An application-driven method for assembling numerical schemes for the solution of complex multiphysics problems

Patrick Zimbrod, Michael Fleck, Johannes Schilp

Angaben zur Veröffentlichung / Publication details:

Zimbrod, Patrick, Michael Fleck, and Johannes Schilp. 2024. "An application-driven method for assembling numerical schemes for the solution of complex multiphysics problems." *Applied System Innovation* 7 (3): 35. https://doi.org/10.3390/asi7030035.

applied system innovation

MDPI

Article

An Application-Driven Method for Assembling Numerical Schemes for the Solution of Complex Multiphysics Problems

Patrick Zimbrod [1,*], Michael Fleck [2] and Johannes Schilp [1,*]

[1] Applied Computer Science, University of Augsburg, 86163 Augsburg, Germany
[2] Metals and Alloys, University of Bayreuth, 95447 Bayreuth, Germany; michael.fleck@uni-bayreuth.de
* Correspondence:patrick.zimbrod@uni-a.de (P.Z.); johannes.schilp@uni-a.de (J.S.)

Abstract: Within recent years, considerable progress has been made regarding high-performance solvers for partial differential equations (PDEs), yielding potential gains in efficiency compared to industry standard tools. However, the latter largely remains the status quo for scientists and engineers focusing on applying simulation tools to specific problems in practice. We attribute this growing technical gap to the increasing complexity and knowledge required to pick and assemble state-of-the-art methods. Thus, with this work, we initiate an effort to build a common taxonomy for the most popular grid-based approximation schemes to draw comparisons regarding accuracy and computational efficiency. We then build upon this foundation and introduce a method to systematically guide an application expert through classifying a given PDE problem setting and identifying a suitable numerical scheme. Great care is taken to ensure that making a choice this way is unambiguous, i.e., the goal is to obtain a clear and reproducible recommendation. Our method not only helps to identify and assemble suitable schemes but enables the unique combination of multiple methods on a per-field basis. We demonstrate this process and its effectiveness using different model problems, each comparing the resulting numerical scheme from our method with the next best choice. For both the Allen–Cahn and advection equations, we show that substantial computational gains can be attained for the recommended numerical methods regarding accuracy and efficiency. Lastly, we outline how one can systematically analyze and classify a coupled multiphysics problem of considerable complexity with six different unknown quantities, yielding an efficient, mixed discretization that in configuration compares well to high-performance implementations from the literature.

Keywords: simulation; multiphysics; finite difference; finite volume; finite element; discontinuous Galerkin

Citation: Zimbrod, P.; Fleck, M.; Schilp, J. An Application-Driven Method for Assembling Numerical Schemes for the Solution of Complex Multiphysics Problems. *Appl. Syst. Innov.* **2024**, *7*, 35. https://doi.org/10.3390/asi7030035

Academic Editor: Teen-Hang Meen

Received: 19 February 2024
Revised: 5 April 2024
Accepted: 19 April 2024
Published: 24 April 2024

1. Introduction

Within the discipline of modeling physical processes by partial differential equations (PDEs), one is confronted with an increasing number of choices of numerical methods to perform this job. To an application-oriented expert, that is, engineers as well as scientists who are familiar with the physics to be modeled, but not necessarily with the numerics of PDE approximation, this abundance of choices may quickly appear daunting due to the growing amount of research in the numerics community. Modern, general, and mathematically rigorous implementations of the finite element method (FEM), such as `deal.II` [1], `FEniCS` [2], `Firedrake` [3], or `MFEM` [4], offer a wide variety of formulations that one can choose from, with varying degrees of customizability. The number of different options is best illustrated by considering the various finite elements and corresponding function spaces that the application expert typically can and must choose from nowadays. For instance, the popular website DefElement, which summarizes a vast amount of finite element types along their characteristics and shape functions, offers over 45 different choices of element to approximate scalar and over 40 for vector quantities [5]. These elements are, for the most part, readily implemented in the abovementioned software libraries. However,

not all of them necessarily produce good or even stable approximations for any given PDE problem [6]. Even if one were to have prior knowledge on a good choice of function space, e.g., $H(\mathrm{div})$ to approximate divergence-free quantities [7], one would still have to choose between more than 20 different kinds of finite element. Choosing the right tool for the problem naturally becomes even more of an issue when other standard tools that are used in practice are additionally considered, such as finite difference and finite volume methods. These recent advancements are, however, in contrast to established numerical techniques that are oftentimes still used as a standard in practice. For example, the finite volume method is still widely considered the standard practice for solving problems in computational fluid dynamics [8]. In contrast, there have been several recent developments based on the discontinuous Galerkin method that have shown to perform noticeably better for fluid dynamics problems [9]. We, thus, note that a gap has emerged between industry standard tools and modern, high-performance numerical methods that we attribute to the increasing complexity and insight required to properly assemble the latter.

Additionally, apart from these outlined recent developments, choosing an approximation method that offers the right number and type of degrees of freedom is essential for obtaining a solution that can be efficiently computed, especially under the requirement of good convergence and stability. Where the former might be negligible in practice, since achieving the utmost performance is not always important, having a stable approximation is paramount. This line of reasoning not only applies to problems governed by a single PDE but even more so to multiple equations forming a system with sometimes notably different dynamics. Thus, using one discretization method in a monolithic way will not necessarily perform equally well for each PDE of that system, creating bottlenecks regarding either stability or accuracy and, thus, further complicating the question of which specific method is best suited for the job. Using multiple numerical methods within one multiphysics problem, however, requires the interoperability of all discretizations. Coupling different solvers has been shown to quickly become tedious regarding implementation and may even yield severe bottlenecks due to large amounts of data transfer. Having a common formulation for some schemes is, thus, desirable, such that one may recover specific methods by imposing abstractions, for example, by omitting some steps in assembling a global linear system.

In this work, we make an initial effort towards closing the outlined, increasing gap between state-of-the-art research in the numerics community and best practices in applications. As covering the entirety of numerical methods available is impossible in practice, especially for nonexperts in every single aspect, we initially restrict the scope of view to a rather narrow subset of methods. The added benefit of this approach is that this enables us to remove methods that would otherwise render making a problem-oriented choice largely ambiguous.

The contributions of this work to address the abovementioned problems are twofold: First, we propose a unifying approximation taxonomy that enables recovering the most prevalent grid-based numerical methods by imposing some well-defined abstractions, albeit with a relatively narrow scope for now. Secondly, we propose a generalized framework for choosing an appropriate, that is, stable and performant, numerical scheme given a fixed set of inputs. These include the system of PDEs and the triangulation where the problem is defined, as well as the available computing hardware. As such a method necessitates a unified view of all the numerical methods considered to enable quantifiable comparisons, this heavily builds on the first part of this article. In combination, this enables an application expert to make an informed choice of which scheme to use on a per-field basis given some equally well-defined inputs. Since in an application setting, stability is typically more important than convergence rate, we focus our effort on providing methods that produce stable solutions but do not fall behind too much compared to the utmost performant alternatives.

The remainder of this work is, thus, structured as follows: We first present a brief review of works in the literature that are concerned with comparing the mentioned numerical schemes that we subsequently build upon. The theory necessary to construct such

a baseline will be covered in the following sections. We will then proceed by analyzing typically given inputs for a PDE problem and assemble a method to systematically derive suitable numerical schemes. The results of choosing and implementing numerical methods according to our developed framework will be demonstrated afterward. We will investigate two distinct benchmark problems, each comparing two methods that would be closest to being optimal in that specific case. Special attention will be given to accuracy concerning the analytical solution as well as computational complexity and performance. Finally, we will demonstrate the effectiveness of the presented framework by assessing a complex and currently relevant multiphysics problem. We show how to systematically arrive at a mixed choice of discretization schemes using the proposed decision method. Finally, we indicate some relevant works in the literature that employ similar approximations, highlighting the validity and relevance of this work.

2. Previous Works

In this section, we outline some prior efforts aimed at comparing different grid-based numerical methods or drawing connections between them.

Some authors rigorously showed the equivalence of the finite volume method to either mixed finite element [10] or Petrov Galerkin finite element methods [11,12]. With regards to the finite difference and finite element method, Thomèe showed early on that the FEM can be understood as a somewhat equivalent, yet generalized, variant of taking finite differences on arbitrary grids [13]. Some general differences between these schemes were outlined by Key and Krieg [14]. In the work of Shu, some analogies were brought up between finite volume and finite difference schemes in WENO formulation [15]. A theoretical and numerical comparison between higher-order finite volume and discontinuous Galerkin methods was conducted by Zhou et al. [9]. Additionally, Dumbser et al. constructed a unifying framework to accommodate high-order finite volume and discontinuous Galerkin schemes [16]. In the context of elliptic PDEs, Lin et al. presented a theoretical and empirical comparison between the comparably new weak Galerkin, discontinuous Galerkin, and mixed finite element schemes [17]. A comparative study between discontinuous Galerkin and the streamline upwind Petrov Galerkin method for flow problems can be found in [18]. These works in summary draw point-wise comparisons between some grid-based approximation schemes. Despite being quite useful for disseminating individual advantages and disadvantages for a given application, one may still lack an understanding of the general properties. Furthermore, Bui-Thanh presented an encompassing analysis and application of the hybridizable discontinuous Galerkin method (HDG) to solve a wide variety of PDE-governed problems. It was, therefore, shown that this numerical scheme is general and powerful enough to form a unified baseline [19]. In addition, due to the generality of this method, there have been works that attempt to benchmark DG methods to more conventional and widely adopted continuous Galerkin methods (CG) [20–22]. Some authors proposed combinations of numerical schemes that operate optimally to solve hyperbolic [23] or parabolic [24] systems of PDEs.

In summary, we draw the following conclusion from this brief review of the relevant literature. To this date, there only exist a few comparisons between grid-based approximation schemes that outline common properties in a mostly ad hoc or point-wise manner. We have, however, outlined in the previous section that it would be beneficial from an application-oriented perspective to have an encompassing taxonomy for these numerical schemes. Furthermore, many different variants of numerical schemes have been proposed to tackle a wide variety of PDE problems. What appears to be missing, though, is a general guideline on how to choose between these vast alternatives to obtain a method, or possibly a combination of different methods, to solve a system in a stable (above all) and reasonably efficient way.

3. Theoretical Baseline

The general procedure of this chapter is as follows. We first introduce the most general scheme considered here, which is the discontinuous Galerkin method. Then, for each additional scheme considered, we individually work out the necessary simplifications to arrive at that numerical method, starting at the DGM. In the remaining sections of this article, we use underline notation ($\underline{\cdot}$, $\underline{\underline{\cdot}}$) to indicate vectors and matrices and Roman indices (i, j) to denote elements of lists or arrays at the computational level. We assume the reader of this article to have a coarse overview of the presented methods, but not much insight into the specifics of each. We, thus, present the necessary theory in a comparably coarse manner that focuses on the qualitative characteristics.

3.1. Discontinuous Galerkin Method

This method was originally proposed in 1973 to solve challenging hyperbolic transport equations in nuclear physics [25]. In spirit, it can be held as a synthesis of finite element and finite volume schemes and poses a generalized variant of both.

To derive such a scheme, we begin by stating the strong form of a given PDE. The most straightforward example in this case would be a first-order linear advection equation with a homogeneous von Neumann boundary condition:

$$\partial_t \alpha + \underline{u} \cdot \nabla \alpha = 0, \qquad \frac{\partial \alpha}{\partial n} = 0 \quad \forall x \in \partial\Omega, \tag{1}$$

where ∂_t signifies the temporal derivative, x is the set of spatial coordinates, $\partial\Omega$ denotes the boundary of the computational domain Ω, and n is the unit normal with respect to $\partial\Omega$. In this case, and henceforth in this article, we assume for reasons of simplicity that the velocity field is divergence-free, i.e., $\nabla \cdot \underline{u} = 0$.

We now state the weak form of Equation (1), that is, we multiply with a test function v, integrate over the entirety of the domain Ω, and apply partial integration to the second term on the left-hand side that contains the nabla operator. For a divergence-free velocity field, one may set $\underline{u} \cdot \nabla \alpha = \nabla \cdot (\underline{u}\alpha)$, which results in the following formulation: Find $\alpha \in V$ such that

$$\int_\Omega v \cdot \partial_t \alpha \, dx + \int_{\partial\Omega} v\alpha(\underline{u} \cdot \underline{n}) \, ds - \int_\Omega \nabla v \cdot \underline{u}\alpha \, dx = 0, \qquad \forall v \in W, \tag{2}$$

where we need to make an appropriate choice for the solution space V and the test space W which may, but do not need to, differ from each other.

Due to partial integration, we now encounter an additional term that has to be integrated over the domain boundary $\partial\Omega$, where $(\underline{u} \cdot \underline{n})$ denotes the velocity component normal to the boundary.

To make such a problem solvable by a computer, one must additionally choose the discretization of the solution space V, denoted V_h. A particularly popular choice of space is the set of Lagrange polynomials. In addition, the physical space must be discretized in the form of a triangulation. The DG scheme then consists of assembling the finite-dimensional, linear system on the element level. This enables high locality of the solution process, which leads to efficient computation on parallel architectures, as less data transfer is required.

One resulting key feature of the DG scheme is that the elements now do not overlap anymore in terms of their degrees of freedom. Thus, the global problem is broken up into individual problems. This, in general, leads to large systems that are, however, sparse and, in the case of the mass matrix, even block diagonal. The remaining term, often denoted the numerical flux, is the only term within the physical domain that ensures coupling across elements. Through evaluation of this surface integral, adjacent degrees of freedom are coupled and, thus, global conservation of quantities can be assured.

As the polynomial space of DG schemes only belongs to the L^2 space of functions but not H^1, the basis functions are discontinuous; thus, the derivative at boundaries is not well defined. Solving PDEs involving second derivatives is, thus, not possible as is. As a

consequence, there have been many successful extensions of this method to circumvent that problem. At this point, we name the most prevalent schemes, namely the symmetric interior penalty [26], hybridizable [27] and local DG scheme [28]. These methods, despite having different approaches, have been extensively studied and compared to each other [29]. As it turns out, all methods work well and have individual advantages and disadvantages. For this work, we will take the hybridizable DG scheme as a general framework. We note here that the proposed method would, however, work with any of the other schemes given above.

Within the abovementioned methods, one introduces an additional term in the weak form that serves as a penalty for discontinuous solutions. An alternative approach that is also pursued within the HDG scheme is the algebraic manipulation of the PDE system by splitting. One recursively introduces new dependent variables for quantities that appear in higher-order derivatives such that each quantity is differentiated at most once. We illustrate this using the Laplace equation

$$\Delta u = 0. \tag{3}$$

The corresponding, well-known weak form is as follows: Find $u \in V$, such that for all $v \in V$

$$\int_{\partial\Omega} v(\nabla u \cdot \underline{n})\, ds - \int_{\Omega} \nabla v \nabla u\, dx = 0. \tag{4}$$

By introducing the auxiliary variable $\sigma = \nabla u$, Equation (4) is extended to the following system:

$$\int_{\partial\Omega} v(\sigma \cdot \underline{n})\, ds - \int_{\Omega} \nabla v \sigma\, dx = 0, \tag{5}$$

$$\sigma = \nabla u. \tag{6}$$

Employing such an approach enables splitting PDE systems of arbitrary order, resulting in larger systems of first-order PDEs.

3.2. Continuous Galerkin Finite Element Method

The most straightforward step to conduct is to derive the continuous Galerkin (CG) from the DG method. The former is oftentimes also referred to as the classic finite element method, being the original formulation used to solve problems in structural mechanics [30].

In this case, all degrees of freedom (DoFs) in the domain are global, in contrast to being local to each cell. However, each basis function associated with a given degree of freedom has compact support and is, thus, only nonzero within the direct vicinity. The resulting linear system hence remains sparse but has considerably fewer DoFs than an equivalent discretization produced by a DG method.

One may obtain a CG method starting from the DGM by strongly coupling the degrees of freedom at cell interfaces. In other words, the previously discontinuous approximation must be made continuous. In terms of the weak form of a given problem, the numerical flux that has been introduced by partial integration has to vanish. This step is exactly taken in deriving weak forms for the CG scheme. The equivalent weak form of the advection equation given by Equation (2) is, then, the following: Find $u \in V$ such that

$$\int_{\Omega} v \cdot \partial_t \alpha\, dx - \int_{\Omega} \nabla v \cdot \underline{u} \alpha\, dx = 0, \qquad \forall v \in V. \tag{7}$$

By coupling coinciding DoFs, one may equivalently introduce shared DoFs between cells. This results in comparison to the DGM in a smaller global system that is in turn more coupled, yielding more nonzero entries per row and column in the system matrices.

The condensation of such a system by coupling DoFs is illustrated in Figure 1. From the numbering of DoFs in both figures, it becomes apparent that the number of additional allocations grows drastically with increasing dimensionality of the problem.

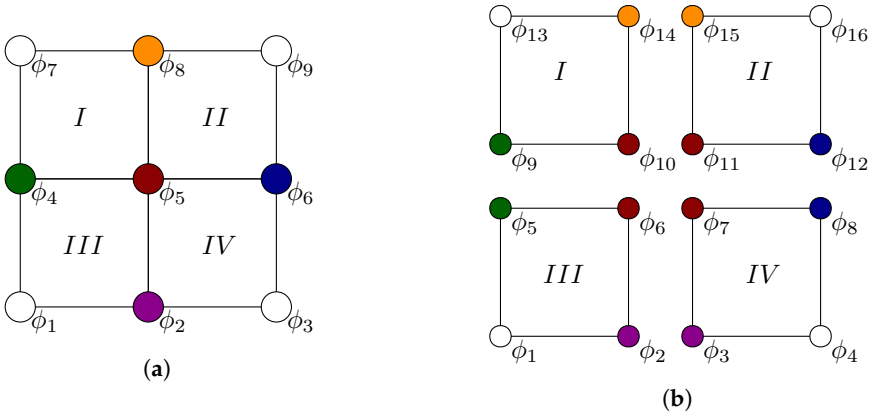


Figure 1. Coupling of global DoFs in the continuous Galerkin (**a**) versus discontinuous Galerkin FEM (**b**), both of first order. In the latter case, DoFs are entirely local to the cell and, thus, receive no contribution from neighboring cells. Weak coupling is only introduced by the additional numerical flux. Coupled DoFs are drawn in identical colors.

