An Application of Gaussian Process Modeling for High-order Accurate Adaptive Mesh Refinement Prolongation

Steven I. Reeves[a,e], Dongwook Lee[a,*], Adam Reyes[c], Carlo Graziani[b], Petros Tzeferacos[c,d]

[a]*Department of Applied Mathematics, The University of California, Santa Cruz, CA, United States*
[b]*Mathematics and Computer Science, Argonne National Laboratory, Argonne, IL, United States*
[c]*Department of Physics and Astronomy, University of Rochester, Rochester, NY, United States*
[d]*Laboratory for Laser Energetics, University of Rochester, Rochester, NY, United States*
[e]*Machine Learning Software Engineering, Advanced Micro Devices Inc., Santa Clara, CA, United States*

Abstract

We present a new polynomial-free prolongation scheme for Adaptive Mesh Refinement (AMR) simulations of compressible and incompressible computational fluid dynamics. The new method is constructed using a multi-dimensional kernel-based Gaussian Process (GP) prolongation model. The formulation for this scheme was inspired by the two previous studies on the GP methods introduced by A. Reyes *et al.*, Journal of Scientific Computing, 76 (2017), and Journal of Computational Physics, 381 (2019). In this paper, we extend the previous GP interpolations and reconstructions to a new GP-based AMR prolongation method that delivers a third-order accurate prolongation of data from coarse to fine grids on AMR grid hierarchies. In compressible flow simulations, special care is necessary to handle shocks and discontinuities in a stable manner. To meet this, we utilize the shock handling strategy using the GP-based smoothness indicators developed in the previous GP work by A. Reyes *et al.* We compare our GP-AMR results with the test results using the second-order linear AMR method to demonstrate the efficacy of the GP-AMR method in a series of test suite problems using the AMReX library, in which the GP-AMR method has been implemented.

Keywords: Adaptive Mesh Refinement; Prolongations; High-order methods; Gaussian processes; Computational fluid dynamics;

*Corresponding author
Email addresses: steven.reeves@amd.com (Steven I. Reeves), dlee79@ucsc.edu (Dongwook Lee), adam.reyes@rochester.edu (Adam Reyes), carlo@mcs.anl.gov (Carlo Graziani), p.tzeferacos@rochester.edu (Petros Tzeferacos)

1. Introduction

In the fields of geophysics, astrophysics, and laboratory plasma astrophysics, simulations have become essential to characterizing and understanding complex processes (e.g., [24, 29, 37, 58]). As increasingly complex systems are considered for computer modeling, modern simulation codes face increasingly versatile challenges to meet expected metrics in vast parameter space. Computational fluid dynamics (CFD) has been and will continue to be an indispensable tool to improve our capabilities to investigate scientifically challenging conditions, particularly where simplified theoretical models fail to adequately capture the correct physical behavior, or physical experiments become too observationally difficult and/or prohibitively expensive to be the sole pathways for discovery.

In computer simulations significant imbalances in length and temporal scales can cause the underlying physics to become extremely challenging to simulate with available computational resources. To alleviate such conditions, practitioners have explored approaches by which a simulation can focus on localized flow regions when the dynamics exhibit confined features that evolve on a much shorter length scale relative to the flow dynamics on the rest of the computational domain.

Adaptive mesh refinement (AMR) is one such approach that allows computer experiments to focus on localized dynamical changes by adaptively changing the simulation's grid resolutions to focus resources on localized regions in space and time. Since the 1980s, AMR has been an exceptional tool. It has become a powerful strategy in utilizing CFD simulations for computational science across many disciplines such as astrophysics, geophysics, atmospheric sciences, oceanography, biophysics, engineering, and many others [41].

There have been many advancements in AMR since the seminal paper by Berger and Oliger [8]. In their paper, the primary concern was to focus on a strategy for generating subgrids and managing the grid hierarchy for scalar hyperbolic PDEs in one and two spatial dimensions (1D and 2D). In the subsequent work by Berger and Colella [7], further improvements were made possible for numerical solutions of the 2D Euler equations to provide a robust shock-capturing AMR algorithm that satisfies the underlying conservation property on large-scale computer architectures. The novel innovations in their work have become the AMR standards, namely refluxing (or flux correction) between fine-coarse interface boundaries, conservative (linear) prolongation, and restriction on AMR hierarchies timestep subcycling. Bell *et al.* extended the precedent 2D AMR algorithms of [7, 8] to a 3D AMR algorithm and applied it to solve 3D hyperbolic systems of conservation laws [6]. They demonstrated that the AMR algorithm reduced the computational cost by more than a factor of 20 than on the equivalent uniform grid simulations in simulating a 3D dense cloud problem interacting with a Mach 1.25 flow on Cray-2. By far, this is the main benefit of using AMR, particularly in large 3D simulations, in that one could

gain such a computational speed-up by focusing computational resources on the dynamically interesting regions of the simulation.

In [27], Jameson addressed the efficiency of AMR schemes. In his work, the overall computational efficiency of using AMR is assessed based on the accuracy of the AMR scheme and the fraction of the region of the domain requiring grid refinements. Jameson estimated that small scale features, such as shocks, vortices, eddies, rotating flows, turbulence, etc., should not exceed more than a third of the computational domain in order for low-order AMR schemes (e.g., the traditional second-order AMR methods) to be computationally more competitive than a non-adaptive high-order calculation because the computational workload of AMR is larger than that of non-adaptive high-order numerical methods. In other words, a non-adaptive high-order calculation is practically beneficial for a simulation that is abundant of small scales over the majority of the domain, as the low-order AMR simulation would become computationally burdensome without gaining further enhancements in solution accuracy compared with the non-adaptive high-order simulation. He also points out that matching the numerical order of accuracy of the AMR scheme to a high-order baseline solver makes AMR competitive for larger fractions of the domain requiring the highest refinement. This study provides a criterion that helps determine how much one could leverage computational efficiency in utilizing AMR schemes against high-order non-AMR schemes.

The computational gains of AMR are only useful if there is consistency between the AMR calculations and those on a uniform grid. Mathematically speaking, AMR calculations should converge to the corresponding uniform grid solutions in the limit of grid convergence. In a numerical comparison study, Schmidt *et al.* [48] considered conditions upon which statistical agreement can be achieved between AMR and uniform grid calculations at the same effective grid resolutions.

Modern AMR implementations may be categorized into two main types: structured and unstructured. Unstructured AMR, and meshes in general, are very useful for problems with irregular geometry (e.g., many structural engineering problems) but are often computationally complex and difficult to handle when regridding. On the other hand, structured AMR (SAMR, or block-structured AMR) offers practical benefits over unstructured such as ease of discretization, a global index space, accuracy gain through cancellation terms, and ease of parallelization.

In block-structured AMR, a PDE solution is constructed on a hierarchy of levels with different resolutions. Each level is composed of a union of logically rectangular grids or patches. These patches can change dynamically throughout a simulation. Different AMR schemes feature different patch/block shapes so that some AMR codes allow logically different shapes of patch/block; some do not. Fig. 1 illustrates the use of AMR in a block-structured environment.

The approach presented by Berger and Oliger [8] and Berger and Colella [7] has set the foundation on the patch-based SAMR. An alternative to the patch-based formulation is the octree-based approach, which has evolved into the fully-threaded tree (FTT) formalism (or cell-based) of Khokhlov [31]; the

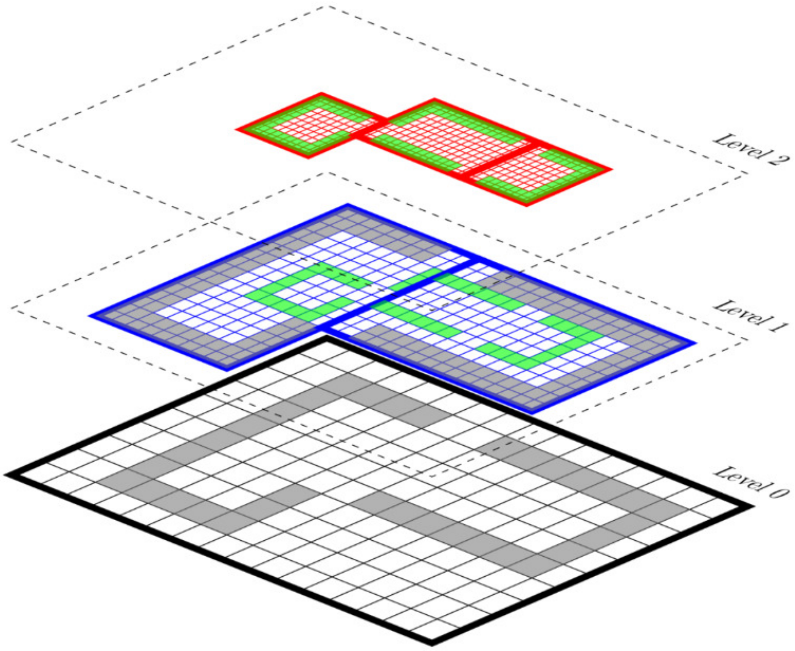


Figure 1: Multiple levels in a block-structured AMR grid hierarchy.

block-based octree formalism of MacNiece *et al.* [34] and van der Holst *et al.* [59].

AMR methods have gained popularity over the past 30 years, and various codes have adopted them in astrophysics. Some of the well-known examples implementing the patch-based AMR include AstroBEAR [18], ENZO [10], ORION [32], PLUTO [38], CHARM [39], CASTRO [2], and MAESTRO [40]; the octree-based AMR has been implemented in FLASH [21, 23], NIRVANA [64], and BATS-R-US [25, 42]; the FTT AMR in RAMSES [57] and ART [33]. The AMRVAC code [30] features both the patch-based and octree-based AMR schemes.

In contrast to these codes that incorporate AMR to deliver specific astrophysics applications, other frameworks have pursued a more general functionality. Examples include PARAMESH [34] that supplies the octree-based block-structured mesh capability independent of any governing equations solely; AMReX [62], Chombo [14, 1], and SAMRAI [26], on the other hand, supply both AMR capabilities and broader support for solving general systems of equations of hyperbolic, parabolic, and elliptic partial differential equations (PDEs) without necessarily being tied to a physical application. A more compressive survey on the block-structured AMR frameworks can be found in [20].

Recently, there have been many noticeable efforts aimed at designing high-order accurate solvers for governing systems of equations (e.g., [44, 36, 61, 11,

35, 22, 5, 53, 45, 47]) per a trend of decreasing memory per compute core in newer high-performance computing (HPC) architecture [4, 19, 55]. Such high-order (4th or higher) PDE solvers are then combined with the AMR strategies described above.

Traditionally, when combined with second-order PDE solvers, a second-order linear interpolation scheme has been commonly adopted for data prolongation from coarse to finer AMR levels. As recent developments have advanced the order of PDE solvers to high-order accuracy beyond second-order, there have also been efforts to push the accuracy of AMR interpolations correspondingly [60, 35]. Such algorithms reduce accuracy gaps that would exist between an underlying high-order PDE solvers and the traditional second-order AMR interpolation, which otherwise can degrade the quality of solutions from the high-order PDE solvers when the solutions are projected to AMR grids that are progressively undergoing refinements and de-refinements. In addition, another accuracy loss inevitably happens at fine-coarse boundaries. Therefore, it is natural to close the accuracy gap in the direction of providing high-order models in AMR interpolations to serve better to maintain the overall solution accuracy integrated as a whole on AMR grid configurations. The high-order AMR prolongations of Shen *et al.* [50] and Chen *et al.* [12] are in this vein. These authors coupled high-order finite difference method (FDM) PDE solvers with fourth- or fifth-order accurate prolongations based on the well-known high-order polynomial interpolation schemes of WENO [28] and MP5 [56], respectively. These studies have shown that the AMR simulations with a higher-order coupling can produce better results in terms of increasing solution accuracy and lowering numerical diffusion, thereby resolving fine-scale flow features as much as possible.

The present work focuses on developing a new high-order polynomial-free interpolation scheme for AMR data prolongation on the block-structured AMR implementation using the AMReX library. Our high-order prolongation scheme stems from the previous studies on applying Gaussian Process Modeling [43] in designing high-order reconstruction/interpolation in finite volume method (FVM) [45] and in finite difference method (FDM) [47].

The main goals of this paper are two-fold: (i) we aim to show how the previous GP reconstruction and interpolation methods can be extended to a volume-conserving third-order AMR prolongation method, and (ii) we aim to demonstrate how the new GP-AMR method can be seamlessly integrated with an existing FVM code framework. To meet our goals, we utilize a public AMR framework called AMReX for our testbed, in which we have implemented our third-order GP-AMR prolongation scheme. The new GP-AMR prolongation method's performance and accuracy are compared with those of the standard second-order linear prolongation method in AMReX. Both prolongation methods are combined with a second-order PDE solver [†]. We acknowledge that our

[†] Our results are represented in 5.2, where the tests derived in AMReX use a second-order advection routine and Castro [2] tests use PPM [15] with the corner transport upwind algorithm. It should be noted that the PPM method implemented in Castro only allows

third-order GP-AMR scheme is ideally better integrated with an underlying PDE solver that is at least third-order accurate to match the overall solution accuracy. However, such a work pre-requires to implement a third- or higher-order FVM PDE solver, an attentive task [35] in general and beyond the scope of the current work. For this reason, we leave such a task for future work.

This paper is organized as follows. In Section 2, we overview the relevant AMR framework, AMReX, as our computational toolkit in which we integrate our new GP-based prolongation algorithm. In Section 3, we provide a mathematical overview of the GP modeling specific to high-order AMR prolongation. We give step-by-step execution details of our algorithm in Section 4. Section 5 shows the code performance of the new GP prolongation on selected multi-dimensional test problems, and finally, in Section 6, we summarize the main results of our work. A brief introduction to our GP modeling is described in Appendix A.

2. Overview of AMReX

AMReX is developed and managed by the Center for Computational Science and Engineering at Lawrence Berkeley National Laboratory. AMReX is funded through the Exascale Computing Project (ECP) as a software framework to support the development of block-structured AMR applications focusing on current and next-generation architectures [62]. AMReX supports for many operations involving adaptive meshes, including multilevel synchronization operations, particle and particle/mesh algorithms, solution of parabolic and elliptic systems using geometric and algebraic multigrid solvers, explicit/implicit mesh operations, to name a few. As part of an ECP funded project, AMReX takes the hybrid MPI/OpenMP CPU parallelization along with GPU implementations (CUDA). AMReX is mostly comprised of source files that are written in C++ and Fortran. Fortran is solely used for mathematics drivers, while C++ is used for I/O, flow control, memory management, and mathematics drivers (when portability is a requirement).

The novelty of the current study is the new GP-based prolongation method implemented within the AMReX framework. The GP implementation furnishes an optional high-order prolongation method from coarse to fine AMR levels, alternative to the default second-order linear prolongation method in AMReX. In this way, the GP results in Section 5 naturally inherit all the generic AMReX operations such as load balancing, guardcell exchanges, refluxing, AMR data, and grid management, except for the new GP prolongation method. We display a suite of test comparisons between the two prolongation methods. Scientific motivations for adopting the GP algorithm will also be briefly discussed in the next section.

AMR restriction is another important operation on the AMR data management in the opposite direction, from fine to coarse levels. We use the default

second-order accuracy for finite volume simulations in multiple spatial dimensions.

averaging-based restriction method that maintains conservation on AMR grid hierarchies. This approach populates data on coarse levels by averaging down the corresponding fine level data according to

$$\mathbf{U}^C = \frac{1}{R}\sum_{i}^{R}\mathbf{U}_i^f, \tag{1}$$

where $\mathbf{U}^C$ and $\mathbf{U}^f$ are conservative quantities on the coarse and fine grids respectively, and $R = \prod_d r_d$ is the normalization factor with r_d, integer values in the form of powers of two, representing the refinement ratio in each direction $d = x, y, z$.

