authorship contribution statement

Sebastian D. Proell: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation. **Peter Munch:** Writing – review & editing, Software, Methodology. **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 the following financial interests/personal relationships which may be considered as potential competing interests: Christoph Meier reports financial support was provided by German Research Foundation.

Data availability

Data will be made available on request.

Acknowledgments

The authors would like to thank Katharina Kormann for her work on the matrix-free infrastructure and Marc Fehling for his work on the distributed hp-adpative infrastructure in `deal.II`. Furthermore, we thank Maximilian Bergbauer and Niklas Fehn for valuable discussions on high-performance computing and the `deal.II` matrix-free implementation.

Funding

This work was supported by funding of the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within project 437616465 and project 414180263.

Appendix A. Temporal convergence

A representative example of two layers with a single track each is used to analyze temporal convergence of the time stepping scheme for the parameters used in the numerical examples. The geometry consists of a $1.0 \times 0.2 \times 0.2\,\mathrm{mm}^3$ base plate ($x \times y \times z$ dimensions) with two powder layers of $40\,\mu\mathrm{m}$ on top as illustrated in Fig. 17. Starting at ($x = 0$, $y = 0$), a single track is scanned along the $(1, 0, 0)$ direction in both layers with a $0.06\,\mathrm{s}$ cool down time in between layers. All other material and scan parameters are identical to Tables 2 and 5. The temperature at the observation point $(0.5\,\mathrm{mm}, 0.0, 0.24\,\mathrm{mm})$ (top of first layer, middle of track) is shown in Fig. 18 for three different time step sizes, where Δt_{ref} is the time step size used in the numerical examples ($2 \times 10^{-5}\,\mathrm{s}$ in the explicit phase, $2 \times 10^{-2}\,\mathrm{s}$ in the implicit phase). The switch between explicit and implicit time stepping and activation of a new layer is robust and the results are well converged for the desired level of accuracy in a part-scale model.

To judge the rate of convergence, we extend the example described above to another small time step size $\Delta t_{\mathrm{ref}}/4$ and compare the L2 norm of the difference between the temperature fields of every experiment and a reference solution computed with a time step size of $\Delta t_{\mathrm{ref}}/8$. The results for the explicit and implicit scheme are shown in Fig. 19 at selected points in time. The plot shows the expected linear convergence of both the explicit and implicit scheme. Note that we integrate the boundary terms explicitly, even in the implicit scheme, as indicated in (22). Due to their low contribution in the cool down phase, we still obtain the expected linear convergence rate. A more detailed discussion and proof of convergence for such a combined implicit–explicit Euler scheme can be found in [79].

The convergence rate appears to accelerate towards the right end of the graphs. It has been verified that this effect is caused by the finite precision of the reference solution. For the intermediate results (further to the left in the graphs), for which the reference solution is accurate enough, the expected $\mathcal{O}(\Delta t)$ behavior can be seen. The numerical values for the rate of convergence for the three points in time shown in Fig. 19 are, from left to right, 1.02, 0.96 and 1.04.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.addma.2023.103921. **Video 1:** Scan-resolved thermal simulation of the manufacturing of all 312 layers of a cantilever specimen. The interlayer cool down phase of $1\,\mathrm{s}$ is not visualized.

References

[1] I. Gibson, D. Rosen, B. Stucker, M. Khorasani, D. Rosen, B. Stucker, M. Khorasani, Additive Manufacturing Technologies, Vol. 17, Springer, 2021.

[2] D. Herzog, V. Seyda, E. Wycisk, C. Emmelmann, Additive manufacturing of metals, Acta Mater. 117 (2016) 371–392.

[3] M. Grasso, B.M. Colosimo, Process defects and in situ monitoring methods in metal powder bed fusion: a review, Meas. Sci. Technol. 28 (4) (2017) 044005.