As the numerical flux is zero by definition for a CG scheme, we may also omit it from computation. Thus, the CGM is noticeably less arithmetically intensive in this regard. However, this computational saving is offset by the strong coupling of DoFs, resulting in a more dense linear system and possibly a more complex assembly process in terms of memory management.

As equivalence can be shown here based on the weak form and, thus, early on in the model assembly process, the choice of finite element is unaffected. This, in consequence, also applies to the chosen type of triangulation or the order of approximation.

3.3. Finite Difference Method

At first glance, the finite difference method (FDM) might appear to be conceptually different from the finite element methods given above. Instead of treating the discretized problem in an element-wise manner, the FDM operates on discrete points directly and, per se, lacks a notion of cells in the domain. Yet, both methods still may yield identical results in discretization. A comparison of both approaches is shown in Figure 2.

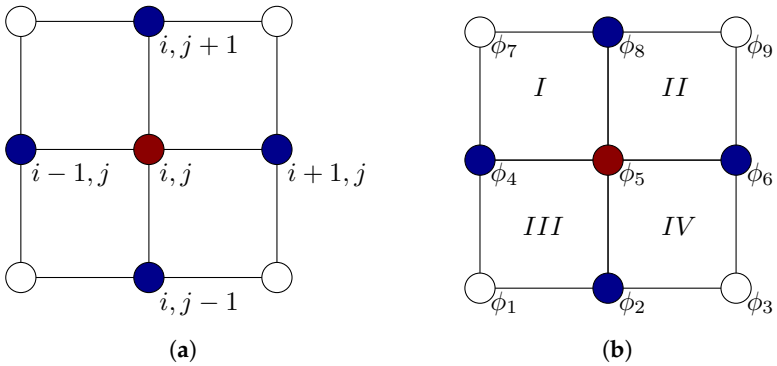


Figure 2. Comparison of the nodal nature of the FDM (**a**) versus the cell-wise assembly used in the CG FEM (**b**) for an identical, Cartesian triangulation with 9 nodes. Both methods are formulated as first-order approximations. Red-colored nodes signify the points where the PDE is evaluated. Contributions to this node are taken from blue nodes, whereas no contribution—white nodes.

In the figures, i and j denote vertical and horizontal indices of grid nodes, ϕ_i are the FE basis functions, and Roman letters denote indices of cells. Forming, for example, the Laplacian for an FDM requires access to the vicinity of vertex i, j (red node) in all Cartesian directions (blue-colored nodes). For both methods, gray-marked nodes do not pose a contribution to the value of the central node. For a special case of FEM with quadrilateral elements, the same nodes form contributions to the global basis function ϕ_5. However, we

now do not evaluate the Laplacian operator directly but, instead, gather contributions from weak form integrals. In the case of ϕ_5, we have to gather contributions from cells I to IV.

However, one may still show the equivalence of CGM and FDM by investigating the resulting global linear system. We exemplify this claim using the Laplacian as a differential operator and the second-order central stencil. This formulation continues to be widely used as an approximation technique. As will be shown later, this particular choice of operator is especially straightforward to compare with the CG-FEM due to the choice of trial and test functions. On a Cartesian, two-dimensional grid with uniform spacing h in both directions, the approximation reads

$$\Delta u \approx \frac{u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1}}{h^2}. \tag{8}$$

Such a system in stencil notation will produce a global matrix with main diagonal values 4 and four off-diagonals with entries 1.

We now proceed to construct an equivalent CG finite element scheme, where the global system matrix is required to be exactly equivalent to the FD formulation.

The weak (CG) Laplacian can be formulated as follows: Find $u_h \in V$ such that for all $v_h \in V$

$$\int_\Omega v \Delta u \, dx = \underbrace{\int_{\partial\Omega} v(\nabla u \cdot \underline{n}) \, ds}_{=0} - \int_\Omega \nabla v \nabla u \, dx. \tag{9}$$

We have in this case introduced the additional restriction that trial and test space be identical, that is, we use a Bubnov Galerkin method. Now, let Ω be an identical triangulation to the FD variant using quadrilateral $\mathbb{Q}^1$ elements, that is linear Lagrange elements.

Then, the four basis functions spanning the reference element are

$$\phi_1(x, y) = xy - x - y + 1; \tag{10}$$
$$\phi_2(x, y) = x(1 - y); \tag{11}$$
$$\phi_3(x, y) = y(1 - x); \tag{12}$$
$$\phi_4(x, y) = xy. \tag{13}$$

The finite difference stencil given by Equation (8) only takes into account contributions from nodes that lie strictly horizontally or vertically from the node of interest. As a consequence, the node on the reference quadrilateral that is positioned diagonally from the center node must not have any contribution to the weak form integral, otherwise the resulting linear system cannot be equal. We thus need to evaluate the weak form in a way such that the resulting matrix $\phi_i \cdot \phi_j$ becomes sparse. It turns out that this can be achieved by choosing a collocation method for quadrature. In that case, quadrature points are chosen to coincide with the node coordinates, and as a consequence, the mass matrix $\phi_i \cdot \phi_j$ becomes the identity matrix.

From the family of Gaussian quadrature schemes, one can achieve this using a Gauss–Lobatto quadrature of order equal to the polynomial order of the finite element. We note at this point that choosing this particular combination of quadrature method and number of nodes will lead to inexact integration and, thus, a numerical error is introduced. In this case, to produce a collocated scheme, one must pick the second-order Gauss–Lobatto variant using two quadrature nodes per coordinate direction. As this type of integration is known to be accurate up to degree $2n - 3$, this scheme will only integrate linear polynomials exactly. However, the above-listed basis functions are bilinear and have a combined polynomial order of 2. Thus, integration will not be accurate in this case. However, the modern FEM in general does not prescribe any particular method of integrating the weak formulation, per se [31]. Thus, although the results of a properly implemented FEM in the sense of exact integration will be slightly more accurate, one can still show equivalence regarding a particular instance of the FEM.

We now evaluate the element-wise stiffness matrix $-\int_{\Omega^{(e)}} \nabla_k \phi_i \nabla_k \phi_j \, dx$ within the reference domain $[0;1] \times [0;1]$ for the given first order Lagrange element using Gauss–Lobatto quadrature, more specifically the variant using two quadrature points per coordinate direction. This results in

$$\underline{\underline{K}}^{(e)} = \begin{bmatrix} -1 & 1/2 & 1/2 & 0 \\ 1/2 & -1 & 0 & 1/2 \\ 1/2 & 0 & -1 & 1/2 \\ 0 & 1/2 & 1/2 & -1 \end{bmatrix}. \tag{14}$$

As such, $\underline{\underline{K}}^{(e)}$ does not yet equal Equation (8). The final step consists of assembling the linear system in the physical domain using the reference stiffness matrix. In a Cartesian mesh in two dimensions, an interior node is owned by four quadrilateral elements. If one carries out this assembly process, an equivalent formulation can be obtained:

$$\underline{\underline{K}} = \begin{bmatrix} \ddots & \ddots & & & \vdots & \vdots & \vdots & & & & \\ \ddots & \ddots & \ddots & & \vdots & 1 & \vdots & & & & \\ & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots & & & & \\ & & \ddots & \ddots & \ddots & 1 & \vdots & & & & \\ \cdots & \cdots & \cdots & \ddots & \ddots & \ddots & \vdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 1 & \cdots & 1 & \ddots & -4 & \ddots & 1 & \cdots & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \vdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots \\ & & & & \vdots & 1 & \ddots & \ddots & \ddots & & \\ & & & & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \\ & & & & \vdots & 1 & \vdots & & \ddots & \ddots & \ddots \\ & & & & \vdots & \vdots & \vdots & & & \ddots & \ddots \end{bmatrix}. \tag{15}$$

The exact position of the one entry in the typically large and sparse matrix depends on the mesh topology as well as the global numbering of the degrees of freedom.

For the discretization of other operators, a similar argument holds, as the shown procedure is irrespective of the choice of weak form or basis function. For example, one could discretize the gradient of a function ∇u using an upwind finite difference formulation in fluid mechanics for resolving convective terms. An equivalent finite element method can be assembled by producing a weak form, as given in the above example, choosing the same collocation method and carrying out the integration numerically. However, one important difference is that one cannot choose the test space to be equivalent to the trial space. This would yield a symmetric system that does not correspond to an upwind finite difference formulation and is also not stable in the case of solving a pure advection equation. One must instead choose a test space with asymmetric test functions to account for the notion of an upwind node, thus yielding a Petrov Galerkin scheme [32].

We can, as a result, summarize the FDM to be a special instance of the CG FEM. On one hand, integration is restricted to a collocation method, and on the other hand, the Jacobian mapping from reference to physical elements is constant throughout the domain. This close relationship has also been hinted at by analysis of boundary value problems by Thomèe [13].

For the sake of achieving the same discretization, the use of finite differences over finite elements becomes apparent from the discussion above. Most strikingly, the process of producing a local stencil is vastly more straightforward than performing element-wise assembly and gathering the weak form integrals in a global, sparse linear system. Each element-wise operation in assembly would otherwise require the evaluation of the mesh

Jacobian for the requested element, that is, the mapping from the reference to physical space. Furthermore, this constant stencil enables finite difference schemes to operate in a matrix-free manner easily. For larger systems, this can help to avoid a large amount of allocated memory, thus being well suited for modern hardware architectures that are typically memory-bound.

These advantages are, however, offset by some topological restrictions on the mesh. The simplicity of a constant stencil also implies that the mesh must not deviate from a Cartesian geometry. Otherwise, additional complexity is introduced since Equation (8) becomes a stencil in the reference domain that has to be mapped to the physical domain. This would still save the computational effort to assemble the weak form. However, since this process only has to be carried out once for the reference element, the computational impact can be held low by precomputing the integrand.

3.4. Finite Volume Method

In a similar vein to the FDM, the use of finite volumes might appear distinctly different from the idea of finite elements. Here, we make extensive use of Stokes' theorem to replace volume with hull integrals in conservation laws [33]. There exist different formulations of this method, most namely, a cell- and vertex-centered form. The main difference lies in where the solution is stored. In the former case, the solution is stored at the polygonal cell centers that are spanned by the mesh vertices. The latter, instead, directly uses these vertices as solution points [34]. In this case, one does not operate on the computational mesh directly but, rather, on its dual. As the cell-centered formulation is considerably more widely used, we investigate this variant further in the following.

It can be shown, however, that the FVM can simply be considered a Bubnov discontinuous Galerkin method of polynomial order zero. To illustrate this, we again turn to Equations (1) and (2) describing the strong and weak form of the advection equation. A finite volume approximation in conservation form is

$$\int_\Omega \partial_t \alpha \, dx + \int_\Omega \underline{u} \nabla \alpha \, dx = \int_\Omega \partial_t \alpha \, dx + \int_{\partial\Omega} \alpha (\underline{u} \cdot \underline{n}) \, ds. \tag{16}$$

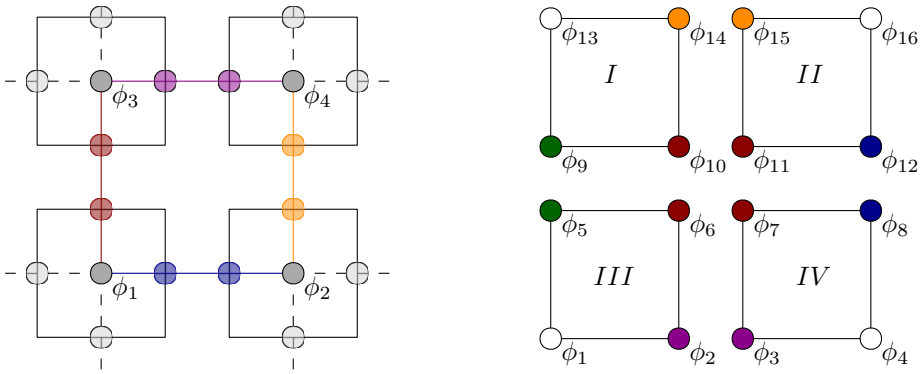
Apart from the presence of a test function v in Equation (2), the second integrand simply represents the net flux of the conserved quantity $\underline{u}\alpha$ over the set of element boundaries.

For Equations (2) and (16) to be equivalent in this case, the third integrand resulting from partial integration has to vanish in addition. However, this can be shown trivially by setting the order of the polynomial space for the trial and test function to zero. Then, the derivative of the test function vanishes and, thus, the entire term does not contribute to the weak form.

After performing this step, the test function is still present in the remaining parts of Equation (2). For the remaining terms to be equivalent, they must vanish out of the equation as well. This can be accomplished straightforwardly by fixing the value of the test function to be unity. In the weak form, this step is admissible since it must hold for all instances of V. As $1 \in V^0$ where V^0 is the space of constant polynomials, this statement holds in particular for a Bubnov Galerkin scheme, as trial and test space must be identical. The qualitative similarity of both schemes is illustrated in Figure 3.

For both schemes, DoFs are entirely local to the cell and coupling happens through the calculation of a numerical flux—or, in more formal terms, through the evaluation of the hull integral in the corresponding weak form. However, the FVM only stores one DoF per cell, which has notable implications for the calculation of the numerical flux. This means that as a first step, the cell values have to be reconstructed at the mesh facets. These reconstructed DoFs which depend on the cell values that they interpolate between can then be coupled to their counterparts at opposing mesh facets. These relationships are denoted by DoFs being colored identically (coupled) and being transparent (reconstructed) in Figure 3a. For a DGM scheme of first order or higher, this interpolation step is oftentimes not necessary if the quadrature scheme is chosen carefully. For a collocation method (see Section 3.3 for a

more thorough discussion), one does not need to tabulate the full list of DoF values at the set of facet quadrature points, but rather, only a small subset of DoFs that are owned by the facet [35].



(**a**) (**b**)

Figure 3. Comparison of FVM (**a**) versus DGM (**b**) on an identical quadrilateral triangulation. Coupled DoFs are marked in identical colors. For the FVM, one must first reconstruct the values of the DoFs at the mesh facets to then compute the numerical flux.

In summary, the FVM can again be considered as a special instance of the Bubnov type DGM, where the shared polynomial space V is taken to be of constant order and the test function v is set to be unity.

We note, similarly to the discussion on the FDM, that this simplification of Equation (2) brings with it some computational advantages that can be offset by sacrificing flexibility. The absence of a true weak form in a finite volume formulation again means that actual assembly is not needed. In addition, one may omit the transformation from the reference to physical space, as interpolating degrees of freedom to mesh facets and forming a finite sum of these contributions can be performed on the mesh directly. The caveat of this approach is that FVM in principle is bound to be at most first-order accurate. In practice, this does not hold as the FVM can be extended to higher orders by applying higher-order flux reconstruction techniques [9,15]. Such techniques can, however, quickly become computationally expensive as well with increasing order. This is achieved in this case by widening the stencil for polynomial reconstruction, increasing memory and time complexity by a considerable amount [36].

3.5. Summary

In this section, we established a common framework to formulate the most prevalent grid-based numerical schemes for the solution of PDEs.

It turns out that the DG method possesses enough flexibility to incorporate the CGM, FDM, and FVM by imposing a set of restrictions. A summary of the results presented in this section on how the schemes compare overall is given in Table 1.

This framework is not only of theoretical use; rather, such a common formulation also enables us to combine these schemes arbitrarily to solve larger problems. As each scheme possesses strengths and means to gain computational efficiency, this is an important result since it enables efficiently mixed discretizations of multiphysics problems. Establishing a practical method to achieve exactly this will be the content of the next section.

Before concluding the discussion on relating the above numerical schemes, we add an important remark. There do exist several extensions to these methods that in general do not fit into the framework that has been established. We will list a few examples for the sake of illustration.

Table 1. Comparison of the individual restrictions that the presented schemes impose. Certain simplifications bring with them computational advantages, as discussed above.

Scheme	Geometry	Function Space	Weak Form	Quadrature
DGM	Arbitrary	Discontinuous (L^2)	Full	Arbitrary
CGM	Arbitrary	Continuous (H^1)	No hull integrals over interior facets	Arbitrary
FDM	Cartesian	Continuous (H^1)	No hull integrals over interior facets	Collocation
FVM	Arbitrary	Discontinuous ($\mathbb{P}^0 \in L^2$), Bubnov Galerkin	No volume integrals	None required

There do exist formulations of the FDM that can capture domains with less regularity; see, for example, [37–40]. One can also find alternative discretization methods based on FDM in the literature that encompass the notion of missing structure in grids more naturally, such as understanding vertices as centroids of Voronoi cells [41].

As mentioned previously, there exist various formulations of the FVM that extend far beyond the original restriction of being first-order accurate. The cell-averaged flux is then determined in terms of reconstructing polynomials that, in theory, can be of arbitrary order. Such approaches, per se, do not fit well into the above given DG scheme but do, however, achieve similar results.

4. Method for Assembling Numerical Schemes

The overarching goal of this section is to identify a suitable combination of numerical schemes for a given multiphysics problem that is stable and accurate on one hand, but also performant with regards to a specific choice of hardware on the other hand.

With the set relations between methods discussed in Section 3, we can now use the simplifications and, thus, computational advantages that each scheme presents. That is, we follow the guideline to impose as many restrictions as possible whilst sustaining enough degrees of freedom to accurately capture the behavior of a given PDE. In this way, we aim to provide the application expert with a recommendation on which schemes to use for a particular computational problem. Our key concerns are, above all, to make this recommendation unambiguous. Also, we focus on providing recommendations that will produce a stable solution and do not require tuning of artificial parameters. If at least either of both requirements were not given, the usefulness of this method would be lost as one would have to undergo substantial experimentation to attain a valid solution. Thus, by proposing such a method, we aim to give a sensible trade-off between practicality and reproducibility, on one hand, and utmost performance at the cost of possibly many model-building iterations and tuning, on the other hand.

4.1. Preliminary Assumptions

As a starting point, it has to be stated that encompassing the entire state of research on such schemes would be an impossible task. The likewise formalization of a common framework is equally challenging as a consequence and, thus, is not considered in this work.

Instead, we follow the path of introducing some restrictions that are, on one hand, enough to construct a unifying scheme but, on the other hand, not too strict such that the efficient solution of real-world problems would be out of scope.

Thus, we propose the following restrictions to arrive at a one-to-one choice of numerical schemes. These apply to single-field, as well as to multiphysics problems.

