SANDIA REPORT

SAND2005-7981
Unlimited Release
Printed January 2006

Using high-order methods on adaptively refined block-structured meshes — discretizations, interpolations, and filters

J. Ray, C. A. Kennedy, S. Lefantzi, and H. N. Najm,
Sandia National Laboratories, CA

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories. operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2005-7981
Unlimited Release
Printed January 2006

Using high-order methods on adaptively refined block-structured meshes – discretizations, interpolations, and filters

J. Ray
Advanced Software R. & D.,
Sandia National Laboratories
P. O. Box 969, Livermore CA 94550
jairay@ca.sandia.gov

C. A. Kennedy, S. Lefantzi and H. N. Najm
Reacting Flow Research
Sandia National Laboratories
P. O. Box 969, Livermore CA 94550
cakenne@speakeasy.net, lefantzi@comcast.net, hnnajm@ca.sandia.gov.

Abstract

Block-structured adaptively refined meshes (SAMR) strive for efficient resolution of partial differential equations (PDEs) solved on large computational domains by clustering mesh points only where required by large gradients. Previous work has indicated that fourth-order convergence can be achieved on such meshes by using a suitable combination of high-order discretizations, interpolations, and filters and can deliver significant computational savings over conventional second-order methods at engineering error tolerances. In this paper, we explore the interactions between the errors introduced by discretizations, interpolations and filters. We develop general expressions for high-order discretizations, interpolations, and filters, in multiple dimensions, using a Fourier approach, facilitating the high-order SAMR implementation. We derive a formulation for the necessary interpolation order for given discretization and derivative orders. We also illustrate this order relationship empirically using one and two-dimensional model problems on refined meshes. We study the observed increase in accuracy with increasing interpolation order. We also examine the empirically observed order of convergence, as the effective resolution of the mesh is increased by successively adding levels of refinement, with different orders of discretization, interpolation, or filtering.

Acknowledgment

This work was supported by the US Department of Energy (DOE), Office of Basic Energy Sciences, Division of Chemical Sciences, Geosciences, and Biosciences, SciDAC (Scientific Discovery through Advanced Computing) Computational Chemistry Program. Support was also provided by the DOE Office of Advanced Scientific Computing Research SciDAC program.

Contents

1 Introduction 7
2 Structured Adaptive Mesh Refinement 9
2.1 Derivatives 11
2.2 Filters 15
2.3 Hyperviscosity 18
2.4 Interpolants 18
3 Test Problems 23
3.1 Appropriate interpolation orders 23
3.2 Achieving high-order convergence 27
4 Conclusions 32
References 35

Appendix

A Stencil coefficient computation 37
B Interior Stencils for 3D interpolants 41
C Rules for pairing a derivative and an interpolation stencil 43

Figures

1 A vertex-centered grid. The large circles are the coarse mesh, and the fine mesh points at edge-centers, face-centers and body-centers are represented by triangles, squares and a (small) circle. 10

2 A soliton solution of Eq. 26 at $t = 0$ and 1. Symbols are used to plot the solution on L_0 and lines on L_1. The red line shows the solution at $t = 1.0$. Inset: A detail of the L_1 solution.We see that the movement of soliton during the simulation is small vis-a-vis its size. 24

3 Convergence of the "right-hand-side" of the KdV equation on L_1 with $p_D = 4$ and $p_I = 4, 6$ and 8. The time is $t = 1.0$. A uniform mesh run is plotted as a guide for the convergence slope. $p_I = 8$ shows a convergence steeper than fourth order while the others do not. Note that the resolution here is L_0 resolution; L_1 is a factor of 2 finer. The effective resolution of a $L_0 = 50$ run is a uniform mesh of 100 and is compared with a uniform mesh run corresponding to the effective resolution. E_1 is the RMS error (with respect to the exact solution) on Level1; E_0 is the error for the uniform mesh run. 25

4 A traveling wave solution of Eq. 27 at $t = 0$ and 0.5. Symbols are used to plot the solution on L_0 and lines on L_1. The red line shows the solution at $t = 0.5$. 26

5 Convergence of the "right-hand-side" of the KS equation on L_1 with $p_D = 4$ and $p_I = 6$ and 8. The time is $t = 0.5$. A ideal fourth-order convergence plot is provided for guidance. $p_I = 8$ shows a convergence steeper than fourth order while the others do not. Note that the resolution here is L_0 resolution; L_1 is a factor of 2 finer.E_1 is the RMS error on Level1. 27

6 Runge phenomenon on a 2-level block-structured mesh. The patch outlined in black is the Level1 patch. Level0 is a 100×100 mesh on a unit square. The solution is at $t = 5 \times 10^{-5}$ and was computed with $p_D = 4$ and $p_I = 6$. The correct solution should not contain the structure/oscillations one observes at southeast and northwest corners of the Level1 patch. .. 28

7 Numerical errors (calculated using the exact solution) for the FitzHugh-Nagumo equations solved using fourth-order discretizations. Numerical solutions computed with $p_F = 6, 8$ and 10 on grid hierarchies with up to 3 levels of refinement are plotted with □, △ and ▽ respectively. $(p_D)^{th}$ and $(p_F)^{th}$-order convergence are plotted with solid and dashed lines respectively. $p_F = 6, 8$ and 10 are represented by red, blue and black respectively. For $p_F \geq 8$, we see fourth-order convergence as filter errors are smaller than discretization errors. The y-axis has the Level0 RMS error (E_0) and the "ideal" errors if the discretization or the filter errors dominate. 29
8 Numerical (RMS) errors observed when ($p_D = 4, p_F = 8$). Solutions were done on 50×50 (black), 100×100 (red) and 200×200 (blue) meshes with different levels of refinement. Errors on Level 0, 1, 2 and 3 are plotted with □, △, ▽ and ◇ respectively. Ideal fourth-order convergence is plotted using solid lines. Apart from the 50×50 uniform mesh run, the errors follow the ideal convergence closely. $\mathrm{E_i}, \mathrm{i} = 0, \ldots 3$ refer to the RMS error on level i. 30
9 The RMS difference between numerical and exact solution for the FitzHugh-Nagumo equations solved on a $N \times N$ uniform mesh using $p_D = 6$ with filters of order $p_F = 8, 10$ and 12. Errors from runs without filtering are also plotted. △, ▽, ○ are used to denote $p_F = 8, 10$ and 12 results; □ is used for runs without filtering. 31
10 Comparison of the RMS difference between exact and numerical solutions of the FitzHugh-Nagumo equations when computed with $p_D = 4, p_I = 8$ and $p_D = 6, p_I = 8$ and filtered with filters of order $p_F = 8$ and 10. $\Delta t = 10^{-6}$ and the problem is integrated till $t = 4 \times 10^{-4}$. Results are plotted for a uniform mesh and meshes with 1 to 3 levels of refinement with a Level0 mesh of 100×100. △ and ▽ indicate $p_F = 8$ and 10; dashed lines indicate $p_D = 4$ and solid ones $p_D = 6$. E_0 is the RMS error on Level0. 32

Tables

1 Number of grid points per wavelength required to obtain a chosen relative error tolerance ε using explicit, centered-difference, midpoint, interpolant operators of orders two through ten. See table 13 for stencil coefficients. 11
2 Stencil coefficients for centered and upwinded $\partial/\partial x$ operators on uniform grids. 13
3 Boundary stencil coefficients of lower-order, $\partial/\partial x$ operators. 14
4 Stencil coefficients for centered $\partial^2/\partial x^2$ operators on uniform grids. 14
5 Coefficients of $\partial^2/\partial x^2$ operators near boundary points. 14
6 Stencil coefficients for centered $\partial^3/\partial x^3$ operators on uniform grids. 15
7 Coefficients of $\partial^3/\partial x^3$ operators near left boundary points. 15
8 Stencil coefficients for centered $\partial^4/\partial x^4$ operators on uniform grids. 15
9 Coefficients of $\partial^4/\partial x^4$ operators near left boundary points. 16
10 Stencil coefficients for centered $\partial^5/\partial x^5$ operators on uniform grids. 16
11 Coefficients of $\partial^5/\partial x^5$ operators near left boundary points. 16
12 Stencil coefficients for centered $\partial^{2n}/\partial x^{2n}$ operators on uniform grids. 17
13 Coefficients of one-dimensional, vertex-centered, midpoint interpolants in terms of ξ. 19
14 Coefficients and truncation errors of two-dimensional, vertex-centered interpolants in terms of ξ and η. 21

Using high-order methods on adaptively refined block-structured meshes – discretizations, interpolations, and filters

1 Introduction

Numerical solution of space-time, partial differential equations (PDEs) is a very common undertaking. Using the method of lines and possibly finite-difference techniques applied to the spatial operators, one solves a large system of ordinary differential equations (ODEs). A critical aspect of solving these sometimes enormous problems is the efficiency of the spatial discretization. Ideally, one first establishes an error tolerance requirement for the simulations and then sets about the task of finding the method that will reliably deliver this tolerance at the minimum cost. In general, there are two simple guiding ideas to consider; high-order methods become relatively more efficient as error tolerances are reduced and grid points should be allocated more generously to less smooth parts of the solution. Combining these two in simple geometries, we will consider high-order spatial discretizations in conjunction with adaptive mesh refinement (AMR) for solution of PDEs having substantial spatial variation in smoothness. In particular, we will use Structured Adaptive Mesh Refinement (SAMR) [1, 2], where a layered hierarchy of rectangular uniform-mesh patches is used to discretize the domain. While this approach resolves a domain efficiently by refining a grid only where required, one would also like to minimize the number of mesh levels and this is facilitated by higher-order methods. Large multiphysics problems are typically solved on distributed memory parallel computers. These require that the domain of the solution be decomposed amongst processors in a load-balanced manner. If one tries to achieve a given spatial error tolerance by refining repeatedly, one gets localized regions of very high mesh density, distributed within an otherwise coarse mesh. This strongly inhomogeneous grid poses a challenge to domain-decomposition parallel load-balancing algorithms, which frequently fail to achieve a scalable decomposition. Thus, while AMR does allow one to concentrate mesh density in regions of interest, a certain moderation in its use adds tremendously to the ease and speed of computation.

Although adaptive mesh spatial discretizations are not new, high-order (i.e. > 2) versions of them are not common. If one forgoes domains with intricate geometries, one may use block-structured adaptive meshes and exploit the mesh's regularity to consider high-order finite difference methods. [3] addresses the problem of solving the Poisson equation to fourth order accuracy on block-structured adaptively refined meshes. Starting with a classical Mehrstellen method, the authors develop and test a fourth order solution methodology for the Poisson equation in 2D and 3D. Tests were done on an adaptive mesh (coarse base mesh with one extra level of refinement) and fourth order convergence was predicted theoretically and demonstrated empirically. Interpolations at coarse-fine boundaries were done using a sequence of high-order 1D interpolations, some fourth and others sixth order accurate, depending upon the configuration of patches. Their method requires that the refinement ratio between successive levels be four.

While differentiations and interpolations in structured AMR may be done using wavelets, we follow the approach of [4, 5, 6, 7, 8] where they are performed in physical space. [6] points out that AMR is more efficient than a traditional single grid method only when the higher wavenumber content of the flowfield is relatively nonuniform and resides within smaller regions of the domain. Because integration variables on each grid are eventually the result of interpolations based on local polynomials, one cannot differentiate this interpolated data indefinitely. Interpolated data based on an order p_I interpolant, which is differentiated k times (differentiation order), results in a field with maximum differentiation order of $p_I - k + 1$, i.e. $C^{p_I - k + 1}$

(see appendix C). Therefore if, e.g., viscous terms are included in Navier-Stokes computations using interpolants and derivative operators of identical order, one may observe an order reduction to order, $p_D - 1$, depending on the significance and resolution of the viscous terms. Alternative approaches to high-order AMR are discussed by Holmström [9] where wavelets are often central to the AMR procedure.