Lastly, maintaining conservation across fine-coarse interface levels is done by the operation called refluxing. This process corrects the coarse grid fluxes by averaging down the fluxes computed on the fine grids abutting the coarse grid. In practice, the conservation is managed as a posterior correction step after all fluid variables $\mathbf{U}^C$ on a coarse cell are updated. For other AMR operations related to AMReX, interested readers are encouraged to refer to [62, 63, 65].

3. Gaussian Process Modeling for CFD

In this paper we present a new prolongation method based on Gaussian Process (GP) Modeling. Our approach of designing a GP-based prolongation method for CFD is primarily based on the two previous foundational works on GP high-order methods: the 1D finite-volume GP method prescribed in [45] and the 3D finite-difference GP method in [47].

Readers wishing to pursue the subject in greater detail are referred to the sections in Appendix A of this paper as well as [9, 43]. Also, readers who are more interested in the direct CFD applications of our new GP modeling in finite difference and finite volume frameworks are encouraged to review our previous studies [45, 47].

We describe two main modeling algorithms of the GP-based AMR prolongation in the following sections. Section 3.1 describes the first method that prolongs pointwise state data from coarse to fine levels. For AMR simulations in which the state data is represented as volume-averaged, conserving such quantities become crucial to satisfy the underlying conservation laws. To meet this end, we introduce the second method in Section 3.2, which preserves volume-averaged quantities in prolongation. We will refer to our GP-based AMR prolongation as GP-AMR for the rest of this paper.

3.1. The first version: a pointwise GP-AMR prolongation

We introduce the first GP-AMR prolongation method that is suitable for AMR applications where the state data is comprised of pointwise values. In this case, the GP-AMR model samples are given as pointwise evaluations of the underlying function. Let Δd denote the grid distance between two distant points in a *coarse* level in each $d = x, y, z$ direction. Using the posterior mean

function in Eq. (A.4), we first devise a pointwise prolongation scheme for AMR, i.e., an AMR prolongation of pointwise data from coarse to fine levels. The choice of $\mathbf{x}_*$ will depend on the *refinement ratio* $\mathbf{r} = [r_x, r_y, r_z]$ and there will be $\prod_d r_d$ new points generated for the new level in general. For example, if we wished to refine a single coarse grid by two in all three directions in 3D (i.e., $r_x = r_y = r_z = 2$), we would generate eight new grid points as well as the eight new associated data values at those grid points in a newly refined level.

To illustrate the process, we consider a simple example of a two-level refinement in 1D. In this refinement, two refined data values are to be newly generated for every coarse value. Assume here that we utilize a stencil with the GP radius of one (i.e., $R = 1$), in which case the local 3-point GP stencil $\mathbf{f}_i$, centered at each i-th cell for interpolation, is laid out as

$$\mathbf{f}_i = [q_{i-1}, q_i, q_{i+1}]^T,$$

where each q_i is a pointwise data value located x_{i-1}, x_i, x_{i+1} respectively. We wish to generate two finer pointwise data values $q_{i\pm 1/4}$ for each ith coarse cell from this given stencil data at the coarse level. To do this, we use the posterior mean function in Eq. (A.4) on three 3-point GP stencils, $\mathbf{f}_i$, to populate two new data points $q_{i\pm\frac{1}{4}}$ at two new grid locations, $x_* = \{x_{i\pm\frac{1}{4}}\}$,

$$q_{i\pm\frac{1}{4}} = \mathbf{k}_{i\pm\frac{1}{4}}^T \mathbf{K}^{-1} \mathbf{f}_i, \tag{2}$$

where we used a zero mean prior, $\bar{\mathbf{f}} = 0$. In 2D or 3D, multidimensional data values are to be reshaped into a 1D local array $\mathbf{f}_i$ in an orderly fashion. This strategy will be fully described in Section 4.

A common practice with GP modeling is to assume a zero prior mean as we did with Eq. (2), which we also follow in our current implementation. Something to note is that the GP weights, $\mathbf{k}_*^T \mathbf{K}^{-1}$, where $*$ denotes $i \pm 1/4$ for the present purpose, are independent of the samples $\mathbf{f}_i$. Moreover, the GP weights are solely constructed based on the choice of kernel function $\mathbf{K}$ and the location of the samples, $\mathbf{x}_i$, and prediction point, $\mathbf{x}_*$, alone. This is particularly useful in block-structured AMR applications, as we can compute the weights for each level a priori, based on the minimum and maximum levels prescribed for each run. Otherwise, we can generate the model weights at the initialization of level is used and save them for later uses. This is in contrast to polynomial based methods for high-order prolongation that requires a least squares solution for each cell to be refined [35], which we explore further in Section 3.5.

Since the matrix $\mathbf{K}$ is symmetric and positive-definite, we can use the Cholesky decomposition to compute the GP weights. In practice, we compute and save $\mathbf{w}_*^T = \mathbf{k}_*^T \mathbf{K}^{-\mathbf{1}}$ using Cholesky decomposition followed by back-substitution only once per refinement level and per simulation, either at an initial grid configuration step or at the first time an AMR level is newly used. In this way, the computational cost of the prolongation is reduced to a dot product between $\mathbf{w}$ and $\mathbf{f}$ for each newly prolonged point.

Finally, we arrive at a compact form for a pointwise version of the GP prolongation for each ith coarse cell, producing two new data points $q_{i\pm\frac{1}{4}}$ at

$x_{i\pm 1/4}$,

$$q_{i\pm\frac{1}{4}} = \mathbf{w}^T_{s\pm\frac{1}{4}}\mathbf{f}_i. \tag{3}$$

3.2. The second version: a volume-averaged GP-AMR prolongation

For most AMReX and fluid dynamics application codes, the state data is volume-averaged (or cell-averaged), as per the formulation of FVMs. The above GP-AMR prolongation for pointwise data has to be modified for the volume-conserving relation between fine and coarse data. This conservation property is implicit in the integral formulation of the governing equations used in FVM and is to be preserved in AMR prolongation steps; otherwise, simulations will disobey the underlying principles of the conservation laws in hyperbolic systems of PDEs.

The key observation from [45] is that the averaging over cells constitutes a "linear" operation on a function $f(\mathbf{x})$. As is done for finite-dimensional Gaussian Processes, linear operations on Gaussian random variables yields a new Gaussian random variable with linearly transformed mean and covariance functions. Accordingly, given that integration is a linear operator, we achieve an integrated covariance GP kernel $\mathbf{C}$ to calculate the covariance between two cell-averaged quantities, G_k and G_h, as described in [45]. That is,

$$\begin{aligned}
[\mathbf{C}]_{kh} = C_{kh} &= \mathbb{E}[(G_k - \bar{G}_k)(G_h - \bar{G}_h)] \\
&= \int \mathbb{E}[(f(\mathbf{x}) - \bar{f}(\mathbf{x}))(f(\mathbf{y}) - \bar{f}(\mathbf{y}))] dg_k(\mathbf{x}) dg_h(\mathbf{y}) \\
&= \iint K(\mathbf{x}, \mathbf{y}) dg_k(\mathbf{x}) dg_h(\mathbf{y}),
\end{aligned} \tag{4}$$

where, with abuse of notation, $\mathbf{x} = [x_x, x_y, x_z]^T \in \mathbb{R}^D$ with $D = 3$,

$$dg_s(\mathbf{x}) = \begin{cases} d\mathbf{x} \displaystyle\prod_{d=x,y,z}^{D} \frac{1}{\Delta x_d} & \text{if } \mathbf{x} \in I_s, \\ 0 & \text{else,} \end{cases} \tag{5}$$

is the cell-volume measure, and $G_i = \langle f(\mathbf{x}_i) \rangle = \frac{1}{\mathcal{V}} \int_{I_i} f(\mathbf{x}_i) d\mathcal{V}$ are the cell-averaged data over the ith cell $I_i \subset \mathbb{R}^D$ with its cell volume $\mathcal{V}$.

With the use of the squared-exponential kernel, Eq. (4) is written analytically

as,

$$C_{kh} = \prod_{d=x,y,z}^{D} \sqrt{\pi}\left(\frac{\ell}{\Delta x_d}\right)^2 \Bigg\{ \left(\frac{\Delta_{kh}+1}{\sqrt{2}\ell/\Delta x_d}\text{erf}\left[\frac{\Delta_{kh}+1}{\sqrt{2}\ell/\Delta x_d}\right] + \frac{\Delta_{kh}-1}{\sqrt{2}\ell/\Delta x_d}\text{erf}\left[\frac{\Delta_{kh}-1}{\sqrt{2}\ell/\Delta x_d}\right]\right) + \frac{1}{\sqrt{\pi}}\left(\exp\left[-\left(\frac{\Delta_{kh}+1}{\sqrt{2}\ell/\Delta x_d}\right)^2\right] + \exp\left[-\left(\frac{\Delta_{kh}-1}{\sqrt{2}\ell/\Delta x_d}\right)^2\right]\right) - 2\left(\frac{\Delta_{kh}}{\sqrt{2}\ell/\Delta x_d}\text{erf}\left[\frac{\Delta_{kh}}{\sqrt{2}\ell/\Delta x_d}\right] + \frac{1}{\sqrt{\pi}}\exp\left[-\left(\frac{\Delta_{kh}}{\sqrt{2}\ell/\Delta x_d}\right)^2\right]\right)\Bigg\}, \tag{6}$$

where we used $\Delta_{kh} = (x_{d,h} - x_{d,k})/\Delta x_d$ with $d = x, y, z$.

Following similar arguments as for the covariance kernel function, the prediction vector $\mathbf{k}_*$ with one point anchored at $\mathbf{x}_*$ must also be linearly transformed to reflect the relationship between the input data averaged over coarse cells and the output prolonged data averaged over the fine cells, which requires double integrals over the cell I_* and over each I_k. This leads to

$$[\mathbf{T}_*]_k = T_{*k} \equiv T(\mathbf{x}_*, \mathbf{x}) = \int_{I_*}\int_{I_k} K(\mathbf{x}_*, \mathbf{x}) dg_k(\mathbf{x}) dg_*(\mathbf{x}_*), \tag{7}$$

where

$$I_* = \bigtimes_{d=x,y,z}^{D} \left[x_{d,*} - \frac{\Delta x_d}{2r_d},\ \ x_{d,*} + \frac{\Delta x_d}{2r_d}\right],$$

in which $\times$ denotes the Cartesian production on sets. Using the SE kernel, we have a closed analytical form for T_{*k},

$$T_{*k} = \pi^{D/2} \prod_{d=x,y,z}^{D} r_d \left(\frac{\ell}{\Delta x_d}\right)^2 \sum_{\alpha=1}^{4} (-1)^\alpha \left[\phi_{\alpha,d}\,\text{erf}(\phi_{\alpha,d}) + \frac{1}{\sqrt{\pi}}\exp(-\phi_{\alpha,d}^2)\right], \tag{8}$$

where, for each $\alpha = 1, \ldots, 4$ and for each $d = x, y, z$,

$$\phi_{\alpha,d} = \frac{\Delta^\alpha}{\sqrt{2}\ell/\Delta x_d}, \tag{9}$$

with each numerator Δ^α being given by

$$\Delta^\alpha = \begin{cases} \Delta_{*k} + \frac{r_d - 1}{2r_d} & \text{if } \alpha = 1, \\ \Delta_{*k} + \frac{r_d + 1}{2r_d} & \text{if } \alpha = 2, \\ \Delta_{*k} - \frac{r_d - 1}{2r_d} & \text{if } \alpha = 3, \\ \Delta_{*k} - \frac{r_d + 1}{2r_d} & \text{if } \alpha = 4. \end{cases} \tag{10}$$

Finally, with the combination of the cell-averaged kernel in Eq. (6) and the cell-averaged weight vector in Eq. (8), we obtain our second GP-AMR formula given in the integral analog of Eq. (A.4) for cell-averaged data prolongation from coarse to fine levels,

$$\langle f(\mathbf{x}_*)\rangle = \mathbf{T}_*^T \mathbf{C}^{-1} \mathbf{G}_i, \tag{11}$$

where we used the zero mean as before. The ith data vector $\mathbf{G}_i$ of cell-averaged samples within the GP radius R is given as $\mathbf{G}_i = [G_{i-R}, \ldots, G_{i+R}]^T$. Analogous to the pointwise method, we cast $\mathbf{T}_*^T \mathbf{C}^{-1}$ into a new GP weight vector $\mathbf{z}_*$ to rewrite Eq. (11) into a compact form,

$$\langle f(\mathbf{x}_*)\rangle = \mathbf{z}_*^T \mathbf{G}_i. \tag{12}$$

Many multidimensional numerical methods often perform interpolation in a dimension-by-dimension manner for simplicity. In contrast, the above two pointwise and volume-averaged GP-AMR methods are inherently multidimensional in their spatial operations. Moreover, the use of the SE kernel as a base in each d-direction facilitates the closed analytical form obtained in Eq. (6) for multiple spatial dimensions. Therefore, our GP-AMR methods provide a genuine framework where all interpolation procedures in AMR grid hierarchies naturally support multidimensionality, as the evaluation of the covariance matrices depends only on the distance between data points but not on the data themselves. Furthermore, it is worth pointing out that the two prolongation schemes in Eqs. (3) and (12) are merely a straightforward calculation of dot products between the GP weight vectors and the grid data ($\mathbf{f}$ and $\mathbf{G}$).

This is the novelty of the use of GP modeling in AMR prolongation, which reveals two new compact prolongation methods that are computed in the same way for any stencil configuration in any number of spatial dimensions without any added complexity. This is in stark contrast to polynomial-based methods that each different interpolation algorithm requires the use of explicit basis functions and have strict requirements on stencil sizes and configurations.

We summarize this section by noting two distinct advantages in the present GP-AMR methods. First, the algorithm offers unique flexibility that can vary its order of accuracy, $(2R+1)$, by varying the size of the GP data according to the GP radius R. Second, the baseline 1D GP scheme can be seamlessly extended to any higher dimension by including the corresponding higher dimensional term(s) in the products in Eqs. (6) and (8).

3.3. Examples of GP prolongation in 1D and 2D using $R = 1$

We present two simple examples of GP prolongation with $R = 1$ to illustrate the relevant data layout and notations of our GP algorithm.

Example 1: *GP prolongation in 1D with $R = 1$.* Fig. 2 shows that two parent cells I_2 and I_3 are prolonged to a total of six finer children cells, two from a refinement level of two ($r_x = 2$) and the other four from a refinement level of four

($r_x = 4$). The left-most child cell depicted as I_* is centered at new prolonged data point $x_* = x_2 - \Delta x/4$ with a uniform grid-scale Δx of the coarse cells I_i. In Eq. (10), we compute three relative distances of Δ_{*k} with $k = 1, 2$, and 3, each of which becomes $\Delta_{*1} = 3/4$, $\Delta_{*2} = 1/4$, and $\Delta_{*3} = 5/4$.

To obtain a new prolonged data at x_*, GP-AMR uses either $\mathbf{f}_2 = [q_1, q_2, q_3]^T$ for pointwise data, or $\mathbf{G}_2 = [\langle q_1 \rangle, \langle q_2 \rangle, \langle q_3 \rangle]^T$ for volume-averaged data, over which the GP interpolation takes place at third-order accuracy. With these three 3-point data configurations under $R = 1$, the 3×3 GP kernel is either a pointwise matrix $\mathbf{K}$ or a volume-averaged matrix $\mathbf{C}$, while the pointwise weight vector $\mathbf{w}_*$ and the volume-averaged weight vector $\mathbf{z}_*$ are of length three. The GP data vector $\mathbf{f}_2$ (or $\mathbf{G}_2$) are applied to the constant (precomputed) weight vector $\mathbf{w}_*^T$ or $\mathbf{z}_*^T$ according to Eq. (3) (or Eq. (12)) to obtain a new GP-prolonged data at x_*, for which the three coarse data on I_1, I_2 and I_3 have been correlated to produce a third-order accurate GP prolonged prediction. The GP data prolongations of the coarse cell I_3 with $r_x = 4$ can be obtained in a similar manner.