1. Only Bubnov Galerkin schemes are considered, that is, we omit Petrov Galerkin methods. The former restricts the choice of test space to be identical to the trial space. As such, we omit schemes that, for instance, use weighted functions or stencils to account for flow fields. An example of such schemes would be the streamline

upwind Petrov Galerkin (SUPG) method [32]. This restriction is essential to obtain an unambiguous choice of method, as the notion of Petrov Galerkin methods does not imply any particular choice of function space. Naturally, omitting the use of arbitrary test function spaces contains a trade-off as we, in theory, restrict the solution space by doing so. However, the majority of numerical methods developed so far indeed fall within the category of Bubnov Galerkin schemes. Thus, we mainly disregard special instances of numerical methods that are tailored to particular use cases, such as the SUPG method outlined above.

2. We omit function spaces for approximation other than the L^2 and H^1 Sobolev spaces. There exist a vast variety of so-called mixed finite element schemes that use finite elements based on different or composite function spaces with unique properties [42]. For example, one may construct function spaces that can exactly fulfill divergence-free properties ($H(\mathrm{div})$) or conditions based on the rotation of a field ($H(\mathrm{curl})$). The specific choice of finite element, then, would require a considerable amount of expertise and would warrant a complex decision process of its own. Furthermore, the number of elements available for such spaces is vastly ambiguous, as outlined in Section 1. We thus focus on scalar-, vector-, and tensor-valued Lagrange elements solely. They have been shown to encompass a similar solution space as well and perform comparably for fluid and electromagnetic problems [43,44]. In summary, we similarly sacrifice some freedom in choosing possibly very-well-suited function spaces for the benefit of attaining a decision metric that allows for an unambiguous recommendation.
3. Closely related to the previous statement, we restrict the solution space further by requiring that only finite elements utilizing Lagrange polynomials should be used. As the standard scalar- and vector-valued $\mathbb{P}^k$ and $\mathbb{Q}^k$ finite elements, being by far the most popular choices, use exactly this family of polynomials, this requirement is weaker in practice than it might seem at first glance [45,46].
4. We impose a coarse taxonomy to classify the qualitative behavior of a given PDE, that is, we specify limits regarding the leading coefficients of the differential operators. This should indicate whether the physical process described by the PDE is either more dissipative or more convective by nature. We thus introduce a more physical interpretation than the considerably stricter coercivity measures employed in functional analysis. Our taxonomy closely follows the classes that were proposed by Bitsadze [47]. We do not claim this classification to be universally accurate. In practice, it has been shown, however, that having discrete cut-off values to disambiguate classes of PDE eases the choice of numerical scheme for application experts considerably. Hence, we choose to follow this path despite some shortcomings regarding generality.
5. We only investigate systems of PDEs with differential operators up to second order. These are most common within physical processes and enable a wider range of numerical schemes to be used. For instance, equations of higher order, such as the Cahn–Hilliard equation, would require the use of finite elements where up to third-order derivatives are defined. Such elements of high continuity are cumbersome to derive and are rarely used. Instead, we propose that in such cases the system should be reformulated as a mixed problem, where in the mentioned example, one could represent the quantity of interest as two fields with second derivatives each. This technique is also regularly used in practice.
6. For finite volume methods, we use the cell-centered variant, as already outlined in Section 3.4, instead of the vertex-centered or cell-vertex formulation. This is due to this form being the most popular choice in practice. Furthermore, it resembles the other schemes more naturally, as has been shown in previous sections.

4.2. PDE Classification

To find a numerical scheme that produces stable results, knowing the qualitative behavior of the system is oftentimes a necessity. In particular, this means that the specific

capabilities that a chosen numerical scheme possesses need to reflect the properties that the system presents.

We illustrate this by example. We once again investigate the simple advection equation (Equation (1)), which is known to be first-order hyperbolic. Such systems are prone to either preserving or even amplifying discontinuities given in the initial condition; thus, the capability of accurately representing these should be incorporated into the choice of numerical schemes. Suitable candidates would then be a finite volume or discontinuous Galerkin method. However, the finite difference method using a centered stencil or the continuous Galerkin method would give suboptimal results. The strong imposition of continuity in the domain would then yield spurious oscillations that affect stability.

We hence require the system of PDEs to be classified firsthand. We follow the popular taxonomy of second-order PDEs that can, for example, be found in the book by Bitsadze [47], but follow a more general method for determining the appropriate class [48]. That is, we define a singular governing equation in the form of a PDE to be either elliptic, parabolic, or hyperbolic, depending on the shape that its characteristic quadric takes in space. More formally, consider a differential operator L of the form

$$L[u] = \sum_{i,j=1}^{n} a_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j}, \tag{17}$$

where x_i are the dependent variables and a_{ij} is the matrix forming the coefficients of the highest spatial derivatives. Considering the eigenvalues λ_i of a_{ij}, L is called

- Elliptic, if all λ_i are either positive or negative;
- Parabolic, if at least one eigenvalue is zero and all others are either positive or negative;
- Hyperbolic, if at least one eigenvalue is positive and at least one is negative.

The characterization of first-order differential operators is more straightforward, however. It can be shown that first-order PDEs with constant, real coefficients are always hyperbolic. This condition is met for most cases relevant to engineering or physical applications. More precisely, a first-order PDE is hyperbolic if the resulting Cauchy problem is uniquely solvable. In the case of real, constant coefficients, the polynomial equation for each variable has to admit n solutions for an equation of order n while keeping all other variables constant. In the present case, this is trivially true.

We apply this classification for each governing equation of the independent variables for a given multiphysics problem. In practice, one may oftentimes identify the class by the differential operators that frequently appear in a given PDE. For example, a PDE that only has a Laplacian as a spatial differential operator—such as the Laplace equation $\Delta u = 0$ or the heat equation $\partial_t u - \Delta u = 0$—exhibits dissipative behavior and is prototypical for elliptic and parabolic PDEs. Oftentimes, one can easily identify a differential operator as parabolic if it has an elliptic operator in its spatial derivatives and an additional temporal derivative, as is exactly the case for the heat equation.

Both the abovementioned classes of PDE are dissipative, with the reason being that PDEs of second order can only have discontinuous derivatives along their characteristics. Since elliptic differential operators lack any characteristics, they strictly admit smooth solutions in that sense [48]. Thus, we associate this qualitatively dissipative behavior with elliptic and parabolic PDEs, as defined above.

However, the advection equation (Equation (1)) only has the gradient as a spatial differential operator, representing purely convective behavior. Exactly this behavior of transporting information through the domain with finite speed is associated with the wave-like character of hyperbolic equations. Figure 4 gives an overview of the classes of PDEs considered.

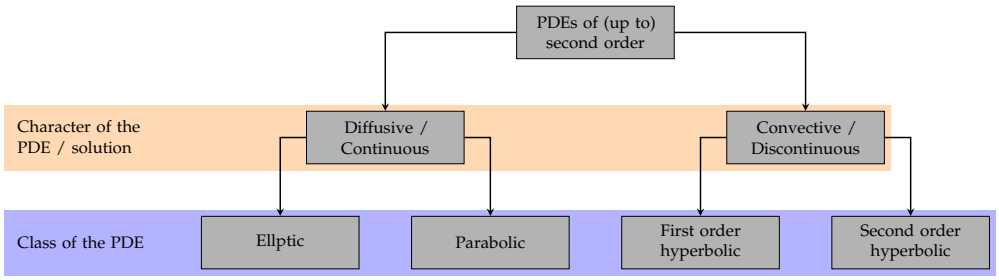


Figure 4. Classification of PDEs up to second order by qualitative nature and types following [47].

In alignment with the postulate at the beginning of this section, we aim to solve a given class of PDEs with as few degrees of freedom as possible whilst not overconstraining the solution.

Most importantly, discontinuities that might appear in the solution should be properly accounted for and reflect the choice of numerical scheme. The direct consequence is that methods enforcing continuity should be used for problems that qualitatively exhibit high regularity and continuity. From the previous discussion, it becomes apparent that this is the case for finite differences and continuous Galerkin finite elements. Problems that either conserve or even develop shocks, however, should be solved using methods that naturally allow for such. This means that either finite volumes or discontinuous Galerkin finite elements suit this requirement most naturally.

4.3. Domain Geometry

As discussed in Section 3.3, the discretization using finite differences inherently assumes an even grid with uniform spacing between nodal points. The direct consequence of this simplification is that assembly can be performed in the computational domain directly and in an equal manner for every node point.

In general, if the domain has a particularly simple shape, for example, a hypercube, and does not contain any holes, it can be triangulated using a Cartesian grid. Thus, if the discrete domain fulfills these conditions and the differential operators form an elliptic or parabolic PDE, using the FDM to efficiently assemble the global system is advisable.

For FVM, CGM, and DGM, regularity of the computational domain, in general, does not pose any considerable advantages that may accelerate the assembly of the discretized system.

4.4. PDE Linearity

Another crucial property to assess is its linearity. In this case, we disambiguate strictly linear, semilinear, quasilinear, and fully nonlinear equations, following the definition given in Evans [49]:

A k-th order partial differential equation of the form

$$F(D^k u(x), D^{k-1} u(x), ..., Du(x), u(x), x) = 0$$

is called

1. Linear, if it has the form
$$\sum_{|\alpha| \leq k} a_\alpha(x) D^\alpha u = f(x)$$
for given functions $a_\alpha (|\alpha| \leq k), f$. The PDE is homogeneous if $f \equiv 0$.
2. Semilinear, if it has the form
$$\sum_{|\alpha| = k} a_\alpha(x) D^\alpha u + a_0(D^{k-1} u, ..., Du, u, x) = 0$$

3. Quasilinear, if it has the form

$$\sum_{|\alpha|=k} a_\alpha D^\alpha u(D^{k-1}u, ..., Du, u, x) + a_0(D^{k-1}u, ..., Du, u, x) = 0$$

4. The PDE is fully nonlinear if it depends nonlinearly upon the highest-order derivatives.

While linearity does not pose much of a problem for elliptic or parabolic equations, it plays an important role in whether a discretization is stable for hyperbolic equations. The theory of nonlinear flux limiters is, in general, well researched for DG methods and largely profits from extensive developments that originally stem from the FVM. However, accurate computation and implementation remain to be a hurdle in practice. There have thus been several approaches to circumvent this issue, for example, by switching to an FV scheme in regions where there might be problems regarding the stability of the solution [50,51].

As the overarching goal of this method is to provide straightforward guidance for end users, we will omit such approaches that must in most cases be implemented in a custom and rather particular fashion in favor of simplicity.

We thus recommend that, for equations where the solution is not likely to require many nonlinear iterations per time step, one may safely use a DG scheme. In other cases where stability cannot be assured universally, one should, rather, switch to a finite volume formulation that may be overly diffusive, but on the upside is guaranteed to yield a stable solution.

4.5. Computing Environment

Within the last decade, advancement of computer hardware has been known to slowly hit the so-called memory wall [52]. That is, applications tend to be bound by the capability of the hardware to transfer memory instead of performing arithmetic operations. This, in particular, holds for numerical simulations that are performed using many workers or large problems. In such cases, the evaluation of sparse matrix-vector products poses high loads regarding memory bandwidth [53].

Then, there are numerical schemes that naturally lend themselves toward parallelism and others that are more memory-bound by design. Thus, for a given computing hardware that places enough emphasis on massive parallelism and two numerical schemes performing (nearly) identically, one should prefer the one that handles parallelism better. We thus naturally arrive at the question of where one should disambiguate between massively parallel and other, regular hardware.

There are essentially two factors that would affect such a classification. First, the hardware architecture itself plays an important role. We may, on one hand, solve a PDE on the classic CPU architecture that is capable of performing arithmetic on many precision levels and use many specialized instruction sets, such as AVX or fused multiply–add (FMA). Another possibility is the use of highly parallel computing units, such as general-purpose graphics processing units. Those, however, have a memory layout and instruction set that is much more tailored toward one purpose. In the case of a GPU, this is medium- to low-precision operations with comparably low memory intensity but instead high arithmetic effort.

The other deciding factor is the number of workers involved in the simulation process. The more workers that exist, the more processor boundaries that are present and, thus, more information has to be shared between processors. For some schemes, this overhead due to the exchange of memory between workers can become prohibitive. Within the finite volume method, for instance, parallel efficiency measured in GFlops/s starts to drop notably within the regime of 50 to 100 workers [54]. The quantitative drop-off also depends on the specific implementation, since other authors report slightly different results. Fringer et al., for instance, note a decline in parallel efficiency for a finite volume solver starting at 32 workers [55]. Thus, as a general guideline, we recommend employing methods that are suited for highly parallel environments at roughly 50 or more CPU workers. For

execution on massively parallel architectures, such as GPUs, the switch to such algorithms is considered necessary to obtain good efficiency.

4.6. Problem Scale

Another deciding factor for whether adaptivity is needed or not is the presence of multiple length scales in a multiphysics model.

We follow the definition given in [56,57] and characterize a PDE-governed problem to have a Multiscale nature if models of multiple spatial or temporal scales are used to describe a system. Oftentimes, this is the case if equations are used that originate from different branches of physics, such as continuum mechanics versus quantum mechanics or statistical thermodynamics.

This may, on one hand, be a physical process with slow and fast dynamics, for example, in chemical reaction networks. Then, the multiscale nature shows itself in the time domain of the problem. Another example of a commonly encountered problem in alloy design is the evolution of the temperature field and phase kinetics during heating and solidification. In this case, various length scales can be involved, such as in processes involving laser heating. The temperature gradients then involve resolutions at a scale of around 1×10^{-5} m, whereas the width of a solidification front, rather, reduces to a submicrometer scale, that is, around 1×10^{-7} m [58]. Regarding the previous definition, we have one model that is governed by laws of macroscale thermodynamics (that is, the heat equation). The other part of physics present is typically described by the evolution of a phase field. The corresponding equations of this model are, however, derived from the formulation of a free energy functional from Landau theory [59].

Due to the wide variety of physical processes and combinations thereof, formulating general criteria for the presence of a multiscale problem from a mathematical point of view is challenging. To the knowledge of the authors, there do not exist any metrics in the literature that would enable such a classification. We instead rely on the knowledge of the application expert who we assume to be familiar with the physics that should be captured. For a rough disambiguation, however, one may use the definition given above.

Such multiscale phenomena are prohibitively expensive to resolve on a uniform mesh due to the nonnegligible difference in the dynamics of the system. One option to efficiently resolve the physics at multiple scales is to employ different grids and solve the resulting problem in parallel. This has, for example, been conducted for the abovementioned case, specifically metal additive manufacturing [60].

A rather effective, alternative approach is the modification of the governing equations such that they become tailored to a specific numerical scheme. For instance, the well-known phase field model has been adapted using specialized stencils to the FDM such that spurious grid friction effects are eliminated [61,62]. This approach, however, requires extensive knowledge about the numerics as well as the physical nature of a given problem.

Another possibility that requires fewer adaptions of the code to the specific problem is to make use of grid adaptive algorithms. This approach for the problem presented is a popular alternative and has been implemented multiple times [63–66]. Thus, grid adaptivity plays a key role in creating solutions to such problems, if the domains are not to be resolved on different discretizations entirely. Numerical methods as a consequence need to reflect on this requirement; as such, finite difference methods are not suitable for such types of problems.

CG finite element methods do enable grid as well as polynomial degree adaptivity. Yet, the imposition of hanging node constraints is oftentimes not trivial. Though there have been considerable strides toward easy and intuitive handling of hanging nodes for continuous elements [67,68], these methods naturally fall short of the inherently decoupled nature of DoFs present in discontinuous methods.

Whereas grid adaptivity is easily realizable within FVM, there is little room for adaptivity regarding the order of approximation and it can, at best, be achieved using varying reconstruction stencils [15].

By far, the most naturally suited method for h- as well as p-adaptivity is the DG FEM. The locality of DoFs enables the splitting of cells without the need for hanging node constraints. The same argument applies to altering the degree of a finite element, as additional DoFs within the cell need not be attached to a counterpart on its neighbors.

4.7. Summary

We may now condense the various aspects of choosing appropriate numerical schemes as follows into a unifying method, given the restrictions we posed in Section 4.1.

First, we take three sets of inputs that are of practical relevance: the mathematically formulated, continuous problem, the computational domain that one wishes to solve the former on, and the configuration of the target hardware.

To design the intended decision process, we start by evaluating the decision metrics that impact the target scheme in the most general manner at first. The general question of whether the prescribed system of PDEs requires an efficient solution on a large scale fulfills this requirement here. By the term large scale, we understand state-of-the-art computing hardware on massively parallel architectures. That decision, in turn, is influenced by two factors: One may directly intend to efficiently solve the system of PDEs on that hardware, or the multiscale nature of the problem demands such a computing environment. If either is the case, solving the entire system using the HDG method is advisable due to the resulting locality of the problem.

The remaining parts of the decision process depend on the class of PDE present. From here on, we operate in a field-wise manner and classify the system of PDEs for each independent variable separately. If a PDE is convective in character, that is, hyperbolic, we recommend the use of numerical schemes that incorporate discontinuous approximations. But, if a problem is diffusive by nature, the solution will be continuous and, thus, the use of continuous approximations is more advisable.

In the case of the former, following the discussion in Section 4.4, a final disambiguation must be made regarding linearity. If the PDE is linear or semilinear, a DG scheme can be applied due to the unlikeliness of stability issues. Otherwise, the use of a simple FV scheme is more advisable to obtain a stable solution without having to iterate through many different choices of flux limiters in a trial-and-error fashion.

Regarding the continuous schemes, as was explained in Sections 3.3 and 4.3, the configuration of the domain geometry plays an important role in the efficiency of the overall scheme. If the domain is Cartesian, irrespective of dimensionality, the FDM can deliver accurate results with a considerably decreased number of arithmetic operations. The conceptual flexibility of the FEM regarding the domain is then unnecessary. In the other case, though, where the domain is topologically more complex, relying on FEM algorithms that account for the necessary global mappings is more appropriate. It would, of course, be possible to identify a middle ground between both schemes, for example, when a simple and prescribed transformation can be applied to the entire domain. This would, for example, be the case for systems that can be described by polar coordinates. However, few computer codes implement such functionality. As the focus of this method lies on practicality and usefulness, we rather choose a method that can make use of widespread and established computer codes and thus omit these possibilities.

As a result, we obtain one process that guides the user through iteratively selecting the most appropriate combination of numerical schemes for a given, fixed, and well-defined set of inputs. This method may be summarized in a flow chart, which is depicted in Figure 5.