[4] C. Meier, S.L. Fuchs, N. Much, J. Nitzler, R.W. Penny, P.M. Praegla, S.D. Proell, Y. Sun, R. Weissbach, M. Schreter, et al., Physics-based modeling and predictive simulation of powder bed fusion additive manufacturing across length scales, GAMM-Mitt. 44 (3) (2021) e202100014.

[5] 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, Annu. Rev. Heat Transfer 20 (2017).

[6] S.L. Fuchs, P.M. Praegla, C.J. Cyron, W.A. Wall, C. Meier, A versatile SPH modeling framework for coupled microfluid-powder dynamics in additive manufacturing: binder jetting, material jetting, directed energy deposition and powder bed fusion, Eng. Comput. (2022) 1–25.

[7] S.A. Khairallah, A. Anderson, Mesoscopic simulation model of selective laser melting of stainless steel powder, J. Mater Process. Technol. 214 (11) (2014) 2627–2636.

[8] C. Körner, E. Attar, P. Heinl, Mesoscopic simulation of selective beam melting processes, J. Mater Process. Technol. 211 (6) (2011) 978–987.

[9] M. Markl, R. Ammer, U. Rüde, C. Körner, Numerical investigations on hatching process strategies for powder-bed-based additive manufacturing using an electron beam, Int. J. Adv. Manuf. Technol. 78 (2015) 239–247.

[10] C. Meier, S.L. Fuchs, A.J. Hart, W.A. Wall, A novel smoothed particle hydrodynamics formulation for thermo-capillary phase change problems with focus on metal additive manufacturing melt pool modeling, Comput. Methods Appl. Mech. Engrg. 381 (2021) 113812.

[11] M. Russell, A. Souto-Iglesias, T. Zohdi, Numerical simulation of laser fusion additive manufacturing processes using the SPH method, Comput. Methods Appl. Mech. Engrg. 341 (2018) 163–187.

[12] H. Wessels, C. Weißenfels, P. Wriggers, Metal particle fusion analysis for additive manufacturing using the stabilized optimal transportation meshfree method, Comput. Methods Appl. Mech. Engrg. 339 (2018) 91–114.

[13] W. Yan, Y. Qian, W. Ge, S. Lin, W.K. Liu, F. Lin, G.J. Wagner, Meso-scale modeling of multiple-layer fabrication process in selective electron beam melting: inter-layer/track voids formation, Mater. Des. 141 (2018) 210–219.

[14] E. Herbold, O. Walton, M. Homel, Simulation of Powder Layer Deposition in Additive Manufacturing Processes Using the Discrete Element Method, Technical Report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2015.

[15] C. Meier, R. Weissbach, J. Weinberg, W.A. Wall, A.J. Hart, Critical influences of particle size and adhesion on the powder layer uniformity in metal additive manufacturing, J. Mater Process. Technol. 266 (2019) 484–501.

[16] C. Meier, R. Weissbach, J. Weinberg, W.A. Wall, A.J. Hart, Modeling and characterization of cohesion in fine metal powders with a focus on additive manufacturing process simulations, Powder Technol. 343 (2019) 855–866.

[17] R.W. Penny, P.M. Praegla, M. Ochsenius, D. Oropeza, R. Weissbach, C. Meier, W.A. Wall, A.J. Hart, Spatial mapping of powder layer density for metal additive manufacturing via transmission X-ray imaging, Addit. Manuf. 46 (2021) 102197.

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

[19] L.-E. 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.

[20] 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.

[21] 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.

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

[23] J. Zhang, F. Liou, W. Seufzer, J. Newkirk, Z. Fan, H. Liu, T.E. Sparks, Probabilistic simulation of solidification microstructure evolution during laser-based metal deposition, in: 2013 International Solid Freeform Fabrication Symposium, University of Texas at Austin, 2013.

[24] 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.

[25] N. Hodge, R. Ferencz, J. Solberg, Implementation of a thermomechanical model for the simulation of selective laser melting, Comput. Mech. 54 (1) (2014) 33–51.