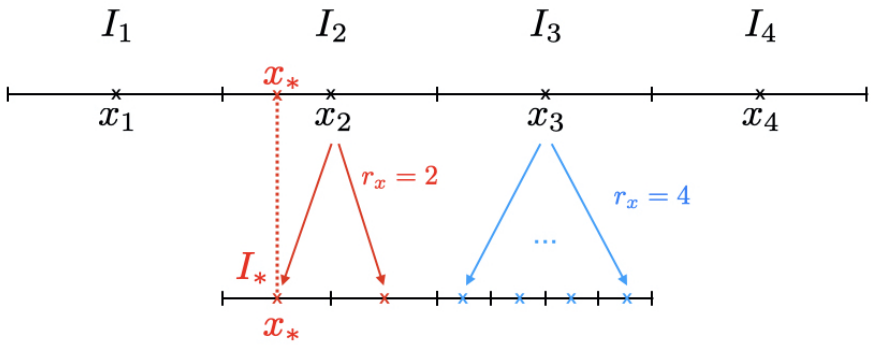


Figure 2: The figure illustrates two different data prolongations with two different refinement levels. The parent cell I_2 is refined with a refinement level of two, $r_x = 2$, generating two new children cells, while the parent cell I_3 is refined with a refinement level of four, $r_x = 4$, producing four new children cells. The new children cells are shown at the bottom.

Example 2: *GP prolongation in 2D with* $R = 1$. The overall GP prolongation in 2D can be processed in a similar way as we have discussed in the previous 1D example. The only important point of discussion here is the local GP stencil shape that takes a 5-point cross-shaped configuration with $R = 1$. The GP data array, either $\mathbf{f}_2$ or $\mathbf{G}_2$, can be constructed by rearranging the five coarse data on $I_1, \ldots, I_5$ into a 1D array in an orderly fashion. This data layout leads to a 5×5 GP covariance kernel, including all five coarse cell data on $I_1, \ldots, I_5$, in either pointwise or volume-averaged. Accordingly, the GP weight vector is of length five. The rest should follow easily by considering the 2D nature of the grid configurations.

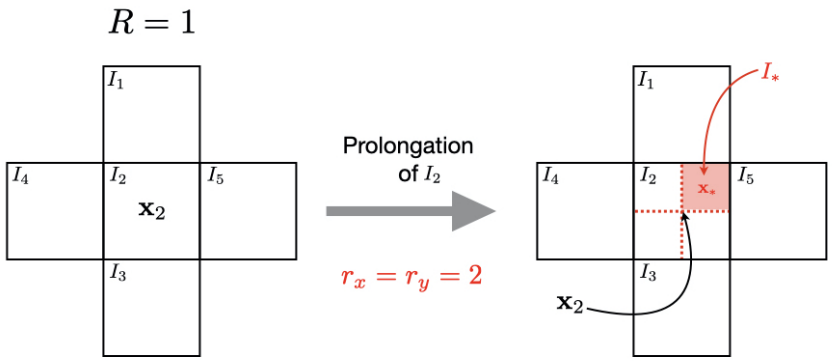


Figure 3: The figure illustrates a 2D GP prolongation of the cell I_2 with a refinement level of two, $r_x = r_y = 2$, generating four new children cells.

3.4. Handling discontinuities: a nonlinear multi-substencil method of GP-WENO

Both of the above GP modeling techniques can suffer from non-physical oscillations near discontinuities. The pointwise SE kernel $\mathbf{K}$ and the cell-averaged SE kernel $\mathbf{C}$ work very well for smoothly varying continuous data, but we need to implement some "limiting" process to suppress 'nonphysical' oscillations in flow regions with sharp gradients. As an example, the linear polynomial interpolations used by default in AMReX adopts the monotonized central (MC) slope limiter to produce *limited* slopes that do not introduce any new extrema into the solution.

This study utilizes the GP-based smoothness indicator approach studied in GP-WENO [45, 47]. The first step is to build $(2D + 1)$ substencil data on each substencil S_m, $m = 1, \ldots, 2D + 1$. In the second step, these data are combined using linear weights γ_m. The linear weights are derived from an overdetermined linear system relating the weights generated by building a GP model on all substencils S_m and the weights generated from a GP model on a total stencil S. The last step is to take the linear weights γ_m to define nonlinear weights ω_m using the GP-based smoothness indicators β_m [46, 47].

We now describe in detail the GP-AMR method for the two-dimensional case. Extensions to other dimensions are readily obtained due to the isotropic SE kernel choice, with the only difference being in the number of stencil points used. We begin with a total stencil S, taken as all cells whose index centers are within a radius of two (i.e., $R = 2$) of the central cell $I_{i,j}$. The total stencil S is then subdivided into $(2D + 1)$ candidate stencils, $S_m, m = 1, \ldots 2D + 1$, such that

$$\bigcap_{m=1}^{2D+1} S_m = \{\mathbf{x}_{i,j}\} \quad \text{and} \quad \bigcup_{m=1}^{2D+1} S_m = S. \tag{13}$$

A schematic of these stencil configurations is given in Fig. 4, which illustrates a prolongation of the grid data on the coarse cell $I_{i,j}$ to a newly refined data

point at $\mathbf{x}_*$ will have the form:

$$f_* = \sum_{m=1}^{2D+1} \omega_m \mathbf{w}_m^T \mathbf{f}_m, \tag{14}$$

where $\mathbf{w}_m^T = \mathbf{T}_{*,m}\mathbf{C}_m^{-1}$ for the cell-averaged prolongation, or $\mathbf{w}_m^T = \mathbf{k}_{*,m}^T\mathbf{K}_m^{-1}$ for the pointwise prolongation.

The coefficients ω_m are defined as in the WENO-JS method [52],

$$\omega_m = \frac{\tilde{\omega}_m}{\sum_s \tilde{\omega}_s}, \quad \text{where} \quad \tilde{\omega}_m = \frac{\gamma_m}{(\epsilon + \beta_m)^p}, \tag{15}$$

where we set $\epsilon = 10^{-36}$ and $p = 2$, following [47].

As in [45, 47], we compute the nonlinear smoothness indicators β_m that are derived as a data-dependent form by considering the negative log of the GP likelihood function in Eq. (A.2), namely, $-\log(\mathcal{L})$ for the stencil data (see [45, 47] for more details). The result is a new set of probability-based GP smoothness indicators β_m, defined by,

$$\beta_m = \mathbf{f}_m^T \mathbf{K}_{m,\sigma}^{-1} \mathbf{f}_m, \tag{16}$$

for the pointwise prolongation, and

$$\beta_m = \mathbf{f}_m^T \mathbf{C}_{m,\sigma}^{-1} \mathbf{f}_m, \tag{17}$$

for the cell-averaged prolongation. The subindices in the covariance kernels $\mathbf{K}_{m,\sigma}$ and $\mathbf{C}_{m,\sigma}$ represent that (i) they are computed over the data points on each S_m to measure the data correlations over the relative distance between the two data points in S_m, and (ii) the kernels' hyperparameter is now a smaller parameter σ, replacing the standard (and larger) length-scale ℓ in the covariance kernels $\mathbf{K}$ and $\mathbf{C}$ on the total stencil S.

As described in [45, 47], the GP-based smoothness indicators β_m defined in this way are derived by taking the negative log of the GP likelihood in Eq. (A.2). This gives rise to the statistical interpretation of β_m, which relates that if there is a shock or discontinuity in one of the substencils, say S_k, such a short length-scale (or rapid) change on S_k makes $\mathbf{f}_k$ unlikely. Here, this likeliness is relative to a GP model that assumes that the underlying function is smooth on the length scale set by σ. In other words, the GP model whose smoothness is represented by the smoothness property of its covariance kernel, $\mathbf{K}_{m,\sigma}$ or $\mathbf{C}_{m,\sigma}$, gives a low probability to $\mathbf{f}_k$, in which case β_k – given as the negative log likelihood of $\mathbf{f}_k$ – becomes relatively larger than the other β_m, $m \neq k$. In this way, analogous to the original Weighted Essentially Non-Oscillatory (WENO) [28] approach, the nonlinear GP-WENO adaptively chooses a non-oscillatory stencil by nonlinearly weighing GP predictions trained on a set of substencils according to a GP-based local smoothness indicator β_m for each m. The smoothness is determined using the GP likelihood to measure the compatibility of the substencil data $\mathbf{f}_m$ with the smooth SE kernel. Effectively, each β_m indicates how well the data in

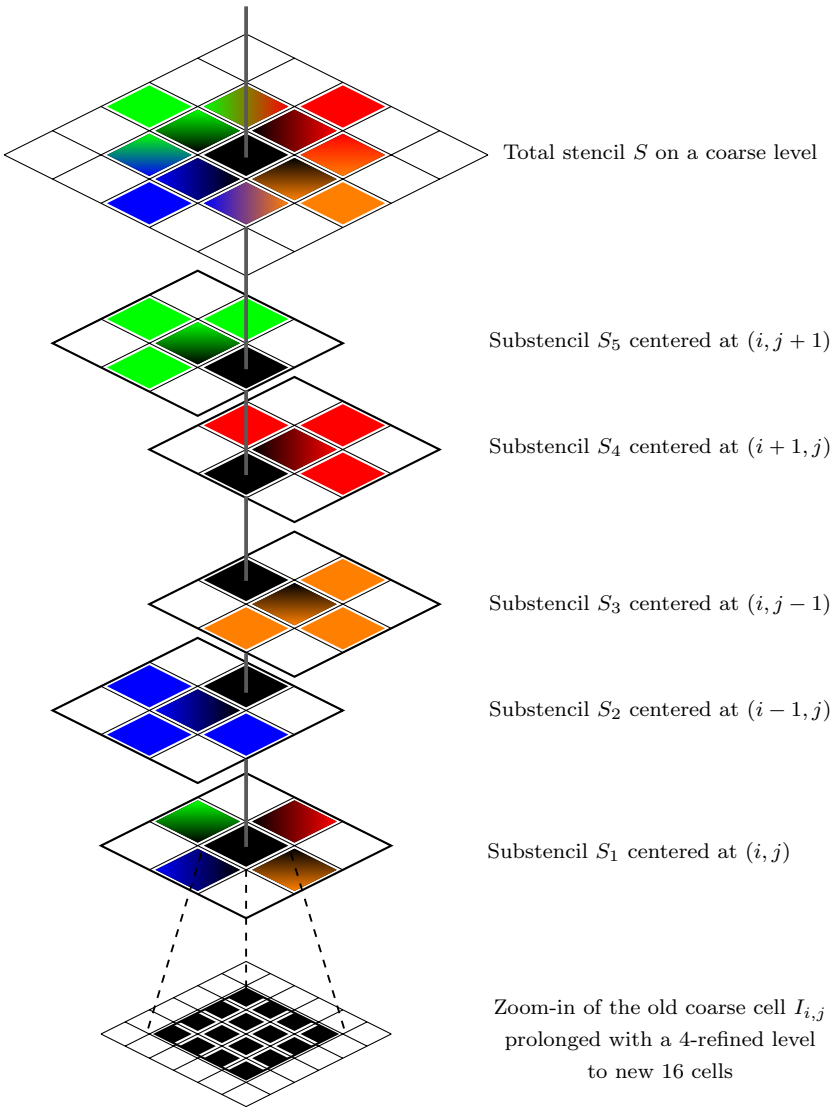


Figure 4: An illustration of the GP prolongation using five GP substencils, all of which are combined to produce 16 new prolonged data on a 2D finer grid. The 4-refinement ratio in both x and y directions is considered here to prolong the single data from the old coarse cell $I_{i,j}$ to 16 newly refined locations, $\mathbf{x}_*$, pictured in the bottom. The vertical line is displayed to indicate the position of the coarse cell $I_{i,j}$ in the total stencil S as well as the five substencils $S_1, \ldots, S_5$. The color schemes (e.g., green, red, orange, and blue) in each substencil S_m are used to match the corresponding areas of each S_m as part of the total stencil S.

$\mathbf{f}_m$ matches with the GP model assumptions encoded in the *smooth, infinitely differentiable SE kernels* $\mathbf{K}_{m,\sigma}$ (or $\mathbf{C}_{m,\sigma}$).

There two differences between $\mathbf{K}$ and $\mathbf{K}_{m,\sigma}$ in that (i) $\mathbf{K} \in \mathbb{R}^{M\times M}$ and $\mathbf{K}_{m,\sigma} \in \mathbb{R}^{(2D+1)\times(2D+1)}$, where $M = 2D^2 + 2D + 1$ for each spatial dimension $D = 1, 2, 3$, and (ii) the scale-length hyperparameter σ for $\mathbf{K}_{m,\sigma}$ is a length scale corresponding to the narrow shock-widths spread over a couple of grid spacing; hence σ is much smaller than the length hyperparameter ℓ of $\mathbf{K}$ (i.e., $\sigma \ll \ell$). We typically set $\sigma \sim c_\sigma \min_d(\Delta x_d)$ with $1 \leqslant c_\sigma \leqslant 3$, and $\ell \sim c_\ell \min_d(\Delta x_d)$ with $6 \leqslant c_\ell \leqslant 12$ for our simulations. The same differences hold between $\mathbf{C}$ and $\mathbf{C}_{m,\sigma}$ as well.

Notice that, due to the properties of the kernel matrices [45], we can cast β_m in Eqs. (16) and (17) into a simpler form,

$$\beta_m = \sum_{i=1}^{2D+1} \frac{1}{\lambda_i} \left(\mathbf{v}_i^T \mathbf{f}_m\right)^2, \tag{18}$$

where $\mathbf{v}_i$ and λ_i are the eigenvectors and eigenvalues of the covariance kernel matrix, $\mathbf{K}_{m,\sigma}$ or $\mathbf{C}_{m,\sigma}$.

In our method, it can be said that we use GP modeling for both a regression (prolongation) and a classification. The regression aspect enables us to prolongate GP samples (i.e., function values or fluid values) over the longer length-scale variability specified by ℓ. On the other hand, the classification aspect allows us to detect and handle discontinuities. This is achieved by employing a much shorter length-scale variability tuned by σ, which is integrated into the eigensystem in Eq. (18) generated with $\mathbf{K}_{m,\sigma}$ or $\mathbf{C}_{m,\sigma}$. Smaller than ℓ, the parameter σ is chosen to reflect the short width of shocks and discontinuities in numerical simulations, which is typically over a couple of grid spacings. In this manner, we use two length scale parameters, ℓ for the interpolation model and σ for shock-capturing.

Another key factor are the linear weights γ_m, $m = 1, \ldots, 2D + 1$. Let $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_{2D+1}]^T$ be a vector containing $(2D + 1)$ linear weights, each corresponding to one of the substencils. These weights are retrieved by solving an over-determined linear system

$$\mathbf{M}\boldsymbol{\gamma} = \mathbf{w}_*, \tag{19}$$

where the nth column of $\mathbf{M}$ is given by $\mathbf{w}_n$, and $\mathbf{w}_*$ is the model weights for the interpolation point $\mathbf{x}_*$ relative to the total stencil S. As mentioned previously, these weights are generated using the length scale parameter ℓ. We should note that $\mathbf{M}$ is a potentially sparse matrix and is constructed using the substencil model weights (see below for an example).