Figure 5. Graphic summary of the proposed process for choosing appropriate numerical schemes. Inputs (**I**) are given by purple trapezoids, decision points (**D**) by white diamonds, and processes (**P**) by orange rectangles. Processes with additional vertical bars denote more complex processes and have references to their respective sections. Results are shown in green trapezoids.

5. Examples

The purpose of this section is to walk through the proposed method, employing two simple example PDEs. Although these are not multiphysics problems, they may be combined in theory.

5.1. Allen–Cahn Equation

First, we consider the following scalar PDE together with zero flux boundary conditions to be imposed at the four borders of a rectangular domain $\Omega : [0; L] \times [0; W]$:

$$\frac{1}{K}\partial_t \phi - \Delta\phi = -\frac{2}{\xi^2}\partial_\phi g(\phi) - \frac{\mu_0}{3\gamma\xi}\partial_\phi h(\phi) \quad \forall \phi \in \Omega, \tag{18}$$

$$\frac{\partial \phi}{\partial n} = 0 \quad \forall x \in \partial\Omega. \tag{19}$$

This equation is called the Allen–Cahn equation and describes the time-evolution of a scalar, nonconserved order-parameter field ϕ, as is often called the phase field. The equation is commonly used in the modeling of self-organized microstructure evolution or complex pattern formation processes, as driven by local thermodynamics and/or mechanics. The phase field variable ϕ can be understood as a coloring function that locally indicates the presence or absence of a certain phase or a certain material state within a given microstructure. For instance, in modeling of microstructure evolution during solidification, $\phi = 1$ may denote the local presence of the solid and $\phi = 0$ may denote the local presence of the liquid phase [61,62]. If applied to the description of crack propagation, the order-parameter field ϕ is understood as the local material state, which can be either broken $\phi = 1$ or not $\phi = 0$ [69,70].

The scalar quantities K, ξ, μ_0, and γ are model constants that determine the evolution of the scalar field ϕ, and we adopt the notation of [71]. The polynomials g and h on the right-hand side of Equation (18) pose a nonlinearity to the equation. Their derivatives are given by $\partial_\phi g(\phi) = 2\phi(1-\phi)(1-2\phi)$ and $\partial_\phi h(\phi) = 6\phi(1-\phi)$. In the following, we will gather those polynomial terms in the joint potential term $f(\phi) = 2\partial_\phi g(\phi)/\xi^2 + \mu_0 \partial_\phi h(\phi)/3\gamma\xi$. Further details on the parametrization of the model are given in the Appendix A Table A1.

Concerning the Allen–Cahn equation, we will consider two different scenarios, highlighting different aspects of the physics behind the equation. For each of these two scenarios, we formulate quantitative measures to be able to adequately question the accuracy of the numerical solutions.

In the first scenario, we consider the motion of a planar interface between two phases at different energy density levels. The low-energy phase is expected to grow at the expense of the high-energy phase, which induces a motion of the interface between them at a velocity proportional to the constant energy density difference μ_0. The scenario is realized as a quasi 1D problem $[0; L = 100] \times [0; W = 1]$, where the interface normal direction is pointing in the x-direction and the use of simple von Neumann boundary conditions with zero phase field fluxes at the borders of the rectangular domain is legitimate. The realization of this scenario with tilted interface orientations, including the formulation of appropriate boundary conditions on the borders of the rectangular domain, is discussed in detail in [62]. In this highly symmetric quasi-1D case, the scenario can be quantitatively evaluated using the existing analytic solution for the phase field:

$$\phi(x,t) = \frac{1}{2}\left(1 - \tanh\frac{x - \tilde{x}(t)}{\xi}\right), \tag{20}$$

where the time dependence of the central interface position $\tilde{x}$ is given by $\tilde{x}(t) = x_0 + M\mu_0 t/\gamma$, with the initial position at $x_0 = 20$. The initial condition of this problem is, thus, formed by evaluating Equation (20) at time zero. To investigate the impact of arithmetic complexity on computational efficiency, we will seek an approximation of this real-valued function $\phi(x,t)$ in one spatial dimension on a $[0; 100]$ grid with equispaced vertices.

We can now start applying the proposed methodology, as described above in Section 3 and following Figure 5. That is, we follow the path of the flowchart from the top to the bottom. We first classify the hardware scale according to P1 in the figure. The given hardware architecture, that is, an 8-core CPU system, falls well below the established

recommendation for the threshold of partitioned problems, which is at least 50 workers. Therefore, there is no need from a hardware side for massive parallelism.

Next, we investigate the problem scale with process P2. The problem is governed by one scalar equation and there are no submodels involved, as defined in Section 4.6. Concerning the length scales, the presented system exhibits one extra physical length scale and that is the width ξ of the diffuse interface. This extra physical length scale originates from the nonlinearity of the Allen–Cahn equation and complements the other length scales, such as the dimensions of the domain, as well as the grid spacing, both being more natural in the numerical solution of PDEs. This poses the issue of numerical resolution of the systems length scales, that is, both the domain dimensions, as well as the width ξ of the diffuse interface, need to be properly represented on the discrete numerical grid [61,62]. However, the fact that the problem is quasi-one-dimensional restricts the computational demands of the scenario. We thus arrive at the first decision point D1, where we can negate the necessity for massive parallelism.

The next process step, P3, involves classifying the problem at hand, following the definition given in Section 4.2. As dependent variables, we encounter the time t as well as the spatial components x and y. The coefficient matrix a_{ij}, summing up all leading coefficients of second derivatives, then becomes for the 2D case

$$a_{ij}^{AC} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{21}$$

In this case, enumerating the eigenvalues λ_i is trivial, since a_{ij}^{AC} is a diagonal matrix, and we have $\lambda_0 = 0, \lambda_1 = -1, \lambda_2 = -1$. We thus find that one eigenvalue is zero since the temporal derivative is only of first order and all other eigenvalues are of the same sign. Therefore, Equation (18) is a second-order PDE of parabolic type and we can proceed in D2 with the left branch.

Moving on in the decision process, we would next classify the problem domain in D3 given input I3. As we use an equispaced grid in 1D, the discretization is Cartesian; thus, solving the problem using finite difference would be the best choice. As there are no other fields to classify according to decision point D5, we conclude the decision process. Within the unified methodological framework, we implement both FD and CG schemes and the scenario is comparatively solved using both schemes. This allows us to compare the schemes concerning numerical resolution capabilities and to investigate differences in the mutual arithmetic complexity and their impact on efficiency.

Evaluating the CG method requires the reformulation of Equation (18) in its weak form, though. The finite dimensional weak statement is, then, as follows: Find $\phi_h \in V_h$, such that

$$\int_\Omega \frac{1}{M} v_h \partial_t \phi_h \, dx + \int_\Omega \nabla v_h \nabla \phi_h \, dx - \int_\Omega v_h f(\phi_h) \, dx = 0 \quad \forall v_h \in V_h, \tag{22}$$

where we have already assumed the solution and test function to lie in the finite-dimensional subspace V_h.

Equation (18) requires the discretization of the Laplacian as its only differential operator. The temporal derivative will be treated using the method of lines approach, that is, we solve a large system of spatially discretized ordinary differential equations.

The finite difference discretization of the Laplacian results in the well-known second-order central difference stencil

$$\Delta\phi \approx \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2}. \tag{23}$$

The nonlinear right-hand side $f(\phi)$ must be updated every time step using the current value of ϕ. As such, we do not need to perform any assembly and can even avoid forming

a global system of equations. Instead, we rely on (23) for the Laplacian, which can be handily vectorized. There is also no need to perform any mapping between the reference and physical domain, as explained in Section 3.3.

For the finite element discretization, we need to perform all these steps, resulting in a global nonlinear system of equations for each time step. The discrete form of Equation (22) then reads

$$\underline{\underline{M}}\partial_t\underline{\phi} + \underline{\underline{K}}\,\underline{\phi} - F(\underline{\phi}) = 0. \tag{24}$$

where we introduced the mass matrix $\underline{\underline{M}}$ and the stiffness matrix $\underline{\underline{K}}$ for the Laplacian. These represent the spatially discretized differential operators that act on the vector of degrees of freedom $\underline{\phi}$. The algebraic terms that are nonlinear in $\underline{\phi}$ are gathered in the discrete vector $F(\underline{\phi})$. For the sake of comparison regarding efficiency, we require the resulting fields of both schemes to be (nearly) identical apart from floating point errors.

Given this requirement, we note that the finite difference formulation lacks an analogous term to the finite element mass matrix. We consequently require M to be the identity matrix in an equivalent finite element formulation, given that all other terms are equal. The latter can easily be verified for a stiffness matrix assembled with first-order Lagrange polynomials and a collocation method. The derivation of such an equivalent scheme was covered in Section 3.3. Using collocated finite elements is chosen here for the sake of comparison as well as for computational efficiency. The resulting mass matrix can then be inverted trivially by taking the element-wise inverse instead of computing the full inverse. Such an operation is considerably more expensive and should thus be avoided if possible.

To compare both schemes regarding efficiency, we implement both schemes from scratch within the Julia programming language [72]. Due to its flexibility, high-level syntax, and simultaneous, granular control over various performance aspects via its rich type system, Julia has gained considerable momentum in the past few years within the scientific community. We carefully set up both schemes using analogous data structures to enable a side-to-side comparison of the computational complexity. The most high-level parts of the codes are given in Listing 1.

We also include the functions that are called within each time step to solve the semidiscrete system, to give a high-level view of which steps are necessary and how they are implemented in particular. Both semidiscrete systems use in-place operations to avoid memory allocations. For the CG-FEM code, we implement a full mesh topology to solve the problem with a first-order method, although both discretizations consist of Cartesian meshes. One could in this case assume a globally constant Jacobian and, thus, save a considerable amount of arithmetic complexity. However, this would skew the results regarding performance and would not make full use of the flexibility of the FEM.

It becomes immediately apparent from the comparison that solving the Allen–Cahn equation using finite elements requires an assembly process that is noticeably more complex. The only arrays that need to be stored for the FD version are the grid coordinates and the solution array. Because the latter can be arranged in memory such that it represents the Cartesian topology of the grid, one can simply point to the neighbors of a vertex in memory without having to look up the vertex–vertex connectivity. This is not the case for the FEM. Instead, we encounter an additional indirection through a cell-vertex list, where we gather all DoFs associated with the currently visited cell.

We furthermore cannot construct the global linear system at once, but need to go through the cell-wise assembly process which effectively leads to most of the nonzero matrix entries being visited multiple times. This is in sharp contrast to the FDM where the global system is only present implicitly through functions that apply the Laplacian stencil. As a consequence, memory requirements are greatly reduced.

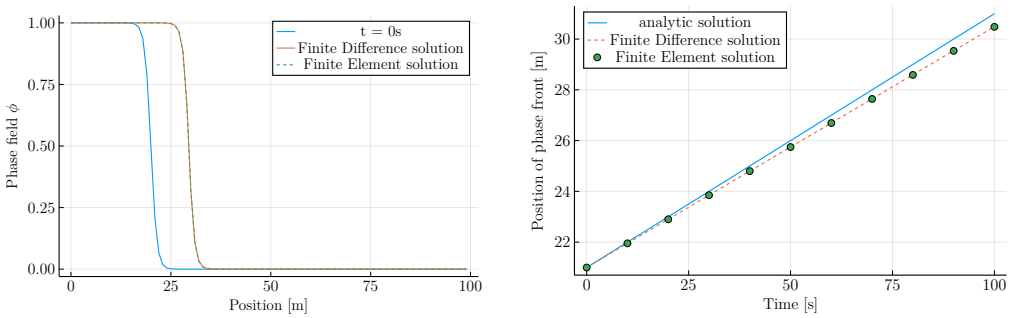
Listing 1. Top-level overview of the necessary data structures and the functions to update the semidiscrete systems for the CG FEM (left) and FDM (right). In the case of the FDM, one can avoid assembling a global linear system entirely; thus, the top-level data structure only holds the solution and grid as large arrays. For the FEM, assembly on general grids in a matrix-free manner is far from trivial. Additionally, the triangulation data structure is more complex due to the necessary topological information. Furthermore, the reference FE needs to be stored and correctly mapped using Jacobian values. The full code is available in the code repository mentioned at the end of this article.

```
struct FETriangulation{V,C}<:Triangulation
    vertices::V
    connectivity::C
    dim::Int
end

struct FiniteElement{E<:ElementType,
        P<:Primitive,B<:AbstractMatrix,
        Q<:AbstractVector,G<:AbstractArray}
    primitive::P
    element_type::E
    order::Int
    ndofs::Int
    basis_coeffs::B
    quadrature_nodes::B
    quadrature_weights::Q
    basis_at_quad::B
    grad_basis_at_quad::G
    grad_monomial_basis::G
end

struct AssemblyCache{T<:AbstractVecOrMat}
    coeffs::T
    loc::T
    glob::T
end

struct CGProblem{T<:Triangulation,J,P,C
      E<:FiniteElement,M<:AbstractMatrix,
      V<:AbstractVector,F<:Function}
    triangulation::T
    referenceElements::E
    detJ::J
    bilinearForm::M
    linearForm::V
    u::V
    massMatrix::Union{M,Nothing}
    parameters::P
    rhs::F
    cache::C
end

function (a::CGProblem)(du,u,p,t)
    mesh        = a.triangulation,
    element     = a.referenceElements
    K           = a.bilinearForm
    F           = a.linearForm
    detJ        = a.detJ
    params,     = a.parameters
    cache,rhs   = a.cache, a.rhs
    M           = a.massMatrix

    mul!(du,K,u,-1.,0.)
    # f(u) changes, thus we reassemble
    assemble_F!(F,cache,u,element,
            mesh,detJ,params,rhs,M)
    du .+= F

end
```

```
struct FDTriangulation{V,D}<:Triangulation
    vertices::D
    h::V    # dx in each dim
    dim::Int
end

struct FDProblem{T<:FDTriangulation,
        B<:Function,L<:Function
        V<:AbstractArray,P,BC<:Function}
    triangulation::T
    order::Int
    bilinearForm::B
    linearForm::L
    u::V
    parameters::P
    boundaryCondition::BC
end

function (a::FDProblem)(du,u,p,t)
    apply_bilinear! = a.bilinearForm
    apply_linear!   = a.linearForm
    apply_bc!       = a.boundaryCondition
    h               = a.grid.h
    params          = a.parameters

    a.bilinearForm(du,u,h)

    a.linearForm(du,u,a.parameters)
    a.boundaryCondition(du)
end
```

For transient problems, one needs to additionally make a suitable choice for the temporal discretization, that is, the choice of method as well as the time step. Here, we make use of the well-optimized Julia library `DifferentialEquations.jl` [73]. As an exemplary implementation of modern, high-performance codes for the solution of ordinary differential equations (ODEs), this package offers various algorithms that are capable of adaptive time stepping such that an application expert does not need to provide any input regarding temporal discretization. Here, in particular, we can even make use of built-in heuristics that automatically select a suitable integration scheme, based on the supplied ODE problem [74]. The resulting effort for the end user can be condensed to selecting a suitable numerical scheme for the spatial discretization, as outlined by Figure 5, and leave the problem of tuning the spatial discretization aside entirely, as it is not required for the application expert in the formulation of the ODE given above. For this particular problem, we prescribe the use of an adaptive, implicit, fourth-order Rosenbrock method for the temporal evolution of both FD and CG-FE systems to achieve a fair comparison between both solutions. This solver is stable and third-order accurate when used on nonlinear parabolic problems [73]. The spatially discretized, nonlinear problems at each time step are solved explicitly according to the timestep update functions given in Listing 1.

Before comparing both schemes regarding computational efficiency, we first verify that the FD and collocated CG-FE schemes produce identical results. Figure 6 shows the solutions of both schemes for solving the phase field evolution (Figure 6a) and for modeling interface position over time (Figure 6b).



(**a**) (**b**)

Figure 6. Solution of the one-dimensional Allen–Cahn equation using the finite difference and finite element method. (**a**) Evolution of the phase front at $t = 100$ s with respect to the initial condition. (**b**) Comparison of finite difference and finite element solutions to the analytical solution given by Equation (20).

As can be observed, both schemes produce visually identical results. The quantitative differences in the numerical results are minimal and can be attributed to floating point errors that accumulate over the process of time integration. However, there is a considerable difference between the analytical and the numerical interface velocity, as visible in Figure 6b. The reason for this discrepancy is grid friction, which results from the limited numerical resolution of the diffuse interface profile and could be reduced by increasing the dimensionless ratio $\xi/\Delta x = 1.5$, where Δx denotes the grid spacing [61,62]. Grid friction and pinning during stationary interface motion has been studied previously, for instance, by [75,76]. So far, this detrimental effect has been only studied using finite-difference-based schemes. It relates to metastabilities that result from a broken translational invariance of the discrete numerical schemes [61,62]. As a consequence, the current interface velocity oscillates as the center of the interface passes one grid point after the other. If, further, the time discretization error is small enough, the average interface velocity turns out to be considerably below the expectation. With decreasing phase-field width, we obtain increasingly larger deviations of the average velocity as well as increasing larger oscillations. This culminates in a vanishing velocity, where the phase field is pinned to the computational grid. Interestingly, spurious grid friction and pinning can be eliminated in

fast Fourier and finite difference implementation of the Allen–Chan equation, using the newly proposed sharp phase field method [77,78]. For phase field models, which use a double-obstacle potential instead of the double-well potential, a comparable technique was proposed by Eiken [79]. It is quite interesting to note that this discretization error, which is very characteristic for the Allen–Cahn equation, turns out to be so very similar for the two different numerical schemes in this case. This again highlights the close relationship of the two different numerical schemes.

Furthermore, we point out that computational resource usage differs considerably. Table 2 reports some descriptive statistics on the performance of both implementations. These differences in run times as well as memory consumption can be attributed to multiple factors. First, the nonlinear right-hand side changes each time step; thus, assembly has to be performed dynamically for the finite element method. The finite difference method in contrast can simply rely on point-wise evaluation of the strong form instead of numerically computing the weak form integrals. Secondly, the finite difference method does not need to perform any mapping during the time step as no assembly is required. During computation of the right-hand side integral, this is a necessity for the finite element method.