[26] S. Kollmannsberger, A. Özcan, M. Carraturo, N. Zander, E. Rank, A hierarchical computational model for moving thermal loads and phase changes with applications to selective laser melting, Comput. Math. Appl. 75 (5) (2018) 1483–1497.

[27] D. Riedlbauer, T. Scharowsky, R.F. Singer, P. Steinmann, C. Körner, J. Mergheim, Macroscopic simulation and experimental measurement of melt pool characteristics in selective electron beam melting of Ti-6Al-4V, Int. J. Adv. Manuf. Technol. 88 (2017) 1309–1317.

[28] 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.

[29] 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 (2021) 1–37.

[30] M. Carraturo, J. Jomo, S. Kollmannsberger, A. Reali, F. Auricchio, E. Rank, Modeling and experimental validation of an immersed thermo-mechanical part-scale analysis for laser powder bed fusion processes, Addit. Manuf. 36 (2020) 101498.

[31] F. Dugast, P. Apostolou, A. Fernandez, W. Dong, Q. Chen, S. Strayer, R. Wicker, A.C. To, Part-scale thermal process modeling for laser powder bed fusion with matrix-free method and GPU computing, Addit. Manuf. 37 (2021) 101732.

[32] E. Neiva, M. Chiumenti, M. Cervera, E. Salsi, G. Piscopo, S. Badia, A.F. Martín, Z. Chen, C. Lee, C. Davies, Numerical modelling of heat transfer and experimental validation in powder-bed fusion with the virtual domain approximation, Finite Elem. Anal. Des. 168 (2020) 103343.

[33] Y. Zhang, G. Guillemot, M. Bernacki, M. Bellet, Macroscopic thermal finite element modeling of additive metal manufacturing by selective laser melting process, Comput. Methods Appl. Mech. Engrg. 331 (2018) 514–535.

[34] N. Hodge, R. Ferencz, R. Vignes, Experimental comparison of residual stresses for a thermomechanical model for the simulation of selective laser melting, Addit. Manuf. 12 (2016) 159–168.

[35] M.F. Zaeh, G. Branner, Investigations on residual stresses and deformations in selective laser melting, Prod. Eng. 4 (1) (2010) 35–45.

[36] E.R. Denlinger, J. Irwin, P. Michaleris, Thermomechanical modeling of additive manufacturing large parts, J. Manuf. Sci. Eng. 136 (6) (2014).

[37] R.K. Ganeriwala, N.E. Hodge, J.M. Solberg, Towards improved speed and accuracy of laser powder bed fusion simulations via multiscale spatial representations, Comput. Mater. Sci. 187 (2021) 110112.

[38] C. Li, E.R. Denlinger, M.F. Gouge, J.E. Irwin, P. Michaleris, Numerical verification of an octree mesh coarsening strategy for simulating additive manufacturing processes, Addit. Manuf. 30 (2019) 100903.

[39] E. Neiva, S. Badia, A.F. Martín, M. Chiumenti, A scalable parallel finite element framework for growing geometries. application to metal additive manufacturing, Internat. J. Numer. Methods Engrg. 119 (11) (2019) 1098–1125.

[40] A. Olleak, F. Dugast, P. Bharadwaj, S. Strayer, S. Hinnebusch, S. Narra, A.C. To, Enabling part-scale scanwise process simulation for predicting melt pool variation in LPBF by combining GPU-based matrix-free FEM and adaptive remeshing, Addit. Manuf. Lett. 3 (2022) 100051.

[41] P. Michaleris, Modeling metal deposition in heat transfer analyses of additive manufacturing processes, Finite Elem. Anal. Des. 86 (2014) 51–60.

[42] D. Arndt, W. Bangerth, M. Feder, M. Fehling, R. Gassmöller, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, et al., The deal.II library, version 9.4, J. Numer. Math. 30 (3) (2022) 231–246.

[43] W. Bangerth, C. Burstedde, T. Heister, M. Kronbichler, Algorithms and data structures for massively parallel generic adaptive finite element codes, ACM Trans. Math. Softw. 38 (2) (2012) 1–28.

