SIAM J. SCI. COMPUT.
Vol. 44, No. 5, pp. C367–C398

MURPHY—A SCALABLE MULTIRESOLUTION FRAMEWORK FOR SCIENTIFIC COMPUTING ON 3D BLOCK-STRUCTURED COLLOCATED GRIDS*

THOMAS GILLIS† AND WIM M. VAN REES‡

Abstract. We present the derivation, implementation, and analysis of a multiresolution adaptive grid framework for numerical simulations on octree-based three-dimensional block-structured collocated grids with distributed computational architectures. Our approach provides a consistent handling of nonlifted and lifted interpolating wavelets of arbitrary order demonstrated using second-, fourth-, and sixth-order wavelets, combined with standard finite-difference-based discretization operators. We first validate that the wavelet family used provides strict and explicit error control when coarsening the grid, and show that lifting wavelets increase the grid compression rate while conserving discrete moments across levels. Further, we demonstrate that high-order PDE discretization schemes combined with sufficiently high-order wavelets retain the expected convergence order even at resolution jumps. We then simulate the advection of a scalar to analyze convergence for the temporal evolution of a PDE. The results shows that our wavelet-based refinement criterion is successful at controlling the overall error while the coarsening criterion is effective at retaining the relevant information on a compressed grid. Our software exploits a block-structured grid data structure for efficient multilevel operations, combined with a parallelization strategy that relies on a one-sided MPI-RMA communication approach with active post-start-complete-wait synchronization. Using performance tests up to 16,384 cores, we demonstrate that this leads to a highly scalable performance. The associated code is available under a BSD-3 license at https://github.com/vanreeslab/murphy.

Key words. multiresolution, adaptive mesh refinement, wavelet, finite-difference, MPI-RMA

MSC codes. 65M04, 65M50, 65M06

DOI. 10.1137/21M141676X

1. Introduction.

Solutions to partial differential equations (PDEs) are typically characterized by unsteady spatial scale separations. In incompressible fluid dynamics, for example, flows within a boundary layer include important structures on the smallest viscous length scales, whereas the wake is characterized by much larger, inertial structures. Moreover, these flows are intrinsically unsteady and often coupled with the motion or deformation of immersed boundaries, making the resolution requirements

*Submitted to the journal's Software and High-Performance Computing section December 14, 2021; accepted for publication (in revised form) July 25, 2022; published electronically September 29, 2022.

https://doi.org/10.1137/21M141676X

Funding: The work of both authors was supported by an Early Career Award from the Department of Energy, Program Manager Dr. Steven Lee, award DE-SC0020998. The work of the first author was supported by the Belgian American Educational Foundation (BAEF) and by a Wallonie-Bruxelles International (WBI) excellence fellowship. Computational resources have been provided by the Consoritum des Équipements de Calcul Intensif (CÉCI) funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under grant 2.5020.11 and by the Walloon Region. Additional resources include the Tier-1 supercomputer of the Fédération Wallonie-Bruxelles, infrastructure funded by the Walloon Region under grant agreement 1117545, the Tier-0 LUMI accessed throught the pilot phase and in collaboration with the CÉCI, and MELUXINA an infrastructure accessed through EuroHPC under grant EHPC-DEV-2022D05-149. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under contract DE-AC02-05CH11231 using NERSC award ASCR-ERCAP20232.

†Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (tgillis@mit.edu).

‡Corresponding author. Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (wvanrees@mit.edu).

difficult to predict a priori. Especially in three-dimensional (3D) problems, where performance and memory constraints of computing resources constrain the range of applications, it is desirable to construct methods that can adapt the local resolution of the grid to the physical requirements of the PDE solution evolved on that same grid. To achieve accurate solutions, such methods need to be able to discretize and evolve the PDE consistently across different resolutions, detect the need to refine or opportunity to coarsen, and perform the actual coarsening or refinement of the field. Further, their (parallel) implementation needs to be sufficiently efficient so that any increase in computational overhead due to the required memory access patterns, load balancing, and synchronizations does not cancel the gains from the reduction in computational elements, compared to uniform resolution grids. Consequently, the algorithms and implementations of such adaptive grid refinement methods are significantly more complex than uniform grid methods and have seen significant development over the last three to four decades of research in scientific computing.

An established family of methods is formed by the patch-based adaptive mesh refinement (pAMR) approach, which considers nested overlapping grids of increasing resolution [3, 2]. This approach is prevalent across many application domains and has mature implementations inside several software frameworks such as `Chombo` [1], `SAMRAI` [25], and `AMReX` [45]. Overlapping grids simplify the use of coarsening and refinement operators and provide straightforward integration with multigrid-based elliptic solvers. Excluding more expensive adjoint approaches [18], grid adaptation decisions in pAMR are typically made on the basis of either a heuristic measure of the field values and/or their derivatives or an estimate of the truncation error through a Richardson extrapolation technique [2], or some combination of both [35]. If the chosen measure is larger than a user-defined threshold, the grid is refined, and if it is smaller than a second user-defined threshold, the grid is coarsened. Such heuristic measures are easy to implement but generally provide no a priori sense of the error made. The Richardson extrapolation method is more rigorous but also more challenging to implement [1] and requires the simultaneous evolution of the discrete equations on multiple levels. After the decision to adapt a grid has been made, the actual coarsening/refinement approaches and the evaluation of differential operators across resolution boundaries are most often based on polynomial interpolation. Most approaches achieve second-order accuracy in space throughout these operations, though extensions to fourth order have been demonstrated as well [44, 41].

Our work falls under a different family of methods which does not consider overlapping grids but instead uses an octree-based approach. This was demonstrated first in `Gerris` [32] and has a more general distributed implementation in `p4est` [6], which itself is successfully used across different application clients such as the finite-volume solver `ForestClaw` [7]. The dyadic recursive structure associated with these approaches can be combined with a wavelet-based multiresolution analysis of any signal on the grid [30, 31]. Combining wavelet-based grid adaptation with node-based collocated PDE solution methods such as finite-difference techniques leads to the wavelet collocation method [24, 8, 36]. Wavelet collocation methods typically rely on interpolating wavelets, which distinguish the information between two levels through the deviation of fine-level values from an interpolating polynomial constructed from coarse-level values [13]. The interpolating wavelets have been cast in a formal basis through the addition of the biorthogonality [9] and extended to include moment preservation and reduce aliasing through the lifting scheme [37, 10, 38]. The lifted interpolating wavelets form the basis of so-called second generation wavelet collocation methods [43]. Existing codes using wavelet-based grid adaptation with finite-difference-based PDE

evolutions are `MRAG` [42, 34], which uses nonlifted interpolating wavelets for incompressible flow simulations with shared-memory parallelism, and `wabbit` [15], using nonlifted and lifted interpolating wavelets on distributed memory architectures for weakly compressible flow simulations. Overall, despite the potential advantages of high-order grid adaptation, formal multiresolution analysis, explicit error control, and parallel performance, there is a significantly smaller body of work on the implementation details and performance analysis of nonlifting and lifting wavelets on block-structured grids, as compared to pAMR methods.

In this work, we describe the derivation, implementation, and analysis of a parallel, scalable implementation of a 3D multiresolution adaptive grid solver for PDEs on collocated grids, supported by nonlifted or lifted wavelets for scale detection, grid adaptation, and ghost reconstruction. In section 2 we provide a brief background of the wavelet theory and show how this translates to block-structured multilevel grids. We emphasize here the consistent treatment of resolution jumps to preserve polynomial order and lifting properties in multiple dimensions. Section 3 details our implementation, using one-sided MPI-RMA communication strategies to handle parallel communication. In section 4 we validate our approach on static grid adaptation tests by demonstrating error control and convergence of high-order finite-difference schemes for all wavelets and moment conservation for lifted wavelets. Section 5 applies the resulting software to solve PDEs, where we provide detailed analysis of the error as a function of the wavelet-based thresholding parameters. Here we limit ourselves to linear and nonlinear scalar advection equations as examples of challenging problems involving dynamically changing scales, though our framework as presented can already handle a wider set of problems including advection-diffusion and reaction-diffusion equations, and will be further extended in future work. In section 6 we demonstrate that our code retains parallel efficiency across more than 16,000 compute cores and conclude our work with a perspective and future work in section 7.

2. Wavelet-based multiresolution.

Our work relies on a few key contributions that have been made to the field of wavelets theory that include the multiresolution analysis, biorthogonal interpolating wavelets, and the lifting scheme. Though a complete overview of wavelet theory is beyond the scope of this manuscript, we provide a concise review of the concepts required to detail the mathematical framework for our multiresolution grid adaptation below.

2.1. Interpolating wavelets.

Throughout our work we use interpolating wavelets. These were first introduced by [13] and generalize the polynomial interpolation procedure on nested dyadic grids presented in [11, 12] using wavelet theory, as detailed in [14]. Interpolating wavelets can be constructed through polynomial interpolation, thus avoiding the Fourier transform. Through the introduction of the orthogonal multiresolution analysis first, and the biorthogonal multiresolution analysis second, they have been formalized within the framework of second-generation wavelets, and through the lifting scheme they can be generalized to achieve moment conservation properties and reduce aliasing. Here we will briefly touch upon these concepts and their mathematical background.

2.1.1. The orthogonal multiresolution analysis.

The multiresolution analysis [29, 9, 37] defines nested orthogonal subspaces decomposition of $L_2(\mathbb{R})$ indexed by L. Mathematically the nested spaces can be written as $V^L \subset V^{L+1}$ for $L \in \mathbb{Z}$, where the union $\cup_{L\in\mathbb{Z}} V^L$ is dense and orthogonality implies that the intersection $\cap_{L\in\mathbb{Z}} V^L$ is empty. The subspaces are further defined with dilation and translation characteristics

that guarantee the existence of a unique function $\varphi(x)$, such that for any $L \in \mathbb{Z}$ the translated and dilated family of functions $\varphi_k^L(x) = \sqrt{2^L}\, \varphi(2^L\, x - k)$ for $k \in \mathbb{Z}$ is an orthonormal basis of V^L [29]. Orthonormality here means that $\left\langle \varphi_k^L(x)\ ,\ \varphi_i^L(x) \right\rangle = \delta_{i,k}$ $\forall\, \{k, i\} \in \mathbb{Z}$, where $\langle f(x)\ ,\ g(x) \rangle = \int_{-\infty}^{\infty} f(x) g(x)\ \mathrm{d}x$. The difference between two spaces V^L and V^{L+1} is characterized by a new subspace W^L as the orthogonal complement of V^L to V^{L+1}, so that $W^L = \left(V^L \cap V^{L+1}\right)^{\perp}$, hence $V^L \oplus W^L = V^{L+1}$. Similarly to V^L, a basis for W^L is obtained through the dilatation and translation of the wavelet function $\psi(x)$, such that $\psi_m^L(x) = \sqrt{2^L}\, \psi(2^L x - m)$ with $m \in \mathbb{Z}$.

With these definitions, a given function $f(x) \in L_2(\mathbb{R})$ can be projected onto either basis to define the scaling coefficients λ_k^L and detail coefficients γ_k^L,

$$\lambda_k^L \triangleq \left\langle f(x)\ ,\ \varphi_k^L(x) \right\rangle \qquad \text{and} \qquad \gamma_m^L \triangleq \left\langle f(x)\ ,\ \psi_m^L(x) \right\rangle \ . \tag{2.1}$$

We can then build a hierarchy of projections of $f(x)$ into the wavelet subspaces. We start with the projection of $f(x)$ onto level L denoted as

$$P^L[f]\,(x) \triangleq \sum_k \lambda_k^L\ \varphi_k^L(x). \tag{2.2}$$

Further, given that $V^L \oplus W^L = V^{L+1}$, we can relate the projection of $f(x)$ onto level $L+1$ to lower levels through the refinement relation:

$$P^{L+1}[f]\,(x) \triangleq \sum_j \lambda_j^{L+1}\ \varphi_j^{L+1}(x) = \sum_k \lambda_k^L\ \varphi_k^L(x)\ +\ \sum_m \gamma_m^L\ \psi_m^L(x)\ . \tag{2.3}$$

Applied recursively, (2.3) can be used to create a hierarchy of nested decompositions from level L_0 to level L:

$$P^{L+1}[f]\,(x) = \sum_j \lambda_j^{L+1} \varphi_j^{L+1}(x) = \sum_k \lambda_k^{L_0}\ \varphi_k^{L_0}(x)\ +\ \sum_{L_0 \le l \le L} \sum_m \gamma_m^l\ \psi_m^l(x)\ . \tag{2.4}$$

2.1.2. Biorthogonality and linear filters. To generalize the multiresolution analysis to a broader class of wavelet functions such as the symmetric or interpolating ones, one can relax the criteria for finding scaling and corresponding wavelet functions using the dual multiresolution analysis based on biorthogonality [9, 38]. With biorthogonal wavelets, the basis of V^L, $\varphi_k^L(x)$, no longer needs to be orthonormal. Instead, one uses another subspace $\tilde{V}^L$ and the associated basis functions $\tilde{\varphi}_k^L(x)$ such that

$$\left\langle \varphi_i^L(x)\ ,\ \tilde{\varphi}_k^L(x) \right\rangle = \delta_{i,k} \quad \text{with} \quad \{i, k\} \in \mathbb{Z}\ . \tag{2.5}$$

The spaces V^L and $\tilde{V}^L$ have nonorthogonal complements W^L and $\tilde{W}^L$, respectively, such that $V^L \perp \tilde{W}^L$ and $\tilde{V}^L \perp W^L$. This leads to the definition of the primal (dual) scaling functions φ ($\tilde{\varphi}$) and the primal (dual) wavelet functions ψ ($\tilde{\psi}$), which form bases of their respective subspaces and satisfy

$$\begin{aligned} \left\langle \tilde{\varphi}_i^L(x)\ ,\ \psi_k^L(x) \right\rangle &= \left\langle \tilde{\psi}_i^L(x)\ ,\ \varphi_k^L(x) \right\rangle = 0 \qquad \text{and} \\ \left\langle \tilde{\varphi}_i^L(x)\ ,\ \varphi_k^L(x) \right\rangle &= \left\langle \tilde{\psi}_i^L(x)\ ,\ \psi_k^L(x) \right\rangle = \delta_{i,k}. \end{aligned} \tag{2.6}$$

The definitions of the scaling and detail coefficients become

$$\lambda_k^L \triangleq \left\langle f(x)\ ,\ \tilde{\varphi}_k^L(x) \right\rangle \qquad \text{and} \qquad \gamma_m^L \triangleq \left\langle f(x)\ ,\ \tilde{\psi}_m^L(x) \right\rangle \ , \tag{2.7}$$

and the refinement relation remains unchanged:

$$P^{L+1}[f](x) = \sum_j \lambda_j^{L+1}\, \varphi_j^{L+1}(x) = \sum_k \lambda_k^L\, \varphi_k^L(x) \;+\; \sum_m \gamma_m^L\, \psi_m^L(x) \quad . \tag{2.8}$$

Following the nested subspace decomposition a linear filter can be associated to each primal/dual basis function [38], binding two levels together:

$$\varphi_k^L(x) = \sum_j h_{k,j}\, \varphi_j^{L+1}(x), \qquad \psi_m^L(x) = \sum_n g_{m,n}\, \varphi_n^{L+1}(x), \tag{2.9}$$
$$\tilde{\varphi}_k^L(x) = \sum_j \tilde{h}_{k,j}\, \tilde{\varphi}_j^{L+1}(x) \qquad \text{and} \qquad \tilde{\psi}_m^L(x) = \sum_n \tilde{g}_{m,n}\, \tilde{\varphi}_n^{L+1}(x).$$

Combined with the biorthogonal refinement relation, the filters provide the relations for the forward wavelet decomposition (also known as the analysis operation):

$$\lambda_k^L = \sum_j \tilde{h}_{k,j}\lambda_j^{L+1} \;\triangleq\; \tilde{H}_{k,j}\, \lambda_j^{L+1} \qquad \text{and} \qquad \gamma_m^L = \sum_j \tilde{g}_{m,j}\lambda_j^{L+1} \;\triangleq\; \tilde{G}_{m,j}\, \lambda_j^{L+1}, \tag{2.10}$$

where we used the Einstein summation convention to simply the notation. The inverse wavelet decomposition (also known as synthesis operation) is obtained as

$$\lambda_j^{L+1} = \sum_k h_{j,k}\, \lambda_k^L \quad + \quad \sum_m g_{j,m}\, \gamma_m^L \;\triangleq\; H_{j,k}\, \lambda_k^L + G_{j,m}\, \gamma_m^L \quad . \tag{2.11}$$

Both the forward and inverse transforms have linear computational complexity in the number of degrees of freedom and are easily represented as a block diagram, as illustrated in Figure 2.1(b).