Table 2. Run times of the finite element and finite difference model of the 1D Allen–Cahn equation. Both models were run on identical hardware and Julia 1.8.5 with LLVM 13.0.1 [72]. Time stepping was performed using the `DifferentialEquations.jl` library [73]. Linear algebra operations were performed using OpenBLAS [80,81] on a single-threaded Apple M1Pro ARM processor. Fast evaluation of fused array expressions was provided by the `Tullio.jl` library [82]. Allocated memory refers to the physical size of the problem-specific data structures given in Listing 1. The sample size for each scheme is $n = 100$.

	FDM	**FEM**	**Relative**
Median run time	0.450 ms	9.503 ms	21.1×
Mean run time $\pm 1\sigma$	0.578 ms ± 0.759 ms	9.643 ms ± 0.741 ms	16.7×
Allocated memory	1.446 kB	15.598 kB	10.8×

The largest performance discrepancy, however, can be attributed to the fact that the finite difference method can operate in a matrix-free manner due to the Cartesian grid it is applied on. As all vertices are equispaced, there exists one global stencil that can be applied on each vertex independent of all other members of the grid. The finite element method, in contrast, uses the grid topology to accumulate the weak form integrals into corresponding entries of the global system matrices and vectors. Thus, it always produces a typically very sparse global system that cannot be vectorized similarly. It should be noted that the discrepancy in results should not be expected to be as drastic as shown for linear problems, as then the FEM does not require the reassembly of the right-hand side. The computational advantage then reduces to the matrix-free evaluation of the linear system.

In the second scenario, we investigate a more practically relevant benchmark in two dimensions and turn to the well-known vanishing grain problem, leaving all other aspects of the problem as is. Here, the dissolution of a circular-shaped nucleus under the interface energy density pressure under two-phase equilibrium condition $\mu_0 = 0$ is simulated. These dynamics are also governed by the Allen–Cahn equation and denote the complementary physical effects as compared to the above scenario. In a sharp interface picture, with a constant and isotropic interface energy density γ, we expect the temporal evolution of the grain radius to be given by

$$r(t) = \sqrt{R_0^2 - 2Mt}, \tag{25}$$

where R_0 indicates the initial radius and M is the phase field mobility. Snapshots of the phase field ϕ at initial and terminal times are given in Figure 7.

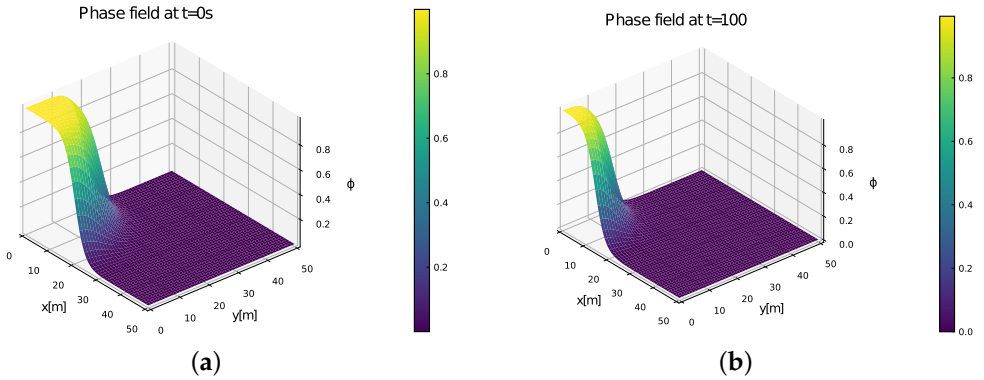


Figure 7. Solution of the two-dimensional Allen–Cahn equation using the finite difference and finite element method. (**a**) Initial configuration of the phase field. The domain shows a quarter-slice of the nucleus. (**b**) Phase field distribution within the quarter domain after 100 s. The nucleus has shrunk to a smaller radius whilst retaining the interface profile.

We also report the temporal evolution of the radius function for solving this scenario using both numerical methods in Figure 8. As in the one-dimensional simulation, the collocated FE and FD solutions behave identically to each other. Both exhibit a notable discrepancy towards the sharp interface behavior, which again relates to known issues of finite numerical resolution in the phase field simulation [62]. We note at this point that the solution generated by a common FE model, i.e., with fully accurate quadrature, as explained in Section 3.3, also produces very similar results given that all other parameters are chosen the same, albeit with a large computational disadvantage due to the full inversion of the resulting mass matrix. The linked code repository at the end of this article contains the necessary data structures to reproduce these results.

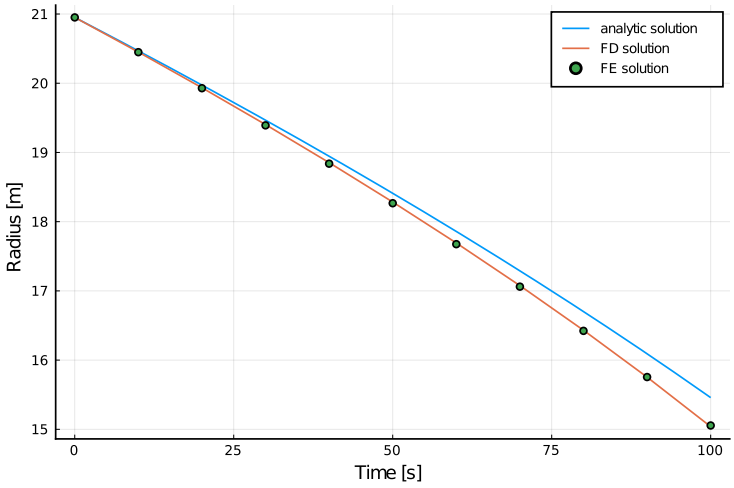


Figure 8. Evolution of radius over time of the vanishing grain problem. Both finite difference and finite element solutions show a considerable, accumulating error toward the analytical solution.

To assess the performance gap of both schemes for higher dimensions, we again benchmark both codes against each other. The results are given in Table 3. Comparing the results from the 2D simulation benchmark in Table 3 with its 1D counterpart (Table 2), we find that the discrepancy in performance becomes noticeably more drastic with increasing dimensionality of the problem. This can be attributed to the increased scattering of DoFs in memory. Thus, memory access is less stridden, increasing the lookup time. For a hardware architecture that demands more parallelism and has a shared memory architecture, this could quickly evolve into a serious bottleneck.

Table 3. Run times of the finite element and finite difference model of the 2D Allen–Cahn equation. Both models were run on identical hardware and Julia 1.8.5 with LLVM 13.0.1 [72]. Time stepping was performed using the `DifferentialEquations.jl` library [73]. Linear algebra operations were performed using OpenBLAS [80,81] on a single-threaded Apple M1Pro ARM processor. Fast evaluation of fused array expressions was provided by the `Tullio.jl` library [82]. Allocated memory refers to the physical size of the problem-specific data structures given in Listing 1. The sample size for each scheme is $n = 100$.

	FDM	**FEM**	**Relative**
Median run time	4.743 ms	192.837 ms	40.7×
Mean run time $\pm 1\sigma$	4.731 ms ± 0.059 ms	192.787 ms ± 0.219 ms	40.7×
Allocated memory	21.490 kB	494.730 kB	23.0×

Of course, this scenario has been discussed already in many different works. Quite often, it is used to highlight accuracy gains or performance improvements of advanced numerical techniques, that are particularly suggested to solve the Allen–Cahn problem. For instance, Gräser et al. discussed the scenario in the context of solving the anisotropic Allen–Cahn equation using fully implicit or a linearized time discretization and semi-implicit time discretizations and globally convergent truncated nonsmooth Newton methods. They provide information on resulting differences in the achieved accuracies and concerning the complexity of the schemes [83]. Another example is the nonlinear preconditioning for diffuse interface models based on the Allen–Cahn equation, as first suggested by Glasner [84]. Interestingly, this preconditioning technique seems to be related to the abovementioned sharp phase field method suggested by Finel et al. [77]. Both methods provide a tremendous improvement potential, as demonstrated by a comparative study using this scenario.

5.2. Two-Phase Advection

As a second model problem, we will investigate the advection equation in two dimensions. This problem is well studied in the literature and is known as challenging to solve accurately. Due to the absence of dissipative terms, numerical algorithms oftentimes struggle to converge towards the entropy solution and either produce spurious oscillations, rendering the solution unstable, or yield overly diffusive approximations, where conservation laws are violated [85]. We choose this problem in particular due to being simple yet challenging enough to study. In addition, the advection equation frequently arises in modeling multiple phases in an Eulerian framework and the motion of immersed immiscible fluids in general. It is, thus, of high relevance in a multitude of multiphysics problems.

In particular, we investigate a pure advection problem involving two phases. We choose to describe the motion of two fluids and track the volume fractions α_i, as is common for the volume-of-fluid (VoF) formulation [86]:

$$\partial_t \alpha + \underline{u} \cdot \nabla \alpha = 0, \tag{26}$$

$$\alpha_1 + \alpha_2 = 1, \tag{27}$$

$$\Omega \in [0;5] \times [0;5], \tag{28}$$

$$t \in [0;5]. \tag{29}$$

The initial condition to this problem is given as a rectangle function that is one in the interval $x \in [2;3] \times [2;3]$ and zero everywhere else. One may alternatively track only the motion of the interface using a coloring function ϕ. This is common for the level set method; the governing equation, however, is the same as Equation (26).

We would like to solve this problem on three different architectures to showcase the effect of parallelism on the efficiency of numerical schemes. The choices of hardware along with important quantities are given in Table 4.

Table 4. Hardware configurations for the advection equation model problem. The three setups mimic popular computing environments in applied settings: a mobile computer, a stationary workstation grade tower, and a server tailored to numerical computing.

CPU Name	Number of Cores	Core Clock Speed [GHz]	Memory Size [GB]	Memory Bandwidth [GB/s]	Memory Speed [GHz]
Apple M1 Pro	8	3.2	16	200	6.4
Intel Xeon W-2295	18	3.0	128	94	2.9
2x AMD EPYC 7763	128	2.45	512	204	3.2

We once again follow the process summarized in Figure 5. Regarding the system of PDEs (I1), Equation (27) is simply an algebraic constraint and thus can be calculated in a simple postprocessing step. Thus, (27) is not a governing equation in the sense of a PDE, and α_2 will consequently not be considered an independent variable, as detailed in Section 4.2. We are then left to solve a single scalar advection equation for α_1.

Proceeding in the flow chart, we next classify the hardware scales within P1. Here, we find that the last hardware configuration listed in Table 4 necessitates the use of schemes that are tailored for high parallelism, as the given amount of 128 processes is above the specified regime where the use of parallelizable algorithms is worthwhile using. Therefore, this configuration should be run using a discontinuous Galerkin method. For both other configurations using 8 and 18 processes, this does not apply. Continuing with process P2, we find that the given problem only exhibits one length scale; thus, this criterion for parallelism can be omitted. Thus, we arrive at decision D1 and find that for the problem statement involving the largest of the three computing architectures, the use of the discontinuous Galerkin method is advised.

For the remaining two configurations, we can proceed by classifying the PDE according to process P3. With the temporal derivative and gradient as the only differential operators, Equation (1) is a first-order PDE. The advection velocity vector $\underline{u}$ has constant and real components. Thus, following Section 4.2, we find that the advection equation presented here is hyperbolic and proceed with the right branch of the flow chart after decision D2.

Consequently, we need to evaluate the linearity of Equation (26) for process P4 as a next step. As the terms including the differential operators are linear and there is no right-hand side, it may be classified straightforwardly to be linear. Due to its linearity, the most efficient choice for the remaining configurations turns out to be the discontinuous Galerkin method as well. As stated previously, the original two-equation system only consists of one PDE; thus, we conclude the decision process here, as all fields governed by a PDE have been assigned (decision D5). It should be noted here that when the VoF method is used together with the finite volume method, one can expect the interface of two phases to diffuse considerably. Within the numerics community, this problem is well known and has led to the practice of introducing an additional interface compression term into the advection equation [87].

$$\partial_t \alpha + \underline{u} \cdot \nabla \alpha + \nabla \cdot [\underline{u}_c \alpha (1 - \alpha)] = 0, \tag{30}$$

$$\underline{u}_c = \min(c_\alpha |\underline{u}|, \max(|\underline{u}|)) \frac{\nabla \alpha}{|\nabla \alpha|}, \tag{31}$$

where the newly introduced, artificial interface compression velocity $\underline{u}_c$ is used. In this case, c_α is a positive, scalar constant somewhere around unity. As can easily be observed by combining Equations (30) and (31), this makes the resulting PDE fully nonlinear, which can be regarded as a severe disadvantage in this case. This additional, artificial interface motion counteracts the diffusion that the finite volume method has shown to exhibit. As

will be shown, such an additional term can be omitted if the problem is solved using the recommended DG scheme due to the considerably lowered numerical diffusion. In summary, choosing the FV over the recommended DG method, in this case, would not only require evaluating a fully nonlinear variant of the advection equation to gain accurate results, but it would also force the application expert to pick an appropriate value for an artificial model constant, which can be subject to tedious calibration work.

In the following, we will compare this particular choice of method with the finite volume method, which would be the next alternative and is, in principle, also well suited to tackle such problems. The continuous Galerkin and finite difference methods do not lend themselves well to solving such equations and will thus be omitted from this benchmark. In particular, the CG method is known to be unstable for first-order hyperbolic equations, as the stability of the scheme can be shown to be dependent on mesh size [42].

One must add that in principle, the finite difference method could be applied here, where, however, two different limitations apply. First, the only choice of stencil that would be stable for this equation is the forward difference (or upwind) approximation. This choice, however, is not covered by the proposed decision process, as it is formally equivalent to a continuous Petrov Galerkin method. One can show that this scheme corresponds to a simplified streamline upwind Petrov Galerkin (SUPG) method. As we have restricted ourselves for the sake of decidability to Bubnov Galerkin methods, this stencil is not admissible here. Secondly, using the finite difference method here implies the strict use of a Cartesian grid.

We thus proceed to write the weak form for Equation (26) by multiplying with a discrete test function v_h, integrating over the whole domain Ω and subsequently performing integration by parts. Find $\alpha_h \in V_h$ such that

$$\int_\Omega v_h \frac{\partial \alpha_h}{\partial t}\, dx + \int_\Gamma v_h \alpha_h (\underline{u} \cdot \underline{n})\, dS - \int_\Omega (\nabla v_h \cdot \underline{u}) \alpha_h \, dx = 0, \qquad \forall v_h \in V_h, \tag{32}$$

$$\alpha_{h,\tilde{\Gamma}_{b,l}} = \alpha_{h,\tilde{\Gamma}_{t,r}}, \quad \forall x \in \partial\Omega, \tag{33}$$

where Γ denotes the union of all interior and exterior facets of the domain and $\tilde{\Gamma}$ are the subsets of the domain boundary $\partial\Omega$. In this case, specifically, $\tilde{\Gamma}_{b,l}$ are the slave facets at the bottom and left boundary that the values of the slave facets from the top and right master facets $\tilde{\Gamma}_{t,r}$ are mapped to. This PDE in combination with periodic boundaries possesses an analytical solution of the form

$$\alpha(x,t) = \alpha(x - ut, 0). \tag{34}$$

That is, after traversing the quadratic domain with the given velocity $\underline{u} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$, the solution field must exactly correspond to the initial condition. Verification of numerical results is, thus, very straightforward.

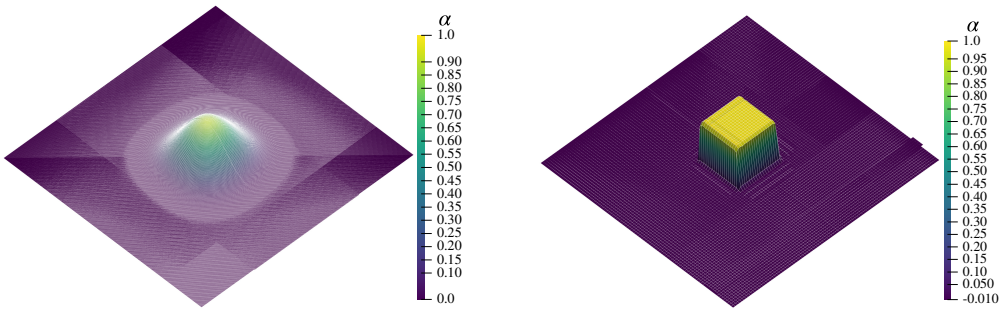
We solve this problem using the Firedrake problem-solving environment along with the popular libraries PETSc and Scotch for efficient parallel computing [3,88–95]. As the finite volume method can simply be understood as a discontinuous Galerkin method of polynomial degree zero, the implementation is virtually the same for both schemes.

Note that for the finite volume method, the last term in Equation (32) becomes zero since the derivative of a constant vanishes. Thus, we omit this term from the assembly to save computations and to more accurately represent the arithmetic intensity posed by the original formulation of this scheme.

For the sake of visualization, we project the solution onto a first-degree space with H^1 continuity. The equations are solved by a three-stage implicit Runge–Kutta method. In both cases, careful attention has to be paid regarding the time step. For hyperbolic problems of such time, the time step where stability is given is strictly bounded by the Courant–Friedrichs–Lewy number CFL $\leq \frac{1}{2k+1}$ for a scheme of degree k [96]. One can easily verify that for a finite volume scheme, this corresponds to the well-known condition that the CFL number must stay at or below unity. Not only does that mean, regarding

arithmetic complexity, that the FVM has a simplified assembly process, but it also means that the admissible time step is, in general, larger than for DG methods. This discrepancy drastically increases with the polynomial order taken for the DGM. The corresponding linear systems are each solved using the same matrix-free algorithm with Krylov subspace preconditioning.

The corresponding results of the simulation are shown in Figure 9.



(**a**) (**b**)

Figure 9. Solution of the two-dimensional advection equation using the finite volume and discontinuous Galerkin method. Both models were run using the identical amount of global degrees of freedom set to 147.456. Element facets are drawn in white to illustrate the difference in grid size. (**a**) Finite volume solution at end time on a 384 × 384 grid. The solution is heavily diffused to a parabolic profile. (**b**) Discontinuous Galerkin solution at end time on a 96 × 96 grid using third-order polynomials. One can observe the presence of spurious oscillations that remain stable in magnitude throughout the simulation.

The differences in accuracy are obvious. The DG method can capture the rectangular profile throughout the simulation with relatively good accuracy, while the FV simulation is strongly diffused. The latter is due to the missing artificial interface compression, given in Equation (30), resulting in a substantial numerical loss of the conserved quantity α.