A key element of typical AMR constructions is that the more accurate solution from the finest grids is periodically interpolated onto the coarser ones and solutions from the coarse mesh are used at coarse-fine interfaces. The coarse-fine interface takes the form of a "halo" of cells/grid points around fine patches where the solution is interpolated from the underlying coarse mesh. Thus, an important issue in high-order AMR is dealing with the potential for the *Runge phenomenon* associated with interpolation on uniform grids. A key result of Trefethen and Weideman [10] is that for spatial modes of the grid variables, the error in interpolation of modes, $\exp(i\xi x)$, decreases to zero as the grid density tends to infinity if and only if ξ is small enough to provide six grid points per wavelength. This implies that high-order interpolation will only succeed if there is active control of the spatial wavenumbers that live on the computational grid. Central to doing this are robust refinement/coarsening criteria combined with careful addition of numerical dissipation for an otherwise low-dissipation numerical method. This robust refinement/coarsening strategy might act to ensure that the energy content of a grid variable above some chosen cut-off wavenumber is essentially zero. This also places the requirement that reliable spatial error estimates be available. Wavelets are one of the means for doing this [11]. One could, alternatively, employ right-hand-side (RHS) evaluations using stencils of different spatial accuracies to compute a relative error measure. One could also carry on two simultaneous computations, one on an "actual" grid and a second one on a "ghost" grid with a factor-of-two coarser resolution. The difference of the two solutions gives an estimate of the spatial discretization error [1]. However, approximate error estimation techniques based on first and second derivatives are more commonly used [12]. To facilitate the interpolations required in both refinement and coarsening, several approaches to high-wavenumber dissipation may be utilized. Filtering grid variables may be done directly using stand-alone filters. Indirectly, it may be accomplished by using upwinded derivative operators rather than centered-difference approximations in evaluating the RHS or a hyperviscosity term may be added. A robust refinement/coarsening strategy along with control of numerical dissipation is essential to an efficient high-order AMR method, but one does not appear to fully exist currently.

Several other outstanding issues must be considered when contemplating high-order AMR. One is the proper specification of interpolant boundary closures. It is well known that elaborate boundary closures must be used if one seeks uniformly high-order, finite-difference, first-derivative operators and time-stable integration[13, 14, 15, 16]. What is less clear is what, if anything, is required of finite-difference interpolant boundary closures above and beyond satisfying accuracy requirements. Do interpolant boundary closures affect time stability of the overall method? The literature offers no guidance on this matter. Another topic that may affect the efficiency of an AMR procedure is the selection of the Level0 grid spacing. Large initial grid spacings imply more grid levels, more interpolations, but fewer total grid points. Lower grid point count can be advantageous for expensive source terms like chemical reaction rates, more so if the chemical mechanism is stiff. If the refinement/coarsening apparatus is of high quality, interpolations are relatively cheap, and the domain decomposition is good then a relatively larger Level0 grid spacing seems appropriate. On the other hand, to the degree to which these three items are not met, a smaller Level0 grid spacing seems more prudent. When determining the number of mesh levels, one must also remember that the smallest acceptable grid spacing on any grid level is a function not just of the governing equations but also of the numerical dissipation.

Although AMR with high order discretizations is appealing and potentially useful, it complicates the time integration strategy. Each mesh level has its own grid spacing and consequently the stiffness of the convec-

tive and diffusive eigenvalues vary inversely and inversely squared with the grid spacing, respectively. If convection and diffusion are treated explicitly in time, then there will be a large range in stepsizes corresponding to the stability limit based upon convection and diffusion on each grid. One may approach this by integrating all grids with a uniform stepsize. In this way, the formal order of traditional ODE integrators is maintained, however, the global timestep is restricted by the stability constraint on the finest grid. Alternatively, a "time-refined" approach can be used, where different time step sizes are employed for each mesh level [2]. Typically, this requires *recursive* integration on the various mesh levels, starting with the coarsest level (at a timestep determined by the stability constraint of the mesh level itself), followed recursively by the integration of its children mesh levels, which are processed more often but at their (smaller) stability-constrained timesteps. For example, in a viscous CFL dominated problem on a 4-level SAMR mesh, the finest grid takes 2^8 steps for every one on the coarsest. Generally, this approach constitutes what is known as *subcycling*.but may also be approached more rigorously using partitioned multi-rate [17] methods. This approach constitutes partitioning of the right-hand-side (RHS) at the mesh level. Applying a non-multirate integrator in a multirate context may often return less than the formal rate of convergence. One may further optimize the integration strategy by treating convection, diffusion, and reaction using different integration methods. A partitioning like this is often referred to as an additive partitioning [18]. Alternatively, A- and L-stable [19] implicit schemes may ignore stability restrictions and select step-sizes based exclusively on accuracy and iteration concerns. Such methods usually result in linear systems which are solved by iterative techniques.

In [20] we demonstrated, using a 1D FitzHugh-Nagumo equation, that fourth-order convergence could be achieved on block-structured adaptively refined meshes. Details of the choice of interpolations and filters, and the interactions between them, were omitted; instead we extended our method to more realistic problems by solving (on 28 processors) a coupled system of 10 reaction-diffusion equations modeling the ignition of H_2 in air. The goal of this paper is to demonstrate high-order spatial convergence in multidimensions on adaptively-refined Cartesian grids by using finite difference differentiation, interpolation, and dissipation. It is beyond the scope of the paper to address all of the topics relevant to an efficient, high-order spatial scheme - for example, we will not address high-order temporal discretizations or time stable boundary closures for interpolant operators. Instead, we focus on demonstrating methods that give high-order convergence, and begin to address effective strategies for retaining high order but avoiding Runge phenomena. Runge phenomena are avoided by using dealiasing filters. In Sec. 2 we discuss specific numerical and computational issues associated with SAMR constructions and establish the stencil coefficients of the derivatives, interpolants, and filters. In Sec. 3 we use these tools in different problems (the Korteweg-de Vries, and the Kuramoto-Sivashinsky equations in 1D and a 2D FitzHugh-Nagumo equation) and analyze their efficacy. We draw our conclusions in Sec. 4.

2 Structured Adaptive Mesh Refinement

Structured Adaptive Mesh Refinement (SAMR) [1, 2] is a particularly appealing approach to geometrically simple spatial domains. The starting point for the method consists of laying a relatively coarse Cartesian mesh over a rectangular domain. Based on some suitable metric, regions requiring further refinement are identified, and the grid points are flagged and collated into *rectangular* children patches on which a denser Cartesian mesh is defined. The refinement factor between parent and child mesh is usually kept constant for a given problem. In this work, it is always specified as 2. The procedure is done recursively, so that one ultimately obtains a hierarchy of patches with different grid densities, with the finest patches overlaying a small part of the domain. This hierarchy will be referred to henceforth as a *Grid Hierarchy* or the *Mesh*;

individual patches will be termed *grids* or *patches*. Individual patches may be viewed as a wireframe laid over a rectangular domain with the dependent variables of a PDE defined at the intersection of the"wires" (referred to as "vertex-centered grids") or as a division of a domain into small cells, with the variables stored at cell centers ("cell-centered" grids). We will only study *vertex-centered* grids in this paper. Such grids have the property that the finer mesh grid points lie either on top of the coarse mesh grid points or at their geometric midpoints. This may be seen in Fig. 1 where large circles correspond to the coarse mesh, and fine mesh points are denoted by small triangles, squares, and circles requiring 1D, 2D, and 3D interpolations, respectively. The solution at these edge-centered (triangles), face-centered (squares) and body-centered (small circle) points, at coarse-fine interfaces/halos, are interpolated from the underlying coarse mesh (the large circles) using a variety of interpolations techniques.

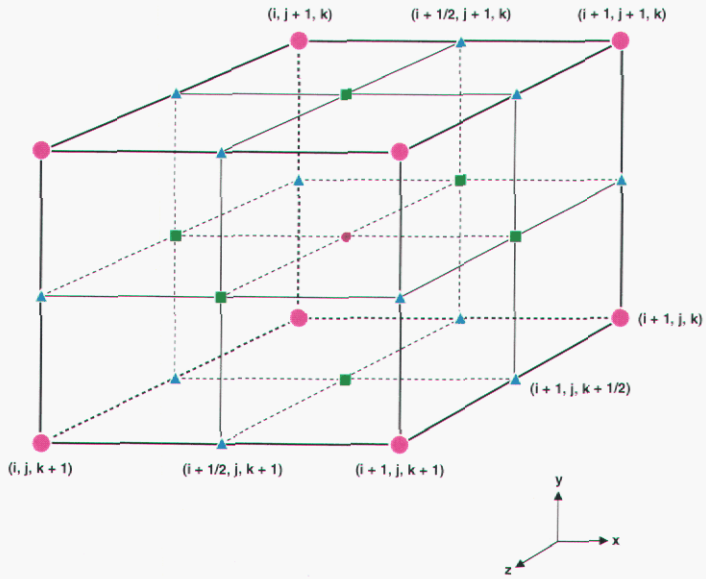

Figure 1. A vertex-centered grid. The large circles are the coarse mesh, and the fine mesh points at edge-centers, face-centers and body-centers are represented by triangles, squares and a (small) circle.

If a constant refinement factor is used (e.g. 2), the number of grid points rises rapidly as one refines. When used with time-refined explicit integration, SAMR-based simulations spend almost all their time in the finest levels. Further, in parallel simulations, the requirement of keeping parents and children on the same processor results in poor domain-partitioning by conventional partitioners. Thus, in practice, time-refined simulations usually have shallow hierarchies (e.g. 4 deep) with a fairly dense "coarse" grid. The resulting timestep size, determined by the stability constraints of the fine "coarse" grid is relatively small. To date, second-order spatial discretizations have been typical for SAMR simulations on vertex- and cell-centered grids. Accuracy has been achieved mainly by increasing mesh density, which has exacerbated the problem of a rather fine starting mesh (henceforth called Level0 mesh) or resulted in deep hierarchies. The high-order spatial discretizations employed in the present work will remedy this situation, reducing the requisite resolution requirement on the Level0 mesh.

High-order SAMR has a tremendous potential to reduce resolution requirements to achieve a given level of

accuracy. Jameson[21] has attempted to quantify the relative efficiencies of different orders-of-accuracy in one-dimension. Table 1 in [20] contains a listing of the number of grid points needed by centered derivative operators of various order to resolve a particular wavenumber mode to a given error tolerance; a similar table for high-order interpolants is in Table 1. In both cases, we see that a modest increase in order (from second to fourth) reduces the resolution requirements by a factor of three at 1% accuracy; at tighter tolerances a factor of 20 is achieved and at higher orders, one achieves almost 2 orders of magnitude. These savings are magnified as one proceeds to higher spatial dimensions.

ε	2E	4E	6E	8E	10E
$10^{-2.0}$	22.20	7.666	5.381	4.486	4.006
$10^{-2.5}$	39.49	10.29	6.597	5.262	4.577
$10^{-3.0}$	70.24	13.76	8.055	6.144	5.205
$10^{-3.5}$	124.9	18.39	9.810	7.153	5.900
$10^{-4.0}$	222.1	24.55	11.93	8.308	6.671
$10^{-4.5}$	395.0	32.76	14.48	9.635	7.530
$10^{-5.0}$	702.5	43.70	17.57	11.16	8.488

Table 1. Number of grid points per wavelength required to obtain a chosen relative error tolerance ε using explicit, centered-difference, mid-point, interpolant operators of orders two through ten. See table 13 for stencil coefficients.