2.1.3. Interpolating wavelets. Our work relies on the interpolating wavelets first proposed by [13] based on the Deslauriers–Dubuc interpolation filters [11, 12]. This wavelet family provides a nonorthogonal basis [13] but their construction can be framed within the context of a biorthogonal multiresolution analysis as detailed in [37, 38].

Like the orthogonal wavelets, interpolating wavelets are characterized by the dilatation and translation of a (nonorthonormal) scaling function $\varphi_k^L = \varphi(2^L x - k)$ [13]. The interpolating property of the scaling function is given by $\varphi(i-k) = \delta_{i,k}$ for $i, k \in \mathbb{Z}$. It is convenient at this point to define $x_{L,k} \triangleq k2^{-L}$ as the coordinate associated to an index $k \in \mathbb{Z}$ at level L such that $\varphi_k^L(x_{L,i}) = \varphi(i-k)$. The interpolating

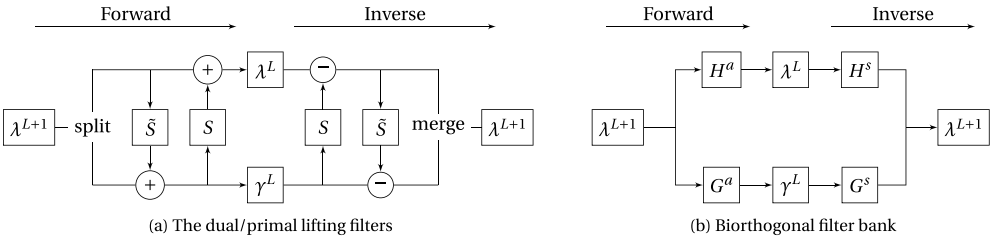


FIG. 2.1. *The forward and inverse wavelet transform as the combination of the dual and primal lifting steps, expressed through the dual/primal filters* (a) *and through the full biorthogonal filter bank* (b).

nature then implies that the evaluation of the projection of the function at $x_{L,i}$ is equal to the associated scaling coefficient at that coordinate, λ_i^L:

$$P^L[f](x_{L,i}) = \sum_k \lambda_k^L \, \varphi_k^L(x_{L,i}) = \lambda_i^L \quad . \tag{2.12}$$

Interpolating wavelets can be classified by their degree of interpolation N, which corresponds to the number of moments of the scaling function,

$$\int_{-\infty}^{\infty} x^p \, \varphi(x) \, \mathrm{d}x = \delta_p, \qquad 0 \leq p < N \quad . \tag{2.13}$$

This relation guarantees the ability of the scaling functions to exactly reproduce polynomials of order $N-1$.

Interpolating wavelets are well suited to wavelet collocation methods because it is convenient to use function evaluations at $x_{L,k}$ interchangeably with scaling coefficients λ_k^L. However, in general this does incur an error associated with the truncation of detail coefficients at any given level. This error can be captured by comparing the exact function with the function $\overline{P}^L[f](x)$ defined as $\overline{P}^L[f](x) \triangleq \sum_k f(x_{L,k}) \, \varphi_k^L(x)$. This corresponds to a similar projection as (2.12), but replacing λ_k^L with $f(x_{L,k})$. A bound on the difference can be found as [39]

$$\left| \overline{P}^L[f](x) - f(x) \right| \leq \mathcal{O}\left(2^{-L\,N}\right) \quad . \tag{2.14}$$

Specifically at location $x_{L,k}$ we then find $\left| f(x_{L,k}) - \lambda_k^L \right| \leq \mathcal{O}\left(2^{-L\,N}\right)$. Since $2^{-L} = x_{L,k+1} - x_{L,k} \sim h$ relates to the grid spacing on level L, this relation implies that using function values in an Nth order interpolating wavelet-based projection incurs a discretization error of $\mathcal{O}(h^N)$.

The simplest family of interpolating wavelets is the Donoho interpolating wavelets [13, 37, 38], which are classified here with the code $N.0$, with N the degree of interpolation. For the Donoho interpolating wavelets, the dual scaling function is a Dirac impulse located at the origin [37], $\tilde{\varphi}_k^L(x) = \delta(x - x_{L,k})$. Hence, it follows that at any level

$$\lambda_k^L = \left\langle f(x) \, , \, \tilde{\varphi}_k^L(x) \right\rangle = f(x_{L,k}) \quad , \tag{2.15}$$

which means the scaling coefficients at level L do not just equate the function *projection* evaluation at level L, as in (2.12), but they equate the function evaluation itself: $\overline{P}[f](x_{L,k}) = P[f](x_{L,k}) = f(x_{L,k}) = \lambda_k^L$. However, these wavelets do not conserve moments when compressing information, and are characterized by considerable aliasing in the wavelet transform as reported in [37]. One potential avenue to address these issues is by increasing $\tilde{N}$, the number of zero moments of the wavelet function. This can be done through the lifting approach proposed in [37] as discussed in the next section.

2.1.4. Lifted interpolating wavelets. The lifting scheme has been introduced as a general and convenient way to construct biorthogonal and second generation wavelets and their associated linear filters [37, 40, 10, 38]. Although the scheme can be generalized to any wavelet family, we restrict ourselves here to the interpolating wavelet. Starting with a set of scaling coefficients on level $L+1$, the lifting scheme uses the following three different steps to obtain the set of scaling and detail coefficients on level L:

1. The *splitting* step splits the fine scaling coefficients into temporary coarse scaling (even indices) and detail (odd indices) coefficients. This step is also known as the application of the "lazy wavelet" and can be captured by the filters $\tilde{H}_{k,j} = \delta_{2k,j}$ and $\tilde{H}_{m,j} = \delta_{2m+1,j}$. After this first step, we have a set of coarse scaling and detail coefficients

$$\lambda_k^L = \lambda_{2k}^{L+1} \quad , \quad \gamma_m^L = \lambda_{2m+1}^{L+1} \ . \tag{2.16}$$

2. The *dual lifting* applies the filter $\tilde{S}$ to the scaling coefficients and uses the result to update the detail coefficients:

$$\gamma_m^L \leftarrow \gamma_m^L + \tilde{S}_{m,k}\ \lambda_k^L \ . \tag{2.17}$$

3. The *primal lifting* applies the filter S to the detail coefficients and uses the result to update the scaling coefficients:

$$\lambda_k^L \leftarrow \lambda_k^L + S_{k,m}\gamma_m^L \ . \tag{2.18}$$

The successive application of the three steps is illustrated in Figure 2.1 and can be expressed through composite filters H^a and G^a that combine all stages into single operators. Reversing the sequence of operations and individual steps leads to the corresponding inverse transform, captured by the H^s and G^s filters.

The Donoho interpolating wavelets discussed in the previous section can be cast in the format of the lifting scheme by setting $\tilde{S}$ to the filter coefficients of [11, 12], which ensure exact interpolation for polynomials of degree up to $N-1$, and setting all primal lifting filter coefficients $\tilde{S} = 0$.

To "lift" these wavelets, following [37], one can choose the primal filter $\tilde{S}$ in such a way that the first $\tilde{N}$ moments of the primal wavelet function vanish:

$$\int_{-\infty}^{\infty} x^p\ \psi_k^L(x)\ \mathrm{d}x = 0, \qquad 0 \le p < \tilde{N} \ . \tag{2.19}$$

If this holds, we ensure the conservation of the first $\tilde{N}$ moments across levels:

(2.20)

$$\begin{aligned}\int_{-\infty}^{\infty} x^p \left[\sum_k \lambda_k^{L+1}\ \varphi_k^{L+1}(x)\right] \mathrm{d}x &= \int_{-\infty}^{\infty} x^p \left[\sum_k \lambda_k^L\ \varphi_k^L(x) \ + \ \sum_m \gamma_m^L\ \psi_m^L(x)\right] \mathrm{d}x \\ &= \int_{-\infty}^{\infty} x^p \left[\sum_k \lambda_k^L\ \varphi_k^L(x)\right] \mathrm{d}x\end{aligned}$$

for $0 \le p < \tilde{N}$. Using the moment properties on $\varphi(x)$ from the interpolating wavelet, the pth moment on level L can be expressed as

$$\int_{-\infty}^{\infty} x^p P^L[f]\,(x)\ \mathrm{d}x = \sum_k\ \lambda_k^L \int_{-\infty}^{\infty} x^p\ \varphi_k^L(x)\ \mathrm{d}x = 2^{-L}\ \sum_k\ \lambda_k^L\ \ (x_{k,L})^p \quad , \tag{2.21}$$

which allows us to rewrite the moment conservation identity of lifted interpolating wavelets (2.20) between two levels $L+1$ and L as

$$\sum_j\ \lambda_j^{L+1}\,(x_{L+1,j})^p = 2\ \sum_k\ \lambda_k^L\ \ (x_{L,k})^p \tag{2.22}$$

for $0 \le p < \tilde{N}$.

TABLE 2.1
Dual-lifting coefficients, $\tilde{S}_i$ [4, Appendix B.3]. For uniform grids the filter is symmetric, i.e., $\tilde{S}_{-i} = \tilde{S}_i$.

N	S_{-2}	S_{-1}	S_0	S_1	S_2	S_3
2			$-1/2$	$-1/2$		
4		$1/16$	$-9/16$	$-9/16$	$1/16$	
6	$-3/256$	$25/256$	$-75/128$	$-75/128$	$25/256$	$-3/256$

To satisfy (2.19), on a uniform grid and assuming $\tilde{N} \leq N$ it can be shown that $S^{\tilde{N}} = -1/2\ \tilde{S}^{\tilde{N}}$ [37, Theorem 12], where $\tilde{S}$ are the dyadic interpolation coefficients from [11, 12] and given in Table 2.1. The lifting scheme thus results in an interpolating filter bank for interpolating wavelets indexed by the corresponding interpolation and moment properties $N.\tilde{N}$, which can be used for refinement ($\tilde{H} = H^a$ and $\tilde{G} = G^a$) and coarsening ($H = H^s$ and $G = G^s$) operations. Nonlifted (Donoho) interpolating wavelets have $\tilde{N} = 0$, whereas lifted wavelets have $\tilde{N} > 0$; numerical values for the filters are given in section SM1. All our results in this work are restricted to $N \in \{2, 4, 6\}$ and $\tilde{N} \in \{0, 2\}$.

Last, we note that lifting the interpolating wavelets leaves the primal scaling function $\varphi(x)$ unaffected, but does change the dual scaling function $\tilde{\varphi}(x)$ from a delta function to a continuous distribution. This means the identity $f(x_{L,k}) = \lambda_k^L$ of the Donoho interpolating wavelets is lost, and instead we fall back on the general error bound $\left|\lambda_k^L - f(x_{L,k})\right| \leq \mathcal{O}\left(2^{-L}\, N\right)$ provided in (2.14).

2.1.5. Compression. Relying on the multiresolution theory, compression can be achieved by discarding all the detail coefficients whose absolute values are smaller than a tolerance ϵ. This yields a coarser (or compressed) representation of the information,

$$P^L[f]_\epsilon(x) = \sum_k \lambda_k^{L_0}\ \varphi_k^{L_0}(x)\ +\ \sum_{L_0 \leq l \leq L}\ \sum_{|\gamma_m| > \epsilon} \gamma_m^l\ \psi_m^l(x)\ . \tag{2.23}$$

It can be shown [13, 43, 26] that the error committed by this approximation is of the order of ϵ,

$$\left\| P^L[f](x) - P^L[f]_\epsilon(x) \right\|_\infty \leq C_1 \epsilon\ , \tag{2.24}$$

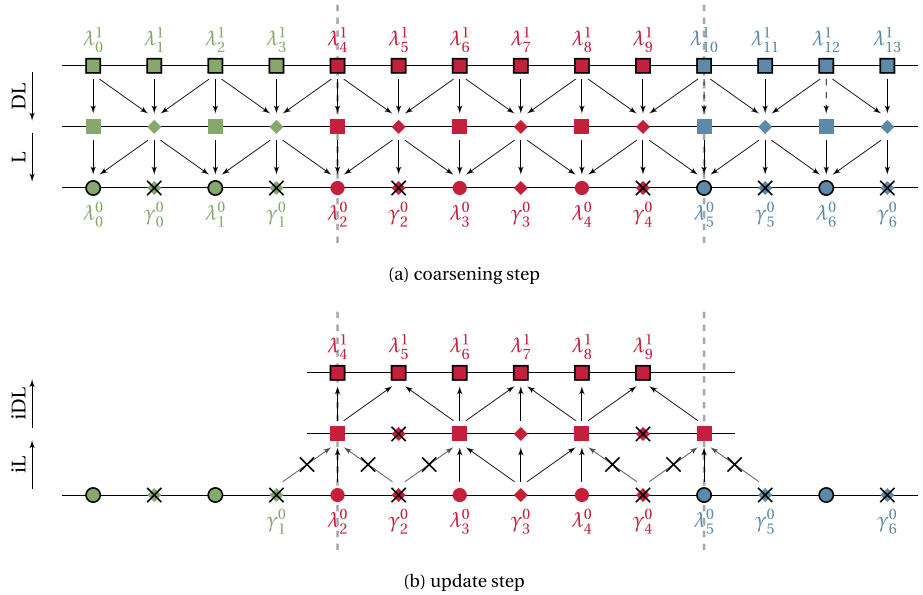
where C_1 depends on $f(x)$. In practice, with a reasonably smooth function the value of $C_1 \approx 1$, which means that ϵ is an accurate estimate of the local error committed.