We note at this point that there are formulations of the FVM that capture shocks much more accurately whilst controlling oscillations. Due to the maturity of this method, the field of constructing weighted essentially nonoscillatory (WENO) schemes is quite advanced and, also, in this case, will yield better approximations. Such schemes are also applicable to considerably more complex systems of equations [97]. However, this argument also applies to the DGM since it is, as has been shown, an extension of the FVM. To make the comparison fair and to be able to rely on existing tools, we chose to compare both schemes with the same, relatively simple reconstruction technique. For both cases, the use of WENO schemes would increase the computational load considerably, as these need to reconstruct polynomial approximations for each flux using relatively wide local stencils—similar to a high-order finite difference scheme [36]. The choice of higher-order reconstruction techniques, however, should not affect the qualitative difference in approximation properties and performance. For example, Zhou et al. reported very similar findings for higher-order WENO schemes for the advection equation in two dimensions [9].

In addition to the previous comparison regarding accuracy, we also benchmark the capabilities for parallel computing using the three machines given in Table 4. To properly scale up this problem, we aim to keep the number of degrees of freedom per core constant for the three environments, that is, in this case around 18.400 DoFs/core. This is in the vicinity of the 25.000 DoFs/core regime, where beyond that point, considerable drop-offs in performance are to be expected [98]. According to the theory, the DGM should perform with a noticeably higher efficiency due to less communication overhead between processes. This is due to the reduced number of cells for the same number of DoFs; as such, the DGM has a much higher ratio of DoFs that lie inside the cell instead of at the boundary. As a consequence, there are fewer DoFs in relative terms that require the evaluation of a

numerical flux and, thus, not as much overhead due to MPI efforts. This relationship is visualized for reference in Figure 10.

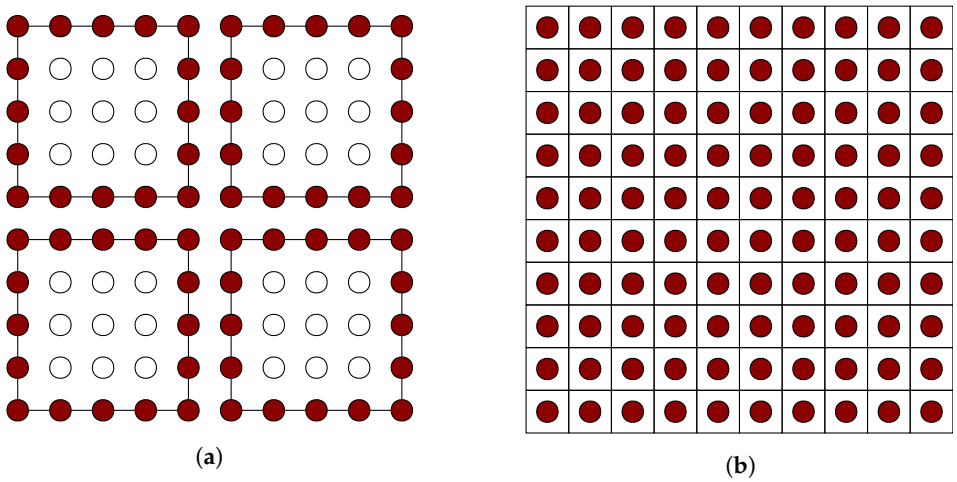
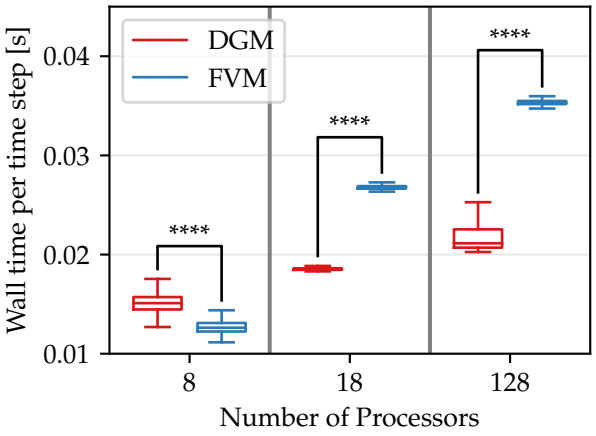


Figure 10. Comparison of shared DoFs (colored dots) between cells for a fourth-order DG method with an FV method with an equal number of total DoFs (100). As the FVM is restricted to one DoF per cell, there are no interior DoFs, increasing the necessary MPI effort. The DGM, on the other hand, has an increasing number of interior DoFs that, for a collocation method, do not contribute to the numerical flux. For the present fourth-order scheme, the ratio of interior to total DoFs is 36%.

The benchmarking results for all three hardware configurations are given in Figure 11. We report the total run times as well as the solution time for one singular time step. This is since, as explained, the DG model has a considerably smaller admissible time step. Thus, comparing overall run times is not suitable to empirically validate the above claims regarding parallel efficiency. The overall computation time is, nonetheless, an important factor in terms of practicality since this is the main quantity one is interested in when performing a simulation.



	Wall time [s]	
Cores	**FVM**	**DGM**
8	45.01	77.76
18	114.77	136.92
128	409.12	434.22

Figure 11. Comparison of run times for the DGM and FVM advection benchmark case, run on the three configurations given in Table 4. *Left:* Weak scaling of both models, measured as wall time per time step solve. The number of DoFs/core was fixed at 18,400. For each sample population, $n \approx 3000$. Horizontal lines: median, boxes: IQR, whiskers: quartiles ±1.5 IQR, ****: $p < 0.0001$. *Right:* Total wall times for the solution over all time steps.

It becomes evident from the reported wall times that the FV model only has an advantage on the small desktop machine concerning solution time. As predicted, scaling up the problem size and number of workers will yield an increasing advantage for the higher-order DGM. The differences in computation time which were found to be empirically

significant grow with increasing problem size, thus supporting the claim that DG methods are more favorable for highly parallel architectures.

A similar trend is visible from the table reporting overall runtimes. The time step restrictions, as discussed, weaken the computational advantage gained in the solution of the semidiscrete system. However, the difference in run time still decreases noticeably with increasing problem and hardware size. For the largest machine with 128 cores, solution times are almost comparable, whereas in the case of the smallest machine, the FV model computed a solution about 42% faster.

5.3. Laser Powder Bed Fusion

As a last example, we will investigate a real multiphysics problem to demonstrate the proposed method for more complex settings. We outline the results from the problem analysis summarized in Figure 5 and indicate how one may implement the resulting numerical scheme.

The problem of interest is the fluid flow and temperature field that is present during the laser-based heating of metallic powder. Combined with a moving heat source that travels along a set path and layer-wise re-application of fresh powder, this technique results in the additive manufacturing process of powder bed fusion of metallic materials using a laser beam (PBF-LB/M).

Having detailed knowledge about the fluid flow and temperature fields is crucial to obtain dense, that is, pore-free parts with a favorable microstructure. The former is governed by the morphology of the solidified melt pool. The latter is heavily influenced by the spatial and temporal temperature gradients.

This specific problem has been tackled many times in the literature using various numerical methods. An encompassing review of the technology as well as recent simulation approaches can be found in [99].

5.3.1. Physics and Governing Equations

Due to the presence of solid, liquid, and gaseous phases, resolving the thermo fluid dynamics of the process involves a multitude of physics. Most prominently, research has shown that the accurate representation of surface tension forces due to temperature gradient plays a key role in obtaining realistic results regarding the morphology of the solidified tracks, temperature gradients, and presence of pores—all of which have shown to be important indicators for process quality [100].

The multiphase problem that arises from this application is discretized using the volume of fluid method. The equations for this model were introduced in Section 5.2. In total, we have one solid metallic, one liquid metallic, and two gaseous phases for vaporized metal and the shielding gas, yielding four phase fractions that need to be tracked. Following the previous discussion in Section 5.2, one may omit to model one of these phases using a conservation law as one can calculate it using the compatibility condition in Equation (27) as well.

With the phase fractions in place, one then obtains the mixed material constants at a given degree of freedom by a rule of mixture law. For thermal conductivity κ, for instance, the phase-averaged value at DoF i is

$$\kappa_{\text{VOF}} = \sum_{k=0}^{N_{\text{Phases}}} \kappa_k \alpha_k. \tag{35}$$

We continue by enumerating the governing equations of this problem. For the melting of metallic materials, one finds that the flow field can be described by the incompressible Navier–Stokes equations in the low Reynolds number regime, plus some source terms that account for the additional physics:

$$\partial_t(\rho \underline{u}) + \nabla \cdot \left[(p - p_{\text{recoil}})\underline{\underline{I}}\right] - \rho \underline{k} - \eta \Delta \underline{u} + \nabla \cdot \underline{\underline{T}}_{\text{capillary}} = 0, \tag{36}$$

with the additional condition for incompressibility

$$\nabla \cdot \underline{u} = 0. \tag{37}$$

Here, $\underline{k}$ is the vector of gravity and other volumetric forces.

One important driving force for melt flow is the temperature-dependent surface tension force, also called the Marangoni force. We can summarize all capillary forces normal and tangential to the interface using the so-called capillary stress tensor

$$\underline{\underline{T}}_{\text{capillary}} = -\sigma(\underline{\underline{I}} - \underline{n} \otimes \underline{n}), \tag{38}$$

where σ denotes the temperature-dependent coefficient of surface tension, I is the identity tensor, and n is the unit normal vector to the capillary surface. The divergence of this quantity represents the capillary force acting on interfaces [101].

This, in turn, needs to be computed using the phase fractions and the surface gradient operator ∇_s

$$\underline{n} = \frac{\nabla_s \alpha}{|\nabla_s \alpha|}, \tag{39}$$

$$\nabla_s(\alpha_1, \alpha_2) = \alpha_1 \nabla \alpha_2 - \alpha_2 \nabla \alpha_1. \tag{40}$$

For the gaseous phase of vaporized material, we additionally consider the recoil pressure

$$p_{\text{recoil}}(T) = 0.53 p_0 \exp \frac{L_v}{R}\left(\frac{1}{T_v} - \frac{1}{T}\right). \tag{41}$$

The only conserved quantity left is the total energy of the system. The corresponding balance equation that governs the evolution of temperature reads

$$\partial_t(\rho c_p T) + \underline{u} \cdot \nabla(\rho c_p T) + \sum_i \rho L \partial_t \alpha_i - \nabla \cdot (\kappa \nabla T) + Q_{\text{Laser}} - Q_{\text{Vap}} - Q_{\text{Radiation}} = 0, \tag{42}$$

with the following (nonlinear) source terms for the laser heat input, vaporization loss, and radiation loss, respectively

$$Q_{\text{Laser}}(x, y) = \frac{2P}{\pi r_0^2} \exp\left(-\frac{2(x^2 + y^2)}{r_0^2}\right), \tag{43}$$

$$Q_{\text{Vap}}(T) = 0.82 \frac{p_{\text{recoil}}}{\sqrt{2\pi MRT}}, \tag{44}$$

$$Q_{\text{Radiation}}(T) = \sigma \epsilon (T^4 - T^{\text{amb}}). \tag{45}$$

The laser that provides Q_{Laser} travels along a straight line throughout the domain with constant velocity and laser power P with beam radius r_0. Thus, this source term represents a transient heat source where its position can be easily interpolated given the current time. Further details on the physical models can, for example, be found in [97].

5.3.2. Computational Domain

We solve the given problem on a grid that is overall three-dimensional and hexahedral in shape. The bounding box has dimensions 2.0 mm in length, 0.3 mm in width, and 0.5 mm in height.

In this problem setting, steep temperature gradients are to be expected due to the concentrated heat input. As a result, it makes sense to employ a finer grid at locations where the melt pool dynamics take place. One possible way to account for that is to use a discretization technique that can incorporate adaptive grids.

However, we would instead like to make use of our previous knowledge and only use a finer grid around the vicinity of the laser path—where the actual melting takes place

and a finer grid is needed to appropriately resolve the flow field. As such, there is no need to use algorithms that flag candidate cells and employ remeshing at every time step. The rest of the domain can be meshed using coarser cells, as they are primarily present in the simulation to properly account for heat conduction to surrounding solidified material.

A slice of the resulting discretization is depicted in Figure 12. The actual mesh is box-shaped, and the visible regions are colored to illustrate the different phases present as well as the varying cell sizes. One can observe that the hexahedral cells become coarser with increasing depth of the powder bed.

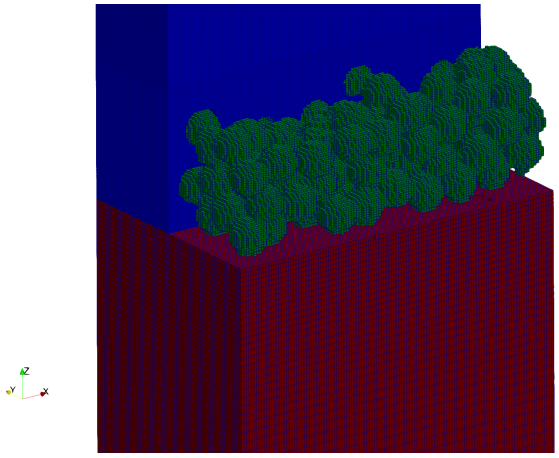


Figure 12. Slice of the computational domain for the laser powder bed fusion test case. The grid consists of hexahedral elements with varying sizing. Colors denote different phases in the initial condition. This figure is adapted from [97].

5.3.3. Computing Resources

We start the selection process for suitable numerical schemes as in the previous sections by investigating the computational hardware, as stated within process P1 in Figure 5. For this problem, we use the same hardware given in Table 4. That is, we aim to solve this problem on the 18-core CPU machine. As given in Section 4.5, such a system of desktop scale does not, per se, necessitate the use of massively parallel hardware, as the amount of processes falls below the recommended amount of 50 to 100 workers.

5.3.4. Problem Scale

Using the system of PDEs as input, we can proceed with analyzing the problem length scales according to process P2 in Figure 5. Given the coefficients and material properties of the PDE system, we note that all physics are expected to operate on a length scale of a few micrometers. We can, thus, conclude that this particular problem does not exhibit multiscale properties that would necessitate adaptivity. Therefore, in combination with the given hardware, we can negate the first decision point D1 in Figure 5 and proceed with the field-wise analysis of the PDE system.

5.3.5. Classification

We begin with the field-wise classification of the PDE system, that is, decisions D2 to D5, along with the corresponding processes as follows: We consider all governing equations of the system (Equations (38)–(45)) individually for each field quantity. That is, we extract the individual equations that one needs to solve. Afterward, we follow the classification methods described by P3 and P4, whereas the latter is only relevant for hyperbolic systems, as illustrated in Figure 5.

We first examine the phase fraction fields. As all variables are governed by the same set of equations, and, thus, the resulting classification applies to every field, we abbreviate

the discussion by only classifying one phase fraction α. The relevant governing equation then is exactly the passive advection equation that was introduced in Section 5.2.

$$\partial_t \alpha + \underline{u} \cdot \nabla \alpha = 0. \tag{46}$$

It then immediately follows from the previous discussion that the phases are governed by first-order, linear, hyperbolic systems. We consequently arrive at the discontinuous Galerkin method for these variables.

Next, we will deal with the velocity vector $\underline{u}$. Extracting all components from Equation (36) that contain differentials of $\underline{u}$, we have

$$\rho \partial_t \underline{u} - \eta \Delta \underline{u} = 0 \tag{47}$$

This system of equations, for each component, possesses the exact structure of the heat diffusion equation, which is known to be a model parabolic equation. Following the more rigorous approach given in Section 4.2, we would obtain for the coefficient matrix a_{ij}

$$a_{ij}^{\underline{u}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\eta & 0 \\ 0 & 0 & -\eta \end{bmatrix}. \tag{48}$$

The first zero-diagonal element is due to the time component only appearing in a first, but not in any second, derivative. We consequently arrive at the same conclusion, being that the governing equation for velocity in this case is parabolic. Thus, we follow the left branch after D2 and continue by classifying the regularity of the domain. The question of whether the grid is strictly regular and Cartesian has already been answered in the previous section; therefore, we arrive at the continuous Galerkin method for the velocity field.

For the pressure p, we find that this quantity only appears within the momentum balance in Equation (36) within the expression $\nabla \cdot p\underline{\underline{I}}$, which is equal to the gradient ∇p. That is, we have $\partial_x p$ as a source term in the components in this equation; thus, the evolution of p is not, per se, given by a separate governing equation. We therefore fall back to the qualitative classification that was described in Section 4.2 and choose an appropriate class based on the continuity requirements of the pressure field. The present problem consists of multiple phases; therefore, we may expect steep jumps in pressure at interfaces. We consequently assume the pressure field to be discontinuous at some points during the simulation and, thus, follow the process in Figure 5 along the path of hyperbolic equations. As there are no additional source terms or nonlinearities in p, we classify this constraint as linear in D4 and therefore also arrive at the discontinuous Galerkin method.

For the remaining temperature field T, we again gather the terms containing differential operators on T from the governing equation, Equation (42)

$$\rho c_p \partial_t T + \rho c_p \underline{u} \cdot \nabla T - \kappa \Delta T = \mathrm{RHS}(T). \tag{49}$$

The remaining terms in Equation (42) that have been gathered in $\mathrm{RHS}(T)$ are either constant, or depend on T itself, but not in its derivatives. Therefore, we may omit them for the sake of classification. As such, we have a PDE of second order and proceed analogously to the classification of the velocity $\underline{u}$. The coefficient matrix a_{ij} is then

$$a_{ij}^{T} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\kappa & 0 \\ 0 & 0 & -\kappa \end{bmatrix}. \tag{50}$$

Similarly to the previous discussion, this resembles a parabolic system and we thus recommend the discretization of the temperature field using the continuous Galerkin method, as the question of domain regularity (D3) has already been addressed and is the same for all fields. If the coefficient of thermal diffusion κ would be zero in this case, the

governing equation would then collapse into a hyperbolic PDE and another classification would apply. We can, however, state that the amount of heat conduction present in the model cannot be neglected, and, thus, we have that $\kappa > 0$ everywhere in the domain.

5.4. Resulting Discretization

With all independent variables of the model addressed in the previous section, we briefly summarize the chosen combination of numerical schemes in Table 5.

Table 5. Resulting schemes of the proposed decision method for the laser powder bed fusion example problem. Columns D1 to D4 refer to the corresponding decision points in Figure 5.