We cast the multidimensional stencil S as a 'flattened' 1D array for our GP modeling procedure in multiple spatial dimensions. To illustrate this concept, we explore a $2D$ example where the coarse level cells are refined by the 4-refinement ratio in both x and y directions, i.e., $r_x = r_y = 4$. Suppose $D = 2$, in which case and the total stencil S is in the 5×5 patch of cells centered at (i, j)

and contains 13 data points. The total stencil is subdivided into five 5-point substencils S_m, $m = 1, \ldots, 5$. We take the natural cross-shape substencil for each S_m on each of which GP will approximate function values (i.e., state values of density, pressure, etc.) at 16 new refined locations, i.e., $(i \pm 1/4, j \pm 1/4)$, $(i \pm 1/4, j \pm 3/4)$, $(i \pm 3/4, j \pm 1/4)$, and $(i \pm 3/4, j \pm 3/4)$.

For instance, let's choose $\mathbf{x}_{i+1/4,j+1/4}$ as the location we wish GP to compute function values for prolongation. Explicitly, five 5-point substencils are chosen as,

$$\begin{aligned}
S_1 &= [\mathbf{x}_{i,j-1}, \quad \mathbf{x}_{i-1,j}, \quad \mathbf{x}_{i,j}, \quad \mathbf{x}_{i+1,j}, \quad \mathbf{x}_{i,j+1} \quad], \\
S_2 &= [\mathbf{x}_{i,j-2}, \quad \mathbf{x}_{i-1,j-1}, \mathbf{x}_{i,j-1}, \quad \mathbf{x}_{i+1,j-1}, \mathbf{x}_{i,j} \quad], \\
S_3 &= [\mathbf{x}_{i+1,j-1}, \mathbf{x}_{i,j}, \quad \mathbf{x}_{i+1,j}, \quad \mathbf{x}_{i+2,j}, \quad \mathbf{x}_{i+1,j+1}], \\
S_4 &= [\mathbf{x}_{i,j}, \quad \mathbf{x}_{i-1,j+1}, \mathbf{x}_{i,j+1}, \quad \mathbf{x}_{i+1,j+1}, \mathbf{x}_{i,j+2} \quad], \\
S_5 &= [\mathbf{x}_{i-1,j-1}, \mathbf{x}_{i-2,j}, \quad \mathbf{x}_{i-1,j}, \quad \mathbf{x}_{i,j}, \quad \mathbf{x}_{i-1,j+1}].
\end{aligned} \tag{20}$$

In this example, the total stencil S is constructed to satisfy $\bigcap_{m=1}^{5} S_m = \{\mathbf{x}_{i,j}\}$ and $\bigcup_{m=1}^{5} S_m = S$, containing 13 data points whose local indices range from $i-2, j-2$ to $i+2, j+2$, excluding the 12 cells in the corner regions. See Fig. 4, for a detailed schematic of the multi-substencil approach.

Using these data points, we build a 13×5 over-determined system

$$\begin{pmatrix}
w_{1,1} & 0 & 0 & 0 & 0 \\
w_{1,2} & w_{2,1} & 0 & 0 & 0 \\
w_{1,3} & 0 & w_{3,1} & 0 & 0 \\
w_{1,4} & 0 & 0 & w_{4,1} & 0 \\
0 & w_{2,2} & 0 & 0 & 0 \\
0 & w_{2,3} & w_{3,2} & 0 & 0 \\
w_{1,5} & w_{2,4} & w_{3,3} & w_{4,2} & w_{5,1} \\
0 & 0 & w_{3,4} & w_{4,3} & 0 \\
0 & 0 & 0 & w_{4,4} & 0 \\
0 & w_{2,5} & 0 & 0 & w_{5,2} \\
0 & 0 & w_{3,5} & 0 & w_{5,3} \\
0 & 0 & 0 & w_{4,5} & w_{5,4} \\
0 & 0 & 0 & 0 & w_{5,5}
\end{pmatrix}
\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \end{pmatrix} =
\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \\ w_{11} \\ w_{12} \\ w_{13} \end{pmatrix}, \tag{21}$$

which is solved using the QR factorization method for least squares.

Notice that both the pointwise SE kernel and the integrated SE kernel in Section 3.1 and Section 3.2 are both isotropic kernels. Hence, every $\mathbf{K}_{m,\sigma}$ and $\mathbf{C}_{m,\sigma}$ are identical over each substencil, illustrating that the WENO combination weights (i.e., $\mathbf{w}_m^T$) and GP model weights (i.e., $\mathbf{w}_*^T$ and $\mathbf{T}_*^T$) only need to be computed and saved once per level, and reused later.

We remark here that the size of the total GP stencil S for GP-WENO is larger than the stencil size of five in the case for the single-stencil GP calculation

for smooth, non-discontinuous flows in Fig. 3. We design the grid configuration of GP-WENO this way so that the size of each multi-substencil S_m is the same as the single-stencil size for the smooth GP case, as illustrated in Fig. 4. Given that the main mechanism of the nonlinear WENO scheme [28] is basically to select the least oscillatory solution at discontinuities among multiple candidate solutions represented from multiple substencils S_m, the accuracy of the GP-WENO prolongation on each five-point substencil S_m of radius one $R = 1$ is to be kept at least third-order when the local flow experiences sharp gradients. Away from discontinuities, the single-stencil GP with $R = 1$ described in Section 3.1 and Section 3.2 guarantees to prolong coarse solutions to finer cells at third-order.

3.5. Comparison with polynomial based prolongation

Let us comment on comparing the proposed GP-WENO prolongation with existing polynomial-based high-order prolongations, such as the work of McCorquodale & Colella [35] and Zhang *et al.* [60]. Multidimensional polynomial interpolation suffers from some well known complications, namely that not all stencils result in a well conditioned or solvable linear system. This issue may be alleviated by utilizing a tensor product stencil [51, 54] or solving a least squares problem with more than the minimum stencil points. The latter pathway is taken by [35, 60], requiring a new least squares solve for the prolongation polynomial coefficients at each coarse cell to be refined throughout a simulation. By contrast, the non-parametric form of the GP method we have presented allows for the prolongation to be expressed in terms of a set of pre-computed weights (Eqs. (3) and (12)) for the volume averages on the prolongation stencil. This offers a considerable performance advantage for the GP approach, taking full advantage of the structured AMR mesh. In addition to precomputing weights, the covariance kernel is isotropic since this application is with block structured AMR. As a result, the kernels for each substencil are identical; hence the same kernel is used for all substencils under consideration.

Additionally, in contrast to the use of slope-limiting in the traditional second-order linear prolongations or the nonlinear weighting of the present GP-WENO approach, the least squares approach in high-order prolongation [35, 60] does not leave a natural place to introduce any shock-controlling mechanism. As such, it is distinctive in our approach that the capability to make use of GP-WENO as a high-order prolongation allows the method to furnish essentially non-oscillatory prolongations.

3.6. Further tuning of GP-WENO for enhanced performance

The nonlinear weighting approach of GP-WENO discussed in the previous section has proven robust and accurate in treating discontinuities [45, 47]. Regardless, the nature of its non-linearity requires frequent calculations of nonlinear weights, which take place over the entire computational domain in practice, consequently consuming an extra computing time. As such, one can save the overall computation by applying the GP-WENO weighting only where needed,

i.e., to regions involving sharp gradients, identified by a shock-detector. In our GP formulation, we already have a good candidate for a shock-detector, that is, the GP-based β_m. To meet this, we slightly modify Eq. (18) to introduce an optional switching parameter α, defined by

$$\alpha = \frac{\sum_{i=1}^{2D+1} \frac{1}{\lambda_i} (\mathbf{v}_i^T \mathbf{f})^2}{\mathbb{E}^2_{arith}[\mathbf{f}] + \epsilon_2}. \tag{22}$$

Here, the data array $\mathbf{f}$ includes the $(2D+1)$ data solely chosen from the center-most substencil, e.g., S_1 in Fig. 4, $\mathbb{E}^2_{arith}$ is the squared arithmetic mean over the sampled coarse grid data points over $(2D+1)$ sized substencil centered at the cell $I_{i,j}$, or S_1, that is,

$$\mathbf{E}^2_{arith}[\mathbf{f}] = \left(\frac{1}{2D+1} \sum_{\mathbf{x} \in S_1} f(\mathbf{x}) \right)^2, \tag{23}$$

and finally, ϵ_2 is a safety parameter in case the substencil data values are all zeros. Notice that this is just a scaled version of the β_m in Eq. (18) for the central substencil S_1. Since the GP model is built with the smoothness of data in mind, prescribed by the smooth SE kernel, this parameter α will detect "unlikeliness" of the data $\mathbf{f}$ with respect to the GP model.

We choose a critical value, α_c, so that shocks and high variability in $\mathbf{f}$ are detected when $\alpha > \alpha_c$; smooth and low variability when $\alpha \leqslant \alpha_c$. We heuristically set $\alpha_c = 100$ in this strategy. Using this α parameter, we have a switching mechanism between the more expensive "nonlinear multi-substencil" GP-WENO method in Section 3.4 and the "linear single-stencil" GP model in Sections 3.1 and 3.2.

To illustrate, we show the α values associated with a Gaussian profile elevated by the circular cylinder of height 0.25 defined by the following simple function f in 2D,

$$f(x,y) = \begin{cases} 1 + \exp\left(-(x^2+y^2)\right), & \text{if } (x^2+y^2) < 0.5, \\ 0.25, & \text{else.} \end{cases} \tag{24}$$

In Fig. 5, we demonstrate how α varies over the profile, combining the smooth continuous profile with the abrupt discontinuity. We observed that the α value is close to 2 over the continuous region defined by $(x^2+y^2) < 0.5$. However, α soars to over 300 at the points corresponding to the sharp discontinuity, $(x^2+y^2) = 0.5$, resulting in the full engagement of the multi-substencil GP-WENO model near the discontinuity. In the rest of the smooth region, α becomes much smaller, which justifies well for the use of the linear GP model. This also tells us that the linear GP model would be a sufficient AMR prolongation algorithm for incompressible flow simulations.

If needed, the parameter α_c can be tuned to a different value to alleviate the GP performance relating to sensitivity to shock-detection. By lowering α_c,

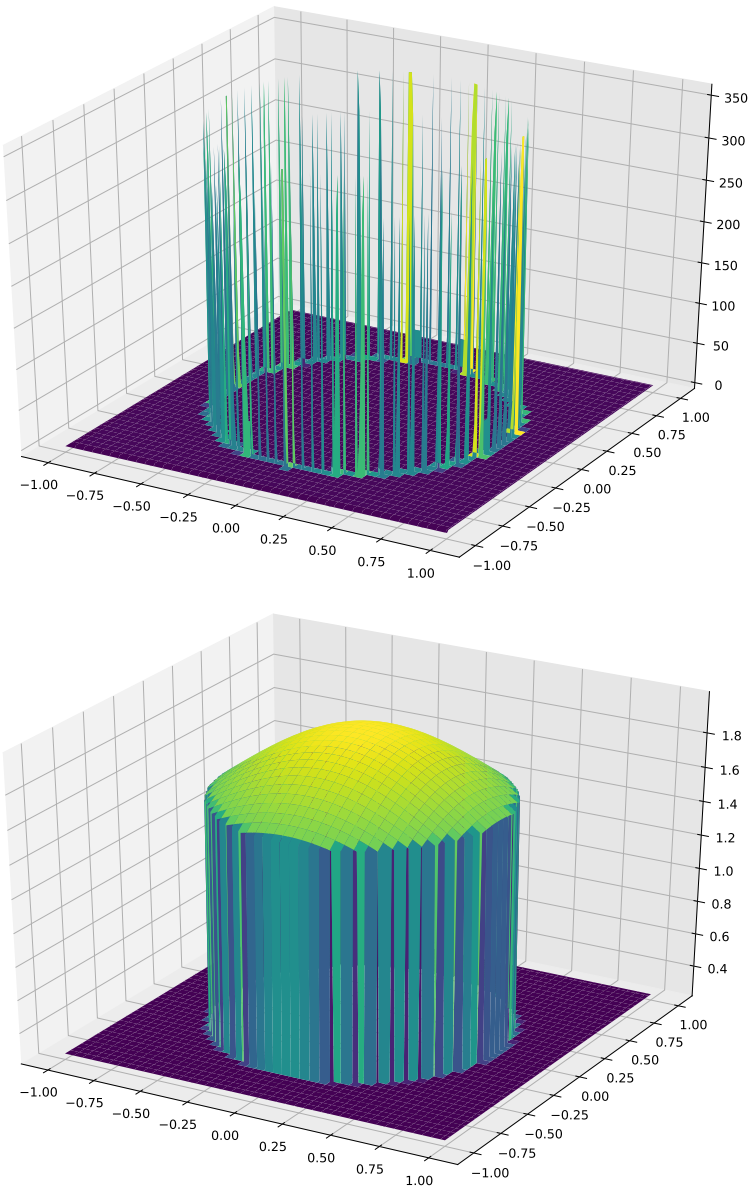


Figure 5: The top plot displays the α values associated with the data from the function $f(x, y)$ depicted in the bottom plot.

shocks will be detected more frequently, leading the overall computation to increase since GP-WENO will be activated on an increased number of cells. In most practical applications, such a tuning would be unnecessary considering that strong shocks are fairly localized. In such regions, α would retain a value much larger than α_c. Therefore, the condition $\alpha > \alpha_c$ for nonlinear GP-WENO would be met most likely over a wide span of possible values of α_c. Nonetheless, shocks' localized nature allows the computationally efficient linear GP model to be used in simulations that do not require the frequent shock handling mechanism. As such, we set $\alpha_c = 100$ in the numerical test cases presented in Section 5.2.

Note that determining a critical value of α will be based on the choice of GP covariance kernels. Without the normalization by the squared arithmetic mean, this factor will vary based on the mean value of the data. In this regard, dividing by the average value of the data, $\mathbf{f}$, helps to normalize the factor without changing the variability detection. From the statistical interpretation of the GP model $\mathbf{E}^2_{arith}[\mathbf{f}]$ may be viewed as a likelihood measure for a GP that assumes uncorrelated data (i.e., $\mathbf{K}_{ij} = \delta_{ij}$), so that α becomes normalized relative to the likelihood of another model.

In the multi-substencil GP-WENO method, there are generally $(2D+1)$ dot products of the stencil size for each prolonged point, $\prod_d r_d$. In patch-based AMR, even though refined grids are localized around the regions containing shocks and turbulence, there are often smooth flow areas in every patch. The switch α allows us to reduce the computational complexity to one dot product of the stencil size for each coarse stencil that has smooth data, therefore reducing the computational cost to one dot product of the stencil size for each prolonged point. This method is extremely useful in 3D, and when the refinement ratio is larger than two.

We conclude this section by making a remark on one significant feature of GP that we do not explore in our current study. The multi-substencil GP-WENO methods on smooth flows, outlined in [45, 47], can variably increase/decrease the order of accuracy. However, in the application for AMR prolongation, large regions of grids may be refined, so the increased computational cost can become undesirable. Note that the linear single-stencil GP interpolation is a third-order scheme, which means that it can serve sufficiently well as a high-order accurate prolongation that often matches the order of discrete solution accuracy in simulations. Reyes et al. [45, 47] discuss how to vary accuracy as a tunable parameter within the GP methodology. The studies show that the GP radius R of the stencil dictates the order of accuracy. The method illustrated in this paper utilizes a GP radius $R = 1$ and is $\mathcal{O}(\Delta x^3)$. However, if one uses $R = 2$ one can retrieve a method that is $\mathcal{O}(\Delta x^5)$.