2.2. Extension to block-structured grids. In this section we describe how to adapt the multiresolution analysis described above to block-structured grids. Throughout this work, we limit the jump of resolution between two adjacent blocks to 1 (denoted as the 2:1 constraint), which considerably simplifies the operations and the implementation complexity. Below we discuss three fundamental operations that are required in the implementation: *coarsening* describes the compression of data in 2^d blocks into a single block at the next lower resolution level, with d the number of spatial dimensions of the grid; *refinement* describes the refinement of data in a single block into 2^d new blocks at the next higher resolution level; and *ghost point reconstruction* relates to the construction of ghost points for blocks across a resolution jump, so that finite-difference stencils can be evaluated on each block at its own local uniform resolution.

We describe each of these operations in one dimension in more detail below, focusing on the wavelet 2.2 for simplicity. Subsequently, we will discuss how implementation choices lead to the treatment of grid points near resolution jumps, and how we define the criteria for compressing or refining a block. We note beforehand that our sketches use grid "blocks" and associated numbering that do not reflect a practical setting, but rather provide the minimum number of points needed to explain the respective operations for ease of interpretation.

2.2.1. Coarsening. Starting from a uniform resolution one can coarsen a block using the filter H_a. The coarsening pattern is illustrated in Figure 2.2 for wavelet 2.2 where the "left" (green) and "right" (blue) fine scaling coefficients at level $L+1=1$ (top row) are converted into coarse scaling coefficients at level $L=0$ (bottom row) through subsequent application of the dual lifting ($\tilde{S}$) and lifting (S) filters. After these steps, only the scaling coefficients at level L are retained while the detail coefficients are discarded. Due to the lifting step, ghost points are required for a block to coarsen when using wavelets with $\tilde{N}>0$, with the precise number specified in the first column of Table 2.2. In the example of Figure 2.2(a) for wavelet 2.2, the green region needs one ghost point at the back (λ_4^1, needed to compute λ_1^0) and the blue region needs two ghost points at the front (λ_8^1 and λ_9^1, required to compute λ_5^0).

Discarding the detail coefficients on level L does not affect the scaling coefficients on that level; however, when $\tilde{N}>0$ discarding these details does affect the scaling coefficients of adjacent blocks whose resolution has not changed. This can be seen and accounted for by performing an inverse wavelet transform from level L back to level $L+1$ after discarding the details, and updating the values of the affected scaling coefficients:



(a) coarsening step

(b) update step

FIG. 2.2. *Sketch of the steps required to coarsen the left (green) and right (blue) grid regions using wavelet* `2.2`*, with the central (red) region remaining at fine resolution. First we apply the dual lifting (DL) and lifting (L) steps successively to compute the coarse-level scaling coefficients, and discard the neglected detail coefficients* (a). *When* $\tilde{N}>0$ *the information encoded in the discarded detail coefficients must also be removed from the fine-level information on the central (red) region, by using the inverse lifting (iL) and inverse dual lifting (iDL) filters during the update step* (b).

TABLE 2.2

Ghost points requirement for a block when coarsening (see Figure 2.2(a)*), for a block at level* $L+1$ *performing the update step when a neighbor at level* L *has coarsened (see Figure* 2.2(b)*), and when refining a block (see Figure* 2.4*). The coarse region extension indicates how many additional scaling and detail coefficients at level* L *have to be computed outside of the blocks boundaries to perform the update step. The associated ghost points are required at the level* $L+1$ *of the block whose neighbor has coarsened. The size of these regions can vary between one and three dimensions, as explained in subsection* 2.3*.*

	Coarsening		**Update after neighbor has coarsened**				**Refinement**	
Wavelet order	*# ghost points*		*coarse region extension*		*# ghost points*		*# ghost points*	
			front	*back*	*front*	*back*		
N $\tilde{N}$	*front*	*back*	(1D / 3D)	(1D / 3D)	(1D / 3D)	(1D / 3D)	*front*	*back*
2 0	0	0	0 / 0	0 / 1	0 / 0	0 / 1	0	1
2 2	2	1	1	2	2	3	0	1
4 0	0	0	0 / 2	0 / 3	0 / 4	0 / 5	1	2
4 2	4	3	3	4	6	7	1	2
6 0	0	0	0 / 4	0 / 5	0 / 8	0 / 9	2	3
6 2	6	5	5	6	10	11	2	3

$$\lambda_j^{L+1} \leftarrow \lambda_j^{L+1} - G_{j,n}^s \gamma_n^L, \tag{2.25}$$

where γ_n^L are all the detail coefficients that we have discarded. To perform the update step on a fine block whose neighbor has coarsened, the fine block needs to have enough ghosts points to compute the values of γ_n^L that are discarded, which increases significantly the ghost point requirements of the update step as shown in Table 2.2. The distance (in index space) to the farthest detail to be discarded depends on the wavelet order and $\tilde{N}$ and is shown in Table 2.2 under the column *coarse region extension*.

In the specific example of Figure 2.2, though the middle red region does not change resolution, we must still remove all the information associated with the discarded detail coefficients of the coarsened green and blue regions. For wavelet 2.2, this corresponds to discarding the information associated with γ_1^0 on the left and γ_5^0 on the right. To achieve this, starting from the original uniform grid on the top line of Figure 2.2(a), the red region requires two ghost points in front (λ_2^1 and λ_3^1, needed to compute γ_1^0) and three ghost points in the back (λ_{10}^1, λ_{11}^1, and λ_{12}^1, needed to compute γ_5^0), in order to perform the update step associated with the coarsening of both its neighboring grid regions. Last, as indicated in Figure 2.2, for wavelet 2.2 we additionally discard the detail coefficient γ_2^0 when the left region coarsens and γ_4^0 when the right region coarsens, which will be explained further in the next two sections.

2.2.2. Ghost point reconstruction. For any grid configuration with blocks at multiple levels of resolution, we have to be able to compute ghost points for each block at their local resolution level. We choose here to rely on the wavelets to do so for all ghosting operations, in order to be consistent with the grid adaptation operations.

Figure 2.3 shows the computation of the ghost points for a fine region (in red) surrounded by neighboring coarse regions (in green and blue) for wavelet 2.2 in one dimension. Ghost points to be computed are shown with open circles (on the coarse level) and open squares (on the fine level), whereas known scaling coefficients are shown in colored symbols with black outlines. A naive wavelet transform indicates the immediate problem that the ghost points for the finer region and those for the coarser region are interdependent: for instance, to compute λ_3^1 we would need to apply inverse lifting on λ_2^0, but λ_2^0 is in turn dependent on λ_3^1 through the dual lifting. This

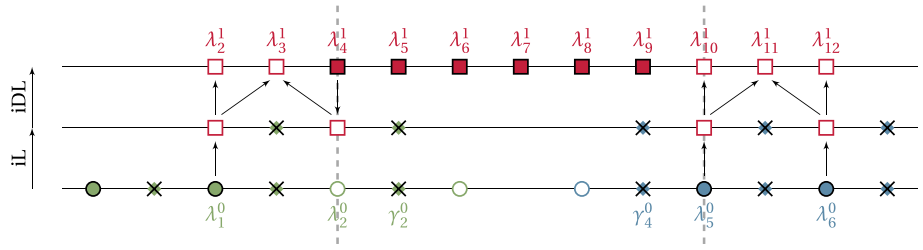


FIG. 2.3. *Sketch of the process used to reconstruct ghost points (open symbols) for a 1D fine-resolution grid region (in red) surrounded by coarse grid regions on the front (green) and back (blue), using wavelet* `2.2`. *The arrows denote the flow of information from grid values to the unknown ghost points. After the ghost points for the fine region have been computed, the ghost points for both coarse regions can be found from a standard coarsening procedure (Figure* 2.2(a)*)*

interdependency gets more intricate in higher dimensions and for higher-order wavelets. Though these systems can be solved as proposed in [34] for nonlifting wavelets, the associated implementation requires expansive look-up tables that significantly increase the memory footprint of the solver, especially in three dimensions [42]. To avoid this complication, we choose to discard the fine-region detail coefficients that cause the interdependency between the wavelet transforms on the two levels, which we denote the "coarse-extension assumption". In the case illustrated in Figure 2.3 specifically, this means we discard the detail coefficients γ_2^0 and γ_4^0. On a more abstract level, this choice means that we effectively extend the coarse-level region across the resolution jump into the first few grid points on the neighboring fine resolution block. To consistently follow through with this assumption requires additional steps in our implementation that we will discuss more below. For now, with this assumption in place the ghost reconstruction across a resolution jump can be done in two steps. First, we can use the wavelet transform to compute ghost values for the fine resolution block through what is essentially a local refinement of the coarse grid, which can now be done explicitly.

In Figure 2.3 we illustrate this approach for the computation of the fine ghost values with the wavelet 2.2. Focusing first on the right resolution jump, our assumption of explicitly discarding γ_4^0 formally sets the ghost point $\lambda_{10}^1 = \lambda_5^0$ and further allows us to directly evaluate λ_{11}^1 as $\lambda_{11}^1 = 1/2(\lambda_5^0 + \lambda_6^0)$. This is essentially a refinement procedure where we take the coarse-level scaling coefficients and refine them under the assumption that all unknown detail coefficients involved in this process are zero, irrespective of the side of the interface where they exist.

For the left resolution jump, the specific grid layout enables us to proceed in one of two ways. The first way is conceptually similar to how we describe the right interface, where by assumption $\gamma_2^0 = 0$ even though this detail resides in the fine resolution grid. Under this assumption, $\lambda_2^0 = \lambda_4^1$ and the ghost point $\lambda_3^1 = 1/2\left(\lambda_1^0 + \lambda_2^0\right)$ is obtained from the inverse wavelet transform of the coarse-level data. The second way relies on the inverse of the dual lifting step: we can directly write $\lambda_3^1 = 1/2\left(\lambda_4^1 + \lambda_1^0\right)$ without explicitly considering γ_2^0. The two approaches are identical if $\gamma_2^0 = 0$. Note that this is possible only on the left interface; on the right interface, the value of λ_{10}^1 is not readily available unless $\gamma_4^0 = 0$, since λ_{10}^1 belongs to the coarse region. In practice, we use the first approach and explicitly assume that both γ_2^0 and γ_4^0 are zero. We do so because the implementation of the inverse dual lifting in multiple dimensions is not trivial (see subsection 2.3), which would complicate the second approach.

This concludes the computation of the ghost points for the fine level; afterward, we can treat the region of the fine resolution as a local uniform grid that we coarsen

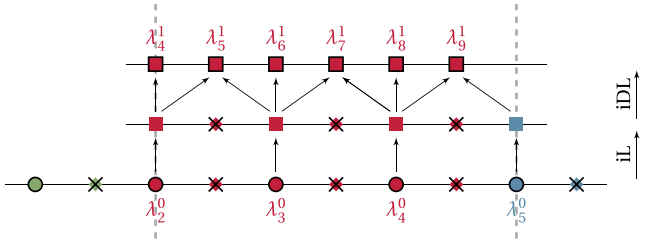


FIG. 2.4. *Sketch of the refinement procedure of a 1D coarse grid region (in red) using wavelet* `2.2`. *Starting from the coarse-level scaling coefficients (bottom line), we use inverse lifting (iL) and inverse dual lifting (iDL) to compute the fine-level scaling coefficients (top line). For this wavelet, only one ghost point on the back of the refined region is required, indicated in blue.*

in order to obtain the ghost points for the coarse grid levels. This procedure poses no further difficulties and is identical to the coarsening described above.

2.2.3. Refinement. The refinement operation of a block away from resolution boundaries is trivially done through the subsequent application of the lifting and dual lifting filters. Near resolution boundaries, we retain the coarse-extension assumption introduced for the computation of ghost points described in the previous subsection, which enables the explicit computation of the fine-level scaling coefficients. This process is illustrated in Figure 2.4 for the special case of the wavelet 2.2, in which case only the ghost point λ^0_5 is required on the right resolution jump to compute the new scaling coefficient λ^1_9. The number of ghost points needed for a block to refine for any other wavelet considered here is shown in the last column of Table 2.2.

2.2.4. Substitution. In the previous subsections we motivated and detailed the coarse-extension assumption, where we neglect specific fine-level detail coefficients near coarse-fine resolution jumps to facilitate explicit ghost reconstruction and refinement operations. We explained that this is essentially equivalent to an extension of the coarse-level region into a small band of the adjacent fine-level block. In practice however, these specific detail coefficients on the fine-level block may not be zero due to field operations on the associated fine-level scaling coefficient, such as during the evolution of a PDE. Without addressing this, we would inconsistently neglect high-frequency information during the grid adaptation and ghost reconstruction due to our coarse-extension assumption.

To avoid that this spurious information persists and leads to an inconsistent wavelet transform on the two sides of the interface, we perform an additional "substitution" step where we overwrite each fine-level scaling coefficient associated with a neglected detail coefficient locally with wavelet-reconstructed values that will enforce a zero detail coefficient. To achieve this step, we use the dual lifting relationship

$$\begin{aligned} \gamma^L_m = 0 \quad &\Rightarrow \quad G^a_{m,j}\ \lambda^{L+1}_j \;=\; G^a_{m,j\neq(2m+1)}\ \lambda^{L+1}_{j\neq(2m+1)} + \lambda^{L+1}_{2m+1} = 0 \\ &\Rightarrow \quad \lambda^{L+1}_{2m+1} \leftarrow -G^a_{m,j\neq(2m+1)}\ \lambda^{L+1}_{j\neq(2m+1)}, \end{aligned} \tag{2.26}$$

where we used the fact that $G^a_{m,0} = 1$ for all wavelets considered in this work. We note that in one dimension, the proposed approach is exactly the inverse of the dual lifting step, as illustrated in Figure 2.5. This substitution step is subtle so we point out that this step does not affect the order of accuracy of the wavelet operations, only removes detail coefficient values that are generated during the PDE evolution starting from values below the coarsening threshold, and can be rigorously understood as the

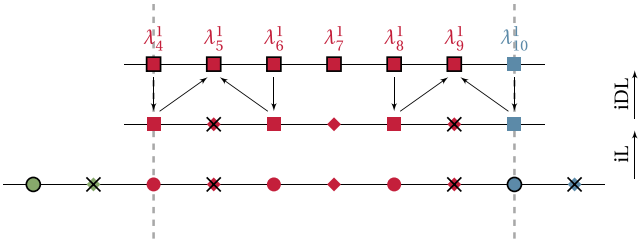


FIG. 2.5. *Sketch of the substitution process, where spurious information associated with neglected detail coefficients in fine resolution grid regions adjacent to coarse resolution regions is explicitly discarded. For wavelet* `2.2` *in one dimension, as shown here, substitution is equivalent to the inverse dual lifting.*

consistent enforcement of the coarse-extension assumption. In practice, we apply the substitution step as part of the ghost point reconstruction, immediately after the computation of the fine ghost points and before computation of the coarser ghost points.