However, the use of high-order discretizations and interpolants is not without its drawbacks. SAMR interpolant operators are evaluated using a linear combination of as many as p_I^d (where p_I is the order of accuracy and d is the dimensionality) values and their respective coefficients. While this renders high-order interpolants at each grid point far more expensive compared to bilinear or biquadratic (trilinear and triquadratic for 3D) interpolations, results in [20] indicate that the computational savings from the sparser grids may far outweigh the heavier per-gridpoint computational requirement in high-order methods. Nevertheless, high-order discretizations and interpolants do lead to certain complications. They may need to be closed to lower order at domain boundaries to preserve time-stability when coupled with explicit time-marching algorithms. Further, they may require smaller timesteps for stability, though this reduction is often not too consequential.

In this study we examine empirically high-order spatial convergence on a multiple-level SAMR mesh to identify correct discretization-interpolation-dissipation sets in conjunction with appropriate refinement/coarsening procedures. As a first step in that direction, we develop the expression for high-order discretizations, interpolants and filters which will be used in Sec. 3.

2.1 Derivatives

Finite-difference differentiation operators are constructed for the purposes of providing high-order derivatives as well as potentially adding dissipation via upwinding. In the case of arbitrary skewed stencils representing the p_Dth-order accurate approximation to the kth-derivative on a uniform stencil of width Δx, we

may write the derivative at grid point i as [22]

$$\cdots + \beta_L f_{i-2}^{(k)} + \alpha_L f_{i-1}^{(k)} + f_i^{(k)} + \alpha_R f_{i+1}^{(k)} + \beta_R f_{i+2}^{(k)} + \cdots = \tag{1}$$
$$\cdots + \frac{c_L f_{i-3}}{(\Delta x)^{(k)}} + \frac{b_L f_{i-2}}{(\Delta x)^{(k)}} + \frac{a_L f_{i-1}}{(\Delta x)^{(k)}} + \frac{\Upsilon f_i}{(\Delta x)^{(k)}} + \frac{a_R f_{i+1}}{(\Delta x)^{(k)}} + \frac{b_R f_{i+2}}{(\Delta x)^{(k)}} + \frac{c_R f_{i+3}}{(\Delta x)^{(k)}} + \cdots$$

where f_i is the value of the function at point i, and $f_i^{(k)}$ is the value of the k-derivative at that point. Following Kennedy and Carpenter [22] we use α, β, $\cdots$, to denote coeffi cients of the derivative terms $f_i^{(k)}$ that are one, two, $\cdots$, grid points on either side of point i, with R and L subscripts denoting the right and left directions on either side of point i respectively. We use a, b, $\cdots$ similarly for the coeffi cients of the function terms f_j for $j = i \pm 1, i \pm 2, \cdots$, and Υ for the coeffi cient of f_i. The Fourier image of this discrete derivative is given by

$$\Psi = \frac{\left\{ \begin{array}{c} [\Upsilon + (a_R + a_L)\cos(\xi) + (b_R + b_L)\cos(2\xi) + (c_R + c_L)\cos(3\xi) + \cdots] + \\ i[(a_R - a_L)\sin(\xi) + (b_R - b_L)\sin(2\xi) + (c_R - c_L)\sin(3\xi) + \cdots] \end{array} \right\}}{\left\{ \begin{array}{c} [1 + (\alpha_R + \alpha_L)\cos(\xi) + (\beta_R + \beta_L)\cos(2\xi) + \cdots] + \\ i[(\alpha_R - \alpha_L)\sin(\xi) + (\beta_R - \beta_L)\sin(2\xi) + \cdots] \end{array} \right\}} \tag{2}$$

or, after expanding the sine and cosine functions as a Taylor series,

$$\Psi = \sum_{m=1}^{\infty} \psi_{(2m-2)} \xi^{(2m-2)} + i \sum_{m=1}^{\infty} \psi_{(2m-1)} \xi^{(2m-1)} \tag{3}$$

where ξ is the Fourier dual variable, and $\Psi(\xi)$ is the approximation of the derivative in Fourier space [22, 23].

For the k-th derivative, and a p_D-th order accurate derivative discretization, we must now solve $(p_D + k)$ simultaneous equations; $\psi_k = (i)^k$ (for k even), $\psi_k = (i)^{k-1}$ (for k odd), and $\psi_l = 0, l = 0, 1, \cdots, (p_D + k - 1), l \neq k$, in $(p_D + k)$ unknowns; $\{\cdots, \beta_L, \alpha_L, \alpha_R, \beta_R, \cdots, c_L, b_L, a_L, a_R, b_R, c_R, \cdots\}$. Table 2 lists various values obtained from solving these equations for $k = 1$ $(\partial/\partial x)$ along with their leading-order truncation errors (L.O.T.E.). The suffi x E is used to denote explicit centered, U is for upwind, UU is for doubly upwind, D is downwind, and the prefi x denotes the order of accuracy. It is interesting to inspect these stencils further. On uniform grids, the difference operators may be considered as linear combinations of centered fi rst-derivative, difference operators plus low-order approximations to higher derivatives. Singly upwinded stencils may be readily decomposed as

$$\left({}^{(1)}\Delta^{+}_{(2n-1)}\right)_i = \left({}^{(1)}\Delta^{c}_{(2n)}\right)_i - (-1)^n \frac{(n-1)!n!}{(2n)!} \left({}^{(2n)}\Delta^{c}_{(2)}\right)_i, \tag{4}$$

$$\left({}^{(1)}\Delta^{+}_{(2n)}\right)_i = \left({}^{(1)}\Delta^{c}_{(2n)}\right)_i + (-1)^n \frac{(n-1)!n!}{(2n)!} \left({}^{(2n+1)}\Delta^{+}_{(1)}\right)_i. \tag{5}$$

where Δ denotes the derivative operator, the northwest superscript is the order of the derivative, the southeast subscript is the order of the accuracy, the northeast superscript denotes (c)-centered, (+)-upwind, or (−)-downwind, and the subscript outside of the parentheses denotes the grid point at which the difference operator is acting. We illustrate this with an example.

Name	d_L	c_L	b_L	a_L	Υ	a_R	b_R	c_R	d_R	L.O.T.E.
2E	0	0	0	-1/2	0	1/2	0	0	0	$-(i/6)\xi^3$
4E	0	0	1/12	-2/3	0	2/3	-1/12	0	0	$-(i/30)\xi^5$
6E	0	-1/60	3/20	-3/4	0	3/4	-3/20	1/60	0	$+(i/140)\xi^7$
8E	1/280	-4/105	12/60	-4/5	0	4/5	-12/60	4/105	-1/280	$-(i/630)\xi^9$
3U	0	0	0	-2/6	-1/2	1	-1/6	0	0	$-(i/12)\xi^4$
5U	0	0	1/20	-4/8	-1/3	1	-2/8	1/30	0	$-(i/60)\xi^6$
7U	0	-1/105	1/10	-6/10	-1/4	1	-3/10	1/15	-1/140	$-(i/280)\xi^8$
4U	0	0	0	-1/4	-5/6	3/2	-1/2	1/12	0	$+(i/20)\xi^5$
6U	0	0	1/30	-2/5	-7/12	4/3	-1/2	2/15	-1/60	$+(i/105)\xi^7$
4UU	0	0	0	0	-25/12	4	-3	4/3	-1/4	$-(i/5)\xi^5$
3UU	0	0	0	0	-11/6	3	-3/2	1/3	0	$+(i/4)\xi^6$
5UU	0	0	0	-1/5	-13/12	2	-1	1/3	-1/20	$+(i/30)\xi^6$
3DUU	0	0	0	-3/10	-19/30	6/5	-3/10	1/30	0	$-(i/20)\xi^4$
5DUU	0	0	3/70	-16/35	-37/84	8/7	-5/14	8/105	-1/140	$-(i/105)\xi^6$
4DUU	0	0	5/42	-6/7	5/12	4/21	3/14	-2/21	1/84	$-(2i/35)\xi^5$

Table 2. Stencil coefficients for centered and upwinded $\partial/\partial x$ operators on uniform grids.

Consider stencil 3U in Table 2. By Eq. 4, ($n = 2$) its stencil coefficients can be obtained from those of a centered stencil (4E, Table 2) and by scaling those of a second-order filter (4E, Table 12) by - 1/12. We can verify this by simply reading the stencil coefficients from the tables and performing the scaling.

More elaborate upwinding stencils may be decomposed as

$$\left({}^{(1)}\Delta^{++}_{(4)}\right)_i = \left({}^{(1)}\Delta^{c}_{(4)}\right)_i + \frac{1}{12}\left({}^{(5)}\Delta^{+}_{(1)}\right)_i - \frac{1}{4}\left({}^{(5)}\Delta^{+}_{(1)}\right)_{i+1}, \tag{6}$$

$$\left({}^{(1)}\Delta^{++}_{(3)}\right)_i = \left({}^{(1)}\Delta^{c}_{(6)}\right)_i + \frac{1}{60}\left({}^{(4)}\Delta^{c}_{(2)}\right)_{i-1} - \frac{1}{12}\left({}^{(4)}\Delta^{c}_{(2)}\right)_i + \frac{19}{60}\left({}^{(4)}\Delta^{c}_{(2)}\right)_{i+1}, \tag{7}$$

$$\left({}^{(1)}\Delta^{++}_{(5)}\right)_i = \left({}^{(1)}\Delta^{c}_{(8)}\right)_i - \frac{1}{280}\left({}^{(6)}\Delta^{c}_{(2)}\right)_{i-1} + \frac{1}{60}\left({}^{(6)}\Delta^{c}_{(2)}\right)_i - \frac{13}{280}\left({}^{(6)}\Delta^{c}_{(2)}\right)_{i+1}, \tag{8}$$

$$\left({}^{(1)}\Delta^{-++}_{(3)}\right)_i = \left({}^{(1)}\Delta^{c}_{(6)}\right)_i + \frac{1}{60}\left({}^{(4)}\Delta^{c}_{(2)}\right)_{i+1} + \frac{1}{60}\left({}^{(4)}\Delta^{c}_{(2)}\right)_{i-1} - \frac{1}{12}\left({}^{(4)}\Delta^{c}_{(2)}\right)_i, \tag{9}$$

$$\left({}^{(1)}\Delta^{-++}_{(5)}\right)_i = \left({}^{(1)}\Delta^{c}_{(8)}\right)_i - \frac{1}{280}\left({}^{(6)}\Delta^{c}_{(2)}\right)_{i+1} - \frac{1}{280}\left({}^{(6)}\Delta^{c}_{(2)}\right)_{i-1} + \frac{1}{60}\left({}^{(6)}\Delta^{c}_{(2)}\right)_i, \tag{10}$$

$$\left({}^{(1)}\Delta^{-++}_{(4)}\right)_i = \left({}^{(1)}\Delta^{c}_{(6)}\right)_i + \frac{1}{84}\left({}^{(5)}\Delta^{+}_{(2)}\right)_{i+1} + \frac{1}{84}\left({}^{(5)}\Delta^{+}_{(2)}\right)_{i-1} - \frac{17}{210}\left({}^{(5)}\Delta^{+}_{(2)}\right)_i + \frac{1}{35}\left({}^{(6)}\Delta^{c}_{(2)}\right)_i \tag{11}$$

It may be seen that the even-order, noncentered operators include dispersive components (e.g. Eq. 6; 4UU in Table 2). Other operators have only dissipative components but of both signs (e.g. Eq. 7; 3UU in Table 2) which can amplify errors. It is thus prudent to only consider odd-order, singly-upwinded derivative operators to add dissipation. Note that the reason for the alternating sign in front of the dissipative term is related to the fact that $\left({}^{(2n)}\Delta^{c}_{(2)}\right) \approx (i\xi)^{2n} = (-1)^n\xi^{2n}$. Similarly, for dispersive terms one finds $\left({}^{(2n+1)}\Delta^{c}_{(1)}\right) \approx (i\xi)^{2n+1} = (-1)^n i\xi^{2n+1}$.