4. Step-by-step implementation of the GP-AMR prolongation

We have implemented our new GP-AMR prolongation method in the AMReX framework. The AMReX framework utilizes a hybrid C++/Fortran library with many routines to support the complex algorithmic nature of patch-based AMR and state-of-the-art high-performance computing. As an example, the

object-oriented programming of C++ is fully utilized to furnish simple data and workflow. In AMReX, there is a virtual base class called *Interpolator*. This class has many derivations, including *CellConservativeLinear*, an object for the functions related to a cell-based conservative linear interpolation. The methods presented in the current work reside in the newly introduced *CellGaussian* class within the AMReX framework. This class constructs a GP object, which contains the model weights for each of the $\prod_{d=x,y,z} r_d$ new points per cell as member data. When a simulation is executed in parallelized format, each MPI rank has an Interpolator class, which helps to avoid unnecessary communication. Computationally, the order of execution is as follows. The data-independent calculations in Step 1 through Step 5 are computed only at the initialization of an AMR level and are saved for the duration of the simulation.

1. The refinement ratio and $\Delta\mathbf{x}$ are passed on to the construction of the GP object.
2. Build GP covariance matrices $\mathbf{K}$ and $\mathbf{K}_m$ for the single-stencil linear GP prolongation; $\mathbf{K}_{m,\sigma}$ for the multi-substencil nonlinear GP-WENO (similarly for $\mathbf{C}, \mathbf{C}_m$, and $\mathbf{C}_{m,\sigma}$ for cell-averaged data) using the SE kernel in Eq. (A.6) for pointwise prolongation or Eq. (6) for volume-averaged prolongation.
   - $\mathbf{K}$ and $\mathbf{K}_m$ (also used for the single-stencil linear GP) are specifically used for prolongation and should be used with the hyperparameter ℓ. The size of ℓ should be on the order of the size of the GP stencil to match our model assumption that the data varies smoothly over the stencil. We typically adopt $\ell = 12\Delta$, where $\Delta = \min_{d=x,y,z}\{\Delta x_d\}$.
   - $\mathbf{K}_{m,\sigma}$ is used for shock detection in the nonlinear multi-substencil GP-WENO via the smoothness indicators β_m. These covariance kernels take the parameter σ as the characteristic length scale. In practice, we set $1.5\Delta \leqslant \sigma \leqslant 3\Delta$ corresponding to the typical shock width in high-order Godunov method simulations.
3. Calculate the GP weights $\mathbf{w}_*$ for pointwise data using Eq. (3) (or $\mathbf{z}_*$ for volume-averaged data using Eq. (12) for all $\prod_d r_d$ prolonged points. These weights in Step 3 are calculated only once for every possible refinement ratio and stored for use throughout the simulation before each simulation begins. The GP-AMR prolongations themselves in Eq. (3) or Eq. (12) are data-dependent, requiring updated calculations when data $\mathbf{f}_i$ or $\mathbf{G}_i$ are updated.
4. Compute the eigensystem of $\mathbf{K}_{m,\sigma}$ as part of building the shock-capturing GP-WENO model. The eigenvectors $\mathbf{v}_i/\sqrt{\lambda_i}$ are stored for re-use in calculating β_m and α during simulations.
5. Solve for the linear weights $\boldsymbol{\gamma}$ for each prolonged point using the weights from S_m and S, as given in Eq. (19) or Eq. (21). The linear weights $\boldsymbol{\gamma}$ are only calculated once and stored for each possible refinement ratio.
6. The switch model parameter α is data-dependent and hence is calculated for each coarse cell $I_{i,j,k}$. The cell-by-cell α values are compared to the critical α_c, for which we choose $\alpha_c = 100$ in this paper.

- When $\alpha < \alpha_c$ the data is determined to be smooth enough, and hence we do not to need the full nonlinear multi-substencil GP-WENO prolongation in Section 3.4. Instead, the GP-weights $\mathbf{w}_{1,*}$ (or $\mathbf{z}_{1,*}$) on S_1 can be used without any nonlinear weighting to produce GP prolonged data on newly created fine cells as in Eq. (3) or Eq. (12), the process of which utilizes only the linear single-stencil GP in Section 3.1 or Section 3.2.
- In the case that $\alpha > \alpha_c$, the points are prolonged using the full nonlinear multi-substencil GP-WENO model (e.g., one of the methods in Sections 3.1 and 3.2, plus the nonlinear controls in Section 3.4).

5. Results

In this section, we present the performance of the new third-order GP-based prolongation model using $R = 1$ on a selected set of numerical test problems. The GP results are compared with those obtained by the default conservative second-order linear polynomial scheme in AMReX. We aim to demonstrate that the GP prolongation method can be seamlessly integrated with an existing FVM code framework, delivering the added third-order solution accuracy without increasing extra computational overheads compared with the second-order linear method. To illustrate the utility of the new GP-based prolongation scheme in fluid dynamics simulations, we integrated the prolongation method into AMReX that can be used by a number of AMReX application codes. More specifically, we have utilized two application codes for the results in this section, including Castro [2], a massively parallel, AMR, compressible astrophysics simulation code, and a simple advection tutorial code built in AMReX.

5.1. Accuracy

To test GP-AMR's order of accuracy, we first deploy a simple Gaussian profile refined with the GP prolongation method. The profile is initialized using the formula

$$f(\mathbf{x}) = \exp(-||\mathbf{x}||_2^2), \tag{25}$$

where $\mathbf{x} \in [-2, 2] \times [-2, 2]$. In Fig. 6, we compare the prolonged solution, denoted as $f_p(\mathbf{x})$, against the analytical value, $f(\mathbf{x})$, associated with the Gaussian profile function. We find that the third-order accuracy of the cell-averaged GP prolongation routine matches the analysis in [45, 47]. The convergence rate of the error in L_1 norm, $E = ||f - f_p||_1$, computed using the GP prolongation model with $R = 1$, exhibits the expected third-order accuracy, following the theoretical slope of third-order convergence on the grid scales, $\mathcal{O}(\Delta x^3)$. Also shown in comparison is the second-order convergence of the linear polynomial prolongation method. As seen in Fig. 6, the third-order accurate GP prolongation delivers a faster convergence than the conventional second-order linear prolongation as the computational grid further refines to finer scales, underlining the sheer value of designing a higher-order scheme in CFD simulations.

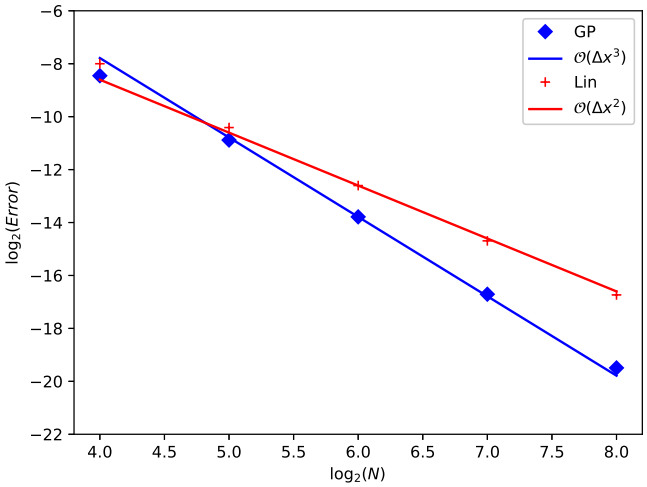


Figure 6: Grid convergence rates of the GP-prolongation method and the linear polynomial method. The quantities are measured in the log of base two to better cope with the two refinement jump ratio. The grid resolutions $N \times N$ used for this test include $N = 16, 32, 64, 128,$ and 256.

5.2. GP-AMR tests

We present three tests to illustrate the efficacy of our new AMR prolongation method. The first problem is a single vortex advection, provided as the *Advection_AmrLevel* tutorial in AMReX. For the second test problem, we present a modified version of the slotted cylinder problem from [13]. In the last problem, we use the Castro astrophysical simulation suite [2] to perform the Sedov implosion test [49]. In all tests we use $\ell = 12 \cdot \min_{d=x,y}\{\Delta x_d\}$. For σ, we use $\sigma = 3 \cdot \min_{d=x,y}\{\Delta x_d\}$ for 2D cases and $\sigma = 1.5 \cdot \min_{d=x,y}\{\Delta x_d\}$ for 3D cases.

5.2.1. Single vortex advection using the AMReX tutorial

The first test is a simple reversible vortex advection run. A radial profile is morphed into a vortex and reversed back into its original shape. This stresses the AMR prolongation's ability to recover the original profile after it has been advected into the coarse cells, focusing on how well the numerical solution can retain its original shape upon its return to the initial location at the final time.

The radial profile initially is defined by

$$f(x, y) = 1 + \exp\Big[-100\left((x - 0.5)^2 + (y - 0.75)^2\right)\Big]. \tag{26}$$

The profile is advected with the following velocity field,

$$\mathbf{v}(x, y, t) = \nabla \times \psi, \tag{27}$$

which is the curl of the stream function,

$$\psi(x, y, t) = \frac{1}{\pi} \sin(\pi x)^2 \sin(\pi y)^2 \cos\left(\frac{\pi t}{2}\right), \tag{28}$$

where $(x, y) \in [0, 1] \times [0, 1]$.

In this demonstration, the level 0 grid size is 64×64, and has two additional levels of refinement (i.e., the level 1 and 2 grids with grid spacing corresponding to 128×128 and 256×256 grids respectively) surrounding the radial profile. The simulation is an incompressible advection problem using the so-called Mac-Projection to compute the incompressibility condition that enforces the divergence-free velocity fields numerically, $\nabla \cdot \mathbf{v} = 0$ [3]. The flux is calculated by a simple second-order accurate upwind linear reconstruction method. Although the overall solution is second-order that is lower than the third-order accuracy of the GP prolongation method, this example still illustrates the computational performance of the GP-prolongation method over the default conservative second-order prolongation.

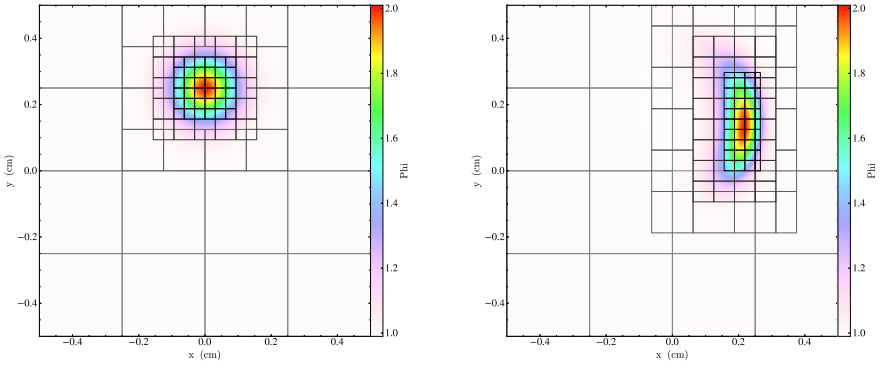
The simulation is finished at $t = 2$. We used sub-cycling of time-steps to improve the overall performance, in which a smaller time-step Δt_f is used on a finer level to advance the regional solutions for stability. The coarser level solutions that advance with a larger time-step Δt_c await until the solutions on the finer levels catch up with the global simulation time $t_g = t^n + \Delta t_c$ over the number of sub-cycling steps $N_{\text{subcycle}} = \Delta t_c / \Delta t_f$.

Table 1: Accuracy and performance of GP-AMR against the default linear AMR for the single vortex test on a workstation with an Intel i7-8700K processor, with 6 MPI ranks.

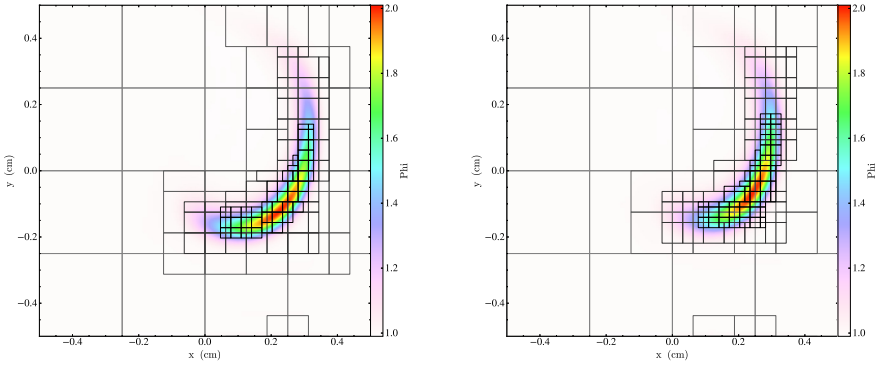
	Execution time	Prolongation time	Number of calls	L_1 error
2D GP-AMR	0.2323s	0.004168s	9115	0.00033
2D Linear	0.2335s	0.008436s	9113	0.00071
3D GP-AMR	1.6523s	0.086361s	21929	0.00151
3D Linear	1.6640s	0.157623s	21893	0.00160

We present the performance and accuracy results for this problem in Table 1. Since the two prolongation methods are of a different order, they can yield different AMR level patterns that can lead to a slight difference in the number of function calls. Regardless, we see that in the fourth column, the numbers of function calls of GP-AMR and the linear method are almost equivalent in both 2D and 3D runs.

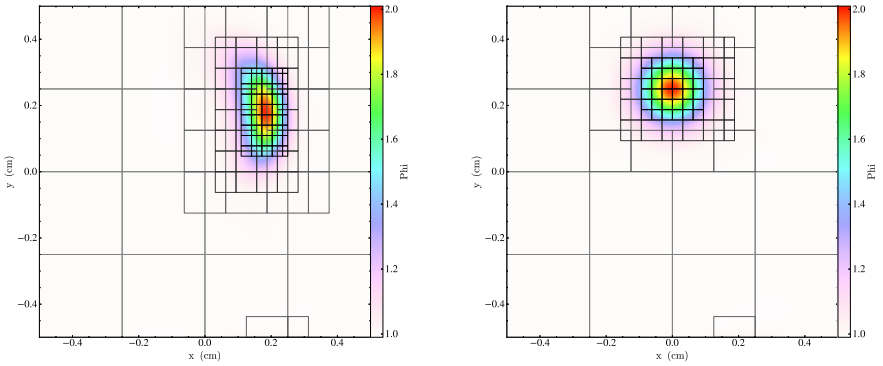
In the third column's prolongation time, we find that the default linear prolongation took approximately twice more time than the GP prolongation. This is due to the smoothness of the solution profile, which does not require the nonlinear multi-substencil GP-WENO treatment in GP-AMR, allowing for the simpler, linear GP algorithm to be used. However, the overall execution times in the second column were equally comparable since there were larger areas (or more cells) that followed the profile and were computed in the finest AMR level in the GP case than in the linear case. This illustrates that the GP-AMR



(a) $t = 0$ (b) $t = 0.282$



(c) $t = 0.651$ (d) $t = 1.447$



(e) $t = 1.785$ (f) $t = 2$

Figure 7: The progression of the 2D radial profile with four levels of refinement using the multi-substencil GP prolongation algorithm. The refinement criterion is set to track the region whose value is larger than 1.5.

algorithm is much less numerically diffusive than the default linear prolongation method. Since the refinement criteria was based on the value of advected profile, the less numerically diffusive algorithm will have larger portions of the domain at higher AMR levels as the solution evolves.

We note that the cost of computing the GP model weights is 0.0002306 seconds on average in our runs and is negligible in comparison to the overall execution time. This metric illustrates that the pre-computation of the GP weights is very inexpensive, being called twice (since there were two levels) per MPI rank.