2.2.5. Block adaptation criteria. Subsection 2.1.5 describes how to compress a signal in one dimension given its wavelet transformation. Here we explain how we transform this condition to compress data on a block-structured grid, detect emerging scales, and manage the need to refine during a simulation.

Compression. Starting with the former, for each block at level L we compute all associated details γ^{L-1} through the forward wavelet transform and take the maximum value for each block b as $\|\gamma^{L-1}\|^b_\infty$. Consistent with our coarse-extension assumption above, we consider within this local infinite norm also the set of details that we require to be negligible when computing ghost points, the refinement relation, and the coarsening steps, even though these detail coefficients might physically reside in adjacent blocks. The number of additional details considered beyond the block boundary is given in the column *coarse region extension* in Table 2.2. Once we have the maximum detail coefficient on each block, we decide on an action to take. Due to the octree nature of our grid, we can only coarsen all leaf blocks within a single tree node simultaneously. Therefore, for each set of 2^d leaf blocks in the grid where d is the spatial dimension, we reduce them to a single coarser level block if each of the leaf blocks b satisfies $\|\gamma^{L-1}\|^b_\infty < \epsilon_c$, with ϵ_c the coarsening threshold. This criterion constitutes a generalization of (2.23) to a block-structured grid. This approach implies that the compression rate of a given signal decreases as the block size increases, due to a reduced granularity in the grid. We emphasize here that by including the details neglected during the coarse-extension assumption within our definition of $\|\gamma^{L-1}\|^b_\infty$, the compression approach remains consistent with (2.23).

Refinement. The criterion to refine is necessarily more ad hoc, as it aims to predict where new scales are expected during the evolution of the equations based on an instantaneous analysis of the field. Different approaches exist within existing wavelet-based adaptive grid methods, such as increasing the resolution of neighboring blocks to take into account the smallest scales created by the PDE [26]. Here we follow [34] to rely on a user-specified tolerance $\epsilon_r > \epsilon_c$ that determines whether refinement is necessary. Using this approach, a block b is refined if $\|\gamma^{L-1}\|^b_\infty > \epsilon_r$, i.e., if the detail coefficients of the current information exceed a user defined threshold. We will analyze this choice in the validation and result sections below.

Under the above compression policy, we are guaranteed by the wavelet framework to discard only information encoded by detail coefficients that do not exceed ϵ_c. With

the refinement approach, the maximum detail coefficients during the evolution of the equations are guaranteed to never exceed ϵ_r, since we would refine when that happens. Specifically, if we refine a block at level L for which $\|\gamma^{L-1}\|_\infty^b > \epsilon_r$, we create new blocks at level $L+1$, each one of which is characterized by $\|\gamma^L\|_\infty^b = 0$. This means we have to make sure we do not coarsen blocks just after they have been refined, even though technically their detail coefficients are smaller than the coarsening threshold. The implementation of this requirement is discussed in section 3.

Ratio between compression and refinement thresholds. From (2.14) we know that $\|\gamma\|_\infty^b \propto h^N$ with N the interpolation order of the wavelet. This means that coarsening a block will generally increase its detail coefficient by a factor 2^N. Consequently, if we choose $\epsilon_r/\epsilon_c < 2^N$ and a block with $\|\gamma\|_\infty^b$ only slightly below the coarsening threshold is coarsened, its details will exceed ϵ_r after coarsening. In this case, the block will be flagged for refinement again, leading to flip-flops in the grid adaptation. Conversely, choosing $\epsilon_r/\epsilon_c > 2^N$ can lead to a nonunique grid: if we consider a block with $\|\gamma\|_\infty^b$ slightly above the coarsening threshold and therefore admissible on the grid, the same block coarsened by one level would also be admissible on the grid. In this case the adaptation is therefore not unique and the obtained grid depends on external factors such as the initial level.

In practice, we observe ϵ_r to be the threshold that determines the overall accuracy of the simulation, since this is the threshold that sets the maximum value of detail coefficient admissible on the grid. Then ϵ_c determines the compression rate, or how much information we are willing to discard for a given ϵ_r. This can be controlled by the ratio ϵ_r/ϵ_c, which we make sure to set to $\epsilon_r/\epsilon_c > 2^N$ to prevent the flip-flopping described above. The effect of both ϵ_r and the ratio ϵ_r/ϵ_c is discussed through numerical experiments below.

Adaptation frequency. For transient problems we have to choose a frequency of adaptation that balances the need to adjust the grid to dynamically evolving scales against the computational cost of changing the grid. In general, lowering the mesh adaptation frequency will lead to wavelet detail coefficients in the grid that fall more and more below the compression threshold in some regions, and increase more and more beyond the refinement threshold in others. The former will not increase the errors made during the simulation, but the latter could potentially lead to the grid not capturing emerging or transported scales that are relevant to the PDE evolution. A counterpoint to this is that the block-structured grid contains significant "inertia," which grows with the block size, due to the fact that we refine even when one single detail in a block exceeds the threshold.

To put the adaptation frequency into context, we can consider a transport problem with characteristic velocity U. In this case, the time it takes for the solution to travel α grid points at level L is $2^{-L}\alpha/U$ (assuming a unit cube as root domain). If we adapt every N_a time steps and follow a CFL-based time step constraint, the solution has traveled $\mathrm{CFL}\, N_a$ grid points between successive adaptations. Though in practice likely problem-dependent our results in subsection 5.1 show that for a smooth signal and suitably small refinement threshold this number is allowed to be of comparable value as the block size N_b leading to $N_a \sim N_b/(\mathrm{CFL})$.

2.3. Extension to multiple dimensions. Here we detail the extension of the above methodology to multiple spatial dimensions, starting with the coarsening operation and subsequently emphasizing some of the implementation details to consider.

The filter application in three dimensions relies on the successive application of the corresponding 1D filters in each dimension. To clarify the notation we use a

superscript on all the filters to denote the direction in which the filter is applied. For coarsening we then obtain

$$\lambda^L_{k_x,k_y,k_z} = [H^a_X \times H^a_Y \times H^a_Z]\, \lambda^{L+1} \quad , \tag{2.27}$$

which exclusively relies on the H^a filter applied tensorially on the scaling coefficients at level $L+1$.

To compute the detail values we alternatively apply the filters H^a or G^a depending on the scaling or detail behavior in the considered dimension, as dictated by whether the associated index in that dimension is even or odd. This means we can distinguish different "degrees" of detail coefficients, given by the number of directions in which the coefficient behaves as detail information. Specifically, the first degree detail coefficients, which we collectively denote as $\gamma^L_{1^\circ}$, have an odd index in one direction only and are given by

$$\begin{gathered}\gamma^L_{m_x} = [G^a_X \times H^a_Y \times H^a_Z]\ \lambda^{L+1}, \qquad \gamma^L_{m_y} = [H^a_X \times G^a_Y \times H^a_Z]\ \lambda^{L+1}, \\ \gamma^L_{m_z} = [H^a_X \times H^a_Y \times G^a_Z]\ \lambda^{L+1}.\end{gathered} \tag{2.28}$$

Similarly the second degree detail coefficients, which we collectively denote as $\gamma^L_{2^\circ}$, have two odd indices and we obtain

$$\begin{gathered}\gamma^L_{m_x,m_y} = [G^a_X \times G^a_Y \times H^a_Z]\ \lambda^{L+1}, \qquad \gamma^L_{m_x,m_z} = [G^a_X \times H^a_Y \times G^a_Z]\ \lambda^{L+1}, \\ \gamma^L_{m_y,m_z} = [H^a_X \times G^a_Y \times G^a_Z]\ \lambda^{L+1}.\end{gathered} \tag{2.29}$$

Finally, the third degree scaling coefficients, which we collectively denote as $\gamma^L_{3^\circ}$, are obtained as

$$\gamma^L_{m_x,m_y,m_z} = [G^a_X \times G^a_Y \times G^a_Z]\ \lambda^{L+1} \quad . \tag{2.30}$$

In order to relax further the notation, for all detail coefficients $\gamma^L = \{\gamma^L_{1^\circ}, \gamma^L_{2^\circ}, \gamma^L_{3^\circ}\}$ we will refer to the fine scaling coefficients located at the same positions as $\{\lambda^{L+1}_{1^\circ}, \lambda^{L+1}_{2^\circ}, \lambda^{L+1}_{3^\circ}\}$, respectively. We also have the "zeroth" degree scaling coefficients associated with even indices in all three directions, which we denote by $\lambda^{L+1}_{0^\circ}$ so that $\lambda^{L+1} = \{\lambda^{L+1}_{0^\circ}, \lambda^{L+1}_{1^\circ}, \lambda^{L+1}_{2^\circ}, \lambda^{L+1}_{3^\circ}\}$.

Revisiting the coarse-extension assumption we made in one dimension, its extension to three dimensions can be formulated as discarding any first, second, and third degree detail coefficients in a fine block adjacent to a resolution jump that are involved in the multidimensional refinement scheme. This is reflected by the higher 3D values in the column coarse region extension of Table 2.2, which represents the index-space distance from the block boundary to the farthest detail coefficient that is discarded. As before, we discard this information through the substitution procedure, which inverts locally the inverse dual lifting step. In three dimensions, the substitution on first degree scaling coefficients is performed as

$$\gamma^L_{1^\circ} = [G^a]_{(i)\neq(0)}\ \lambda^{L+1} + \lambda^{L+1}_{1^\circ} \quad \Rightarrow \quad \lambda^{L+1}_{1^\circ} \leftarrow [G^a]_{(i)\neq(0)}\ \lambda^{L+1} \quad , \tag{2.31}$$

where we used that $[G^a]_{(i)=(0)} = 1$ for all wavelets considered in this work. For second degree scaling coefficients, this becomes

$$\gamma^L_{2^\circ} = [G^a \times G^a]_{(i,j)\neq(0,0)}\ \lambda^{L+1} + \lambda^{L+1}_{2^\circ} \quad \Rightarrow \quad \lambda^{L+1}_{2^\circ} \leftarrow [G^a \times G^a]_{(i,j)\neq(0,0)}\ \lambda^{L+1} \quad , \tag{2.32}$$

and for third degree, we find

$$\gamma^L_{3^\circ} = [G^a \times G^a \times G^a]_{(i,j,k)\neq(0,0,0)}\ \lambda^{L+1} + \lambda^{L+1}_{3^\circ} \tag{2.33}$$
$$\Rightarrow \quad \lambda^{L+1}_{3^\circ} \leftarrow [G^a \times G^a \times G^a]_{(i,j,k)\neq(0,0,0)}\ \lambda^{L+1}.$$

At first it seems that each equation is interdependent since the first, second, and third degree scaling coefficients are all included in λ^{L+1}. However, analyzing the filter G^a reveals that the first degree scaling coefficients $\lambda^{L+1}_{1^\circ}$ only need $\lambda^{L+1}_{0^\circ}$ to be updated, the second degree coefficients $\lambda^{L+1}_{2^\circ}$ need $\lambda^{L+1}_{0^\circ}$ and $\lambda^{L+1}_{1^\circ}$, and so forth. This means we can consistently and explicitly perform the substitution step in three dimensions by first updating the first degree, then the second degree, and finally the third degree scaling coefficients according to the values of the respective neglected detail coefficients.

2.4. Boundary conditions. All theory described above is for infinite signals and grids, and the validation and results cases below rely on fields with compact support, so that boundary conditions are not relevant. In practice, we do need an implementation of boundary conditions for finite signals, which we created by relying on interpolation and extrapolation at the domain boundary. Our solver currently supports zero value (Dirichlet), zero flux (Neumann), plain zeros filling, and extrapolation boundary conditions. They all rely on polynomial interpolation done at the order of the wavelet used, which is compatible with the nonlifted wavelet theory but doesn't conserve the moments when used with lifted wavelets. To improve the numerical properties of the interpolation we use Neville's algorithm [33]. Wavelets on the interval [13] would provide more consistent implementations of such boundary conditions, and the lifting scheme provides avenues for moment conservation [17], but we reserve this for future work.

3. Implementation and algorithms. In this section we discuss high-level implementation choices of the ghost reconstruction and grid adaptation operations, deferring the details to section SM2 in the supplementary materials. Our entire code base relies on the external library `p4est` [6] to handle the meta-data infrastructure of the octree, while all grid adaptation and block operations have been implemented directly in our solver. Within `p4est`, we define each "tree" as a unit cube domain, which forms the root (level $L = 0$) of an octree data structure that can be refined, and the leaves are uniform resolution blocks of size N_b^3. Following `p4est`, the trees can be tiled to create a "forest" of trees, which enables us to create rectangular domains of arbitrary aspect ratios. We currently have implemented wavelets with $N \in \{2, 4, 6\}$ and $\tilde{N} \in \{0, 2\}$. Extension to higher N is straightforward if needed; higher $\tilde{N}$ on the other hand will potentially deteriorate the efficiency of the solver as we will have to significantly increase the number of adjacent detail coefficients that need to be discarded in accordance with our coarse-extension assumption. Throughout our implementation we apply the 2:1 constraint on levels of adjacent blocks, enforcing it during grid adaptation and exploiting it during all multiresolution wavelet operations.

Ghost reconstruction procedure. The implementation of the ghost reconstruction consists of two parts: the first recovers the value from coarser neighbors and same-level neighbors (see Algorithm 1 in section SM2), and the second computes the values from finer neighbors (see Algorithm 2 in section SM2). The interrank communication during both parts is handled using MPI-RMA with a post-start-complete-wait (PSCW) synchronization strategy, chosen to target massively parallel infrastructures [23]. In the first step, we copy (or use `MPI_Get`) the required scaling coefficients from the coarse- and same-level neighbors to the current block, where the actual ghost

points are computed locally once the required values are gathered. This choice reduces the size of the communications and the required memory for the buffers. For similar reasons, to retrieve ghost values from a finer neighbor block, we first compute the required ghost values from the perspective of the neighboring block, which then copies (or uses `MPI_Put`) the coarsened values to our block. Throughout these steps, we have implemented the ghost point computation for vector or tensor fields in a "component-by-component" way, so that we can overlap communication and computation by performing wavelet-based refinement/coarsening operations on one component while the ghost exchange is performed for the next one.