To close the boundaries for these first-derivative operators when insufficient grid points are available for centered-difference operators, Table 3 lists lower-order closures. For situations where $\partial^2/\partial x^2$ is needed, cen-

c_L	b_L	a_L	Υ	a_R	b_R	c_R	L.O.T.E.
0	0	0	-1	1	0	0	$-(1/2)\xi^2$
0	0	0	-11/6	3	-3/2	1/3	$+(1/4)\xi^4$
0	0	-1/3	-1/2	1	-1/6	0	$-(1/12)\xi^4$
0	0	-1	1	0	0	0	$+(1/2)\xi^2$
-1/3	3/2	-3	11/6	0	0	0	$-(1/4)\xi^4$
0	1/6	-1	1/2	1/3	0	0	$+(1/12)\xi^4$

Table 3. Boundary stencil coefficients of lower-order, $\partial/\partial x$ operators.

tered interior stencil coefficients are given in Table 4 along with their leading-order truncation error. Finally,

Name	d_L	c_L	b_L	a_L	Υ	a_R	b_R	c_R	d_R	L.O.T.E.
2E	0	0	0	1	-2	1	0	0	0	$+(1/12)\xi^4$
4E	0	0	-1/12	4/3	-5/2	4/3	-1/12	0	0	$+(1/90)\xi^6$
6E	0	1/90	-3/20	3/2	-49/18	3/2	-3/20	1/90	0	$+(1/560)\xi^8$
8E	-1/560	8/315	-1/5	8/5	-205/72	8/5	-1/5	8/315	-1/560	$+(1/3150)\xi^{10}$

Table 4. Stencil coefficients for centered $\partial^2/\partial x^2$ operators on uniform grids.

lower-order boundary stencils to accompany the stencils listed in Table 4 are given in Table 5. Stencils for ∂_{xxx}, ∂_{xxxx} and ∂_{xxxxx} on vertex-centered grids are in Table 6- 11. The lower-order boundary stencils are also provided. ∂_{xxx} and ∂_{xxxx} are used in Sec. 3 to investigate the Korteweg-de Vries and Kuramoto-Sivashinsky equations [24]; ∂_{xxxxx} could be used to perform a similar investigation for the Kawahara equation.

d_L	c_L	b_L	a_L	Υ	a_R	b_R	c_R	d_R	L.O.T.E.
0	0	0	0	1	-2	1	0	0	$-\xi^3$
0	0	0	0	35/12	-26/3	19/2	-14/3	11/12	$+(5/6)\xi^5$
0	0	0	11/12	-5/3	1/2	1/3	-1/12	0	$-(1/12)\xi^5$
0	0	1	-2	1	0	0	0	0	$+\xi^3$
11/12	-14/3	19/2	-26/3	35/12	0	0	0	0	$-(5/6)\xi^5$
0	-1/12	1/3	1/2	-5/3	11/12	0	0	0	$+(1/12)\xi^5$

Table 5. Coefficients of $\partial^2/\partial x^2$ operators near boundary points.

Name	Υ	$a_R = -a_L$	$b_R = -b_L$	$c_R = -c_L$	$d_R = -d_L$	$e_R = -e_L$	L.O.T.E.
2E	0	-1	1/2	0	0	0	$+(i/4)\xi^5$
4E	0	-13/8	1	-1/8	0	0	$+(7i/120)\xi^7$
6E	0	-61/30	169/120	-3/10	7/240	0	$+(41i/3024)\xi^9$
8E	0	-1669/720	4369/2520	-541/1120	1261/15120	-41/6048	$+(479i/151200)\xi^{11}$

Table 6. Stencil coefficients for centered $\partial^3/\partial x^3$ operators on uniform grids.

b_L	a_L	Υ	a_R	b_R	c_R	d_R	e_R	L.O.T.E.
0	0	-1	3	-3	1	0	0	$+(3/2)\xi^4$
0	-1	3	-3	1	0	0	0	$+(1/2)\xi^4$
0	0	-17/4	71/4	-118/4	98/4	-41/4	7/4	$-(15/8)\xi^6$
0	-7/4	25/4	-34/4	22/4	-7/4	1/4	0	$-(1/8)\xi^6$
-1/4	-1/4	10/4	-14/4	7/4	-1/4	0	0	$+(1/8)\xi^6$

Table 7. Coefficients of $\partial^3/\partial x^3$ operators near left boundary points.

Name	Υ	$a_R = a_L$	$b_R = b_L$	$c_R = c_L$	$d_R = d_L$	$e_R = e_L$	L.O.T.E.
2E	6	-4	1	0	0	0	$-(1/6)\xi^6$
4E	28/3	-13/2	2	-1/6	0	0	$-(7/240)\xi^8$
6E	91/8	-122/15	169/60	-2/5	7/240	0	$-(41/7560)\xi^{10}$
8E	1529/120	-1669/180	4369/1260	-541/840	1261/15120	-41/7560	$-(479/453600)\xi^{12}$

Table 8. Stencil coefficients for centered $\partial^4/\partial x^4$ operators on uniform grids.

2.2 Filters

Finite-difference filters are a useful adjunct to a numerical method lacking spectral accuracy [22, 25]. In a non-AMR context, they are used simply to remove high-wavenumber spatial information that is not resolvable by the numerical method. However, in an AMR context, they may additionally be used to ensure that the high-order interpolants do not encounter the Runge phenomenon. Several criteria exist for a useful filter. Low wave-number information that is resolved should be virtually untouched while the relatively unresolved, high wave-number information should be removed. This constitutes dissipation. Dispersion, which is often introduced by the filter boundary closures making for a non-symmetric dissipation matrix, is to be avoided if possible.

As a filtering function in Fourier space, we seek a function that is equal to 0 at $\xi = \pi$ and is equal to 1 at $\xi = 0$. A simple function to satisfy this is $\left[1 - \sin^{2n}(\xi/2)\right]$. To create this Fourier image using explicit

b_L	a_L	Υ	a_R	b_R	c_R	d_R	e_R	f_R	L.O.T.E.
0	0	1	-4	6	-4	1	0	0	$+2i\xi^5$
0	1	-4	6	-4	1	0	0	0	$+i\xi^5$
0	0	35/6	-186/6	411/6	-484/6	321/6	-114/6	17/6	$-(7i/2)\xi^7$
0	17/6	-84/6	171/6	-184/6	111/6	-36/6	5/6	0	$-(2i/3)\xi^7$
5/6	-18/6	21/6	-4/6	-9/6	6/6	-1/6	0	0	$-(i/6)\xi^7$

Table 9. Coefficients of $\partial^4/\partial x^4$ operators near left boundary points.

Name	Υ	$a_R = -a_L$	$b_R = -b_L$	$c_R = -c_L$	$d_R = -d_L$	$e_R = -e_L$	$f_R = -f_L$	L.O.T.E.
2E	0	5/2	-2	1/2	0	0	0	$-(i/3)\xi^7$
4E	0	29/6	-13/3	3/2	-1/6	0	0	$-(13i/144)\xi^9$
6E	0	323/48	-13/2	87/32	-19/36	13/288	0	$-(139i/6048)\xi^{11}$
8E	0	1039/126	-33853/4032	3011/756	-3125/3024	121/756	-139/12096	$-(37i/6480)\xi^{13}$

Table 10. Stencil coefficients for centered $\partial^5/\partial x^5$ operators on uniform grids.

c_L	b_L	a_L	Υ	a_R	b_R	c_R	d_R	e_R	f_R	g_R	L.O.T.E.
0	0	0	-1	5	-10	10	-5	1	0	0	$-(5/2)\xi^6$
0	0	-1	5	-10	10	-5	1	0	0	0	$-(3/2)\xi^6$
0	-1	5	-10	10	-5	1	0	0	0	0	$-(1/2)\xi^6$
0	0	0	-46/6	295/6	-810/6	1235/6	-1130/6	621/6	-190/6	25/6	$+(35/6)\xi^8$
0	0	-25/6	154/6	-405/6	590/6	-515/6	270/6	-79/6	10/6	0	$+(5/3)\xi^8$
0	-10/6	55/6	-126/6	155/6	-110/6	45/6	-10/6	1/6	0	0	$+(11/144)\xi^8$
-1/6	-2/6	27/6	-70/6	85/6	-54/6	17/6	-2/6	0	0	0	$-(1/6)\xi^8$

Table 11. Coefficients of $\partial^5/\partial x^5$ operators near left boundary points.

finite-difference stencils, one may utilize scaled values of the symmetric $(2n)$th-order derivative of a function f,

$$f_i^{(2n)} = \frac{\Upsilon f_i}{(\Delta x)^{(2n)}} + a\frac{f_{i+1} + f_{i-1}}{(\Delta x)^{(2n)}} + b\frac{f_{i+2} + f_{i-2}}{(\Delta x)^{(2n)}} + c\frac{f_{i+3} + f_{i-3}}{(\Delta x)^{(2n)}} + \cdots \tag{12}$$

where $a = a_L = a_R$, $b = b_L = b_R$, etc ..., with $\Psi = \Upsilon + 2a\cos(\xi) + 2b\cos(2\xi) + 2c\cos(3\xi) + \cdots$. Solving for the stencil coefficients for the second-order accurate version of this operator, $\partial^{2n} f/\partial x^{2n}$, i.e. $\Psi = (-1)^n \xi^{2n} + O(\xi^{2n+2})$, the Fourier image is given by $\Psi = (-1)^n \left[2\sin\left(\frac{\xi}{2}\right)\right]^{2n}$. Actual stencil coefficients for the $\partial^{2n}/\partial x^{2n}$ operator, up to eighth-order, are given in Table 12. Alternatively, one could develop these second-order accurate stencils for arbitrary n using Pascal's triangle.

Name	d_L	c_L	b_L	a_L	Υ	a_R	b_R	c_R	d_R	L.O.T.E.
2E	0	0	0	1	-2	1	0	0	0	$+(1/12)\xi^4$
4E	0	0	1	-4	6	-4	1	0	0	$-(1/6)\xi^6$
6E	0	1	-6	15	-20	15	-6	1	0	$+(1/4)\xi^8$
8E	1	-8	28	-56	70	-56	28	-8	1	$-(1/3)\xi^{10}$

Table 12. Stencil coefficients for centered $\partial^{2n}/\partial x^{2n}$ operators on uniform grids.

These symmetric stencil coefficients are the interior elements of the dissipation matrix. Stencils of order $2n$ give rise to a filter of order, $p_F = 2n$. For the interior, the absolute value of the stencil coefficients is simply equal to $2n!/((2n-i)!i!)$ where i runs from 0 to $2n$. For $i = n$, $|\Upsilon|$ is retrieved. At $i = n \pm 1$ one gets $|a_L = a_R = a|$, at $i = n \pm 2$ one gets $|b_L = b_R = b|$, etc. By a slight rescaling of the dissipation matrix, the filtered value of U may be seen to be $\hat{U} = (1 - \alpha_D \mathbf{D})U$, where $\hat{U}$ is the filtered vector and α_D must be given by $(-1)^{n+1}2^{-2n}$. At the boundaries where $\partial^{2n}/\partial x^{2n}$ cannot be approximated using symmetric stencils, one may either implement skewed stencils and risk dispersive filter errors, or implement $\partial^n/\partial x^n$ rather than $\partial^{2n}/\partial x^{2n}$ in the filter matrix $\mathbf{D}$. We have chosen to adopt the second approach, even though it introduces errors of $O(\Delta x^{n+2})$ at the boundaries while the filter error at interior points is $O(\Delta x^{2n+2})$. Filter coefficients and other details may be found in the literature [22, 25] but are given using the negative of the stencils described in Sec. 2.1 so that $\hat{U} = (1 + \alpha_D \mathbf{D})U$.