In the last column in Table 1, we also report the L_1 errors between the solution at $t = 2$ and the solution at $t = 0$ for both AMR prolongation methods. In the 2D case, we find that the GP-AMR solution is approximately half of the default linear method's error. This highlights the utility of a high-order prolongation method, as smooth features are better recovered after being advected into coarser cells.

Another useful examination is the analogous problem in 3D, in which the computational stencils for both methods grow. For the 3D version, we use a $32 \times 32 \times 32$ base grid with two higher levels of refinement. The details of this simulation can also be found in Table 1. We note that a parallel copy operation becomes slightly more expensive with GP because the need for the GP multi-substencil grows on non-smooth regions to handle discontinuities in a stable manner, as managed by the α_c parameter. This becomes more apparent in 3D, as the computational stencil effectively grows from 7 cells with the single-stencil linear GP to 25 cells with the multi-substencil nonlinear GP-WENO approach. In this 3D benchmark, the difference in error between these methods is smaller than in the 2D case. The GP-AMR simulation still outperforms the simulation with the linear prolongation method but to a smaller degree. The 3D simulation has a base grid coarser than that of the 2D simulation by a factor of two, resulting in a coarser finest grid in the simulation. This is why the L_1 errors are greater in the 3D case.

In Fig. 7, we show the time-dependent evolutions of the 2D single vortex advection with GP-AMR on a base grid of 64×64 with four refinement levels to illustrate the GP method with a more production-level grid configuration.

5.2.2. Slotted cylinder as another AMReX test

Another useful test is the slotted cylinder advection presented in [13]. In this paper, we do not use an exact replica of this problem, but instead, we utilize the initial profile and perform a similar transformation as in the previous problem. That is, the slotted cylinder is morphed using the same velocity used in the single vortex test in Section 5.2.1. Because the advected profile is now piecewise constant, in contrast to the smooth profile in the previous problem, it will require using the multi-substencil nonlinear GP-WENO approach. This will show how the GP-based smoothness indicators prolong non-smooth data stably.

The 2D slotted cylinder is defined as a circle of radius $R = 0.15$ centered at $\mathbf{x}_c = (x_c, y_c) = (0.5, 0.75)$ with a slot of width $W = 0.05$ and height $H = 0.25$

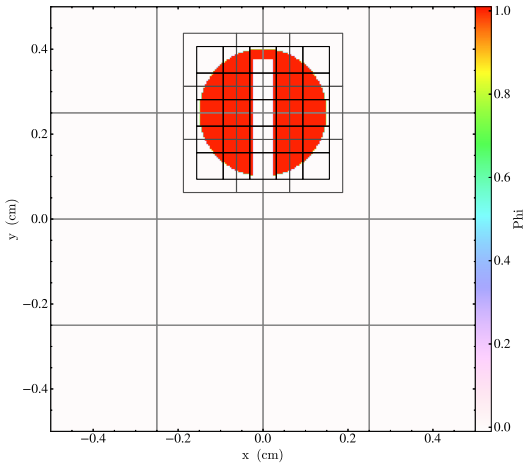


Figure 8: The slotted cylinder at $t = 0$ over the entire domain with three levels of refinement.

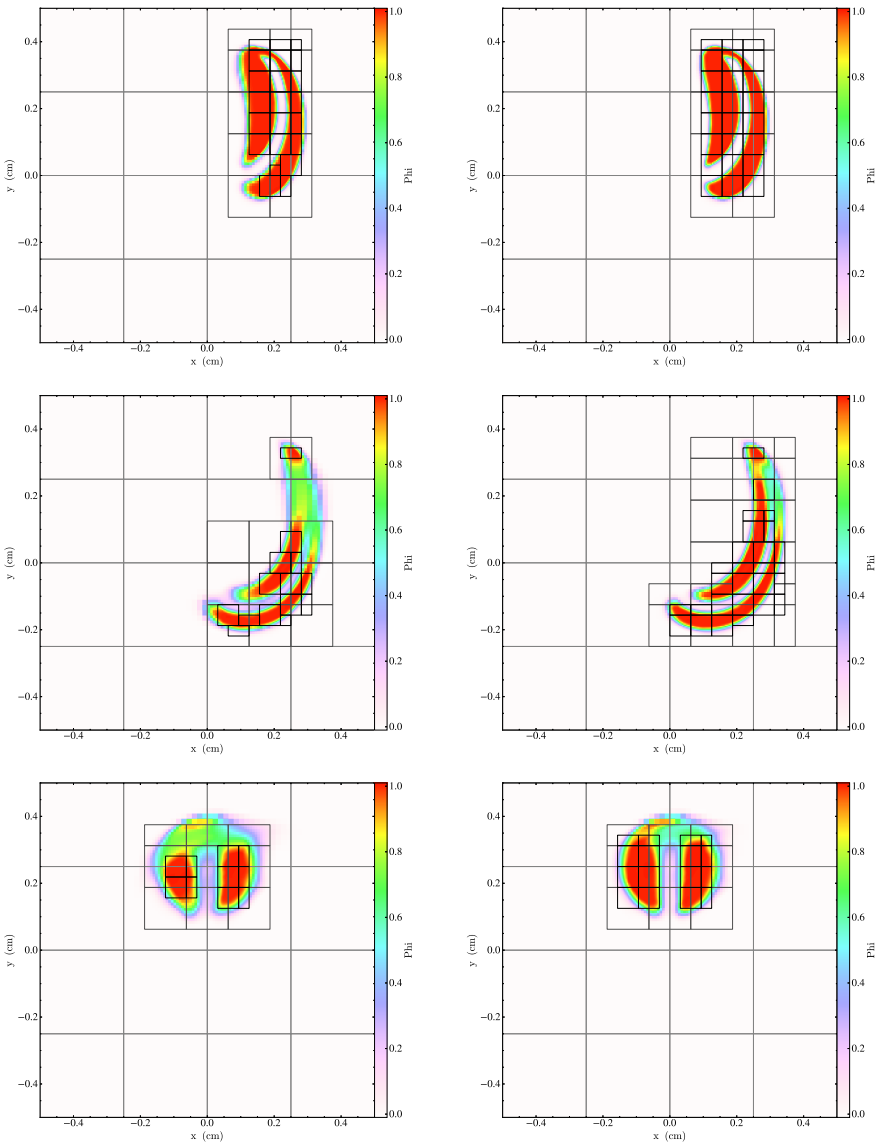
removed from the center of the cylinder. The initial condition is given by,

$$\phi_0(\mathbf{x}) = \begin{cases} 0, & \text{if } R < \sqrt{(x - x_c)^2 + (y - y_c)^2}, \\ 0, & \text{if } |2x_c| < W \text{ and } 0 < y_c + R < H, \\ 1, & \text{else,} \end{cases}$$

where $(x, y) \in [0, 1] \times [0, 1]$. The initial profile is shown in Fig. 8.

In this case, we wish to find the simulation that best retains the profile of the initial condition when it is completed at $t = 2$ as in the previous test. We have two levels of refinement on a base grid of size 64×64 resolution. Fig. 9 contains snapshots of the simulations at times $t = 0.28, 1.44$, and 2. The goal is to retain the initial condition as much as possible, in a similar fashion to the previous 2D vortex advection test.

The result shows that the multi-substencil GP-AMR prolongation preserves the initial condition better than the default conservative linear scheme native to AMReX. Notably, there is far less numerical smearing of the final slotted cylinder at $t = 2$, and the circular nature of the cylinder is better retained with GP-AMR. Furthermore, it should be noted in the AMR patterns at $t = 1.44$ and 2 that a larger area of the slotted cylinder is covered by the finest grid structure with the GP-AMR prolongation. The refinement criterion in this test is set to track the region whose value is larger than 0.99. This is analogous to refining on regions of high density or pressure. Since we kept both the PDE solver and the AMR restriction the same in both cases, the only difference is in the different levels of numerical dissipation within the two AMR prolongation methods. We intentionally set this threshold value 99% close to the initial cylinder value of unity to rapidly reveal the numerical diffusivity of the two AMR prolongation



(a) Linear polynomial prolongation

(b) GP-WENO prolongation

Figure 9: The morphed slotted cylinder problem at times $t = 0.28, 1.44$, and 2, from top to bottom in time. Left: (a) AMReX with the second-order linear polynomial prolongation. Right: (b) AMReX with the third-order multi-substencil GP-WENO prolongation.

methods. This allows us to see how well each method is able to track the critical part of the cylinder. We wish to trace the slotted cylinder's evolution with the finest grid as much as possible for improved solution accuracy. In this way, we can directly assess each method's numerical diffusivity by monitoring how much each method retains the finest grid along with the profile at each evolution step. We see that the default linear prolongation experiences much larger numerical diffusivity than GP-AMR.

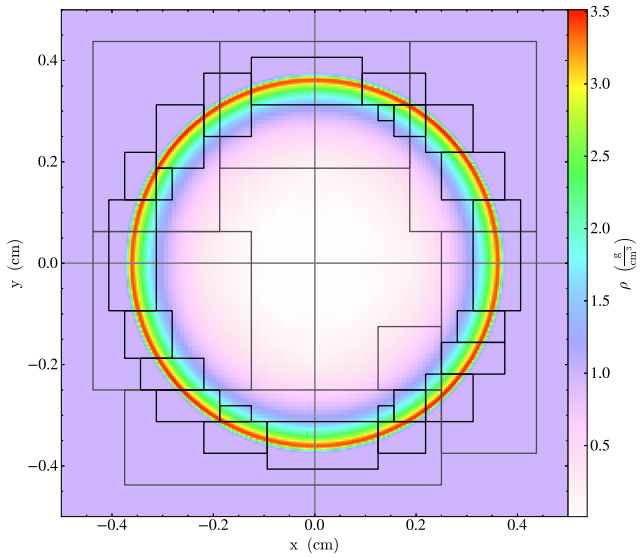
In each panel of the left column in Fig. 9, the area covered by the finest patch resolution is smaller in the linear AMR solution than the corresponding solutions solved by the GP-AMR prolongation on the right column. This behavior can be explained in terms of the numerical diffusivity in each AMR prolongation method. As the initial slotted cylinder morphs into a crescent shape and returns to the original state, the numerical solution repeats its discrete evolutions in two steps: the solution updates of the underlying PDE solver at a given AMR level, followed by the AMR prolongation and restriction to newly configured grid cells according to the AMR's refinement and derefinement criteria. We track the cylinder's critical values larger than 0.99 in this case. Thus, the area covered by the finest patch level is a direct consequence of the second-order and the third-order prolongation algorithms in this comparison. Since we set the refinement criteria as high as 99% to track the initial cylinder profile, the area covered by the finest patch manifests how well the cylinder is dynamically evolved, while maintaining its values close to the original value of unity (colored in red). The more the cylindrical shape gets smeared due to each AMR prolongation's numerical dissipation, the smaller the area will be tagged for further refinements.

The difference is evident in Fig. 9. The larger area of the slotted cylinder with the GP-AMR prolongation remains close to the initial value of unity (the red regions) during the evolution from the initial shape ($t = 0$) to the crescent shape ($t = 1.44$), and finally returning to the initial shape ($t = 2$). The linear method results in a far more blurred cylinder at $t = 2$. While there is some loss with GP-AMR, the profile at $t = 2$ far better resembles the original cylinder at the onset of the simulation, exhibiting the computational advantage of GP-AMR over the linear method, even when paired with a second order algorithm.

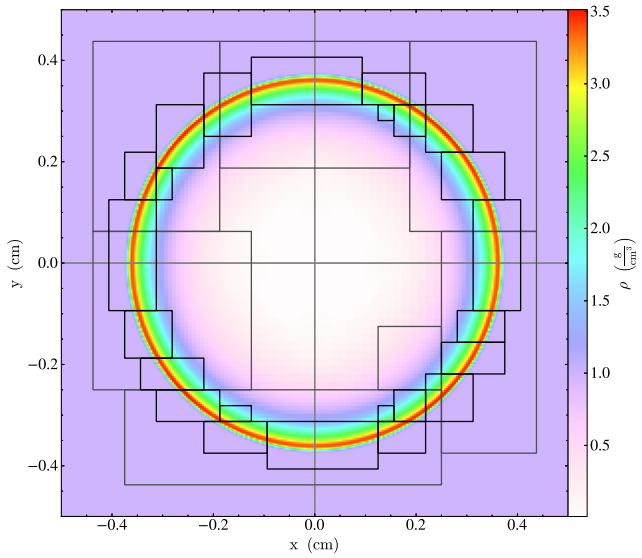
5.2.3. Sedov blast wave using Castro

A perhaps more useful test of the GP-AMR algorithm is a compressible flow condition, where shock-handling becomes necessary. To illustrate the performance of the GP-AMR method in a compressible flow regime, we utilize the Sedov blast wave problem [49] that simulates the evolution of a radially expanding strong pressure wave. This simulation is performed using Castro with the choice of the piecewise parabolic method (PPM) [16] for reconstruction along with the Colella and Glaz Riemann solver [17]. For the 2D test, we have a base grid of 64×64, padded with two additional AMR levels using a $r_x = r_y = 2$.

The results in Fig. 10 illustrate the density propagation of the Sedov blast wave at $t = 0.1$. They demonstrate that the GP solution compares equally well with the solution using the linear prolongation method on a problem whose flow condition evolves into a highly compressible regime. Although simple, the Sedov



(a) Density profile using GP-AMR.



(b) Density profile using the linear AMR.

Figure 10: A comparison of the density profiles of the Sedov blast wave solution at $t = 0.1$ with two levels of refinement.

blast wave is a good test illustrating the shock-handling capabilities of the GP multi-substencil model. Notice, in Fig. 10, that visually, the radial shockwaves

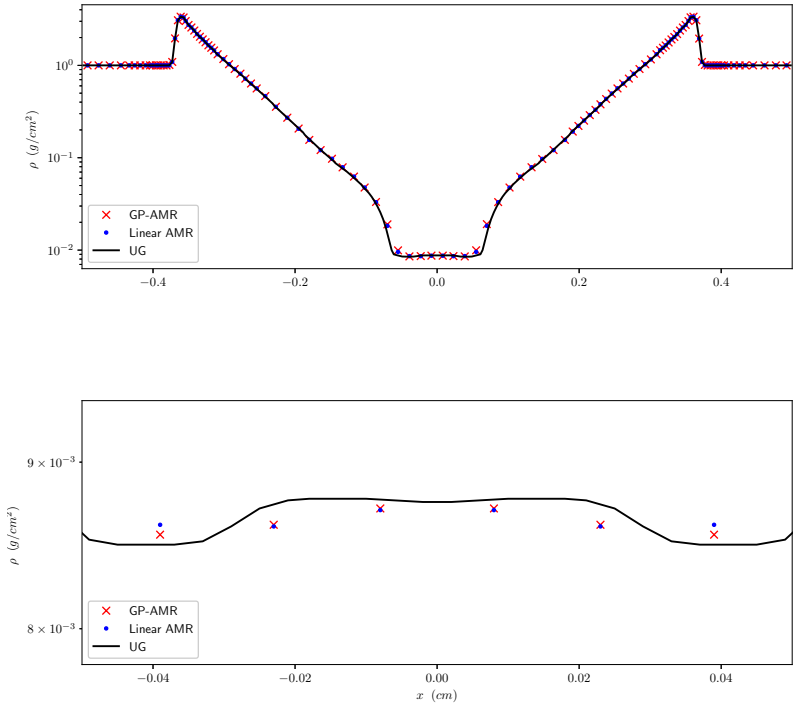


Figure 11: Top: Horizontal cuts over $-0.5 \leqslant x \leqslant 0.5$ holding $y = 0$ of the density profiles of the Sedov test at $t = 0.1$, computed using the GP-AMR prolongation and the linear polynomial prolongation. Also overplotted in the solid curves is the solution computed using the uniform grid solution on a 256×256 grid resolution, analogous to the grid resolution at the highest refinement levels in the AMR solutions. The y-axis is log scaled, while the density values and the x-axis remain unscaled. Bottom: The panel displays a zoomed-in view around the center of the blast wave, $-0.05 \leqslant x \leqslant 0.05$ holding $y = 0$, to make the comparison visually more distinctive at a finer scale. We see that the GP-AMR solution remains closer to the reference uniform grid solution than the linear AMR solution in the central low-density region, demonstrating slight performance improvements.