Grid adaptation. The implementation of the grid adaptation follows an iterative procedure. During each iteration we start by computing the maximum detail criterion for each block as explained in section subsection 2.2.5, which in turn dictates the block's desired actions. Then we enforce global policies that include the 2:1 condition, possible user-defined limits of minimum/maximum level (not used in this manuscript), and the prohibition of coarsening blocks that have been refined at earlier iterations; a detailed description is included in section SM2. This finalizes the adaptation decision on each block, after which we perform the refinement and/or coarsening on the affected blocks and use the update step to adjust the scaling coefficients of blocks whose neighbors have just been coarsened. Finally, we use the `p4est` grid partitioning algorithm to distribute the blocks among ranks and ensure load balancing of the current grid. This ends the current iteration, after which we recompute the ghost values. The iterative process ends when, under our policy, no blocks have changed their resolution, which ensures that $\|\gamma\|_\infty < \epsilon_r$ on all blocks.

4. Validation. We present here the numerical validation of our framework on three different aspects: the grid adaptation and the error control, the moment conservation for the lifted wavelets, and the convergence of finite difference operators on multi-level grids. For all cases, we set the linear dimension of each block to $N_b = 24$, so that each block contains $N_b^3 = 13,824$ unknowns.

4.1. Grid adaptation and error control. The grid adaptation test, referred to as the "epsilon test," measures the error between a coarsened field and the original, uncompressed information. According to the wavelet theory and (2.24) this error must be bounded by $C_1\epsilon_c$, where we observed in practice $C_1 = \mathcal{O}(1)$. For a fixed value of ϵ_c, the epsilon test proceeds as follows:

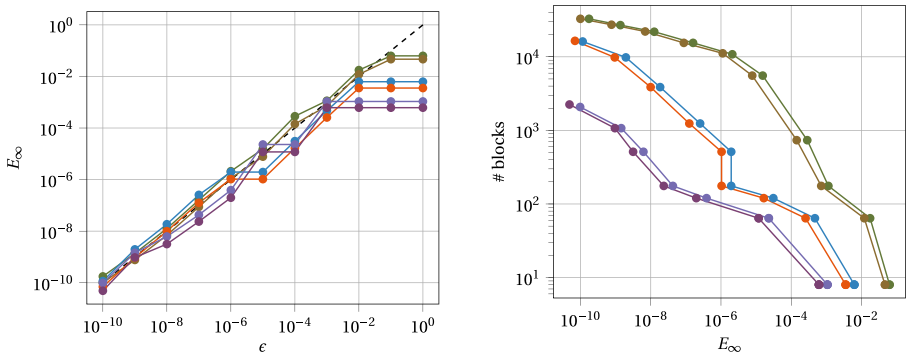
1. initialize an analytic field on a fine level $L_{\max}$,
2. given ϵ_c, coarsen the grid according to the block adaptation policy described above,
3. refine the grid back to the $L_{\max}$ level and compare the error with the initial condition.

We have chosen the analytical field to be a scalar Gaussian function centered within a cubic computational domain of size 2:

$$f(x,y,z) = \exp\left[-\frac{r^2}{\sigma^2}\right], \qquad r^2 = (x-1)^2 + (y-1)^2 + (z-1)^2 \quad , \tag{4.1}$$

where we set $\sigma = 2/15$. The field is initialized at $L_{\max} = 5$. In Figure 4.1(a) we show the evolution of the infinite norm of the error E_∞ depending on the value of ϵ_c, for a range of different wavelets, where the error is defined as

$$E_\infty = \left\| \overline{P}^{L_{\max}}[f](\mathbf{x}) - \overline{P}^{L_{\max}}[f]_{\epsilon_c}(\mathbf{x}) \right\|_\infty \quad . \tag{4.2}$$



(a) error on the field (b) compression of the field

FIG. 4.1. *Epsilon test: effect of the compression threshold ϵ_c on error (left) and number of blocks (right) for a static grid adaptation using wavelet* `2.0` ( ), `2.2` ( ), `4.0` ( ), `4.2` ( ), `6.0` ( ), `6.2` ( ).

The results validate that the ϵ_c is an accurate prediction of the compression error, consistent with the 1D wavelet theory described above. For high values of ϵ_c, the error plateaus as the block granularity in the grid is too low to allow further coarsening. Looking at the number of blocks as a function of the E_∞ in Figure 4.1(b), we observe that for a given error the number of blocks required to represent the compressed field decreases significantly as the wavelet order N increases. Further, the lifting wavelets characterized by $\tilde{N} = 2$ consistently require a slightly smaller number of blocks than their nonlifting counterpart characterized by $\tilde{N} = 0$, for the same error and interpolation order N.

4.2. Moment conservation. The moment-preserving properties of lifting wavelets described above can be validated by comparing the moments on a given uniform level $L_{\max}$, both before and after discarding the detail coefficients according to ϵ_c. Using the same setup as for the epsilon test, we compare moments between $\overline{P}^{L_{\max}}[f](x)$ and $\overline{P}^{L_{\max}}[f]_{\epsilon_c}(x)$ and define their difference as

$$\mathcal{M}_{p,q,r} = \int_{\mathbb{R}^3} \overline{P}^{L_{\max}}[f](\mathbf{x})\, x^p y^q z^r \, \mathrm{d}\mathbf{x} - \int_{\mathbb{R}^3} \overline{P}^{L_{\max}}[f]_{\epsilon_c}(\mathbf{x})\, x^p y^q z^r \, \mathrm{d}\mathbf{x} \ , \tag{4.3}$$

where $0 \leq p, q, r < \tilde{N}$. Each moment can be evaluated from the scaling coefficients using (2.21). The results of this test are shown for the different wavelets in Figures 4.2(a) and 4.2(b), respectively, for the zeroth moment $p = q = r = 0$ and the norm of the three first moments. This validates that the lifted interpolating wavelets conserve both the zeroth and the first moment of the scaling coefficients throughout the adaptation process. When considering the nonlifted wavelet family, we notice that the error in the moments is negligible for the higher ϵ_c values, then suddenly increases at a certain ϵ_c and gradually decrease when ϵ_c is reduced. At the largest values for ϵ_c, the adaptation process coarsens the grid uniformly, and by virtue of the dual scaling functions with $\tilde{N} = 0$ we retain the original function values on the remaining grid points. Refinement does not affect the moments of the field for any interpolating wavelet (see (2.13)) and so in this case our test will compare moments between function values on two uniform grids at different resolutions. On a uniform grid the moment integration rule is equivalent to a spectrally accurate trapezoid quadrature

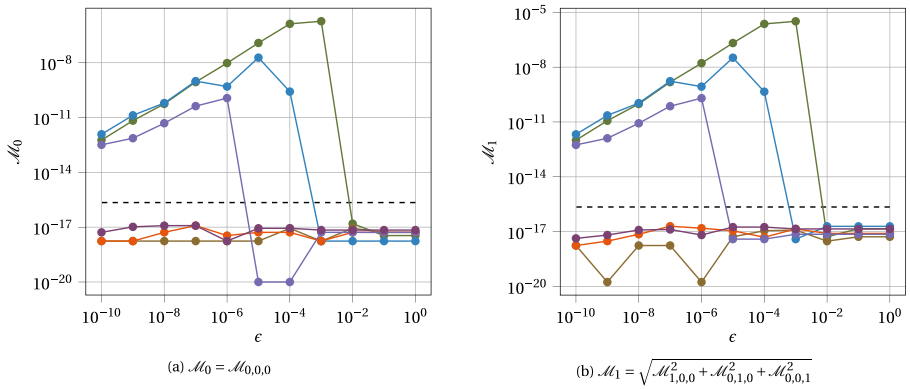


FIG. 4.2. *Moment test: effect of the compression threshold ϵ_c on the conservation of zero (left) and first (right) moments before and after static grid adaptation, for the wavelet* `2.0` ( ), `2.2` ( ), `4.0` ( ), `4.2` ( ), `6.0` ( ), `6.2` ( ).

due to the compactness and smoothness of the Gaussian function, and it turns out that even for the relatively high values of ϵ_c considered, the coarsening does not affect the error of this approximation, leading to zero values in the moment error. For each of the three wavelets with $\tilde{N} = 0$, there exists a "critical" value of ϵ_c for which the coarse grid first contains multiple levels, thus breaking the favorable convergence properties associated with a uniform grid quadrature and showing the real effect of grid adaptation on the lack of moment conservation for these wavelets.

4.3. Wavelets and spatial discretization. As a third measure of a static validation of our 3D wavelet-based multiresolution grid framework, we consider convergence of various finite-difference operators across a resolution jump as a function of the wavelet order and refinement level. We consider an advection term discretized using conservative upwind finite difference schemes of third order (`CONS-3`) and fifth order (`CONS-5`), as well as central laplacian operators of second order (`DIFF-2`) and fourth order (`DIFF-4`). More details about the finite-difference schemes used can be found in section SM4.

The analytical field is the same Gaussian blob as in the previous two subsections and is initialized on a uniform fine level characterized by grid spacing h_f. To simplify the measure of convergence, we do not consider the automatic mesh adaptation in this subsection and instead focus on a grid with two levels of resolution that are fixed in space. Starting from the initial condition at level h_f, we coarsen one eighth of the grid by one level, making sure we cover all possible resolution jumps between blocks (jumps across faces, edges, and corners). We then compute the ghost points as described above and evaluate the finite-difference stencil on the entire grid. To compute the error, we compare the discrete values to the analytic solution of applying the continuous differential operators to the analytic field.

The convergence of the infinite norm (E_∞) of the error is shown in Figure 4.3 as a function of h_f, for different wavelet orders N and $\tilde{N}$. The results show that if the wavelet order is sufficiently high, the expected convergence order is reached for all finite-difference operators even in the infinity norm, indicating a correct treatment of the resolution jump. For lower-order wavelets, the error instead is bound by the polynomial order of the wavelet used to interpolate the fine-level ghost points. Specifically, in this case the error is bound by $N - n$, where N is the wavelet order and n is

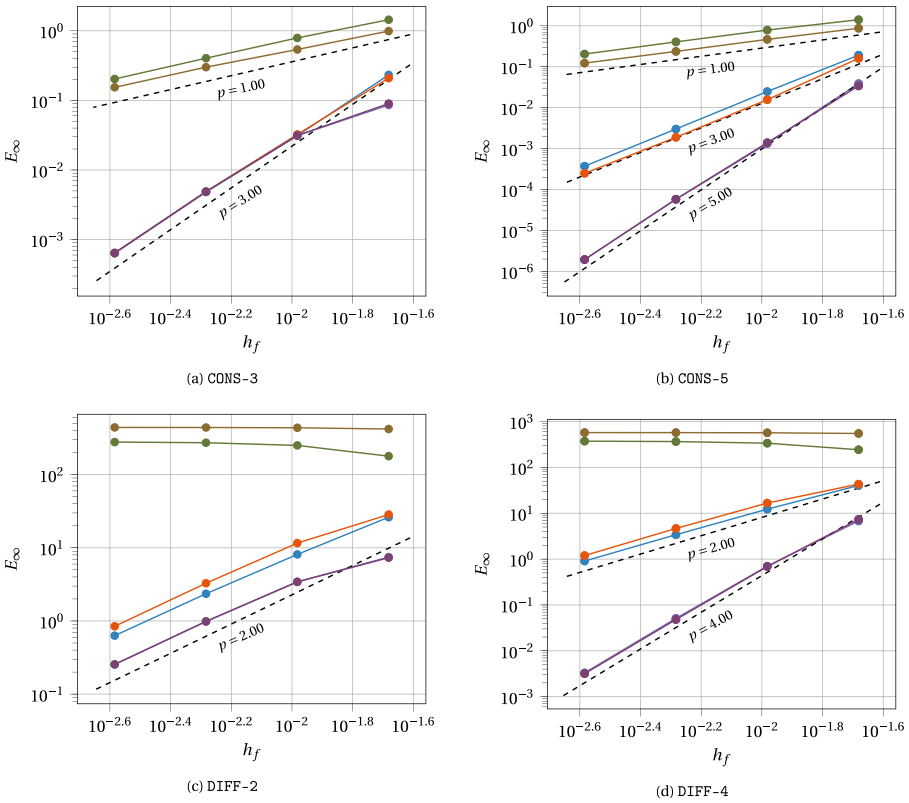


FIG. 4.3. *Convergence order of spatial derivatives for third- and fifth-order conservative advection (top row) and a second- and fourth-order laplacian (bottom row) on a two-level grid with finest level h_f, obtained using ghost points reconstructed with wavelet* `2.0` ( ), `2.2` ( ), `4.0` ( ), `4.2` ( ), `6.0` ( ), `6.2` ( ).

the order of the derivative operator, consistent with the accuracy order of numerically differentiating an Nth degree polynomial n times. Across all cases, the convergence of the error can thus be given as h^p with $p = \min(k, N - n)$, where k is the order of the finite-difference stencil.

In practice we should therefore only consider wavelets with $N \geq 4$ to obtain a scheme that is at least second-order accurate on first- and second-order PDEs.

5. Convergence analysis for a linear advection equation. Having validated the correct implementation of the grid adaptation for nonlifted and lifted wavelets, as well as the ghost point reconstruction and finite-difference operators, we focus here on the behavior of grid adaptation during the evolution of a PDE. We consider the transport of a scalar field in a divergence-free flow field $\nabla \cdot \mathbf{u} = 0$ as a simple case of a hyperbolic conservation law:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{u}\phi) = 0 \quad . \tag{5.1}$$

In this section, we compute the right-hand side using the third-order finite-difference scheme `CONS-3`, perform time integration using a third-order RK3-TVD scheme [21, 22] (detailed in section SM3), and fix the block size to 24^3.