Variable	D1	D2	D3	D4	Result
α_{solid}	no	hyperbolic	n.a.	linear	DGM
α_{liquid}	no	hyperbolic	n.a.	linear	DGM
α_{gas}	no	hyperbolic	n.a.	linear	DGM
$\underline{u}$	no	parabolic	irregular	n.a.	CGM
p	no	hyperbolic	n.a.	linear	DGM
T	no	parabolic	irregular	n.a.	CGM

One may proceed to implement this mixed discretization using any modern finite element code. In the first two examples, we showed that both creating a custom implementation using modern programming languages and implementing a model using popular finite element libraries are viable options to create a corresponding simulation.

In this case, we obtain a coupled, nonlinear problem. This means that instead of solving a discrete linear system of equations, one must solve a global, nonlinear root finding problem of the form $F(u; v) = 0$ in every time step by appropriate means, such as Picard iteration or Newton's method. Such functionality, however, is readily implemented in finite element libraries such as Firedrake [3], which was used in the previous section.

As a final remark, we note that similar models for resolving the melt pool dynamics in laser powder bed fusion have been developed using mixed formulations of the finite element method. For example, Meier et al. developed a custom, high-performance finite element simulation code called `MeltPoolDG` based on the discontinuous Galerkin method as well as a mixed FEM to resolve the melt pool on the powder bed scale [102,103]. Other recent developments include a finite element model on multiple grids to capture the governing physics [104], and a space-time FEM code that is based on a Petrov Galerkin approximation [105]. Furthermore, there also exist implementations of such melt pool models using commercially available software. For instance, the popular COMSOL software package (version 5.5) which implements many variants of the FEM can resolve the relevant thermo fluid dynamics during laser melting [106,107].

6. Conclusions and Future Work

We presented a general framework for determining combinations of interoperable, grid-based approximation schemes that are individually suited for the solution of multi-physics problems. The key findings of this work can be summarized as follows:

- The discontinuous Bubnov Galerkin method can serve as a general mathematical baseline for the finite volume method, finite difference method, and continuous Galerkin method. These form the majority of schemes used in practice, which is why we restrict the scope of this work to this specific subset. We have shown that one can recover all these schemes by introducing certain restrictions to the DG method, which can either be on the algebraic or algorithmic level.
- By using those simplified schemes where appropriate, one can gain stability and performance in numerical simulations that scale with problem size and hardware. In particular, one may even avoid the assembly of a global linear system by leveraging domain restrictions in the case of the finite difference method. If a PDE exhibits strong

nonlinearity, choosing a purely reconstruction-based approach via the finite volume method is then beneficial and assembly of the weak form can be avoided as well.

- This common framework based on the DG method enables interoperability of the mentioned schemes. As such, one can combine them when solving multiphysics problems and thus assemble efficient mixed discretizations. Therefore, manual coupling of different numerical solvers is not needed in theory. Instead, one may implement them in a monolithic way and, thus, avoid costly memory transfer operations.
- One can systematically derive an efficient combination of numerical schemes, given some inputs about the problem, that is, hardware requirements, the mathematical formulation of the problem, and the domain geometry. The method to identify these schemes is based on the field and delivers one-to-one recommendations on which method to use.
- Applying the developed method to the two model problems in this work, the Allen–Cahn equation in 1D and 2D and the advection equation in 2D, yields methods that notably outperform their respective alternatives. In the former case, this difference in computation time exceeds one order of magnitude. In the latter case, we have shown that, as predicted by the theory, the computational advantages scale with problem size and the necessary degrees of freedom for some fixed accuracy differ considerably. We have also shown that for a real multiphysics problem, even for coupled systems of PDEs, one can derive an efficient combination of schemes in a reproducible and for application experts accessible way. This highlights the practicality and usefulness of the established framework for end users. Especially, researchers who are familiar with physical modeling, but not to the same degree with state-of-the-art high-performance numerical methods, are expected to profit from this approach.

Some future work might be attributed to further breaking down the choices of numerical schemes based on a more granular problem classification. The classes of PDE we introduced in this work are rather coarse, albeit the most widely used. Furthermore, this work limits the applicability of this classification to PDEs of up to second order. This naturally formally excludes higher-order PDEs, such as the Cahn–Hilliard equation [108] or the Korteweg de-Vries equation [109] which are also of high practical relevance within their respective fields of physics.

One may also include mixed finite element discretizations into the method. Within the scope of this work, we only included scalar- and vector-valued Lagrange elements. Including mixed finite elements might, however, make the selection process ambiguous; thus, special care must be attributed to keep the simplicity of the method.

Due to the clear recommendation and insight that this method can provide on a per-problem basis, we expect this to have a substantial impact within various scientific communities that apply simulation techniques. For instance, the sharp phase field method, which can describe frictionless interface motion, is thus far only well researched using spectral and finite difference methods [61,62,77]. These, however, limit the applicability to relatively simple geometries as outlined above. It is thus desirable to obtain a formulation that allows for more flexibility, such as the finite element method [110]. In principle, the sharp phase field method is not straightforward to generalize, since the governing equations have to be modified on the discrete level [61]. Our results outlined in Sections 3.3 and 5.1 may, however, provide a natural guideline to derive such a generalization using finite elements from the finite difference formulation given in [62] by computing a reference stencil and applying the finite element specific assembly process afterward. We thus expect that the theoretical framework we described may be of assistance in establishing new simulation models and also provide additional insight into existing implementations within the scientific community.

Author Contributions: Conceptualization, P.Z. and M.F.; method, P.Z.; software, P.Z.; validation, P.Z. and M.F.; formal analysis, P.Z.; investigation, P.Z. and M.F.; resources, J.S.; writing—original draft

preparation, P.Z.; writing—review and editing, M.F. and J.S.; visualization, P.Z. and M.F.; supervision, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All data supporting this publication, including the manuscript source, are available under https://github.com/pzimbrod/multiphysics-pde-methods (accessed on 18 April 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CFL	Courant–Friedrichs–Lewy
CG(M)	Continuous Galerkin (method)
DG(M)	Discontinuous Galerkin (method)
DoF	Degree of freedom
FD(M)	Finite difference (method)
FV(M)	Finite volume (method)
MPI	Message passing interface
PDE	Partial differential equation
VoF	Volume of fluid

Appendix A. Implementation Details

Appendix A.1. Allen–Cahn Equation

We solve both systems of ordinary differential equations (ODEs) using the DifferentialEquations.jl library in Julia. Details about model parametrization are given in Table A1.

Table A1. Parameters used for both 1D and 2D models of the Allen–Cahn equation.

Variable	1D Model	2D Model	Description
Γ	1.0	50.0	Interface energy
ξ	1.5	4.0	Interface width
M	1.0	1.0	Kinetic mobility
μ_0	0.1	0.0	Bulk energy density differential
x_0	20.0	n.a.	Initial interface position
h	1.0	1.0	Grid size

We benchmark both models using the `BenchmarkTools.jl` library. The physical size of both models was measured using the Julia Base function `summarysize`. For timing purposes, each run was repeated 100 times to create reliable empirical data. Both models were run on the same machine in consecutive runs.

Appendix A.2. Advection Equation

Appendix A.2.1. Derivation of the Weak Form

Equation (32) can be obtained by the strong form of the PDE as described in the article, by applying partial integration after multiplying with a test function and integrating over the domain Ω. To arrive at a formulation that can be implemented, there are a few additional steps which are described below.

In most DG codes, the weak form given by Equation (32), more specifically the surface integral, cannot be entered directly but needs to be reformulated using jump and average operators. This is also performed in the advection equation example described in Section 5.2. Such a modified weak form can be derived as follows.

The most straightforward consists of summarizing the convective term of the PDE as the divergence of some flux $\nabla \cdot \underline{f}(\alpha)$. In this case, the flux is simply $\underline{f}(\alpha) = \underline{u}\,\alpha$.

Then, the surface integral of Equation (32) can be formulated as

$$\int_\Gamma v\,\alpha(\underline{u}\cdot\underline{n})\,dS = \int_\Gamma v\underline{f}(\alpha)\cdot\underline{n}\,dS. \tag{A1}$$

We now write down the terms of the surface integral of Equation (32) that contribute to an arbitrary internal facet, that is, we always encounter two integrals from the cells that the facet is owned by. We distinguish between the two cells by introducing the notation "+" and "−"

$$\int_\Gamma v_h^+\alpha_h^+(\underline{u}^+\cdot\underline{n}^+)\,dS + \int_\Gamma v_h^-\alpha_h^-(\underline{u}^-\cdot\underline{n}^-)\,dS, \tag{A2}$$

where we encounter the yet-to-be-determined values for α^+ and α^-. By introducing the common, yet unknown numerical flux $\underline{f}(\alpha)^*$ and utilizing the fact that the facet normal vectors $\underline{n}$ point opposite to each other, Equation (A2) becomes

$$\int_\Gamma \left(\underline{f}(\alpha)^*\cdot\underline{n}\right)(v^+ - v^-)\,dS = \int_\Gamma \left(\underline{f}(\alpha)^*\cdot\underline{n}\right)[\![v]\!]\,dS. \tag{A3}$$

The only issue that is left to deal with is now the choice of an appropriate expression for the numerical flux $\underline{f}(\alpha)^*\cdot\underline{n}$.

Both DG and FV implementations of the advection equation use the Lax–Friedrichs flux function

$$f(\alpha)^* = \{\alpha(\underline{u}\cdot\underline{n}^+)\} + \frac{1}{2}C[\![\alpha]\!], \tag{A4}$$

where $\{\cdot\}$ and $[\![\cdot]\!]$ denote the average and jump operators as defined above. In this case, C is the maximum velocity in the domain and takes the role of the maximum signal velocity, which must be computed separately for more complex problems. It is defined as the Jacobian of the PDE flux $\partial f(\alpha)/\partial\alpha$. Since we have that $f(\alpha) = \underline{u}\alpha$, the Jacobian simply becomes the prescribed advection velocity $\underline{u}$.

Appendix A.2.2. Simulation Parameters

We solve the resulting semidiscrete ODE systems using a three-stage implicit Runge–Kutta formulation using a matrix-free solver.

We time the entire simulation runs from the command line and execute each run six times. Wall times per time step were recorded during the run using the PETSc distributed interface and logged to a text file.

References

1. Bangerth, W.; Hartmann, R.; Kanschat, G. Deal.II—A General-Purpose Object-Oriented Finite Element Library. *Acm Trans. Math. Softw.* **2007**, *33*, 24-es. [CrossRef]
2. Alnaes, M.S.; Blechta, J.; Hake, J.; Johansson, A.; Kehlet, B.; Logg, A.; Richardson, C.; Ring, J.; Rognes, M.E.; Wells, G.N. The FEniCS Project Version 1.5. *Arch. Numer. Softw.* **2015**, *3* . [CrossRef]
3. Ham, D.A.; Kelly, P.H.J.; Mitchell, L.; Cotter, C.J.; Kirby, R.C.; Sagiyama, K.; Bouziani, N.; Vorderwuelbecke, S.; Gregory, T.J.; Betteridge, J.; et al. *Firedrake User Manual*, 1st ed.; Imperial College: London, UK, 2023. [CrossRef]
4. Anderson, R.; Andrej, J.; Barker, A.; Bramwell, J.; Camier, J.S.; Cerveny, J.; Dobrev, V.; Dudouit, Y.; Fisher, A.; Kolev, T.; et al. MFEM: A Modular Finite Element Methods Library. *Comput. Math. Appl.* **2021**, *81*, 42–74. [CrossRef]
5. Scroggs, M.W. DefElement: An Encyclopedia of Finite Element Definitions. In Proceedings of the FEniCS 2023, Cagliari, Italy, 14–16 June 2023. [CrossRef]
6. Brezzi, F.; Bathe, K.J. A Discourse on the Stability Conditions for Mixed Finite Element Formulations. *Comput. Methods Appl. Mech. Eng.* **1990**, *82*, 27–57. [CrossRef]
7. John, V.; Linke, A.; Merdon, C.; Neilan, M.; Rebholz, L.G. On the Divergence Constraint in Mixed Finite Element Methods for Incompressible Flows. *SiAM Rev.* **2017**, *59*, 492–544. [CrossRef]
8. Shademan, M.; Barron, R.M.; Balachandar, R. Evaluation of OpenFOAM in Academic Research and Industrial Applications. In Proceedings of the 21st Conference of the CFD Society of Canada, Vancouver, BC, Canada, 28–30 May 2013.
9. Zhou, T.; Li, Y.; Shu, C.W. Numerical Comparison of WENO Finite Volume and Runge–Kutta Discontinuous Galerkin Methods. *J. Sci. Comput.* **2001**, *16*, 145–171. [CrossRef]

10. Baranger, J.; Maitre, J.F.; Oudin, F. Connection between Finite Volume and Mixed Finite Element Methods. *ESAIM Math. Model. Numer. Anal.* **1996**, *30*, 445–465. [CrossRef]
11. Ye, X. On the Relationship between Finite Volume and Finite Element Methods Applied to the Stokes Equations. *Numer. Methods Partial. Differ. Equ.* **2001**, *17*, 440–453. [CrossRef]
12. Idelsohn, S.R.; Oñate, E. Finite Volumes and Finite Elements: Two 'Good Friends'. *Int. J. Numer. Methods Eng.* **1994**, *37*, 3323–3341. [CrossRef]
13. Thomée, V. The Finite Difference versus the Finite Element Method for the Solution of Boundary Value Problems. *Bull. Aust. Math. Soc.* **1984**, *29*, 267–288. [CrossRef]
14. Key, S.W.; Krieg, R.D. Comparison of Finite-Element and Finite-Difference Methods. In *Numerical and Computer Methods in Structural Mechanics*; Elsevier: Amsterdam, The Netherlands, 1973; pp. 337–352. [CrossRef]
15. Shu, C.W. High-Order Finite Difference and Finite Volume WENO Schemes and Discontinuous Galerkin Methods for CFD. *Int. J. Comput. Fluid Dyn.* **2003**, *17*, 107–118. [CrossRef]
16. Dumbser, M.; Balsara, D.S.; Toro, E.F.; Munz, C.D. A Unified Framework for the Construction of One-Step Finite Volume and Discontinuous Galerkin Schemes on Unstructured Meshes. *J. Comput. Phys.* **2008**, *227*, 8209–8253. [CrossRef]
17. Lin, G.; Liu, J.; Sadre-Marandi, F. A Comparative Study on the Weak Galerkin, Discontinuous Galerkin, and Mixed Finite Element Methods. *J. Comput. Appl. Math.* **2015**, *273*, 346–362. [CrossRef]
18. Yurun, F. A Comparative Study of the Discontinuous Galerkin and Continuous SUPG Finite Element Methods for Computation of Viscoelastic Flows. *Comput. Methods Appl. Mech. Eng.* **1997**, *141*, 47–65. [CrossRef]
19. Bui-Thanh, T. From Godunov to a Unified Hybridized Discontinuous Galerkin Framework for Partial Differential Equations. *J. Comput. Phys.* **2015**, *295*, 114–146. [CrossRef]
20. Yakovlev, S.; Moxey, D.; Kirby, R.M.; Sherwin, S.J. To CG or to HDG: A Comparative Study in 3D. *J. Sci. Comput.* **2016**, *67*, 192–220. [CrossRef]
21. Kronbichler, M.; Wall, W.A. A Performance Comparison of Continuous and Discontinuous Galerkin Methods with Fast Multigrid Solvers. *SIAM J. Sci. Comput.* **2018**, *40*, A3423–A3448. [CrossRef]
22. Kirby, R.M.; Sherwin, S.J.; Cockburn, B. To CG or to HDG: A Comparative Study. *J. Sci. Comput.* **2012**, *51*, 183–212. [CrossRef]
23. Gaburro, E. A Unified Framework for the Solution of Hyperbolic PDE Systems Using High Order Direct Arbitrary-Lagrangian–Eulerian Schemes on Moving Unstructured Meshes with Topology Change. *Arch. Comput. Methods Eng.* **2021**, *28*, 1249–1321. [CrossRef]
24. Yang, F.W.; Goodyer, C.E.; Hubbard, M.E.; Jimack, P.K. An Optimally Efficient Technique for the Solution of Systems of Nonlinear Parabolic Partial Differential Equations. *Adv. Eng. Softw.* **2017**, *103*, 65–84. [CrossRef]
25. Reed, W.H.; Hill, T.R. *Triangular Mesh Methods for the Neutron Transport Equation*; Los Alamos Scientific Lab.: Santa Fe, NM, USA, 1973.
26. Epshteyn, Y.; Riviere, B. Estimation of penalty parameters for symmetric interior penalty Galerkin methods *J. Comput. Appl. Math.* **2007**, *206*, 843–872. [CrossRef]
27. Warburton, T.; Karniadakis, G. A Discontinuous Galerkin Method for the Viscous MHD Equations. *J. Comput. Phys.* **1999**, *152*, 608–641. [CrossRef]
28. Cockburn, B.; Shu, C.W. The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems. *SIAM J. Numer. Anal.* **1998**, *35*, 2440–2463. [CrossRef]
29. Arnold, D.N.; Brezzi, F.; Cockburn, B.; Marini, L.D. Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems. *SIAM J. Numer. Anal.* **2002**, *39*, 1749–1779. [CrossRef]
30. Liu, W.K.; Li, S.; Park, H.S. Eighty Years of the Finite Element Method: Birth, Evolution, and Future. *Arch. Comput. Methods Eng.* **2022**, *29*, 4431–4453. [CrossRef]
31. Ciarlet, P.G. *The Finite Element Method for Elliptic Problems*; Number 40 in Classics in Applied Mathematics; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2002.
32. Brooks, A.N.; Hughes, T.J.R. Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations. *Comput. Methods Appl. Mech. Eng.* **1982**, *32*, 199–259. [CrossRef]
33. Eymard, R.; Gallou'et, T.; Herbin, R. Finite Volume Methods. *Handb. Numer. Anal.* **2000**, *7*, 713–1018. [CrossRef]
34. Moukalled, F.; Mangani, L.; Darwish, M. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*; Fluid Mechanics and Its Applications; Springer International Publishing: Cham, Switzerland, 2016; Volume 113. [CrossRef]
35. Hesthaven, J.S.; Warburton, T. *Nodal Discontinuous Galerkin Methods*; Texts in Applied Mathematics; Springer: New York, NY, USA, 2008; Volume 54. [CrossRef]
36. Liu, Y.; Zhang, Y.T. A Robust Reconstruction for Unstructured WENO Schemes. *J. Sci. Comput.* **2013**, *54*, 603–621. [CrossRef]
37. Fornberg, B. Generation of Finite Difference Formulas on Arbitrarily Spaced Grids. *Math. Comput.* **1988**, *51*, 699–706. [CrossRef]
38. Visbal, M.R.; Gaitonde, D.V. On the Use of Higher-Order Finite-Difference Schemes on Curvilinear and Deforming Meshes. *J. Comput. Phys.* **2002**, *181*, 155–185. [CrossRef]
39. Zhang, W.; Zhang, Z.; Chen, X. Three-Dimensional Elastic Wave Numerical Modelling in the Presence of Surface Topography by a Collocated-Grid Finite-Difference Method on Curvilinear Grids. *Geophys. J. Int.* **2012**, *190*, 358–378. [CrossRef]
40. Perrone, N.; Kao, R. A General Finite Difference Method for Arbitrary Meshes. *Comput. Struct.* **1975**, *5*, 45–57. [CrossRef]