Simulations involving filtering of the solution require a degree of care while filtering. For example, in a multistage time-integration method, filtering the stage value destroys the temporal accuracy of the scheme - filtering can only be done after the step is completed. Further, filtering changes the solution by removing the high wavenumber spectrum of the solution; thus high order filters are to be preferred since their cutoff wave-number is high. High-order filters have wide stencils, which pose a practical difficulty in adaptive mesh schemes since this requires the patches to be at least as wide as the filter stencil. In our studies, we impose a "halo" of grid points n wide around each patch. This ensures that all the "valid" grid points on the patch (i.e. the grid points whose refinement is dictated by the requirements of the physics being simulated) are considered "interior" points by the filter matrix. This is not satisfied at the domain boundaries and fine patches abutting the domain boundary incur filter errors of $O(\Delta x^{n+2})$ at certain points.

2.3 Hyperviscosity

While not used here, we present the following brief comments on hyperviscosity. Beyond using upwinding to apply high-order dissipation during construction of the right-hand-side (RHS) of the governing equations or applying stand-alone filters to remove high wavenumbers from the U-vector itself, one can add a hyperviscosity term to the RHS to both dealias the numerical method and to forestall Runge phenomena. The term would appear as $\mu_H(\nabla^2)^n U$ and may be readily constructed using the symmetric dissipation matrices, $\mathbf{D}^{2n}$, used in the filters. To avoid interfering with the order of the overall method, p, it necessary to use $2n = p_H \geq p$. One may then write $\nabla^{2n}U = (\mathbf{D}_x^{(2)} + \mathbf{D}_y^{(2)} + \mathbf{D}_z^{(2)})^n U$. It is not clear that, for the purposes in mind, this much effort needs to be expended. One could effectively assume that dissipation matrices commute and reduce workload. This would transform

$$\nabla^4 U = \left(\mathbf{D}_x^{(4)} + \mathbf{D}_y^{(4)} + \mathbf{D}_z^{(4)} + \mathbf{D}_x^{(2)}\mathbf{D}_y^{(2)} + \mathbf{D}_x^{(2)}\mathbf{D}_z^{(2)} + \mathbf{D}_y^{(2)}\mathbf{D}_x^{(2)} + \mathbf{D}_y^{(2)}\mathbf{D}_z^{(2)} + \mathbf{D}_z^{(2)}\mathbf{D}_x^{(2)} + \mathbf{D}_z^{(2)}\mathbf{D}_y^{(2)}\right) U$$

to

$$\nabla^4 U \approx \left(\mathbf{D}_x^{(4)} + \mathbf{D}_y^{(4)} + \mathbf{D}_z^{(4)} + 2\mathbf{D}_x^{(2)}\mathbf{D}_y^{(2)} + 2\mathbf{D}_y^{(2)}\mathbf{D}_z^{(2)} + 2\mathbf{D}_z^{(2)}\mathbf{D}_x^{(2)}\right) U.$$

To further reduce work, one could ignore cross-terms and simply approximate $\nabla^{2n}U \approx (\mathbf{D}_x^{(2n)} + \mathbf{D}_y^{(2n)} + \mathbf{D}_z^{(2n)})U$. The remaining difficulty is to select an appropriate value for μ_H.

The use of hyperviscous terms in explicit time-marched schemes can be problematic. Since the temporal stability of the explicit schemes varies as Δx^{2n}, high order hyperviscosity can lead to very small timesteps on fine patches. This problem is removed if one uses implicit schemes on such meshes. Thus, while hyperviscosity appears to be an elegant means of controlling high wavenumber oscillations, practical considerations make filters more attractive.

2.4 Interpolants

Interpolations, in a vertex-centered AMR environment, are used to fill up the coarse-fine interface or "halo" that is kept around all patches. This "halo" enables the the use of symmetric stencils for discretizations and filters uniformly on all patches, except near the domain boundaries. As the order of the filters and discretizations are increased, the width of the "halo" is increased accordingly. The solution in this halo region is not calculated using the equations being solved; rather they are interpolated from the underlying coarse mesh. Since the halo is additive i.e. it is added around the "valid" points whose refinement is dictated by resolution requirement, these interpolations are obtained from coarse grid points where the solution is deemed sufficiently resolved.

Interpolation requirements for structured AMR using a vertex-centered grid arrangement include as many as seven distinct finite-difference subinterpolations for up to three Cartesian coordinate directions. All necessary coarse-to-fine interpolations may be accomplished by using one-dimensional interpolants in the x-, y-, and z-directions, two-dimensional interpolants in the xy-, yz-, and zx-directions, and three-dimensional interpolants in the xyz-direction. Two dimensional simulations require three, say, x-, y-, and xy-interpolations. Three dimensional simulations require all seven to interpolate the $3^3 - 2^3 = 19$ points inside a unit cube. Twelve interpolations require 1D interpolants (the triangles in Fig. 1), six require 2D interpolants (the

squares in Fig. 1), and one requires a 3D interpolant (the small circle). Since all $2^3 = 8$ coarse grid points are also fine grids points, interpolation from fine-to-coarse is merely an injection and requires no special finite difference stencils. Coarse-to-fine interpolations, however, do need specific stencils. Each of these stencils, whether it be for one-, two-, or three-dimensions, interpolates from the coarse grid data to to its geometric midpoint where the fine grid point lies. Although implicitness in finite-difference derivative and interpolant operators provides better high-wavenumber resolution, their added cost makes them ultimately less efficient than explicit methods unless their implicit solves can be made quite economical. Hence, all interpolants considered here will be explicit.

2.4.1 1D - Interior

The stencils in the interior of the domain can symmetrically utilize equal numbers of grid points on either side of the point being interpolated. Hence, the stencil coefficients are identical on each side,

$$f_i^{(0)} = \cdots + cf_{i-5/2} + bf_{i-3/2} + af_{i-1/2} + af_{i+1/2} + bf_{i+3/2} + cf_{i+5/2} + \cdots$$

and give its Fourier image by

$$\Psi = \{\ 2a\cos(\xi/2) + 2b\cos(3\xi/2) + 2c\cos(5\xi/2) + \cdots\ \} = \sum_{l=0}^{\infty} \psi_l \xi^l. \tag{13}$$

To achieve order p_I, we require $\psi_0 = 1$ and $\psi_l = 0$, with $l = 1, 2, \cdots, (p_I - 1)$. Symmetric interior stencils are listed in Table 13 along with their leading-order truncation error. It will be useful to also cast these

Stencil	2nd-order	4th-order	6th-order	8th-order	10th-order
a	1/2	9/16	150/256	1225/2048	39690/65536
b	0	-1/16	-25/256	-245/2048	-8820/65536
c	0	0	3/256	49/2048	2268/65536
d	0	0	0	-5/2048	-405/65536
e	0	0	0	0	35/65536
$\psi_{p_I}\xi^{p_I}$	$-(1/8)\xi^2$	$-(3/128)\xi^4$	$-(5/1024)\xi^6$	$-(35/32768)\xi^8$	$-(63/262144)\xi^{10}$

Table 13. Coefficients of one-dimensional, vertex-centered, midpoint interpolants in terms of ξ.

stencils in Fourier space as

$$\Psi_I^{2(x)} = \cos(\xi/2), \tag{14}$$

$$\Psi_I^{4(x)} = \frac{-1}{4}\cos(\xi/2)[-5 + \cos(\xi)], \tag{15}$$

$$\Psi_I^{6(x)} = \frac{+1}{64}\cos(\xi/2)[89 - 28\cos(\xi) + 3\cos(2\xi)], \tag{16}$$

$$\Psi_I^{8(x)} = \frac{-1}{512}\cos(\xi/2)[-762+299\cos(\xi)-54\cos(2\xi)+5\cos(3\xi)], \tag{17}$$

$$\Psi_I^{10(x)} = \frac{+1}{16384}\cos(\xi/2)[+25609-11528\cos(\xi)+2708\cos(2\xi)-440\cos(3\xi)+35\cos(4\xi)], \tag{18}$$

for the x-direction. In the y- or z-directions, these expressions are modified by simply replacing ξ with $\eta(y)$ or $\zeta(z)$. Using Mathematica [26], one may readily convert between trigonometric and exponential functions using `TrigToExp[]` to retrieve the stencil coefficients.

2.4.2 1D - Boundaries

As boundaries are approached, interpolant stencils must become asymmetrical if higher-order is desired. Stencils are now considered as

$$f_i^{(0)} = \cdots + Cf_{i-5/2} + Bf_{i-3/2} + Af_{i-1/2} + af_{i+1/2} + bf_{i+3/2} + cf_{i+5/2} + \cdots$$

with a Fourier image given by

$$\Psi = \left\{ \begin{array}{c} [(a+A)\cos(1/2\xi) + (b+B)\cos(3/2\xi) + (c+C)\cos(5/2\xi) + \cdots] + \\ i[(a-A)\sin(1/2\xi) + (b-B)\sin(3/2\xi) + (c-C)\sin(5/2\xi) + \cdots] \end{array} \right\} = \sum_{l=0}^{\infty} \psi_l \xi^l. \tag{19}$$

Again, to achieve order p_I, we require $\psi_0 = 1$ and $\psi_l = 0, l = 1, 2, \cdots, (p_I - 1)$. As the stencils are skewed, we use $\Psi_{\mathsf{a}}^{4(x)}$ to denote the fourth-order interpolant in the x-direction that is shifted one grid point to the right, i.e. the nonzero stencil coefficients are $\{A, a, b, c\}$. Shifting two grid points to the right at sixth-order produces $\Psi_{\mathsf{b}}^{6(x)}$ with a stencil determined by $\{A, a, b, c, d, e\}$. To fully close the interpolant boundaries at orders four, six, eight, and ten, the stencils as computed from the Mathematica script provided in Appendix A are required. In the y- or z-directions, these expressions are modified by simply replacing ξ with either η or ζ. For stencils shifted to the left rather than the right, the Fourier image expressions should have each occurrence of I replaced with $-I$ and the lowercase letter used to denote the degree of skewness should be replaced with the uppercase version of that letter.

2.4.3 2D - Interior

Before discussing multidimensional stencils, it will be useful to develop a reasonably clear and compact notation in which to convey both stencils and their coefficients. Let $\{\Gamma_\alpha | \alpha = -p_I^*, -p_I^* + 1, \ldots, p_I^* - 1, p_I^*\} = \{\cdots, C, B, A, a, b, c, \cdots\}$ be the stencil coefficients associated with grid points $\{\cdots, i-(5/2), i-(3/2), i-(1/2), i+(1/2), i+(3/2), i+(5/2), \cdots\}$. Using this, the stencil coefficient associated with the point $(i-(1/2), j+(1/2))$ would be Aa. With this, the preceding symmetric, one-dimensional stencil may be written as

$$f_i^{(0)} = \sum_{\alpha=-p_I^*}^{p_I^*} \Gamma_\alpha f_{i+\alpha} = \cdots + Cf_{i-5/2} + Bf_{i-3/2} + Af_{i-1/2} + af_{i+1/2} + bf_{i+3/2} + cf_{i+5/2} + \cdots \tag{20}$$

where $p_I^* = (p_I - 1)/2$. For a two-dimensional stencil and its Fourier image,

$$f_{i,j}^{(0)} = \sum_{\alpha=-p_I^*}^{p_I^*} \sum_{\beta=-p_I^*}^{p_I^*} \Gamma_\alpha \Gamma_\beta f_{i+\alpha,j+\beta}, \quad \Psi = \sum_{\alpha=-p_I^*}^{p_I^*} \sum_{\beta=-p_I^*}^{p_I^*} \Gamma_\alpha \Gamma_\beta \exp(i\alpha\xi)\exp(i\beta\eta) = \sum_{l=0}^{\infty}\sum_{m=0}^{\infty} \psi_{l,m}\xi^l\eta^m, \quad (21)$$

where we require $\psi_{0,0} = 1$ and $\psi_{l,m} = 0$, with $l,m = 1,2,\cdots,(p_I - 1)$ to achieve order p_I. Symmetry reduces the number of independent stencil coefficients from p_I^2 to $p_I(p_I+2)/8$. For instance, certain off diagonal stencil coefficients satisfy the relations $ce = cE = Ce = CE = ec = Ec = eC = EC$, while diagonal coefficients satisfy $cc = cC = Cc = CC$. These interior stencils may be cast in Fourier space quite simply as $\Psi_I^{2(xy)} = \Psi_I^{2(x)}\Psi_I^{2(y)}$, $\Psi_I^{4(xy)} = \Psi_I^{4(x)}\Psi_I^{4(y)}$, $\Psi_I^{6(xy)} = \Psi_I^{6(x)}\Psi_I^{6(y)}$, $\Psi_I^{8(xy)} = \Psi_I^{8(x)}\Psi_I^{8(y)}$, $\Psi_I^{10(xy)} = \Psi_I^{10(x)}\Psi_I^{10(y)}$, where the one-dimensional stencils are given in Eq. 14- 18. Values for the unique stencil coefficient are listed in Table 14 for even orders from two to ten.