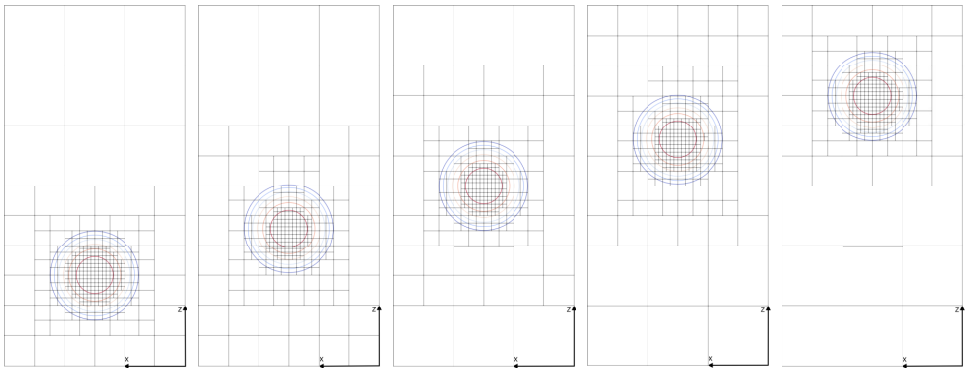


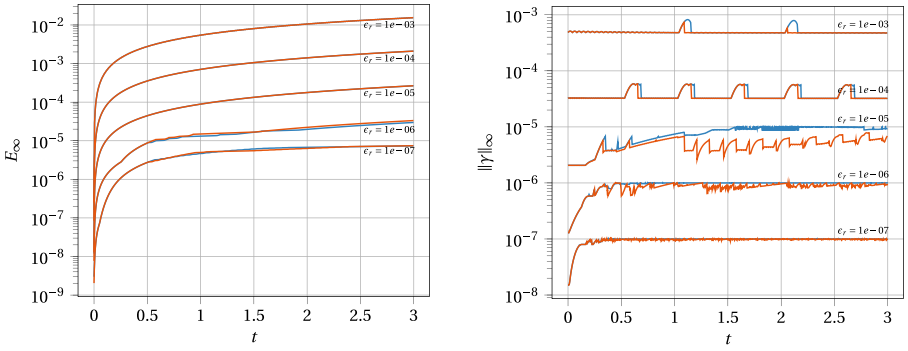
FIG. 5.1. *2D projection in the XZ-plane of the Gaussian blob for wavelet* 4.2 *and* $\epsilon_r = 10^{-6}$ *with* $\epsilon_r/\epsilon_c = 100$. *The 2D projections are illustrated with isolevels at* $[10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$.

We note that this problem poses a sufficiently challenging testcase to allow us to analyze our methodology and differentiate between the wavelets. Nevertheless, as shown above, the presented software framework in its current form is also able to handle time-dependent problems involving diffusion and reaction terms, as well as vector-based quantities.

5.1. Translation of a Gaussian blob. To assess the convergence behavior of our algorithm and implementation, we consider a simple case of the advection of a Gaussian blob in a uniform velocity field. The computational domain is chosen as a rectangular box of size $3 \times 3 \times 6$ with each unit cube represented by a separate tree, leading to 54 trees in the domain. We set the velocity as $[0\ ;\ 0\ ;\ 1]$, and advect a Gaussian blob ($\sigma = 1/5$) initially centered at $(3/2, 3/2, 3/2)$ over a distance of 3, so that we can evaluate the exact solution as a mirror of the initial condition. The time step is controlled by setting the CFL $= 1/4$ (based on the finest-level grid spacing), which is small enough so that the spatial discretization errors dominate the time integration errors. We adapt the grid every 6 time steps, so that the information travels at most 1/16th of the finest-level block between adaptation steps. Within this setting we vary ϵ_r and ϵ_c to control the grid adaptation during the evolution of the PDE, focusing on wavelet 4.0 and wavelet 4.2 only. For context of the discussion, in Figure 5.1 we illustrate the obtained grid for the case of wavelet 4.2, with $\epsilon_r = 10^{-6}$ and $\epsilon_c = 10^{-8}$. In this case the maximum level during the simulation is 4, leading to an effective grid spacing of $h_f = 2.6 \times 10^{-3}$ or, if the grid was uniformly refined to this level, a domain with about 3 trillion grid points. The 2D projections highlight the front/back asymmetry in the grid refinement which is due to the difference between the refinement and coarsening threshold in combination with the moving field: the mesh coarsening will be triggered at larger distances behind the blob than the refinement in front of the blob.

Effect of refinement threshold. We first consider the effect of varying the refinement threshold ϵ_r, keeping the ratio $\epsilon_r/\epsilon_c = 100$ fixed.

The time evolution of the error during the advection is presented in Figure 5.2(a) for wavelet 4.0 (in blue) and wavelet 4.2 (in orange) across a range of ϵ_r values, showing that the error decreases with ϵ_r without significant differences between the two wavelets. The evolution of the maximum detail coefficient on the grid ($\|\gamma\|_\infty$) is



(a) E_∞ over time (b) $\|\gamma\|_\infty$ over time

FIG. 5.2. *Effect of ϵ_r (using $\epsilon_r/\epsilon_c = 100$) on the time evolution of the maximum error (left) and max detail coefficient (right) for a linear advection testcase with* CONS-3*, using wavelet* 4.0 *(* ●— *) and wavelet* 4.2 *(* —● *).*

shown in Figure 5.2(b), which confirms that the maximum detail is always bound by ϵ_r. Further, the time evolution shows that the maximum detail varies over time in a nonsmooth manner, which is explained by noting that the location at which $\|\gamma\|_\infty$ is computed can jump in space as individual blocks refine or coarsen.

Studying the convergence behavior of a simulation on a multilevel grid is not trivial. On a uniform grid, one would show convergence evaluating the maximum error E_∞ as a function of the grid spacing h. On adaptive grids, however, the grid spacing h varies in space and time, we have no direct control on the minimum grid spacing h, and there is no guarantee that the maximum error E_∞ is measured at a single physical location when adapting the grid. Alternatively, one can use a parameter like the effective number of degrees of freedom in the simulation to measure convergence. We show associated results briefly at the end of this subsection. However, we consider this convergence metric less relevant to the point of this work, because the effective number of degrees of freedom is an outcome of the simulation and will likely vary in time. Instead, we control the error primarily by varying ϵ_r, and so a more suitable convergence analysis relates the maximum error at the end time of the simulation as a function of the input parameter ϵ_r (Figure 5.3(a)). The result demonstrates that ϵ_r is successful at controlling the error and moreover its value provides an estimate of the error made in a simulation, albeit with a problem-specific prefactor of $\mathcal{O}(10)$ in this case.

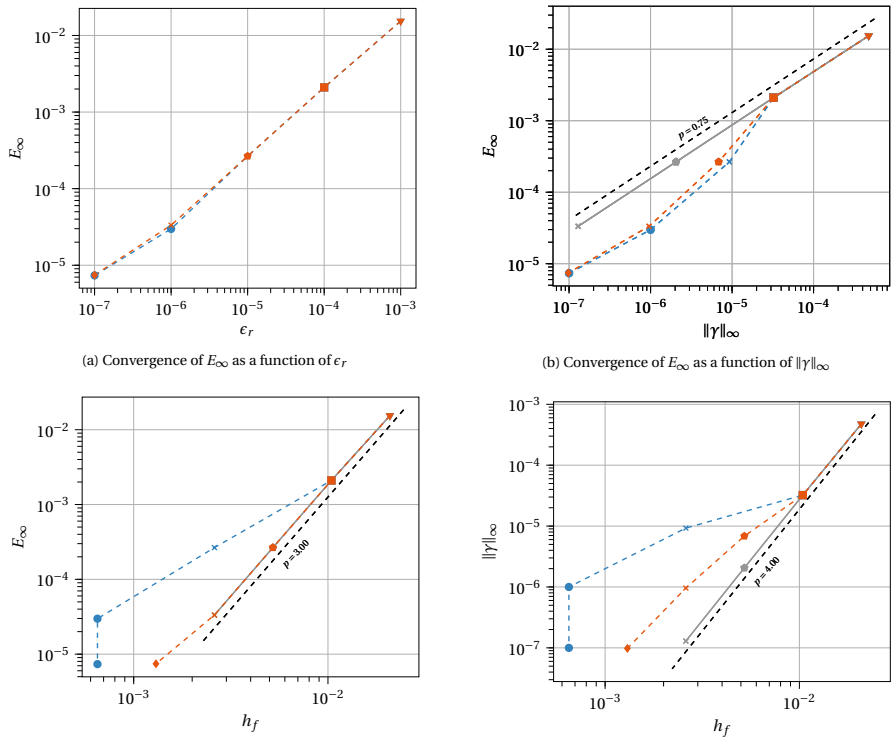
We can decompose this convergence behavior into different components. First, through our adaptation policy we guarantee that ϵ_r bounds the maximum detail coefficient $\|\gamma\|_\infty$. In Figure 5.3(b) we show the relation between the error and the maximum detail coefficient, where we included a uniform resolution line (in gray) obtained by varying the constant grid spacing h, and computing for each h the maximum error E_∞ as well as the maximum detail coefficient evaluated by a single-level wavelet 4.0 transform. Both uniform and multiresolution results show a clear 3/4 slope, where the wavelet results vary as the locations of the maximum scaling coefficient and the maximum error jump independently across different locations in the grid between individual simulations. The 3/4 slope can be explained by two observations. First, we know that the order of the CONS-3 spatial discretization is third, so that $E_\infty \propto h^3$. This is confirmed in Figure 5.3(b), showing the error as a function of the finest-level grid spacing h_f; both the uniform and the wavelet 4.2 lines follow a third-order slope

(wavelet 4.0 will be discussed below). Second, according to (2.14), the detail coefficients associated with a projection of a given smooth function onto a level with spacing h scale as $\|\gamma\|_\infty \propto h^N$, where $N = 4$ is the polynomial interpolation order of the wavelet. For our data this convergence is confirmed in Figure 5.3(c), where we plot the maximum detail coefficient as a function of the finest grid spacing h_f. Combining these relations we find that $E_\infty \propto \|\gamma\|_\infty^{3/4}$.

Looking more closely at Figures 5.3(b) and 5.3(c) indicates that though the overall behavior in Figure 5.3(a) is consistent between wavelet 4.0 and wavelet 4.2, the associated grid adaptation strategies are different. Figure 5.3(b) shows that wavelet 4.2 behaves similarly to the uniform resolution grid, which is impressive as the uniform grid result represents the smallest possible error for any given h_f; the wavelet 4.2 does not compromise that error despite the continuous grid adaptation during the simulation. The behavior of wavelet 4.0 instead demonstrates that this wavelet transform generates detail coefficients that do not predict the error committed by the PDE and therefore cause a spurious coarsening and belated refinement. This is emphasized by the last two points on the left (associated with $\epsilon_r = 10^{-6}$ and $\epsilon_r = 10^{-7}$) where the error goes down even though the finest grid spacing stays the same. This means that at $\epsilon_r = 10^{-6}$ the error was associated with a coarser level than the maximum, and refining that level without affecting the finest level successfully reduced the error. Similarly, in Figure 5.3(c) the gaps between the uniform grid and the adapted grids are associated with coarsening, which increases the maximum detail coefficients. We see again that the behavior between wavelet 4.0 and wavelet 4.2 is different: the reduced aliasing of the lifted wavelet 4.2 leads to a better correlation of the maximum detail coefficients, the finest grid spacing, and the maximum error, compared with the nonlifted wavelet 4.0. We emphasize, however, that despite the different strategies both wavelets successfully control the error during the evolution of this PDE as a function of ϵ_r, as evidenced by the overlapping lines in Figure 5.3(a).

The different strategies of wavelet 4.0 and wavelet 4.2 are further reflected by the number of blocks required throughout the simulation, as a function of ϵ_r. The evolution of the number of blocks over time is shown in Figure 5.4(a). For early times, as the Gaussian blob translates through the grid the trailing side of the blob gets coarsened since the small details there fall below ϵ_c. The leading side does not get refined yet as the small details do not yet exceed ϵ_r, so the number of blocks decreases, leading to the front/back asymmetric grid structure as shown in Figure 5.1. Once the leading side of the blob gets picked up by ϵ_r the number of blocks increases again, though the asymmetry persists.. At later times, the number of blocks plateaus for wavelet 4.2 across all values of ϵ_r, indicating that the grid structure is largely constant. For wavelet 4.0, on the other hand, the number of blocks increases throughout the simulation. This is consistent with the convergence analysis above, where we observed that wavelet 4.0 produces detail coefficients that do not accurately reflect the PDE error and thus refines the grid in locations without strongly reducing the error.

Plotting the error made in the simulation as a function of the number of blocks (Figure 5.4(b)) shows a slope of 1 with respect to the effective number of degrees of freedom for both uniform and multiresolution simulations. Since the number of blocks is inversely proportional to an "effective" linear grid spacing to the power of 3, this thus recovers the third-order convergence of the discretization scheme. Comparing the two wavelets, we find that wavelet 4.0 requires up to twice as many blocks compared to wavelet 4.2 at the lowest error values. Both wavelets provide significant gains over the uniform resolution simulation (fewer blocks by a factor of $\approx 30-40$ for wavelet 4.0 and



(a) Convergence of E_∞ as a function of ϵ_r

(b) Convergence of E_∞ as a function of $\|\gamma\|_\infty$

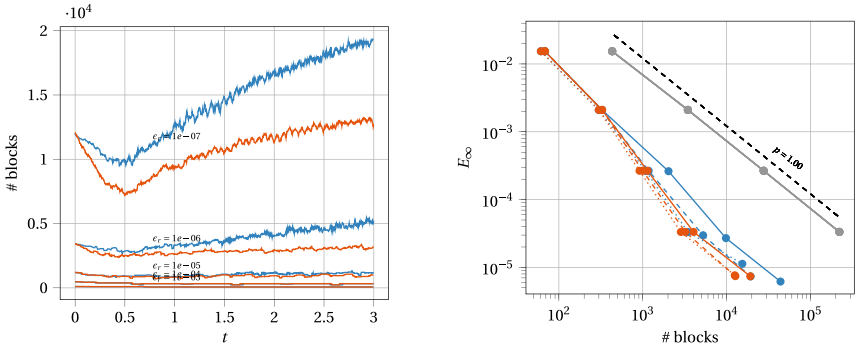
(c) Convergence of E_∞ as a function of the finest grid spacing h_f, showing third-order due to the use of `CONS-3`

(d) Variation of h_f as a function of $\|\gamma\|_\infty$, showing fourth-order due to the use of wavelet `4.X`

FIG. 5.3. *Convergence characteristics for linear advection with* ***CONS-3*** *for wavelet* `4.0` *(* ●— *), wavelet* `4.2` *(* —● *), and a uniform grid (* —● *), as a function of ϵ_r (using $\epsilon_r/\epsilon_c = 100$), with the error evaluated at end time $t = 3$. In each plot, the subsequent data points for multiresolution simulations are obtained by systematically varying ϵ_r, whereas the uniform resolution data points are obtained by systematically varying the grid spacing h_f. Each marker symbol is associated with a unique value of h_f to facilitate the comparison across subgraphs.*

≈ 70 for wavelet 4.2), though this metric is heavily dependent on the scale separation of the simulation. A rough estimate implies that the ratio of the volume occupied by a sphere of radius 3σ and the volume of the rectangular domain is ≈ 70, similar to the compression rate of wavelet 4.2.