41. Sukumar, N. Voronoi Cell Finite Difference Method for the Diffusion Operator on Arbitrary Unstructured Grids. *Int. J. Numer. Methods Eng.* **2003**, *57*, 1–34. [CrossRef]
42. Ern, A.; Guermond, J.L. *Theory and Practice of Finite Elements*; Applied Mathematical Sciences; Springer: New York, NY, USA, 2004. [CrossRef]
43. Cockburn, B.; Li, F.; Shu, C.W. Locally Divergence-Free Discontinuous Galerkin Methods for the Maxwell Equations. *J. Comput. Phys.* **2004**, *194*, 588–610. [CrossRef]
44. Hughes, T.J.R.; Masud, A.; Wan, J. A Stabilized Mixed Discontinuous Galerkin Method for Darcy Flow. *Comput. Methods Appl. Mech. Eng.* **2006**, *195*, 3347–3381. [CrossRef]
45. Arnold, D.N.; Logg, A. Periodic Table of the Finite Elements. *SIAM News* **2014**, *47*, 212.
46. Cockburn, B.; Fu, G. A Systematic Construction of Finite Element Commuting Exact Sequences. *SIAM J. Numer. Anal.* **2017**, *55*, 1650–1688. [CrossRef]
47. Bitsadze, A.V. *Some Classes of Partial Differential Equations*; CRC Press: Boca Raton, FL, USA, 1988; Volume 4,
48. Pinchover, Y.; Rubinstein, J. *An Introduction to Partial Differential Equations*; Cambridge University Press: Cambridge, UK, 2005.
49. Evans, L.C. *Partial Differential Equations*, 2nd ed.; American Mathematical Society: Providence, RI, USA, 2010.
50. Maltsev, V.; Yuan, D.; Jenkins, K.W.; Skote, M.; Tsoutsanis, P. Hybrid Discontinuous Galerkin-finite Volume Techniques for Compressible Flows on Unstructured Meshes. *J. Comput. Phys.* **2023**, *473*, 111755. [CrossRef]
51. Sonntag, M.; Munz, C.D. Shock Capturing for Discontinuous Galerkin Methods Using Finite Volume Subcells. In *Finite Volumes for Complex Applications VII-Elliptic, Parabolic and Hyperbolic Problems*; Fuhrmann, J., Ohlberger, M., Rohde, C., Eds.; Springer: Cham, Switzerland, 2014; pp. 945–953. [CrossRef]
52. McKee, S.A.; Wisniewski, R.W. Memory Wall. In *Encyclopedia of Parallel Computing*; Padua, D., Ed.; Springer: Boston, MA, USA, 2011; pp. 1110–1116. [CrossRef]
53. Arndt, D.; Fehn, N.; Kanschat, G.; Kormann, K.; Kronbichler, M.; Munch, P.; Wall, W.A.; Witte, J. ExaDG: High-Order Discontinuous Galerkin for the Exa-Scale. In *Software for Exascale Computing—SPPEXA 2016-2019*; Bungartz, H.J., Reiz, S., Uekermann, B., Neumann, P., Nagel, W.E., Eds.; Springer International Publishing: Cham, Switzerland, 2020; Volume 136, pp. 189–224. [CrossRef]
54. Marshall, J.; Adcroft, A.; Hill, C.; Perelman, L.; Heisey, C. A Finite-Volume, Incompressible Navier Stokes Model for Studies of the Ocean on Parallel Computers. *J. Geophys. Res. Ocean.* **1997**, *102*, 5753–5766. [CrossRef]
55. Fringer, O.; Gerritsen, M.; Street, R. An Unstructured-Grid, Finite-Volume, Nonhydrostatic, Parallel Coastal Ocean Simulator. *Ocean. Model.* **2006**, *14*, 139–173. [CrossRef]
56. Weinan, E.; Lu, J. Multiscale Modeling. *Scholarpedia* **2011**, *6*, 11527. [CrossRef]
57. Weinan, E. *Principles of Multiscale Modeling*; Cambridge University Press: Cambridge, UK; New York, NY, USA, 2011.
58. Zimbrod, P.; Schilp, J. Modelling of Microstructures during In-Situ Alloying in Additive Manufacturing for Efficient Material Qualification Processes. In Proceedings of the Simulation in Produktion und Logistik 2021: 19. ASIM-Fachtagung Simulation in Produktion und Logistik, Erlangen, Germany, 15–17 September 2021; pp. 177–188.
59. Landau, L. On the Theory of Phase Transitions. *Zh. Eksp. Teor. Fiz.* **1937**, *7*, 19–32. [CrossRef]
60. Ghanbari, P.G.; Mazza, E.; Hosseini, E. Adaptive Local-Global Multiscale Approach for Thermal Simulation of the Selective Laser Melting Process. *Addit. Manuf.* **2020**, *36*, 101518. [CrossRef]
61. Fleck, M.; Schleifer, F. Sharp Phase-Field Modeling of Isotropic Solidification with a Super Efficient Spatial Resolution. *Eng. Comput.* **2023**, *39*, 1699–1709. [CrossRef]
62. Fleck, M.; Schleifer, F.; Zimbrod, P. Frictionless Motion of Diffuse Interfaces by Sharp Phase-Field Modeling. *Crystals* **2022**, *12*, 1496. [CrossRef]
63. Proell, S.D.; Munch, P.; Wall, W.A.; Meier, C. A Highly Efficient Computational Framework for Fast Scan-Resolved Simulations of Metal Additive Manufacturing Processes on the Scale of Real Parts. *arXiv* **2023**, arXiv:2302.05164.
64. Olleak, A.; Dugast, F.; Bharadwaj, P.; Strayer, S.; Hinnebusch, S.; Narra, S.; To, A.C. 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.* **2022**, *3*, 100051. [CrossRef]
65. Olleak, A.; Xi, Z. Part-Scale Finite Element Modeling of the SLM Process with Layer-wise Adaptive Remeshing for Thermal History and Porosity Prediction. *J. Manuf. Sci. Eng.* **2020**, *142*, 121006. [CrossRef]
66. Olleak, A.; Xi, Z. Scan-Wise Adaptive Remeshing for Efficient LPBF Process Simulation: The Thermal Problem. *Manuf. Lett.* **2020**, *23*, 75–78. [CrossRef]
67. Šolín, P.; Červený, J.; Doležel, I. Arbitrary-Level Hanging Nodes and Automatic Adaptivity in the Hp-FEM. *Math. Comput. Simul.* **2008**, *77*, 117–132. [CrossRef]
68. Bangerth, W.; Kayser-Herold, O. Data Structures and Requirements for *Hp* Finite Element Software. *ACM Trans. Math. Softw.* **2009**, *36*, 1–31. [CrossRef]
69. Pilipenko, D.; Fleck, M.; Emmerich, H. On Numerical Aspects of Phase Field Fracture Modelling. *Eur. Phys. J. Plus* **2011**, *126*, 100. [CrossRef]
70. Fleck, M.; Pilipenko, D.; Spatschek, R.; Brener, E.A. Brittle Fracture in Viscoelastic Materials as a Pattern-Formation Process. *Phys. Rev. Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.* **2011**, *83*, 046213. [CrossRef] [PubMed]
71. Fleck, M.; Federmann, H.; Pogorelov, E. Phase-Field Modeling of Li-insertion Kinetics in Single LiFePO4-nano-particles for Rechargeable Li-ion Battery Application. *Comput. Mater. Sci.* **2018**, *153*, 288–296. [CrossRef]

72. Bezanson, J.; Edelman, A.; Karpinski, S.; Shah, V.B. Julia: A Fresh Approach to Numerical Computing. *SIAM Rev.* **2017**, *59*, 65–98. [CrossRef]
73. Rackauckas, C.; Nie, Q. Differentialequations.Jl–a Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *J. Open Res. Softw.* **2017**, *5*, 15. [CrossRef]
74. Rackauckas, C.; Nie, Q. Confederated Modular Differential Equation APIs for Accelerated Algorithm Development and Benchmarking. *Adv. Eng. Softw.* **2019**, *132*, 1–6. [CrossRef]
75. B´osch, A.; Müller-Krumbhaar, H.; Shochet, O. Phase-Field Models for Moving Boundary Problems: Controlling Metastability and Anisotropy. *Z. Phys. B* **1995**, *97*, 367–377. [CrossRef]
76. Karma, A.; Rappel, W.J. Quantitative Phase-Field Modeling of Dendritic Growth in Two and Three Dimensions. *Phys. Rev. Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.* **1998**, *57*, 4323–4349. [CrossRef]
77. Finel, A.; Le Bouar, Y.; Dabas, B.; Appolaire, B.; Yamada, Y.; Mohri, T. Sharp Phase Field Method. *Phys. Rev. Lett.* **2018**, *121*, 025501. [CrossRef] [PubMed]
78. Fleck, M.; Schleifer, F.; Holzinger, M.; Glatzel, U. Improving the Numerical Solution of the Phase-Field Equation by the Systematic Integration of Analytic Properties of the Phase-Field Profile Function. In Proceedings of the 8th GACM Colloquium, Kassel, Germany, 28–30 August 2019; pp. 445–450. [CrossRef]
79. Eiken, J. Numerical Solution of the Phase-Field Equation with Minimized Discretization Error. *IOP Conf. Ser. Mater. Sci. Eng.* **2012**, *33*, 012105. [CrossRef]
80. Wang, Q.; Zhang, X.; Zhang, Y.; Yi, Q. AUGEM: Automatically Generate High Performance Dense Linear Algebra Kernels on X86 CPUs. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 17–22 November 2013; pp. 1–12. [CrossRef]
81. Xianyi, Z.; Qian, W.; Yunquan, Z. Model-Driven Level 3 BLAS Performance Optimization on Loongson 3A Processor. In Proceedings of the 2012 IEEE 18th International Conference on Parallel and Distributed Systems, Singapore, 17–19 December 2012; pp. 684–691. [CrossRef]
82. Abbott, M.; Dilum, A.; N3N5; Schaub, S.; Elrod, C.; Lucibello, C.; Chen, J. Mcabbott/Tullio.Jl: V0.3.5. Zenodo. 2022. Available online: https://zenodo.org/records/10035615 (accessed on 18 February 2024).
83. Gräser, C.; Kornhuber, R.; Sack, U. Time Discretizations of Anisotropic Allen–Cahn Equations. *Ima J. Numer. Anal.* **2013**, *33*, 1226–1244. [CrossRef]
84. Glasner, K. Nonlinear Preconditioning for Diffuse Interfaces. *J. Comput. Phys.* **2001**. *174*, 695–711. [CrossRef]
85. LeVeque, R.J. *Numerical Methods for Conservation Laws*; Birkhäuser: Basel, Switzerland, 1992. [CrossRef]
86. Gopala, V.R.; van Wachem, B.G.M. Volume of Fluid Methods for Immiscible-Fluid and Free-Surface Flows. *Chem. Eng. J.* **2008**, *141*, 204–221. [CrossRef]
87. Okagaki, Y.; Yonomoto, T.; Ishigaki, M.; Hirose, Y. Numerical Study on an Interface Compression Method for the Volume of Fluid Approach. *Fluids* **2021**, *6*, 80. [CrossRef]
88. McRae, A.T.T.; Bercea, G.T.; Mitchell, L.; Ham, D.A.; Cotter, C.J. Automated Generation and Symbolic Manipulation of Tensor Product Finite Elements. *SIAM J. Sci. Comput.* **2016**, *38*, S25–S47. [CrossRef]
89. Homolya, M.; Ham, D.A. A Parallel Edge Orientation Algorithm for Quadrilateral Meshes. *SIAM J. Sci. Comput.* **2016**, *38*, S48–S61. [CrossRef]
90. Hendrickson, B.; Leland, R. A Multilevel Algorithm for Partitioning Graphs. In Proceedings of the Supercomputing '95: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing (CDROM), New York, NY, USA, 3–8 December 1995; p. 28. [CrossRef]
91. Chevalier, C.; Pellegrini, F. PT-SCOTCH: A Tool for Efficient Parallel Graph Ordering. *Parallel Comput.* **2008**, *34*, 318–331. [CrossRef]
92. Rathgeber, F.; Ham, D.A.; Mitchell, L.; Lange, M.; Luporini, F.; McRae, A.T.T.; Bercea, G.T.; Markall, G.R.; Kelly, P.H.J. Firedrake: Automating the Finite Element Method by Composing Abstractions. *ACM Trans. Math. Softw.* **2016**, *43*, 24:1–24:27. [CrossRef]
93. Dalcin, L.D.; Paz, R.R.; Kler, P.A.; Cosimo, A. Parallel Distributed Computing Using Python. *Adv. Water Resour.* **2011**, *34*, 1124–1139. [CrossRef]
94. Balay, S.; Gropp, W.D.; McInnes, L.C.; Smith, B.F. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In *Proceedings of the Modern Software Tools in Scientific Computing*; Arge, E., Bruaset, A.M., Langtangen, H.P., Eds.; Springer: Basel, Switzerland, 1997; pp. 163–202.
95. Balay, S.; Abhyankar, S.; Adams, M.F.; Benson, S.; Brown, J.; Brune, P.; Buschelman, K.; Constantinescu, E.; Dalcin, L.; Dener, A.; et al. *PETSc/TAO Users Manual. Technical Report ANL-21/39*; Revision 3.19; Argonne National Laboratory: Lemont, IL, USA, 2023.
96. Cockburn, B.; Shu, C.W. Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems. *J. Sci. Comput.* **2001**, *16*, 173–261. [CrossRef]
97. Zimbrod, P.; Schreter, M.; Schilp, J. Efficient Simulation of Complex Capillary Effects in Advanced Manufacturing Processes Using the Finite Volume Method. In Proceedings of the 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 6–18 November 2022; pp. 1–6. [CrossRef]
98. Badia, S.; Martín, A.F.; Neiva, E.; Verdugo, F. A Generic Finite Element Framework on Parallel Tree-Based Adaptive Meshes. *SIAM J. Sci. Comput.* **2020**, *42*, C436–C468. [CrossRef]

99. Chowdhury, S.; Yadaiah, N.; Prakash, C.; Ramakrishna, S.; Dixit, S.; Gulta, L.R.; Buddhi, D. Laser Powder Bed Fusion: A State-of-the-Art Review of the Technology, Materials, Properties & Defects, and Numerical Modelling. *J. Mater. Res. Technol.* **2022**, *20*, 2109–2172. [CrossRef]
100. DebRoy, T.; Wei, H.L.; Zuback, J.S.; Mukherjee, T.; Elmer, J.W.; Milewski, J.O.; Beese, A.M.; Wilson-Heid, A.; De, A.; Zhang, W. Additive Manufacturing of Metallic Components–Process, Structure and Properties. *Progress Mater. Sci.* **2018**, *92*, 112–224. [CrossRef]
101. Brackbill, J.; Kothe, D.; Zemach, C. A Continuum Method for Modeling Surface Tension. *J. Comput. Phys.* **1992**, *100*, 335–354. [CrossRef]
102. Meier, C.; Fuchs, S.L.; Much, N.; Nitzler, J.; Penny, R.W.; Praegla, P.M.; Pröll, S.D.; Sun, Y.; Weissbach, R.; Schreter, M.; et al. Physics-Based Modeling and Predictive Simulation of Powder Bed Fusion Additive Manufacturing Across Length Scales. *arXiv* **2021**, arXiv:2103.16982.
103. Kronbichler, M.; Diagne, A.; Holmgren, H. A Fast Massively Parallel Two-Phase Flow Solver for Microfluidic Chip Simulation. *Int. J. High Perform. Comput. Appl.* **2018**, *32*, 266–287. [CrossRef]
104. Caboussat, A.; Hess, J.; Masserey, A.; Picasso, M. Numerical Simulation of Temperature-Driven Free Surface Flows, with Application to Laser Melting and Polishing. *J. Comput. Phys. X* **2023**, *17*, 100127. [CrossRef]
105. Kopp, P.; Calo, V.; Rank, E.; Kollmannsberger, S. Space-Time Hp-Finite Elements for Heat Evolution in Laser Powder Bed Fusion Additive Manufacturing. *Eng. Comput.* **2022**, *38*, 4879–4893. [CrossRef]
106. Liu, L.; Huang, M.; Ma, Y.H.; Qin, M.L.; Liu, T.T. Simulation of Powder Packing and Thermo-Fluid Dynamic of 316L Stainless Steel by Selective Laser Melting. *J. Mater. Eng. Perform.* **2020**, *29*, 7369–7381. [CrossRef]
107. Mayi, Y.A.; Dal, M.; Peyre, P.; Bellet, M.; Metton, C.; Moriconi, C.; Fabbro, R. Transient Dynamics and Stability of Keyhole at Threshold in Laser Powder Bed Fusion Regime Investigated by Finite Element Modeling. *J. Laser Appl.* **2021**, *33*, 012024. [CrossRef]
108. Cahn, J.W.; Hilliard, J.E. Free Energy of a Nonuniform System. I. Interfacial Free Energy. *J. Chem. Phys.* **1958**, *28*, 258–267. [CrossRef]
109. Miura, R.M. The Korteweg–deVries Equation: A Survey of Results. *SIAM Rev.* **1976**, *18*, 412–459. [CrossRef]
110. Dobrzański, J.; Stupkiewicz, S. Towards a Sharper Phase-Field Method: A Hybrid Diffuse–Semisharp Approach for Microstructure Evolution Problems. *Comput. Methods Appl. Mech. Eng.* **2024**, *423*, 116841. [CrossRef]