Stencil	2nd-order	4th-order	6th-order	8th-order	10th-order
aa	1/4	81/256	22500/65536	$1500625/2^{22}$	$1575296100/2^{32}$
ba	0	-9/256	-3750/65536	$-300125/2^{22}$	$-350065800/2^{32}$
bb	0	+1/256	625/65536	$60025/2^{22}$	$77792400/2^{32}$
ca	0	0	450/65536	$60025/2^{22}$	$90016920/2^{32}$
cb	0	0	-75/65536	$-12005/2^{22}$	$-20003760/2^{32}$
cc	0	0	9/65536	$2401/2^{22}$	$5143824/2^{32}$
da	0	0	0	$-6125/2^{22}$	$-16074450/2^{32}$
db	0	0	0	$1225/2^{22}$	$3572100/2^{32}$
dc	0	0	0	$-245/2^{22}$	$-918540/2^{32}$
dd	0	0	0	$25/2^{22}$	$164025/2^{32}$
ea	0	0	0	0	$1389150/2^{32}$
eb	0	0	0	0	$-308700/2^{32}$
ec	0	0	0	0	$79380/2^{32}$
ed	0	0	0	0	$-14175/2^{32}$
ee	0	0	0	0	$1225/2^{32}$
$\psi_{p_I,p_I}\xi^{p_I}\eta^{p_I}$	$-(1/8)\xi^2$ $-(1/8)\eta^2$	$-(3/128)\xi^4$ $-(3/128)\eta^4$	$-(5/1024)\xi^6$ $-(5/1024)\eta^6$	$-(35/32768)\xi^8$ $-(35/32768)\eta^8$	$-(63/262144)\xi^{10}$ $-(63/262144)\eta^{10}$

Table 14. Coefficients and truncation errors of two-dimensional, vertex-centered interpolants in terms of ξ and η.

2.4.4 2D - Boundaries

Boundary stencils to two-dimensional interpolants may be constructed in Fourier space by simply multiplying together the proper one-dimensional interpolants. At orders $p_I = \{2,4,6,8,10\}$, one needs to close the interior method with $\{0,2,5,9,14\} = p_I(p_I+2)/8 - 1$ unique boundary stencils if one wishes to preserve overall order. At fourth-order, when x and y are chosen as the two directions, these are given by

$$\Psi_{I,\mathsf{a}}^{4(xy)} = \Psi_I^{4(x)}\Psi_{\mathsf{a}}^{4(y)}, \qquad \Psi_{\mathsf{a},\mathsf{a}}^{4(xy)} = \Psi_{\mathsf{a}}^{4(x)}\Psi_{\mathsf{a}}^{4(y)}. \quad (22)$$

Note that the other stencils, $\Psi^{4(xy)}_{\mathsf{A,A}}$, $\Psi^{4(xy)}_{\mathsf{a,A}}$, and $\Psi^{4(xy)}_{\mathsf{a,A}}$ are simply rotations of $\Psi^{4(xy)}_{a,\mathsf{a}}$ while $\Psi^{4(xy)}_{\mathsf{a},I}$, $\Psi^{4(xy)}_{\mathsf{A},I}$, and $\Psi^{4(xy)}_{I,\mathsf{A}}$ are rotations of $\Psi^{4(xy)}_{I,\mathsf{a}}$. At sixth-order the unique stencils consist of $\Psi^{6(xy)}_{I,\mathsf{a}}$, $\Psi^{6(xy)}_{I,\mathsf{b}}$, $\Psi^{6(xy)}_{\mathsf{a,a}}$, $\Psi^{6(xy)}_{\mathsf{a,b}}$, and $\Psi^{6(xy)}_{\mathsf{b,b}}$, For eighth-order, one has $\Psi^{8(xy)}_{I,\mathsf{a}}$, $\Psi^{8(xy)}_{I,\mathsf{b}}$, $\Psi^{8(xy)}_{I,\mathsf{c}}$, $\Psi^{8(xy)}_{\mathsf{a,a}}$, $\Psi^{8(xy)}_{\mathsf{a,b}}$, $\Psi^{8(xy)}_{\mathsf{a,c}}$, $\Psi^{8(xy)}_{\mathsf{b,b}}$, $\Psi^{8(xy)}_{\mathsf{b,c}}$, $\Psi^{8(xy)}_{\mathsf{c,c}}$, and at tenth-order, there are $\Psi^{10(xy)}_{I,\mathsf{a}}$, $\Psi^{10(xy)}_{I,\mathsf{b}}$, $\Psi^{10(xy)}_{I,\mathsf{c}}$, $\Psi^{10(xy)}_{I,\mathsf{d}}$, $\Psi^{10(xy)}_{\mathsf{a,a}}$, $\Psi^{10(xy)}_{\mathsf{a,b}}$, $\Psi^{10(xy)}_{\mathsf{a,c}}$, $\Psi^{10(xy)}_{\mathsf{a,d}}$, $\Psi^{10(xy)}_{\mathsf{b,b}}$, $\Psi^{10(xy)}_{\mathsf{b,c}}$, $\Psi^{10(xy)}_{\mathsf{b,d}}$, $\Psi^{10(xy)}_{\mathsf{c,c}}$, $\Psi^{10(xy)}_{\mathsf{c,d}}$, $\Psi^{10(xy)}_{\mathsf{d,d}}$.

2.4.5 3D - Interior

Similar to the derivation of two-dimensional stencils, for three-dimensions we use

$$f^{(0)}_{i,j,k} = \sum_{\alpha=-p_I^*}^{p_I^*} \sum_{\beta=-p_I^*}^{p_I^*} \sum_{\gamma=-p_I^*}^{p_I^*} \Gamma_\alpha \Gamma_\beta \Gamma_\gamma f_{i+\alpha,j+\beta,k+\gamma}, \tag{23}$$

along with its Fourier image

$$\Psi = \sum_{\alpha=-p_I^*}^{p_I^*} \sum_{\beta=-p_I^*}^{p_I^*} \sum_{\gamma=-p_I^*}^{p_I^*} \Gamma_\alpha \Gamma_\beta \Gamma_\gamma \exp(i\alpha\xi)\exp(i\beta\eta)\exp(i\gamma\zeta) = \sum_{l=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \psi_{l,m,n}\xi^l\eta^m\zeta^n, \tag{24}$$

where we require $\psi_{0,0,0} = 1$ and $\psi_{l,m,n} = 0, l, m, n = 1, 2, \cdots, (p_I - 1)$ to achieve order p_I. Stencil coeffi cients may be computed by either solving for the many conditions placed on $\psi_{l,m,n}$ or by constructing the stencils in Fourier space and then converting the trigonometric functions to exponential functions using $\Psi^{2(xy)}_I = \Psi^{2(x)}_I\Psi^{2(y)}_I\Psi^{2(z)}_I$, $\Psi^{4(xy)}_I = \Psi^{4(x)}_I\Psi^{4(y)}_I\Psi^{4(z)}_I$, $\Psi^{6(xy)}_I = \Psi^{6(x)}_I\Psi^{6(y)}_I\Psi^{6(z)}_I$, $\Psi^{8(xy)}_I = \Psi^{8(x)}_I\Psi^{8(y)}_I\Psi^{8(z)}_I$, $\Psi^{10(xy)}_I = \Psi^{10(x)}_I\Psi^{10(y)}_I\Psi^{10(z)}_I$. The $p_I(p_I+2)(p_I+4)/48$ unique stencil coeffi cient values are listed in Table B.1 in the appendix for orders $p_I = \{2,4,6,8,10\}$.

2.4.6 3D - Boundaries

Boundary stencils to three-dimensional interpolants may be constructed by a slight extension of the two-dimensional results, by simply multiplying together the proper three one-dimensional interpolants rather than two. At orders $\{2,4,6,8,10\}$ one needs to close the interior method with $\{0,3,9,19,34\} = p_I(p_I + 2)(p_I+4)/48 - 1$ unique boundary stencils if one wishes to preserve overall order, where each stencil has p_I^3 stencil coeffi cients that may or may not be unique. At fourth-order the unique stencils are given by

$$\Psi^{4(xyz)}_{I,I,\mathsf{a}} = \Psi^{4(x)}_I\Psi^{4(y)}_I\Psi^{4(z)}_{\mathsf{a}}, \qquad \Psi^{4(xyz)}_{I,\mathsf{a,a}} = \Psi^{4(x)}_I\Psi^{4(y)}_{\mathsf{a}}\Psi^{4(z)}_{\mathsf{a}} \qquad \Psi^{4(xyz)}_{\mathsf{a,a,a}} = \Psi^{4(x)}_{\mathsf{a}}\Psi^{4(x)}_{\mathsf{a}}\Psi^{4(z)}_{\mathsf{a}}. \tag{25}$$

Other combinations using the subscripts I and a are merely rotations of the stencils just listed but with identical stencil coeffi cients. At sixth-order the unique stencils consist of $\Psi^{6(xyz)}_{I,I,\mathsf{a}}$, $\Psi^{6(xyz)}_{I,I,\mathsf{b}}$, $\Psi^{6(xyz)}_{I,\mathsf{a,a}}$, $\Psi^{6(xyz)}_{I,\mathsf{a,b}}$, $\Psi^{6(xyz)}_{I,\mathsf{b,b}}$, $\Psi^{6(xyz)}_{\mathsf{a,a,a}}$, $\Psi^{6(xyz)}_{\mathsf{a,a,b}}$, $\Psi^{6(xyz)}_{\mathsf{a,b,b}}$, and $\Psi^{6(xyz)}_{\mathsf{b,b,b}}$, For eighth-order, one has $\Psi^{8(xyz)}_{I,I,\mathsf{a}}$, $\Psi^{8(xyz)}_{I,I,\mathsf{b}}$, $\Psi^{8(xyz)}_{I,I,\mathsf{c}}$, $\Psi^{8(xyz)}_{I,\mathsf{a,a}}$, $\Psi^{8(xyz)}_{I,\mathsf{a,b}}$, $\Psi^{8(xyz)}_{I,\mathsf{a,c}}$, $\Psi^{8(xyz)}_{I,\mathsf{b,b}}$, $\Psi^{8(xyz)}_{I,\mathsf{b,c}}$, $\Psi^{8(xyz)}_{I,\mathsf{c,c}}$, $\Psi^{8(xyz)}_{\mathsf{a,a,a}}$, $\Psi^{8(xyz)}_{\mathsf{a,a,b}}$, $\Psi^{8(xyz)}_{\mathsf{a,a,c}}$, $\Psi^{8(xyz)}_{\mathsf{a,b,b}}$, $\Psi^{8(xyz)}_{\mathsf{a,b,c}}$, $\Psi^{8(xyz)}_{\mathsf{b,b,b}}$, $\Psi^{8(xyz)}_{\mathsf{b,b,c}}$, $\Psi^{8(xyz)}_{\mathsf{b,c,c}}$, $\Psi^{8(xyz)}_{\mathsf{b,c,c}}$, and $\Psi^{8(xyz)}_{\mathsf{c,c,c}}$. This may easily be carried on to tenth-order by including the subscript d.