Effect of coarsening threshold. In the previous section we varied ϵ_r while keeping the ratio ϵ_r/ϵ_c fixed. Repeating the analysis for a range of values for ϵ_r/ϵ_c does not significantly change the results, as shown in Figure 5.4(b). Here the dotted, dashed, and solid lines correspond to $\epsilon_r/\epsilon_c = 16$, $\epsilon_r/\epsilon_c = 100$, and $\epsilon_r/\epsilon_c = 10^4$, respectively, and each data point for each simulation is associated with a given value of ϵ_r. For both wavelets, the number of blocks associated with a given error decreases slightly when ϵ_r/ϵ_c decreases, reflecting the more aggressive coarsening of the grid when ϵ_c increases. At all points, except the finest ϵ_r for wavelet 4.0, increasing ϵ_c for a given ϵ_r decreases the number of blocks without significantly changing the error. This emphasizes the capability of wavelets to detect where to compress information without degrading the overall accuracy of the solution and shows that generally a simulation should take ϵ_r/ϵ_c close to the lower bound of 2^N (explained in subsection 2.2.5), with N the wavelet order.



(a) evolution of the number of blocks over time for various values of ϵ_r, for $\epsilon_r/\epsilon_c = 100$

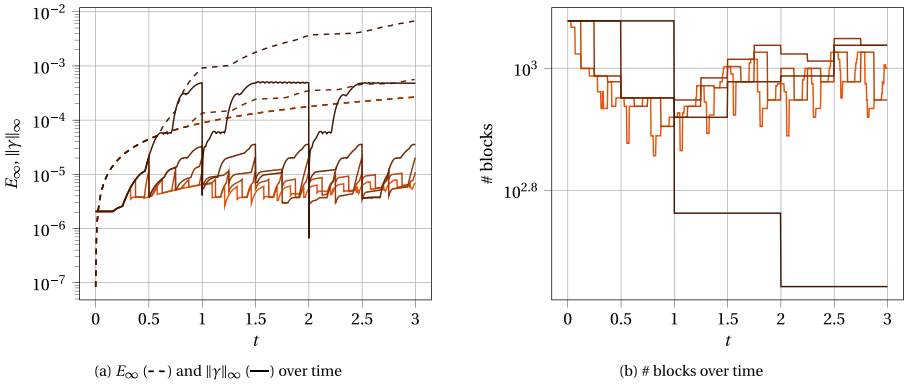
(b) wavelet-based convergence: E_∞ vs the number of blocks (at $t = 3.0$), with $\epsilon_r/\epsilon_c = 16$ (⋯), $\epsilon_r/\epsilon_c = 100$ (- -), and $\epsilon_r/\epsilon_c = 10^4$ (—).

FIG. 5.4. *Evolution of the number of blocks in the grid over time (left), and maximum error against number of blocks (right) for the linear advection testcase with* `CONS-3` *using wavelet* `4.0` *(●—●), wavelet* `4.2` *(●—●), and a uniform grid (●—●), for various values of ϵ_r's. In* (b)*, we also show the effect of varying ϵ_r/ϵ_c using different line styles.*

Effect of adaptation frequency. For a fixed $\epsilon_r = 10^{-5}$, $\epsilon_r/\epsilon_c = 100$, and wavelet 4.2 we vary the adaptation frequencies ranging from every 6 (as in the cases above) to every 768 time steps. With constant value of CFL $= 1/4$ this corresponds to the signal traveling from 1/16th of a block to 8 blocks at the finest scale between adaptations. As mentioned in subsection 2.2.5, there are two expected effects that occur when increasing the adaptation frequency: a delayed coarsening of blocks where the detail coefficient falls below the coarsening threshold, and a delayed refinement of blocks where the detail coefficient exceeds the refinement threshold. Compared to a more frequently adapted grid, the former will reduce the compression rate of the adapted grid whereas the latter will increase the error of the PDE solution.

The results of our numerical experiment are shown in Figure 5.5 with the error and maximum detail evolution for all cases over time (left) and the evolution of the number of blocks (right). Increasing the adaptation frequency up to and including every 192 time steps increases the number of blocks as grid compression is delayed, but does not strongly affect the error. When the adaptation frequency is every 384 time steps or lower, we observe that there are prolonged periods of time where the maximum detail coefficient exceeds the refinement threshold 10^{-5}, and the error starts to increase significantly compared to the other cases. For the lowest frequency (adaptation every 768 time steps), the maximum level in the grid drops to 2 compared to 3 for all other cases, as details of the signal are progressively lost and the grid is compressed accordingly. For this particular testcase, we thus find some robustness in the results to the adaptation frequency between values of 6 and 192, partly due to the smooth nature of the function and partly due to our 2:1 constraint in adaptation approach. Both features ensure that the grid adapted at a single time is able to capture and evolve the solution well when the signal travels up to 2 fine-level blocks afterward.

5.2. Deformation of a Gaussian blob. Here we present the results of our framework on a more challenging scale separation problem and showcase the ability of the wavelet-based adaptation to track the need for computational resources. Specifically, we use the advection equation to transport a Gaussian blob in a nonlinear periodic incompressible flow field defined as



(a) E_∞ (- - -) and $\|\gamma\|_\infty$ (—) over time

(b) # blocks over time

FIG. 5.5. *Effect of adaptation frequency for the linear advection testcase with* `CONS-3` *using wavelet* 4.2. *The number of time steps between successive adaptations shown is* 6 *(—),* 96 *(—),* 192 *(—),* 384 *(—), and* 768 *(—). With* CFL $= 1/4$*, this corresponds to the information traveling* $1/16$*th of a block,* 1 *block,* 2 *blocks,* 4 *blocks, and* 8 *blocks, respectively.*

$$
\begin{aligned}
u(\mathbf{x}) &= \sin^2(\pi\, x)\ \sin(2\pi\, y)\ \sin(2\pi\, z),\\
v(\mathbf{x}) &= \sin(2\pi\, x)\ \sin^2(\pi\, y)\ \sin(2\pi\, z),\\
w(\mathbf{x}) &= \sin(2\pi\, x)\ \sin(2\pi\, y)\ \sin^2(\pi\, z),
\end{aligned}
\tag{5.2}
$$

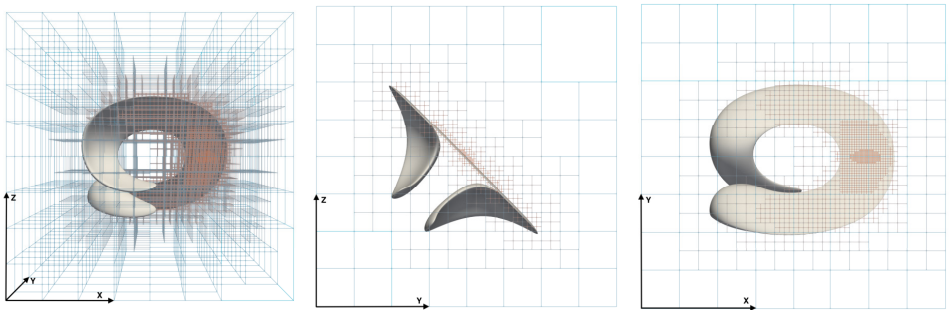
which was proposed originally in [27] and has been used extensively since in the level-set community since [16]. Following the latter we multiply the velocity components by $\cos(\pi t/3)$ and here evaluate only the "forward" evolution up to $t = 1.5$. As initial condition we choose a compact spherically symmetric Gaussian blob defined in terms of the radial coordinate r as

$$
\phi_0(r) = \exp\left[-\frac{r^2/\sigma^2}{1 - r^2/\beta^2}\right] \tag{5.3}
$$

for $r < \beta$ and with $\sigma = 0.1$ and $\beta = 2\sigma$, centered at $[0.35\ ,\ 0.35\ ,\ 0.35]$ within a unit cube. We apply grid adaptation tolerances of $\epsilon_r = 10^{-2}$ and $\epsilon_r/\epsilon_c = 100$, use CFL $= 0.5$, and adapt the grid every 12 time steps.

At $t = 0$, the grid contains only 92 blocks on levels 2 and 3, as the Gaussian is smooth and can be captured by the fourth-order wavelets at relatively coarse resolution. As the blob deforms, it thins rapidly near the center of the domain, as shown through visualization of an isosurface in Figure 5.6. This process triggers refinement throughout the evolution leading eventually to approximately 11,500 blocks, and levels from 2 all the way to 8. The refinement pattern at the final time is interesting, as the grid reflects the subtleties in the wavelet analysis: the maximum levels are localized concentrated exactly where we expect a fourth-order polynomial interpolation to show the largest errors, near the center of the domain. The "tips" of the deformed shape are still smooth and, despite relatively large gradients, still captured well on much coarser resolutions. From a user perspective, we note that we do not need to bound the maximum level; instead, the consistency of the refinement criterion with the wavelet-based grid adaptation ensures that setting a suitable ϵ_r is sufficient to keep the levels in the grid within reasonable bounds.

Finally, we study the ability of the ϵ_r criterion as an input to control the solution error in this nonlinear problem. As the equations have no analytical solutions,



(a) 3D view (b) plane YZ (c) plane XY

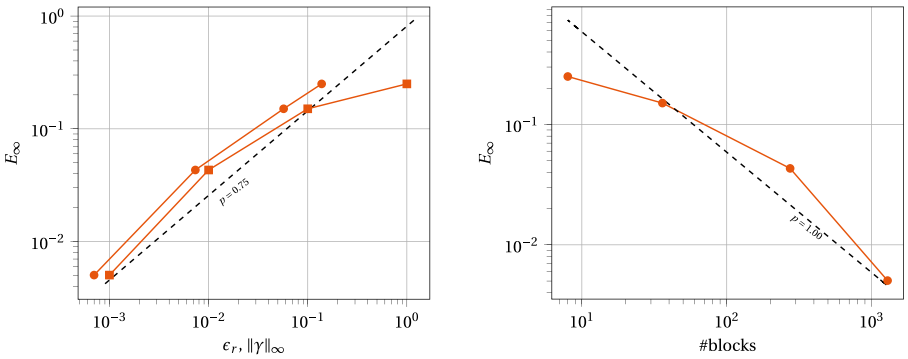
FIG. 5.6. *Advection of a compact Gaussian scalar field using a deformation velocity field: 3D and perspective views of the grid together with an isosurface of the field at value $\phi = 0.25$ for $t = 1.5$. We have visualized the outlines of individual blocks to highlight the structure of the grid.*

we perform a self-convergence study where the result of a uniform-grid simulation at level 5 (or 768^3 grid points) at time $t = 0.5$ is used as reference solution. For the multiresolution cases, we use wavelet 4.2 and consider four different refinement thresholds $\epsilon_r \in \{1\ ;\ 10^{-1}\ ;\ 10^{-2}\ ;\ 10^{-3}\}$. We set the time step as $\Delta t = 1/2\ h_f/U$, with h_f the instantaneous finest spacing in the grid, and U fixed to be $U = 1$ at all times. None of these multiresolution simulations reaches level 5 within this time window. Once the multiresolution simulations have reached $t = 0.5$, we refine their multilevel grids uniformly to level 5 using the chosen wavelet, and then compute the maximum error between the refined uniform grid and the reference uniform resolution simulation result.

The evolution of the error with the refinement criterion is shown in Figure 5.7(a), where we observe that the slope matches the theoretical convergence rate of 3/4 for both the refinement criterion ϵ_r and the measured maximum detail in the domain $\|\gamma\|_\infty$ reasonably well. In Figure 5.7(b) we show the error as a function of the number of blocks at the final time. This metric is more sensitive as the number of blocks varies in time; in this particular case, the second-last point at $\epsilon_r = 10^{-2}$ just refined the grid prior to $t = 0.5$ which has moved the data point to the right compared to the expected slope. Nevertheless, this convergence analysis demonstrates that the main results of the linear transport problem described in subsection 5.1 can still be reproduced for this more challenging, nonlinear transport problem as well.

6. Weak scalability and performance analysis. In this section we present the results of the weak scalability campaign we have done on Cori, a Cray XC40 supercomputer whose CPU partition contains 2388 nodes of 32 cores each.[1] The testcase is the execution of 50 times steps for the advection of a Gaussian tube aligned with the z-direction in a rectangular domain. The size of the domain, and therefore the size of the simulation S, is adapted to maintain the number of blocks per core constant: $S(N_c) = [1 \times 1 \times N_c/32]$, with N_c the number of cores. To get a significant sample of each operation, we perform the adaptation every time step, although in practice we would adapt less frequently. The domain is initialized with 75.5 blocks per rank and one rank per core, which with a block size of 24^3 leads to about 1 million unknowns per core. We used the wavelet 4.0 and CFL = 0.25.

[1]The MPI implementation used is `OpenMPI/4.1.2.`



(a) E_∞ vs ϵ_r ( ), and vs $\|\gamma\|_\infty$ ( ) with $\epsilon_r/\epsilon_c = 100$.

(b) E_∞ vs # blocks.

FIG. 5.7. *Advection of a compact Gaussian scalar field using a deformation velocity field: wavelet-based convergence analysis.*

We study the performance of our framework from 1 nodes to 512 nodes, with the latter corresponding to 16,384 cores and about 17 billion unknowns. In Figure 6.1(a) we show the evolution of the time spent per time step for the grid adaptation (orange) and the stencil computation (blue). We also show separately the time spent in ghost computations (green), which is the communication-heavy part of the stencil computation. Though highly problem-dependent, for this particular problem the grid adaptation takes roughly half the time of the stencil computation within an RK3 time step, which is very reasonable given that we typically would need to adapt the grid only every 10 to 100 time steps. Further, the ghosting takes up about a third of the time of a stencil computation, even on very large partitions.

Based on this timing data, we show in Figure 6.1(b) the weak efficiency η_w defined as

$$\eta_w(N_c) = \frac{T\left(N_{\text{ref}}, S\left(N_{\text{ref}}\right)\right)}{T\left(N_c, S\left(N_c\right)\right)} \quad , \tag{6.1}$$

where $T\left(N, S\left(N\right)\right)$ is the time taken by the simulation to run a problem of size S on N cores and where we used $N_{\text{ref}} = 64$, i.e., 2 nodes.