[44] M. Fehling, W. Bangerth, Algorithms for parallel generic *hp*-adaptive finite element software, 2022, arXiv preprint arXiv:2206.06512.

[45] C. Burstedde, L.C. Wilcox, O. Ghattas, P4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM J. Sci. Comput. 33 (3) (2011) 1103–1133.

[46] S. Essongue, Y. Ledoux, A. Ballu, Speeding up mesoscale thermal simulations of powder bed additive manufacturing thanks to the forward Euler time-integration scheme: A critical assessment, Finite Elem. Anal. Des. 211 (2022) 103825.

[47] T. Moran, D. Warner, N. Phan, Scan-by-scan part-scale thermal modelling for defect prediction in metal additive manufacturing, Addit. Manuf. 37 (2021) 101667.

[48] M. Mozaffar, E. Ndip-Agbor, S. Lin, G.J. Wagner, K. Ehmann, J. Cao, Acceleration strategies for explicit finite element analysis of metal powder-based additive manufacturing processes using graphical processing units, Comput. Mech. 64 (2019) 879–894.

[49] H. Huang, N. Ma, J. Chen, Z. Feng, H. Murakawa, Toward large-scale simulation of residual stress and distortion in wire and arc additive manufacturing, Addit. Manuf. 34 (2020) 101248.

[50] M. Puso Jr., N. Hodge, An assessment of the utility of multirate time integration for the modeling of laser powder bed fusion, Addit. Manuf. 73 (2023) 103657.

[51] D. Soldner, J. Mergheim, Thermal modelling of selective beam melting processes using heterogeneous time step sizes, Comput. Math. Appl. 78 (7) (2019) 2183–2196.

[52] N. Hodge, Towards improved speed and accuracy of laser powder bed fusion simulations via representation of multiple time scales, Addit. Manuf. 37 (2021) 101600.

[53] C.A. Moreira, M.A. Caicedo, M. Cervera, M. Chiumenti, J. Baiges, A multi-criteria h-adaptive finite-element framework for industrial part-scale thermal analysis in additive manufacturing processes, Eng. Comput. 38 (6) (2022) 4791–4813.

[54] Z.-D. Zhang, S.I. Shahabad, O. Ibhadode, C.F. Dibia, A. Bonakdar, E. Toyserkani, 3-Dimensional heat transfer modeling for laser powder bed fusion additive manufacturing using parallel computing and adaptive mesh, Opt. Laser Technol. 158 (2023) 108839.

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

[56] M. Kronbichler, K. Kormann, Fast matrix-free evaluation of discontinuous Galerkin finite element operators, ACM Trans. Math. Softw. 45 (3) (2019) 1–40.

[57] M. Kronbichler, N. Fehn, P. Munch, M. Bergbauer, K.-R. Wichmann, C. Geitner, M. Allalen, M. Schulz, W.A. Wall, A next-generation discontinuous Galerkin fluid dynamics solver with application to high-resolution lung airflow simulations, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2021, pp. 1–15.

[58] A. Satheesh, C.P. Schmidt, W.A. Wall, C. Meier, Structure-preserving invariant interpolation schemes for invertible second-order tensors, 2022, arXiv preprint arXiv:2211.16507.

[59] A. Gusarov, I. Yadroitsev, P. Bertrand, I. Smurov, Model of radiation and heat transfer in laser-powder interaction zone at selective laser melting, J. Heat Transfer 131 (7) (2009).

[60] J. Goldak, A. Chakravarti, M. Bibby, A new finite element model for welding heat sources, Metall. Trans. B 15 (1984) 299–305.

[61] S.I. Anisimov, V.A. Khokhlov, Instabilities in Laser-Matter Interaction, CRC Press, 1995.

[62] K. Kormann, A time-space adaptive method for the Schrödinger equation, Commun. Comput. Phys. 20 (1) (2016) 60–85.