Table 2: Performance of the GP-AMR prolongation against the second-order linear prolongation on the Sedov blast wave with 6 MPI ranks on an Intel i7-8700K processor.

	Execution time	Prolongation time	Number of calls
2D GP-AMR	5.719s	0.07691s	19743
2D Linear	5.698s	0.12610s	19439
3D GP-AMR	64.27s	1.28912s	35202
3D Linear	67.76s	1.96204s	35202
3D Non-Adapitve GP-AMR	72.81s	5.09467s	35202

in both simulations appear identical. To make the comparison more accessible, in Fig. 11, we illustrate a horizontal cross-section through the center of the 2D density profiles in Fig. 10, along with a uniform grid solution on a 256×256 resolution that matches with the resolution of the AMR simulations at the highest refinement level. On inspection of the zoomed-in figure on the bottom, we see that the GP-AMR solution matches the uniform grid solution closer than the linear method's solution. The uniform solution is depicted by a black curve, the linear solution is represented by the blue dots, and the GP-AMR solution is depicted by the red crosses. In Fig. 10, the AMR levels track the shock as it propagates radially, and the shock front is contained at the most refined level. The shock is handled by the multi-substencil nonlinear GP-WENO treatment at the most refined level, which increases the computational complexity in this region. Regardless, the GP algorithm is still less expensive than the linear method because the shock is very well localized; the majority of the domain is computed by the comparatively simpler, single-stencil linear GP model. The linear GP model is no more than a collection of operations involving a series of simple dot-products, utilizing the *pre-computed* weights. This computational benefit of GP appears in the overall computational performance statistics, evidenced in the half-reduced prolongation time on the 2D runs (i.e., 0.07691s for GP vs. 0.12610s for linear), as shown in the third column in Table 2.

However, we see that the entire execution time in the second column is slightly higher with GP-AMR since the GP-AMR prolongation routine was invoked more frequently than the linear prolongation routine over the entire simulation time, due to a larger region (more cells) being tagged for refinements in the GP-AMR run during the simulation, as evidenced by the greater number of calls to the prolongation routine in the third column. The results are obtained using the same workstation as the previous test.

A 3D Sedov blast was also tested, giving us a better look at the multi-substencil GP-WENO cost in the shock regions. For this benchmark, the simulation utilized a base grid of $32 \times 32 \times 32$ with two additional AMR refinement levels, utilizing a refinement factor of two for both levels. The wave was advected until $t = 0.01$ with both simulations (the GP-AMR and the linear prolongations). The performance metrics for the 3D test are also shown in Table 2.

Lastly, by setting $\alpha_c = 0$, we effectively have the multi-substencil nonlinear GP-WENO method engaged on all cells. The metrics for this example

are labeled as "Non-Adaptive GP-AMR" in Table 2. Using the GP-WENO method fully on all computational cells roughly increases GP's computational performance by five times, making it more expensive as a prolongation method. This performance increase is expected because the multi-substencil GP-WENO method combines five sub-GP models, each of which is computed on each sub-stencil S_m, $m = 1, \ldots, 5$.

6. Conclusion

In this paper, we developed an efficient, third-order accurate, AMR prolongation method based on Gaussian Process Modeling. This method is flexible to the type of data being interpolated, as illustrated with a substitution of covariance kernels in Eqs. A.6 and 6. To handle shock waves, a multi-substencil GP-WENO algorithm inspired by WENO [52] was employed based on the previous studies on the GP high-order methods [45, 47]. We recognize that GP-WENO is more computationally expensive than the default linear prolongation method. The α tagging approach in Eq. (22) was proposed as a way to mitigate this situation. This approach uses a normalized smoothness indicator, β_m, furnished from GP to detect regions containing shocks or non-smooth flows.

The purpose of timing the simulations was to illustrate that the GP-AMR based applications were just as computationally performant as instances with the simple linear prolongation. Overall, the GP-AMR method is a balance between speed, stability, and accuracy.

In the scope of this paper, the tunable parameters ℓ and σ are either fixed in relation to the grid-scale Δx_d, or fixed as constant. To further adapt the algorithm, one could try and maximize the log of Eq. A.2 with respect to the hyperparameter ℓ as is done in many applications utilizing Gaussian Process regression (e.g., see Appendix B of [45]). However, in our application, a fixed prescription for ℓ appears to hold the properties we desired.

The algorithm's stability is inherently tied to the σ parameter, which we recommend being no larger than three times the grid-scale (Δx_d in practice). If additional stability is required, we recommend tuning α_c to be smaller or to be zero, requiring the algorithm to fully engage the multi-substencil GP model on all computational cells. This setup gains extra numerical stability at the cost of increased computational loads.

Finally, we remark that an even higher-order GP prolongation method can be obtained by increasing the size of the GP stencil radius, R, from the current value 1 to a larger value. Unlike the typical polynomial-based schemes, our GP algorithm allows this flexible order variation *within* the single algorithmic GP framework. This will be inherently useful as more simulation codes are moving to increasingly accurate methods, yielding fourth or higher-order accurate solutions in delivering high fidelity in predictive science simulations. In this case, the conventional second-order AMR interpolation may degrade the overall quality of the solution.

7. Acknowledgements

This work was supported in part by the National Science Foundation under grant AST-1908834. We acknowledge that the current work has come to fruition with help from the Center for Computational Science and Engineering at Lawrence Berkeley National Laboratory, the home of AMReX. We thank Dr. Ann S. Almgren, Dr. Weiqun Zhang, and Dr. Marcus Day for their insight and advice when completing this research. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. The first and second authors also acknowledge the use of the Lux supercomputer at UC Santa Cruz, funded by NSF MRI grant AST-1828315.

Appendix A. A brief introduction to Gaussian Process Modeling for CFD

For this paper to be self-contained, we give a brief overview of constructing a GP model in this appendix. These step-by-step implementations of the GP-prolongation method are summarized in Section 4.

Gaussian Processes are a family of stochastic processes in which any finite collection of random variables sampled from this process is jointly and normally distributed. In a more general sense, GPs take samples of functions from an infinite-dimensional function space. In this way, the AMR prolongation routine described in detail in Section 4 will be drawn from a data-informed distribution space trained on the coarse grid data. Interested readers are further referred to [43, 9] for more general introductions to GP modeling in general.

Appendix A.1. A statistical introduction to Gaussian Processes

The construction of the posterior probability distribution over the function space is the heart of GP modeling. To construct a GP, one needs to specify a *prior* probability distribution for the function space. This can be done by specifying two functions, a *prior mean function* and a *prior covariance kernel function* (see more details below), by which a GP is fully defined. Samples, namely function values evaluated at known locations, drawn from the GP prior are then used to further update this prior probability distribution. As a consequence, a *posterior* probability distribution, including a new updated posterior mean function and a new updated posterior covariance kernel function, is generated as a combination of the newly updated prior along with these samples, by means of Bayes' Theorem.

Once constructed, one can draw functions from this data-adjusted GP *posterior* space to generate a model for prolongation in AMR, or a model for more general purposes such as reconstruction, regression, or classification. Specifically, the GP posterior mean function could probabilistically predict function values at any arbitrary point where the function has not been previously sampled. In [47, 45], Reyes *et al.* have utilized this *posterior mean function as a*

high-order predictor to introduce a new class of high-order reconstruction/interpolation algorithms for solving systems of hyperbolic equations.

Similarly, from the perspective of designing a probabilistically driven prediction of function values, the posterior mean function becomes an AMR prolongator that delivers a high-order accurate approximation at the desired location in a computational domain resolved by AMR's hierarchical grid structures.

As briefly mentioned, GPs can be fully defined by two functions:

- a mean function $\bar{f}(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$, and
- a covariance function which is a symmetric, positive-definite kernel $K(\mathbf{x}, \mathbf{y}) : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$.

Notationally, we write $f \sim \mathcal{GP}(\bar{f}, K)$ to denote that functions f have been distributed in accordance with the mean function $\bar{f}(\mathbf{x})$ and the covariance $K(\mathbf{x}, \mathbf{y})$ of the GP prior. Analogous to finite-dimensional distributions, we write the covariance as

$$K(\mathbf{x}, \mathbf{y}) = \mathbb{E}\left[\big(f(\mathbf{x}) - \bar{f}(\mathbf{x})\big)\,\big(f(\mathbf{y}) - \bar{f}(\mathbf{y})\big)\right], \tag{A.1}$$

where $\mathbb{E}$ is with respect to the GP distribution.

One controls the GP by specifying both $\bar{f}(\mathbf{x})$ and $K(\mathbf{x}, \mathbf{y})$, typically as some functions parameterized by the so-called hyperparameters. These hyperparameters allow us to give the "character" (i.e., length scales, differentiability, or regularity) of functions generated by the posterior, which will define the underlying pattern of predictions using the posterior GP model. Suppose we have a given GP and N locations, $\mathbf{x}_n \in \mathbb{R}^D$, where $D = 1, 2$, or 3, and $n = 1, \ldots, N$. For samples $f(\mathbf{x}_n)$ collected at those points, we can calculate the likelihood $\mathcal{L}$, viz., the probability of the data $f(\mathbf{x}_n)$ given the GP model. Let us denote the data array in a compact form, $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^T$. The likelihood $\mathcal{L}$ of $\mathbf{f}$ is given by

$$\mathcal{L} \equiv P\big(\mathbf{f}|\mathcal{GP}(\bar{f}, K)\big) = (2\pi)^{-N/2} \det|\mathbf{K}|^{-1/2} \exp\left[-\frac{1}{2}\left(\mathbf{f} - \bar{\mathbf{f}}\right)\mathbf{K}\left(\mathbf{f} - \bar{\mathbf{f}}\right)\right], \tag{A.2}$$

where $\mathbf{K}$ is a matrix generated by $K_{n,m} = K(\mathbf{x}_n, \mathbf{x}_m)$, $n, m = 1, \ldots, N$, and the mean $\bar{\mathbf{f}} = [\bar{f}(\mathbf{x}_1), \cdots \bar{f}(\mathbf{x}_N)]^T$. Since these samples (or functions) are probabilistically distributed according to the GP prior, i.e., $f \sim \mathcal{GP}(\bar{f}, K)$, we now can make a probabilistic statement about the value of any agnostic function f in the GP at a new point $\mathbf{x}_*$, at which we do not know the exact function value, $f(\mathbf{x}_*)$. In other words, the GP model enables us to predict the value of $f(\mathbf{x}_*)$ probabilistically based on the character of likely functions given in the GP model prior.

For AMR, this is especially important as we need to construct data at a finer resolution where we do not know the data values at newly generated grid locations refined from a parent coarse level.

Finally, an application of Bayes' Theorem directly onto the joint Gaussian prior, along with the conditioning property, gives what we desire, namely, the

updated (or data-informed) new posterior distribution of the predicted value f_* conditioned on the prior observations $\mathbf{f}$,

$$P(f_*|\mathbf{f}) = (2\pi U^2)^{-1/2} \exp\left[-\frac{(f_* - \tilde{f}_*)^2}{2U^2}\right], \tag{A.3}$$

where the first main ingredient is the posterior mean $\tilde{f}_*$ given as

$$\tilde{f}_* \equiv \bar{f}(\mathbf{x}_*) + \mathbf{k}_*^T \mathbf{K}^{-1}(\mathbf{f} - \bar{\mathbf{f}}), \tag{A.4}$$

and the second ingredient is the *posterior covariance function* given as

$$U^2 \equiv k_{**} - \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{k}_*. \tag{A.5}$$

The posterior probability given in Eq. (A.3) is maximized by the choice $f_* = \tilde{f}_*$, leading to Eq. (A.4) being taken as *the GP prediction for the unknown* $f(\mathbf{x}_*)$. Meanwhile, the *posterior* covariance in Eq. (A.5) reflects *the GP model's confidence in the prediction* in Eq. (A.4) for the function at $\mathbf{x}_*$.

In this paper, we focus on the posterior mean function in Eq. (A.4), which will become the basis for our interpolation in the GP-based AMR prolongation.

Appendix A.2. Choice of GP kernels, the kernels' hyperparameters, and the GP prior mean function

There are covariance kernel functions available for GP modeling [43, 9]. One of the most widely used kernels in GP modeling is the squared-exponential (SE) covariance kernel function,

$$K(\mathbf{x}, \mathbf{y}) \equiv \Sigma^2 \exp\left[-\frac{(\mathbf{x} - \mathbf{y})^2}{2\ell^2}\right]. \tag{A.6}$$

The SE kernel is infinitely differentiable and, as a consequence, will sample functions that are equally smooth. The kernel contains two model hyperparameters Σ and ℓ. The scaling constant Σ acts as an overall constant factor that has no impact on the posterior mean [‡], which is the key ingredient of our GP-AMR prolongation, and hence we take $\Sigma = 1$. On the other hand, the hyperparameter ℓ controls the length scale on which *likely* functions will vary according to the GP model. It is advisable to choose ℓ so that it is on the order of the size of the prolongation stencil, e.g., $\ell > \Delta d$, and preferably $\ell \geqslant R$, because we want GP's data interpolation to be smooth or underfitting (as opposed to oscillatory or overfitting) on the grid-scale over which the interpolation is considered. In practice, we often choose $\ell = 12 \min_{d=x,y,z} \Delta d$ in this paper.

All that remains to complete the GP model is to specify the prior mean function. The prior mean function is often depicted as a constant mean function

[‡]This reasoning can be seen as a cancellation between the $\mathbf{k}_*^T$ and $\mathbf{K}^{-1}$ terms in Eq. (A.4)

for simplicity, i.e., $\bar{f}(\mathbf{x}) = f_0$. Ideally, the optimal value f_0 can be analytically determined from the data by maximizing the likelihood function in Eq. (A.2) (see Appendix B in [45]). Although, for most practical applications, it is reasonable to take a zero mean function $f_0 = 0$ as we aim for the simplest and most general symmetry of any random samples. As such, we use the zero mean function in this paper.

Appendix A.3. Why is a GP high-order method attractive for CFD?

We outline the key computational advantages of using GP model over the standard polynomial methods in the context of numerical interpolation and reconstruction in finite difference and finite volume methods. The results summarized here are based on the previous studies of the GP high-order methods for FVM [45] and for FDM [47]. These features will hold in the current GP-AMR prolongation method since the key algorithmic building-blocks are common in all three formulations.