3 Test Problems

In this section we will address two issues

1. Given a PDE to solve on a block-structured adaptive mesh, with a discretization of order p_D, what is the order of the interpolations (p_I) that needs to be used for prolongations and restrictions.

2. Once a correct (p_D, p_I) pair have been identified, can the tools developed in the previous section be combined to achieve high order on adaptively refined meshes ?

For the purposes of this paper, we will specify a refinement pattern and *not adapt* the mesh. The issues regarding the criteria for refining-derefining a mesh point will not be addressed here.

3.1 Appropriate interpolation orders

As described in Sec. 1 and 2, the solution on any patch in a *Grid Hierarchy* is the result of interpolations from finer meshes (restrictions) and/or interpolations from the solution on coarser meshes at a coarse-fine boundary (prolongations). In this paper, since we address vertex-centered meshes, restrictions are mere injections and do not involve interpolations; however prolongations do involve interpolations at coarse-fine boundaries.

The position of the coarse-fine boundary is determined by the refining-derefining criteria. These are usually based on an threshold on solution error [1, 2] or a threshold on the gradient [12]. Thus, there exists a small but non-zero gradient at a coarse-fine boundary and an interpolation error is incurred there. Ideally the gradient should be small enough that the interpolation error is negligible vis-a-vis the discretization error regardless of the interpolation chosen, but this is difficult to achieve in practice and still have an efficient grid (we run the risk of refining large areas of the domain and losing the advantage of concentrating resolution only where needed). Since these interpolated values cannot be differentiated indefinitely and still preserve order, it becomes critical to choose an appropriate interpolation order.

In [20], we conducted a study of the appropriate interpolation order p_I given a discretization of order $p_D = 4$. The tests were done with the FitzHugh-Nagumo equation in 1D (where the highest spatial derivative is ∂_{xx}) and we concluded that $p_I \geq (p_D + 2)$ was sufficient. In Appendix C, we present a derivation of the requisite interpolation order p_I for a given order of discretization and arbitrary derivative order. In this analysis, we find that $p_I \geq (p_D + H_D)$ should be sufficient, where H_D is the highest spatial derivative in the equation. We illustrate this empirically by using the Korteweg-de Vries equation (highest spatial derivative is ∂_{xxx}) and the Kuramoto-Sivashinsky equation (∂_{xxxx}).

3.1.1 Korteweg - de Vries equation

The KdV equation is written as

$$U_t + 6UU_x + U_{xxx} = 0 \tag{26}$$

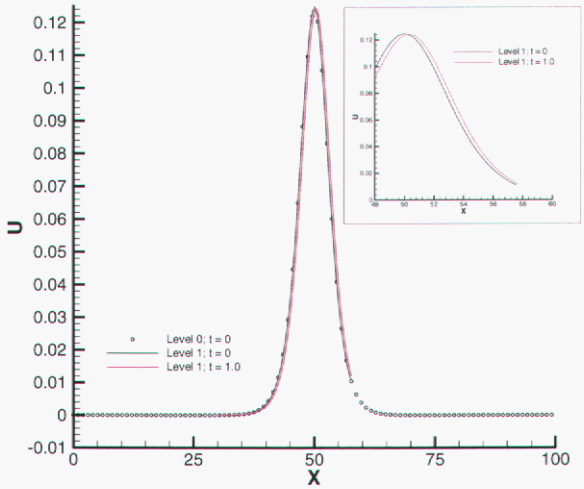

Figure 2. A soliton solution of Eq. 26 at $t = 0$ and 1. Symbols are used to plot the solution on L_0 and lines on L_1. The red line shows the solution at $t = 1.0$. Inset: A detail of the L_1 solution.We see that the movement of soliton during the simulation is small vis-a-vis its size.

Under the assumption that $U_x \to 0$ as $x \to \pm\infty$, we get a traveling wave solution $U = 2k^2(\cosh\{k(x - x_0 - 4k^2t)\})^{-2}$ where k is a wavenumber and the amplitude and velocity of the traveling wave are $2k^2$ and $4k^2$ respectively.

We solve Eq. 26 on a two-level Grid Hierarchy (i.e with a Level0 and Level1, abbreviated as a $\{L_0, L_1\}$ mesh), with a factor of refi nement of 2. The domain is $0 \leq x \leq 100$ with $x_0 = 50$, i.e. the initial condition is a soliton placed at the center of the domain. The region $30 \leq x \leq 56$ is refi ned to a L_1 mesh. We use $p_D = 4$. The width of the halo around the fi ne patches was kept at 6. Runge phenomenon was not observed and we did not use fi lters. Fig. 2 shows the initial condition on L_0 in symbols and the L_1 solution as a solid line. The solution is marched explicitly in a "time-refi ned" manner, using a classic fourth order explicit Runge-Kutta integrator up to time $t = 1.0$ using a timestep (on the coarse mesh) of $\Delta t_0 = 1.0 \times 10^{-3}$. Such a small timestep was used to keep the temporal content in the discrepancy between the numerical and exact solutions small compared to the spatial content. Fig. 2 (inset) shows the movement of the profi le. We see that there is a non-zero gradient at the coarse-fi ne boundary. Such a situation might arise in practice if one is not extremely rigorous about the refi nement-derefi nement criteria. The solution at $t = 1.0$ is also shown.

In Fig. 3 we show the RMS error in the RHS i.e. $-(6UU_x + U_{xxx})$ with respect to the analytical solution on L_1. The x-axis is the resolution of the L_0 mesh. The choice of the RHS for convergence analysis is made in order to bring out clearly the effect of interpolation errors, which are best seen in the higher derivatives. We plot solutions obtained with $p_I = 4, 6$ and 8. A uniform mesh solution, i.e. without using interpolations, is also plotted as a guide for an ideal fourth-order convergence slope. We see that with $p_I = 4$ and 6 the slope is shallower than fourth order, while $p_I = 8$ achieves a greater slope. Thus, using $p_I = 8$ should enable p_D-order convergence.

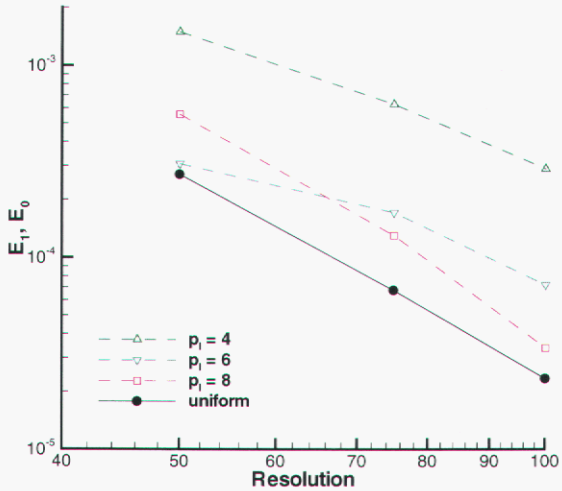

Figure 3. Convergence of the "right-hand-side" of the KdV equation on L_1 with $p_D = 4$ and $p_I = 4, 6$ and 8. The time is $t = 1.0$. A uniform mesh run is plotted as a guide for the convergence slope. $p_I = 8$ shows a convergence steeper than fourth order while the others do not. Note that the resolution here is L_0 resolution; L_1 is a factor of 2 finer. The effective resolution of a $L_0 = 50$ run is a uniform mesh of 100 and is compared with a uniform mesh run corresponding to the effective resolution. E_1 is the RMS error (with respect to the exact solution) on Level1; E_0 is the error for the uniform mesh run.

Note that a $\{L_0, L_1\}$ mesh has an *effective* resolution that is twice the L_0 resolution and should be compared with a uniform mesh solution that is equally resolved. This is done in Fig. 3 where the uniform mesh error plotted against an L_0 resolution of 50 was obtained on a mesh of 100 grid points. Further, the uniform mesh run is uniformly more accurate since the entire domain is well resolved; in the adaptive mesh case, we purposely limit the refi nement to $30 \leq x \leq 56$ to see the effect of the interpolants.

3.1.2 Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky (KS) equation can be written as

$$U_t + UU_x + U_{xx} + U_{xxxx} = 0 \tag{27}$$

Under the assumption that $U_x \to 0$ as $x \to \pm\infty$, we get a traveling wave solution

$$\begin{aligned} U &= \frac{15}{19}k(11H^3 - 9H^2 + 2), \qquad k = \sqrt{\frac{11}{19}} \\ H &= \tanh\left(\frac{k}{2}\left(x - x_0 - \frac{30}{19}kt\right)\right) \end{aligned} \tag{28}$$

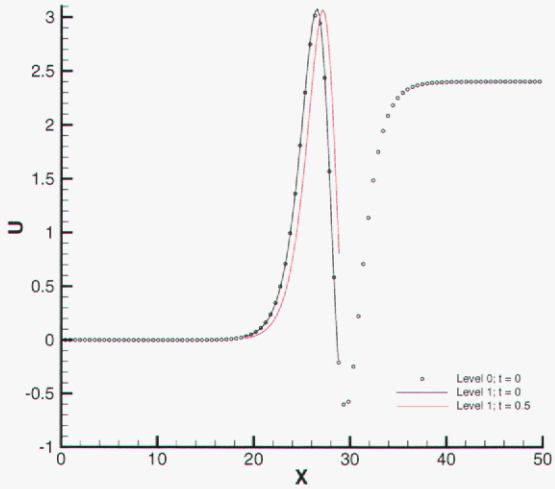

Figure 4. A traveling wave solution of Eq. 27 at $t = 0$ and 0.5. Symbols are used to plot the solution on L_0 and lines on L_1. The red line shows the solution at $t = 0.5$.

As in [20], we solve Eq. 27 on a two-level Grid Hierarchy with a factor of refinement of 2. The domain is $0 \leq x \leq 50$ with $x_0 = 29$. The region $20 \leq x \leq 29$ is refined to a L_1 mesh. $p_D = 4$ is used. The width was the halo was kept at six. Runge phenomenon was not observed and we did not use filters. Fig. 4 shows the initial condition on L_0 in symbols and the L_1 solution as a solid line. The solution is marched explicitly using a fourth order Runge-Kutta integrator to time $t = 0.5$ using a timestep (on the coarse mesh) of $\Delta t_0 = 5.0 \times 10^{-4}$. This solution is also shown.

In Fig. 5 we show the error in the RHS, i.e. $-(UU_x + U_{xx} + U_{xxxx})$ with respect to the analytical solution on L_1. The x-axis is the resolution of the L_0 mesh. We plot solutions obtained with $p_I = 6$ and 8. An ideal fourth-order convergence plot is provided as a guide. We see that with $p_I = 6$ the slope is shallower than fourth order, while $p_I = 8$ achieves a greater slope. Thus, using $p_I = 8$ should enable p_D-order convergence. However, the choice of interpolant order is also dependent on the position of the coarse-fine boundary. We repeated these runs with a refinement region of $20 \leq x \leq 38$. In this case, the gradient at the coarse-fine boundaries is small and both $p_I = 6$ and 8 provided fourth order convergence.

Combining the results obtained above with those in [20], we observe that $p_I \geq (p_D + H_D)$ is sufficient to ensure $(p_D)^{th}$ order spatial accuracy in block-structured adaptive meshes, even when there exists a substantial gradient at a coarse-fine boundary. This expression is seen to hold true at least for $\partial_{xx}, \partial_{xxx}$ and ∂_{xxxx} using fourth-order discretizations. We will use it to achieve high-order convergence in 2D below.

A similar study, performed using the Kawahara equation, could verify the relationship between p_I, p_D and H_D for $H_D = 5$.