[63] M. Chiumenti, X. Lin, M. Cervera, W. Lei, Y. Zheng, W. Huang, Numerical simulation and experimental calibration of additive manufacturing by blown powder technology. Part I: thermal analysis, Rapid Prototyp. J. 23 (2) (2017) 448–463.

[64] J. Irwin, P. Michaleris, A line heat input model for additive manufacturing, J. Manuf. Sci. Eng. 138 (11) (2016) 111004.

[65] E. Burman, S. Claus, P. Hansbo, M.G. Larson, A. Massing, CutFEM: discretizing geometry and partial differential equations, Internat. J. Numer. Methods Engrg. 104 (7) (2015) 472–501.

[66] B. Schott, C. Ager, W.A. Wall, Monolithic cut finite element–based approaches for fluid-structure interaction, Internat. J. Numer. Methods Engrg. 119 (8) (2019) 757–796.

[67] P. Kim, V. Kunc, B. Turcksin, D. Rose, D. Hoskins, K. Rowe, E. Jo, F. Ju, Layer Time Control for Large Scale Additive Manufacturing Using High Performance Computing, Technical Report, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2022.

[68] M. Deville, P. Fischer, E. Mund, D. Gartling, High-order methods for incompressible fluid flow, Appl. Mech. Rev. 56 (3) (2003) B43.

[69] J.M. Melenk, K. Gerdes, C. Schwab, Fully discrete hp-finite elements: Fast quadrature, Comput. Methods Appl. Mech. Engrg. 190 (32–33) (2001) 4339–4364.

[70] S.A. Orszag, Spectral methods for problems in complex geometries, J. Comput. Phys. 37 (1) (1980) 70–92.

[71] M. Kronbichler, W.A. Wall, A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers, SIAM J. Sci. Comput. 40 (5) (2018) A3423–A3448.

[72] S. Schoeder, K. Kormann, W.A. Wall, M. Kronbichler, Efficient explicit time stepping of high order discontinuous Galerkin schemes for waves, SIAM J. Sci. Comput. 40 (6) (2018) C803–C826.

[73] M. Kronbichler, D. Sashko, P. Munch, Enhancing data locality of the conjugate gradient method for high-order matrix-free finite-element implementations, Int. J. High Perform. Comput. Appl. (2022) 10943420221107880.

[74] D. Arndt, N. Fehn, G. Kanschat, K. Kormann, M. Kronbichler, P. Munch, W.A. Wall, J. Witte, ExaDG: High-order discontinuous Galerkin for the exa-scale, in: Software for Exascale Computing-SPPEXA 2016-2019, Springer International Publishing, 2020, pp. 189–224.

[75] P. Munch, K. Ljungkvist, M. Kronbichler, Efficient application of hanging-node constraints for matrix-free high-order fem computations on CPU and GPU, in: High Performance Computing: 37th International Conference, ISC High Performance 2022, Hamburg, Germany, May 29–June 2, 2022, Proceedings, Springer, 2022, pp. 133–152.

[76] P. Kopp, V. Calo, E. Rank, S. Kollmannsberger, Space-time hp-finite elements for heat evolution in laser powder bed fusion additive manufacturing, Eng. Comput. 38 (6) (2022) 4879–4893.

[77] J. Brown, V. Barra, N. Beams, L. Ghaffari, M. Knepley, W. Moses, R. Shakeri, K. Stengel, J.L. Thompson, J. Zhang, Performance portable solid mechanics via matrix-free *p*-multigrid, 2022, arXiv preprint arXiv:2204.01722.

[78] D. Davydov, J.-P. Pelteret, D. Arndt, M. Kronbichler, P. Steinmann, A matrix-free approach for finite-strain hyperelastic problems using geometric multigrid, Internat. J. Numer. Methods Engrg. 121 (13) (2020) 2874–2895.

[79] E. Hansen, T. Stillfjord, Convergence of the implicit-explicit Euler scheme applied to perturbed dissipative evolution equations, Math. Comp. 82 (284) (2013) 1975–1985.