The first major attractive feature in GP is its variable order of convergence within a single algorithmic framework with a minimal code adaptation of increasing or decreasing the size of the GP stencil, controlled by the GP radius R. The GP algorithms have shown novel algorithmic flexibility in which a variable order of spatial accuracy is achieved and given by the convergence rate $(2R+1)$, corresponding to the local stencil size and the resulting $(2R+1) \times (2R+1)$ GP covariance kernel $\mathbf{K}$ on the stencil. The GP radius R is given as a positive integer value representing the radial distance between the central cell $\mathbf{x}_i$ and $\mathbf{x}_{i+R}$. In general, the GP stencil can be either in simple 1D stencils or in genuinely multidimensional stencils, depending on the dimensionality of problems (i.e., 1D vs. 2D/3D) or the way how 2D/3D problems are formulated (i.e., dimension-by-dimension vs. genuinely multi-dimension). Simple 1D GP stencils have been utilized in our previous studies as we focused on a 1D finite volume GP method [45] and a 3D finite difference GP method [47] based on the conventional dimension-by-dimension formalism. This is different in our current study that a genuinely multidimensional GP stencil of $R = 1$ is employed in the GP-AMR prolongation method to solve multidimensional finite volume problems, see Fig. 4. By being inherently multidimensional, our GP method incorporates the local data information simultaneously from all spatial directions. Practically speaking, the GP's multidimensional handling of local data can directly correspond to the strategy including multidimensional cross derivative terms in the traditional polynomial interpolation formalism, delivering the anticipated high-order (higher than second-order) without the deficiency in accuracy in multidimensions that could be bounded by second-order. This multidimensional consideration is particularly essential in finite volume discretization [11, 35, 22, 5].

Secondly, on smooth multidimensional advection problems, GP-WENO methods demonstrate much faster convergence rates than the polynomial-based WENO method [28] to reach the same solution accuracy. For example, the GP solution with $R = 2$ converges at fifth-order to reach the target L_1 error of

5×10^{-3} in only 43% in CPU times relative to the fifth-order WENO companion solution [47].

Lastly, we remark that implementing the key computational components of the GP prolongation method mostly involves a set of simple linear algebra operations and algebraic manipulations such as the posterior mean function in Eq. (A.4); the evaluations of GP kernels $\mathbf{K}$ and $\mathbf{C}$ in Eq. (A.6) and Eq. (4); the volume-averaged GP prolongation in Eq. (11); the GP-based smoothness indicators β_m in Eq. (18); the overdetermined system in Eq. (19); the nonlinear switching parameter α in Eq. (22), all of which are given as straightforward linear algebra problems with simple algebraic manipulations. The step-by-step procedure of how these individual components are integrated as a whole is discussed in Section 4.

References

[1] M. Adams, P.O. Schwartz, H. Johansen, P. Colella, TJ Ligocki, D. Martin, et al. Chombo software package for amr applications-design document (tech. rep.). *Berkeley, CA: Applied Numerical Algorithms Group Computational Research Division Lawrence Berkeley National Laboratory*, 2015.

[2] A. S. Almgren, V. E. Beckner, J.B Bell, M. Day, L. Howell, Joggerst C., M.J. Lijewski, A. Nonaka, M. Singer, and M. Zingale. Castro: A new compressible astrophysical solver. i. hydrodynamics and self-gravity. *The Astrophysical Jounral*, 715:1221–1238, 2010.

[3] A. S. Almgren, J. B. Bell, P. Collela, L. H. Howell, and M. L. Welcome. A conservative adaptive projection method for the variable density incompressible navier-stokes equations. *Journal of Computationl Physics*, 142:1–46, 1998.

[4] N. Attig, P. Gibbon, and T. Lippert. Trends in supercomputing: The European path to exascale. *Computer Physics Communications*, 182(9):2041–2046, 2011.

[5] D.S. Balsara. Higher-order accurate space-time schemes for computational astrophysics—Part I: finite volume methods. *Living Reviews in Computational Astrophysics*, 3(1):2, 2017.

[6] J. Bell, M. Berger, Saltzman J., and M. Welcome. A three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM Journal of Scientific Computing*, 15:127–138, 1994.

[7] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–84, 1989.

[8] M.J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:484–512, 1984.

[9] C. Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*, 2007.

[10] G.L. Bryan, M.L. Norman, B.W. O'Shea, T. Abel, J.H. Wise, M.J. Turk, D.R. Reynolds, D.C. Collins, P. Wang, S.W. Skillman, et al. Enzo: An adaptive mesh refinement code for astrophysics. *The Astrophysical Journal Supplement Series*, 211(2):19, 2014.

[11] P. Buchmüller and C. Helzel. Improved accuracy of high-order WENO finite volume methods on Cartesian grids. *Journal of Scientific Computing*, 61(2):343–368, 2014.

[12] Y. Chen, G. Tóth, and T.I. Gombosi. A fifth-order finite difference scheme for hyperbolic equations on block-adaptive curvilinear grids. *Journal of Computational Physics*, 305:604–621, 2016.

[13] P. Colella, M.R. Dorr, J.A.F. Hittinger, and D.F. Martin. High-order, finite-volume methods in mapped coordinates. *J. Comput. Physics*, 230:2952–2976, 2010.

[14] P. Colella, D.T. Graves, T.J. Ligocki, D.F. Martin, D. Modiano, D.B. Serafini, and B. Van Straalen. Chombo software package for amr applications design document. *Available at the Chombo website: http://seesar. lbl. gov-/ANAG/chombo/(September 2008)*, 2009.

[15] P. Colella and P.R. Woodward. The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of Computational Physics*, 54(1):174–201, 1984.

[16] P. Colella and P.R. Woodward. The piecewise parabolic method (ppm) for gas-dynamical simulations. *Journal of Computational Physics*, 54:174–201, 1984.

[17] P. Collela and H. Glaz. Efficient solution algorithms for the riemann problem for real gases. *Journal of Computational Physics*, 59:264–289, 1985.

[18] A.J. Cunningham, A. Frank, P. Varnière, S. Mitran, and T. W. Jones. Simulating magnetohydrodynamical flow with constrained transport and adaptive mesh refinement: algorithms and tests of the astrobear code. *The Astrophysical Journal Supplement Series*, 182(2):519, 2009.

[19] J.J. Dongarra. *On the Future of High Performance Computing: How to Think for Peta and Exascale Computing*. Hong Kong University of Science and Technology, 2012.

[20] A. Dubey, A. Almgren, J. Bell, M. Berzins, S. Brandt, G. Bryan, P. Colella, D. Graves, M. Lijewski, F. Löffler, et al. A survey of high level frameworks in block-structured adaptive mesh refinement packages. *Journal of Parallel and Distributed Computing*, 74(12):3217–3227, 2014.

[21] A. Dubey, K. Antypas, M.K. Ganapathy, L.B. Reid, K. Riley, D. Sheeler, A. Siegel, and K. Weide. Extensible component-based architecture for flash, a massively parallel, multiphysics simulation code. *Parallel Computing*, 35(10-11):512–522, 2009.

[22] M. Dumbser, O. Zanotti, A. Hidalgo, and D.S. Balsara. Ader-weno finite volume schemes with space–time adaptive mesh refinement. *Journal of Computational Physics*, 248:257–286, 2013.

[23] B. Fryxell, K. Olson, P. Ricker, F.X. Timmes, M. Zingale, D.Q. Lamb, P. MacNeice, R. Rosner, J.W. Truran, and H. Tufo. Flash: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *The Astrophysical Journal Supplement Series*, 131(1):273, 2000.

[24] G.A. Glatzmaiers and P.H. Roberts. A three-dimensional self-consistent computer simulation of a geomagnetic field reversal. *Nature*, 377(6546):203–209, 1995.

[25] A. Glocer, G. Tóth, Y. Ma, T. Gombosi, J.-C. Zhang, and L.M. Kistler. Multifluid block-adaptive-tree solar wind roe-type upwind scheme: Magnetospheric composition and dynamics during geomagnetic storms—initial results. *Journal of Geophysical Research: Space Physics*, 114(A12), 2009.

[26] R.D. Hornung and S.R. Kohn. Managing application complexity in the samrai object-oriented framework. *Concurrency and computation: practice and experience*, 14(5):347–368, 2002.

[27] L. Jameson. Amr vs high order schemes. *Journal of Scientific Computing*, 18(1):1–24, 2003.

[28] G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, 1996.

[29] G.C. Jordan IV, R.T. Fisher, D.M. Townsley, A.C. Calder, C. Graziani, S. Asida, D.Q. Lamb, and J.W. Truran. Three-dimensional simulations of the deflagration phase of the gravitationally confined detonation model of type ia supernovae. *The Astrophysical Journal*, 681(2):1448, 2008.

[30] R. Keppens, M. Nool, G. Tóth, and J.P. Goedbloed. Adaptive mesh refinement for conservative systems: multi-dimensional efficiency evaluation. *Computer Physics Communications*, 153(3):317–339, 2003.

[31] A.M. Khokhlov. Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations. *Journal of Computational Physics*, 143(2):519–543, 1998.

[32] R.I. Klein. Star formation with 3-d adaptive mesh refinement: the collapse and fragmentation of molecular clouds. *Journal of Computational and Applied Mathematics*, 109(1-2):123–152, 1999.

[33] A.V. Kravtsov, A.A. Klypin, and A.M. Khokhlov. Adaptive refinement tree: a new high-resolution n-body code for cosmological simulations. *The Astrophysical Journal Supplement Series*, 111(1):73, 1997.

[34] P. MacNeice, K.M. Olson, C. Mobarry, R. De Fainchtein, and C. Packer. Paramesh: A parallel adaptive mesh refinement community toolkit. *Computer physics communications*, 126(3):330–354, 2000.

[35] P. McCorquodale and P. Colella. A high-order finite-volume method for conservation laws on locally refined grids. *Communications in Applied Mathematics and Computational Science*, 6(1):1–25, 2011.

[36] P. McCorquodale, P. Ullrich, H. Johansen, and P. Colella. An adaptive multiblock high-order finite-volume method for solving the shallow-water equations on the sphere. *Communications in Applied Mathematics and Computational Science*, 10(2):121–162, 2015.

[37] J. Meinecke, H.W. Doyle, F. Miniati, A.R. Bell, R. Bingham, R. Crowston, R.P. Drake, M. Fatenejad, M. Koenig, Y. Kuramitsu, et al. Turbulent amplification of magnetic fields in laboratory laser-produced shock waves. *Nature Physics*, 10(7):520, 2014.

[38] A. Mignone, C. Zanni, P. Tzeferacos, B. Van Straalen, P. Colella, and G. Bodo. The pluto code for adaptive mesh computations in astrophysical fluid dynamics. *The Astrophysical Journal Supplement Series*, 198(1):7, 2011.

[39] F. Miniati and D.F. Martin. Constrained-transport magnetohydrodynamics with adaptive mesh refinement in charm. *The Astrophysical Journal Supplement Series*, 195(1):5, 2011.

[40] A. Nonaka, A.S. Almgren, J.B. Bell, M.J. Lijewski, C.M. Malone, and M. Zingale. Maestro: An adaptive low mach number hydrodynamics algorithm for stellar flows. *The Astrophysical Journal Supplement Series*, 188(2):358, 2010.

[41] T. Plewa, T. Linde, and V.G. Weirs. Adaptive mesh refinement-theory and applications. *Lecture notes in computational science and engineering*, 41:3–5, 2005.

[42] K.G. Powell, P.L. Roe, T.J. Linde, T.I. Gombosi, and D.L. De Zeeuw. A solution-adaptive upwind scheme for ideal magnetohydrodynamics. *Journal of Computational Physics*, 154(2):284–309, 1999.

[43] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation And Machine Learning. MIT Press, 2005.

[44] J. Ray, C.A. Kennedy, S. Lefantzi, and H. N. Najm. Using high-order methods on adaptively refined block-structured meshes: derivatives, interpolations, and filters. *SIAM Journal on Scientific Computing*, 29(1):139–181, 2007.

[45] A. Reyes, D. Lee, C. Graziani, and Tzeferacors P. A new class of high-order methods for fluid dynamics simulation using gaussian process modeling. *Journal of Computational Physics*, 2017.

[46] A. Reyes, D. Lee, C. Graziani, and P. Tzeferacos. A new class of high-order methods for fluid dynamics simulations using Gaussian Process Modeling. *Journal of Scientific Computing*, 76:443–480, 2018.

[47] A. Reyes, D. Lee, C. Graziani, and P. Tzeferacos. A variable high-order shock-capturing finite difference method with gp-weno. *Journal of Computational Physics*, 381:189–217, 2019.

[48] W. Schmidt, J. Schulz, L. Iapichino, F. Vazza, and A.S. Almgren. Influence of adaptive mesh refinement and the hydro solver on shear-induced mass stripping in a minor-merger scenario. *Astronomy and Computing*, 9:49–63, 2015.

[49] L. I. Sedov. Propagation of strong shock waves. *Journal of Applied Mathematics and Mechanics*, 10:241–250, 1946.

[50] C. Shen, J.-M. Qiu, and A. Christlieb. Adaptive mesh refinement based on high order finite difference weno scheme for multi-scale simulations. *Journal of Computational Physics*, 230(10):3780–3802, 2011.

[51] Jing Shi, Changqing Hu, and Chi-Wang Shu. A Technique of Treating Negative Weights in WENO Schemes. *Journal of Computational Physics*, 175(1):108–127, January 2002.

[52] C.-W. Shu. High order eno and weno schemes for computational fluid dynamics. In *High-order methods for computational physics*, pages 439–582. Springer, 1999.

[53] C.-W. Shu. High order WENO and DG methods for time-dependent convection-dominated PDEs: A brief survey of several recent developments. *Journal of Computational Physics*, 316:598–613, 2016.

[54] Chi-Wang Shu. High-order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD. *International Journal of Computational Fluid Dynamics*, 17(2):107–118, 2003.

[55] ASCAC Subcommittee. Top ten exascale research challenges. *US Department Of Energy Report, 2014*, 2014.

[56] A. Suresh and H.T. Huynh. Accurate monotonicity-preserving schemes with runge–kutta time stepping. *Journal of Computational Physics*, 136(1):83–99, 1997.

[57] R. Teyssier. Cosmological hydrodynamics with adaptive mesh refinement-a new high resolution code called ramses. *Astronomy & Astrophysics*, 385(1):337–364, 2002.

[58] P. Tzeferacos, M. Fatenejad, N. Flocke, C. Graziani, G. Gregori, D.Q. Lamb, D. Lee, J. Meinecke, A. Scopatz, and K. Weide. Flash mhd simulations of experiments that study shock-generated magnetic fields. *High Energy Density Physics*, 17:24–31, 2015.

[59] B. van der Holst and R. Keppens. Hybrid block-amr in cartesian and curvilinear coordinates: Mhd applications. *Journal of computational physics*, 226(1):925–946, 2007.

[60] Q. Zhang, H. Johansen, and P. Colella. A fourth-order accurate finite-volume method with structured adaptive mesh refinement for solving the advection-diffusion equation. *SIAM Journal on Scientific Computing*, 34(2):B179–B201, 2012.

[61] R. Zhang, M. Zhang, and C.-W. Shu. On the order of accuracy and numerical performance of two classes of finite volume WENO schemes. *Communications in Computational Physics*, 9(03):807–827, 2011.

[62] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, M. Katz, A. Myers, T. Nguyen, A. Nonaka, M. Rosso, S. Williams, and M. Zingale. Amrex: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, 4(37):1370, 5 2019.

[63] W. Zhang, A. Almgren, M. Day, T. Nguyen, J. Shalf, and D. Unat. Boxlib with tiling: an amr software framework. *arXiv preprint arXiv:1604.03570*, 2016.

[64] U. Ziegler. The nirvana code: Parallel computational mhd with adaptive mesh refinement. *Computer Physics Communications*, 179(4):227–244, 2008.

[65] M. Zingale, A.S. Almgren, M.G. Barrios-Sazo, V.E. Beckner, J.B. Bell, B. Friesen, A.M. Jacobs, M.P. Katz, C.M. Malone, A.J. Nonaka, et al. Meeting the challenges of modeling astrophysical thermonuclear explosions: Castro, maestro, and the amrex astrophysics suite. In *Journal of Physics: Conference Series*, volume 1031, page 012024. IOP Publishing, 2018.