Based on Figures 6.1(a) and 6.1(b), the PSCW strategy allows us to reach a perfect scalability in the case of the ghost and the stencil operation. This observation is confirmed by analyzing the breakdown of the different operations illustrated in Figure SM5.1. This result is only possible because both operations are scalable and done on group of ranks with the PSCW calls, instead of using the entire communicator. We do pay a price to achieve this scalability in the ghost computation, because we have to reinitialize the ghost meta-data structure every time we modify the grid. As observed in Figure 6.1(b), this adaptation process has a lower parallel efficiency. Analyzing the timing of different operations within the adaptation step (Figure SM5.3) shows that the less scalable operations are the synchronization step, which contains a nonblocking `MPI_Allreduce` and various RMA synchronizations, and the `reset` operation of the meta-data for the ghosting, which involves the creation and deletion of the window on `MPI_COMM_WORLD`. For the latter we currently use nondynamic windows as we do not expect the adaptation to be called often compared to the use of the ghosting and the stencils, and so its computation cost will not dominate in practice. However, in the future we could still improve the implementation by considering a dynamic window

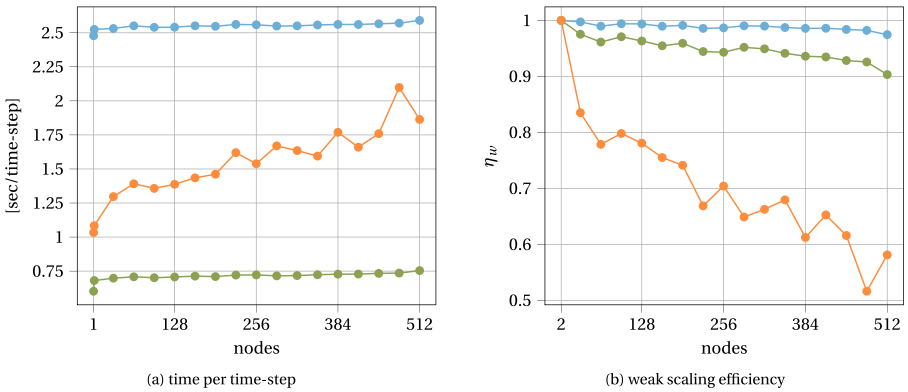


FIG. 6.1. *Weak scaling for the three main operations involved in adaptive grid simulations: the grid adaptation (, performed every time step here), the stencil operations ( ), and the time spent in the ghost computations ( ) during the stencil evaluation. The times are given in seconds per time step, from* 1 *node to* 512 *nodes (*32 *cores to* 16,384 *cores).*

allocation, which might be better suited for a policy that requires very frequent adaptations. Overall, the observed behavior of this weak scalability test is consistent with the implementation choices, as all the global operations are less scalable by definition, while the local operations demonstrate excellent scalability.

7. Conclusion. This work provides a detailed explanation of the mathematical foundation and distributed computational implementation of a 3D block-structured adaptive grid method based on a multiresolution analysis using wavelets, named `murphy`. In our approach we apply significant emphasis on the handling of resolution jumps in block-structured grids to provide consistency with nonlifted and lifted interpolating wavelets of second, fourth, and sixth polynomial order. We validate the implementation through rigorous tests of error control and moment conservation on static grids.

Compared with most existing adaptive mesh refinement approaches, the wavelet-based approach provides explicit grid adaptation metrics that are intrinsically linked to the pointwise error made in the field compared with a polynomial interpolation. The wavelet framework provides a consistent multiresolution perspective for adaptation metric, adaptation procedure, and ghost reconstruction across resolution jumps, with a formal separation of scales in all operations. When combined with finite-difference schemes we demonstrate the ability of our approach to reach pointwise high-order convergence on multilevel grids. Further, the nature of the nonoverlapping octree-based grids with constant-size blocks provides excellent opportunities for scalability. We exploit this in our implementation using state-of-the-art one-sided communication strategies that show excellent scalability of the grid adaptation, ghost reconstruction, and stencil computation processes at least up to 16,384 cores.

We tested our multiresolution adaptive grid algorithm on the convergence of simple linear hyperbolic conservation laws in the form of scalar advection using divergence-free velocity fields. The results demonstrate that the refinement threshold, which provides user control over the error permitted as detected by the wavelet analysis, is an excellent indicator for the global field error even during the evolution of this PDE. As such, reducing the refinement threshold leads to convergence of the error, along a slope that can be captured by the ratio of the convergence order of the

finite-difference scheme and the polynomial order of accuracy of the wavelet used. For lifted wavelets we observe that the global maximum error on any multilevel grid is very close to the error on a uniform resolution grid of the same maximum resolution. Compared to nonlifted wavelets, the reduced aliasing properties of lifted wavelets enable them to detect the need for refinement and opportunity for compression more efficiently, leading ultimately to a smaller number of blocks in the grid by about a factor of two for the same maximum error.

The implementation and performance benefits of our octree-based block-structured grid implementation does lead to constraints in terms of the adaptation patterns that can be achieved, for instance, compared to patch-based adaptive mesh refinement techniques. The use of "forest-of-trees" as offered by `p4est` enables us to create arbitrary aspect ratio rectangular domains, rather than remain restricted to cubic domains as in previous codes [34]. However, our current implementation is still limited to cubic blocks with isotropic grids. To gain more flexibility, one can build on domain mapping techniques [5] or consider extending work in computer graphics that generalizes wavelets to work on arbitrary compact surfaces [28]. Closer to our approach, it should also be relatively straightforward to use a larger set of blocks that enable separate refinement or compression in each of the three cartesian directions. We reserve this extension for future work.

With the fundamentals and implementation of our framework presented, future work will focus on investigating the performance of the wavelet-based refinement criterion to accurately capture refinement requirements for nonlinear PDEs. Previous work on wavelet-adapted grids is promising in this regard (e.g., [34]) but systematic investigations are lacking. At the same time, we note that this level of rigor is also often absent from other adaptive mesh refinement methodologies, which typically rely on heuristic or post hoc criteria for refinement and compression that make it difficult to compare their ability to capture emerging scales, or discard information that does not affect the overall error. In our case, however, the formal wavelet multiresolution theory provides a useful perspective to frame this discussion and analyze convergence with respect to the refinement threshold, compared with other possible policies. Further, we wish to explore the benefits of moment conservation offered by lifted wavelets as demonstrated in our solver, for the solution of conservative PDEs such as the hyperbolic conservation law considered in this work. Finally, we intend to combine our multiresolution framework with an elliptic solver to handle problems such as the free-space incompressible Navier Stokes equations, as well as an immersed interface method [20, 19] which will enable us to perform high-order simulations with embedded interfaces for multiphysics problems.

Acknowledgments. We would like to acknowledge the insightful discussions with James Gabbard, Matthieu Duponcheel, Pierre Balty, and Philipe Chatelain, as well as the help received from Howard Pritchard and Greg Hernandez to run the PSCW MPI-RMA on different infrastructures.

REFERENCES

[1] M. Adams, P. Colella, D. T. Graves, J. Johnson, N. Keen, T. J. Ligocki, D. F. Martin, P. McCorquodale, D. Modiano, P. Schwartz, T. Sternberg, and B. V. Straalen, *Chombo Software Package for AMR Applications Design Document*, Tech. report, Lawrence Berkeley National Laboratory, 2019.

[2] M. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84, https://doi.org/10.1016/0021-9991(89)90035.

[3] M. J. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512, https://doi.org/10.1016/0021-9991(84)90073.
[4] C. Bernard, *Wavelets and Ill Posed Problems: Optic Flow and Scattered Data Interpolation*, PhD thesis, Centre de Mathematiques Appliquees, 1999.
[5] E. Brown-Dymkoski and O. V. Vasilyev, *Adaptive-anisotropic wavelet collocation method on general curvilinear coordinate systems*, J. Comput. Phys., 333 (2017), pp. 414–426, https://doi.org/10.1016/j.jcp.2016.12.040.
[6] C. Burstedde, L. Wilcox, and O. Ghattas, *p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees*, SIAM J. Sci. Comput., 33 (2011), pp. 1103–1133, https://doi.org/10.1137/100791634.
[7] D. Calhoun and C. Burstedde, *ForestClaw: A Parallel Algorithm for Patch-Based Adaptive Mesh Refinement on a Forest of Quadtrees*. arXiv:1703.03116, 2017.
[8] A. Cohen, *Adaptive Methods for PDE: Wavelets or Mesh Refinement?*, arXiv:math/02144, 2002.
[9] A. Cohen, I. Daubechies, and J.-C. Feauveau, *Biorthogonal bases of compactly supported wavelets*, Comm. Pure Appl. Math., 45 (1992), pp. 485–560, https://doi.org/10.1002/cpa.3160450502.
[10] I. Daubechies and W. Sweldens, *Factoring wavelet transforms into lifting steps*, J. Fourier Anal. Appl., 4 (1998), pp. 247–269, https://doi.org/10.1007/BF02476026.
[11] G. Deslauriers and S. Dubuc, *Interpolation dyadique*, in Fractals, Dimensions non-entieres et applications, Masson, Paris, 1987.
[12] G. Deslauriers and S. Dubuc, *Symmetric iterative interpolation processes*, Const. Approx., 5 (1989), pp. 49–68, https://doi.org/10.1007/BF01889598.
[13] D. Donoho, *Interpolating Wavelet Transforms*, Tech. report EFS NSF 408, Stanford University, 1992.
[14] D. Donoho and T. Yu, *Deslauriers Dubuc Ten Years After*, tech. report, Stanford University, 1996.
[15] T. Engels, K. Schneider, J. Reiss, and M. Farge, *A wavelet-adaptive method for multiscale simulation of turbulent flows in flying insects*, Commun. Comput. Phys., 30 (2021), pp. 1118–1149.
[16] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, *A hybrid particle level set method for improved interface capturing*, J. Comput. Phys., 183 (2002), pp. 83–116, https://doi.org/10.1006/jcph.2002.7166.
[17] G. Fernandez, S. Periaswamy, and W. Sweldens, *LIFTPACK: A software package for wavelet transforms using lifting*, Proc. SPIE, 2825 (1996), https://doi.org/10.1117/12.255250.
[18] K. J. Fidkowski and D. L. Darmofal, *Review of output-based error estimation and mesh adaptation in computational fluid dynamics*, AIAA J., 49 (2011), pp. 673–694, https://doi.org/10.2514/1.J050073.
[19] J. Gabbard, T. Gillis, P. Chatelain, and W. M. van Rees, *An immersed interface method for the 2D vorticity-velocity Navier-Stokes equations with multiple bodies*, J. Comput. Phys., 464 (2022), 111339, https://doi.org/10.1016/j.jcp.2022.111339.
[20] T. Gillis, Y. Marichal, G. Winckelmans, and P. Chatelain, *A 2D immersed interface vortex particle-mesh method*, J. Comput. Phys., 394 (2019), pp. 700–718, https://doi.org/10.1016/j.jcp.2019.05.033.
[21] S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge-Kutta schemes*, Math. Comp., 67 (1998), pp. 73–85, https://doi.org/10.1090/S0025-5718-98-00913-2.
[22] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112, https://doi.org/10.1137/S003614450036757X.
[23] W. Gropp, T. Hoefler, R. Thakur, and E. Lusk, *Using Advanced MPI: Modern Features of the Message-Passing Interface*, MIT Press, Cambridge, MA, 2014.
[24] A. Harten, *Multiresolution algorithms for the numerical solution of hyperbolic conservation laws*, Comm. Pure Appl. Math., 48 (1995), pp. 1305–1342, https://doi.org/10.1002/cpa.3160481201.
[25] R. D. Hornung and S. R. Kohn, *Managing application complexity in the SAMRAI object-oriented framework*, Concurrency Computation Practice Experience, 14 (2002), pp. 347–368, https://doi.org/10.1002/cpe.652.
[26] N. K. R. Kevlahan and O. V. Vasilyev, *An adaptive wavelet collocation method for fluid-structure interaction at high reynolds numbers*, SIAM J. Sci. Comput., 26 (2005), pp. 1894–1915, https://doi.org/10.1137/S1064827503428503.

[27] R. J. LeVeque, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM J. Numer. Anal., 33 (1996), pp. 627–665, https://doi.org/10.1137/0733033.
[28] M. Lounsbery, T. D. DeRose, and J. Warren, *Multiresolution analysis for surfaces of arbitrary topological type*, ACM Trans. Graph., 16 (1997), pp. 34–73, https://doi.org/10.1145/237748.237750.
[29] S. G. Mallat, *Multiresolution approximations and wavelet orthonormal bases of $l^2(r)$*, Trans. Amer. Math. Soc., 315 (1989), pp. 69–87, https://doi.org/10.2307/2001373.
[30] S. G. Mallat, *A theory for multiresolution signal decomposition: The wavelet representation*, IEEE Trans. Pattern Anal. Machine Intelligence, 11 (1989), pp. 674–693, https://doi.org/10.1109/34.192463.
[31] Y. Meyer, *Wavelets and Operators*, Vol. 1, Cambridge University Press, Cambridge, UK, 1993, https://doi.org/DOI:10.1017/CBO9780511623820.
[32] S. Popinet, *Gerris: A tree-based adaptive solver for the incompressible euler equations in complex geometries*, J. Comput. Phys., 190 (2003), pp. 572–600, https://doi.org/10.1016/S0021-9991(03)00298-5.
[33] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 3rd ed., Cambridge University Press, Cambridge, UK, 2007.
[34] D. Rossinelli, B. Hejazialhosseini, W. van Rees, M. Gazzola, M. Bergdorf, and P. Koumoutsakos, *MRag-I2d: Multi-resolution adapted grids for remeshed vortex methods on multicore architectures*, J. Comput. Phys., 288 (2015), pp. 1–18, http://dx.doi.org/10.1016/j.jcp.2015.01.035.
[35] C. Roy, *Strategies for Driving Mesh Adaptation in CFD (Invited)*, Aerospace Sciences Meetings, American Institute of Aeronautics and Astronautics, 2009, https://doi.org/doi:10.2514/6.2009-1302.
[36] K. Schneider and O. V. Vasilyev, *Wavelet methods in computational fluid dynamics*, Annu. Rev. Fluid Mech., 42 (2009), pp. 473–503, https://doi.org/10.1146/annurev-fluid-121108-145637.
[37] W. Sweldens, *The lifting scheme: A custom-design construction of biorthogonal wavelets*, Appl. Comput. Harmon. Anal., 3 (1996), pp. 186–200.
[38] W. Sweldens, *The lifting scheme: A construction of second generation wavelets*, SIAM J. Math. Anal., 29 (1998), pp. 511–546, https://doi.org/10.1137/S0036141095289051.
[39] W. Sweldens and R. Piessens, *Quadrature formulae and asymptotic error expansions for wavelet approximations of smooth functions*, SIAM J. Numer. Anal., 31 (1994), pp. 1240–1264, https://doi.org/10.1137/0731065.
[40] W. Sweldens and P. Schroder, *Building your own wavelets at home*, in Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques, 1996.
[41] J. A. van Hooft and S. Popinet, *A fourth-order accurate adaptive solver for incompressible flow problems*, J. Comput. Phys., 462 (2022), 111251, .
[42] W. van Rees, *3D Simulations of Vortex Dynamics and Biolocomotion*, Ph.D. thesis, ETH Zurich, 2014.
[43] O. V. Vasilyev and C. Bowman, *Second-generation wavelet collocation method for the solution of partial differential equations*, J. Comput. Phys., 165 (2000), pp. 660–693, http://dx.doi.org/10.1006/jcph.2000.6638.
[44] 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 J. Sci. Comput., 34 (2012), pp. B179–B201, https://doi.org/10.1137/110820105.
[45] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, M. P. Katz, A. Myers, T. Nguyen, A. Nonaka, M. Rosso, S. Williams, and M. Zingale, *AMReX: A framework for block-structured adaptive mesh refinement*, J. Open Source Software (2019).