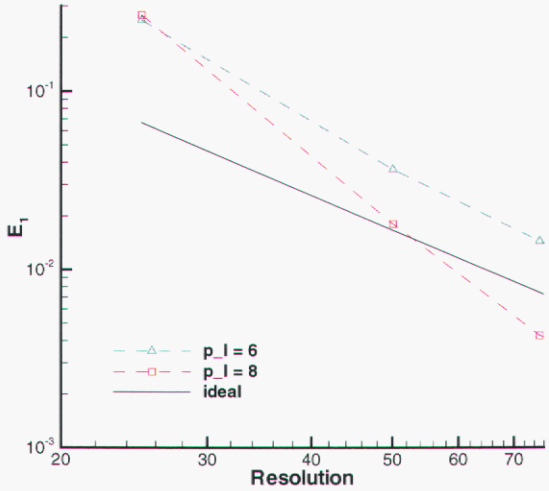

Figure 5. Convergence of the 'right-hand-side" of the KS equation on L_1 with $p_D = 4$ and $p_I = 6$ and 8. The time is $t = 0.5$. A ideal fourth-order convergence plot is provided for guidance. $p_I = 8$ shows a convergence steeper than fourth order while the others do not. Note that the resolution here is L_0 resolution; L_1 is a factor of 2 finer.E_1 is the RMS error on Level1.

3.2 Achieving high-order convergence

In this section we combine the tools developed in Sec. 2 to empirically achieve high order convergence on block-structured adaptive meshes in two-dimensions. Tests will be carried out in 2D using the FitzHugh-Nagumo equation. We choose a Level0 resolution that leads to the Runge phenomenon, because of the coarse effective resolution, thus necessitating the use of fi lters.

The FitzHugh-Nagumo (FN) equation in 2D is

$$U_t = D\nabla^2 U + AU(1-U)(U-\alpha) \tag{29}$$

We look for traveling wave solutions to Eq. 29, similar to the 1D case [27]. We assume that a solution of the form $U = U(\xi + st)$ exists where $\xi = x + \beta y - c$. We can derive the expression for $\nabla^2 U$ and U_t in terms of U' and U''. Under the assumption that

$$U' = aU(1-U) \tag{30}$$

where $U \to 0$ as $\xi \to -\infty$ and $U \to 1$ as $\xi \to \infty$ we get

$$a = \left(\frac{A}{2(1+\beta^2)D}\right)^{1/2}, s = \left(\frac{AD(1+\beta^2)}{2}\right)^{1/2}(1-2\alpha)$$

and

$$U = \frac{1}{2}\left(1 + \tanh\left(\frac{a\xi}{2}\right)\right) = \frac{1}{2}\left(1 + \tanh\left(\frac{\xi}{2\delta_f}\right)\right) \tag{31}$$

The last expression was obtained by solving the differential equation Eq. 30 and completes the solution of Eq. 29. δ_f, a measure of the front thickness, is defi ned as $\delta_f = 1/a$.

We solve Eq. 29 on a unit square. We specify $D = 1, c = 0.5$ and $\beta = 1$. A is set such that $\delta_f = 0.02$. Eq. 31 evaluated at $t = 0$ provides the initial condition, and is also used to specify Dirichlet boundary conditions on the unit square. The analytical solution, Eq. 31, is also used to evaluate the numerical error (difference between the exact and the numerical solutions) for convergence tests, etc. The region $0 \leq x \leq 0.7, 0 \leq y \leq 0.7$ was refi ned (see Fig. 6). The meshes on successive levels were refi ned by a factor of two.

3.2.1 Fourth-order approach

Eq. 29, when solved with fourth-order discretizations ($p_D = 4$) and sixth-order interpolants ($p_I = 6$), led to Runge phenomenon (see Fig. 6). This was not observed in a 1D context in [20], which also supplied the particular choice of (p_D, p_I). Thus fi ltering was required.

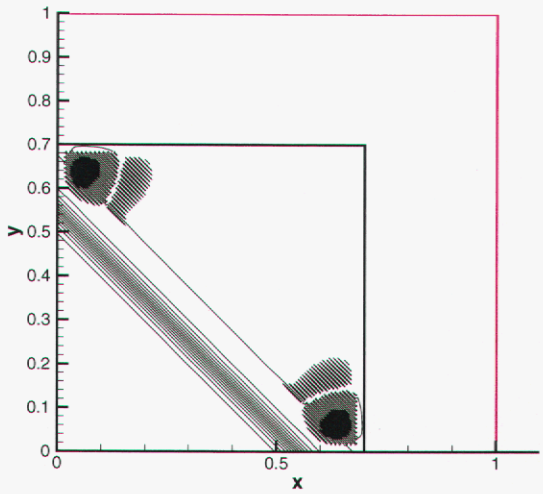

Figure 6. Runge phenomenon on a 2-level block-structured mesh. The patch outlined in black is the Level1 patch. Level0 is a 100×100 mesh on a unit square. The solution is at $t = 5 \times 10^{-5}$ and was computed with $p_D = 4$ and $p_I = 6$. The correct solution should not contain the structure/oscillations one observes at southeast and northwest corners of the Level1 patch.

In order to choose a fi lter of the correct order (p_F), we conducted a series of tests (see Fig. 7). A $(p_F)^{th}$ order fi lter is a $(p_F)^{th}$ spatial derivative in the interior points of the mesh implemented with a centered stencil that

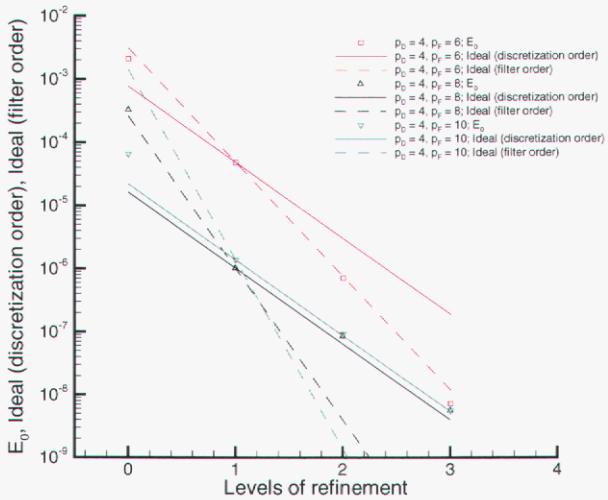

Figure 7. Numerical errors (calculated using the exact solution) for the FitzHugh-Nagumo equations solved using fourth-order discretizations. Numerical solutions computed with $p_F = 6, 8$ and 10 on grid hierarchies with up to 3 levels of refinement are plotted with □, △ and ▽ respectively. $(p_D)^{th}$ and $(p_F)^{th}$-order convergence are plotted with solid and dashed lines respectively. $p_F = 6, 8$ and 10 are represented by red, blue and black respectively. For $p_F \geq 8$, we see fourth-order convergence as filter errors are smaller than discretization errors. The y-axis has the Level0 RMS error (E_0) and the 'ideal" errors if the discretization or the filter errors dominate.

is of second order accuracy. The Level0 mesh is kept at 100×100. The solution was advanced to time $t = 4 \times 10^{-4}$ using a Level0 $\Delta t = 10^{-6}$. The width of the halo is kept at 6. The temporal component in the RMS difference between the numerical and the exact solution is expected to be negligible with respect to the spatial component. In Fig. 7, we plot the Level0 RMS error using symbols, though tests were done with zero, one, two and three levels of refinement (zero levels of refinement is a uniform mesh run). Lines are used to plot the ideal convergence (for the purpose of comparison) at both $(p_D)^{th}$ and $(p_F)^{th}$ order. For $p_F = 6$, it is clear that the numerical errors converge at 6th order; thus the errors introduced by filtering overwhelm the discretization errors. For $p_F = 8, 10$, the convergence of the numerical error is clearly $(p_D)^{th}$ as long as we refine the grid. Also, the difference between the $p_F = 8$ and $p_F = 10$ runs are small (and gets smaller as levels of refinement are introduced) indicating that the filter errors are insignificant. The clear exceptions are the uniform mesh runs, where the numerical errors do not seem to fall on either on the $(p_D)^{th}$ or $(p_F)^{th}$ order convergence plots, but are somewhere in between. Thus, at a 100×100 resolution, the discretization and filter errors are comparable. However, the magnitude of the filter errors (on the uniform mesh runs) decrease with the order of the filters.

Based on Fig. 7 we choose $p_F = 8$ and conduct convergence tests on 50×50, 100×100 and 200×200 Level0 meshes (see Fig. 8). Numerical errors calculated one each level are plotted, as levels of refinement are allowed. We see that the RMS errors on each of the levels are very similar, as might be expected since the solution is restricted from the finer to the coarser meshes. Further, as more levels of refinement are

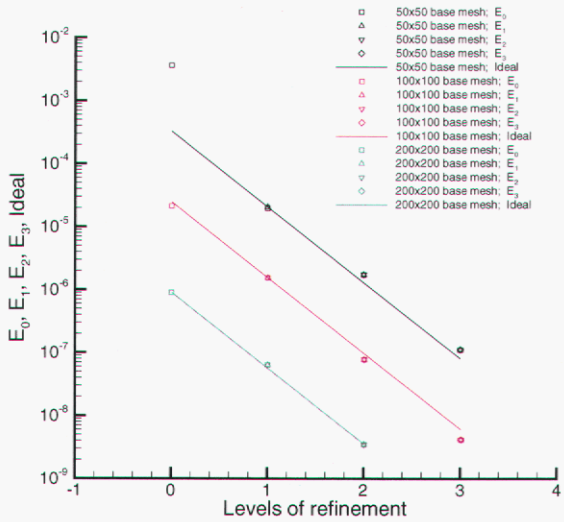

Figure 8. Numerical (RMS) errors observed when $(p_D = 4, p_F = 8)$. Solutions were done on 50×50 (black), 100×100 (red) and 200×200 (blue) meshes with different levels of refinement. Errors on Level 0, 1, 2 and 3 are plotted with □, △, ▽ and ◇ respectively. Ideal fourth-order convergence is plotted using solid lines. Apart from the 50×50 uniform mesh run, the errors follow the ideal convergence closely. $E_i, i = 0, \ldots 3$ refer to the RMS error on level i.

allowed, fourth-order convergence is observed. The only outlier is the 50×50 uniform mesh run; the grid is too coarse to resolve the traveling front and the errors are excessive.

3.2.2 Sixth-order approach

In this section, we address the question of using $p_D = 6$ instead of $p_D = 4$ as was done in Sec. 3.2.1. In Fig. 7, it was seen that $(p_D)^{th}$-order convergence was observed only with levels of refinement, when discretization errors dominated the temporal and filter errors (since it is the RMS difference between exact and numerical solutions that we plot). Since sixth-order stencils are expected to be more accurate than fourth-order ones, we study the interaction between filter and discretization errors in a simplified setting, with no interpolations and no variation in spatial resolution, viz. on a uniform mesh.

In Fig. 9, we plot the RMS difference between exact solutions and numerical solutions computed with $p_D = 6$ along with filters of order $p_F = 8, 10$ and 12. An unfiltered experiment is also plotted for comparison. Runs are done on $N \times N$ grid where $N = 100, 200$ and 400 with $\Delta t = 10^{-7}$. The problem was integrated up to time $t = 4 \times 10^{-4}$ as in Sec. 3.2.1. We see that without filtering, the errors converge ideally $(\mathcal{O}^h)$ order as N is increased. Filtering introduces a significant error and the behavior of the error for filtered runs lies between p_D and p_F. It is only at $N = 400$ that the $p_F = 12$ run approaches the unfiltered runs in accuracy. Thus, it is unlikely that a counterpart of Fig. 8 could be plotted; as can be seen in Fig. 9, filtering errors would dominate the discretization ones unless one started with a Level0 mesh of 400×400 - at which point, the solution will be resolved well enough on the Level0 mesh not to result in Runge phenomenon (and consequently, there

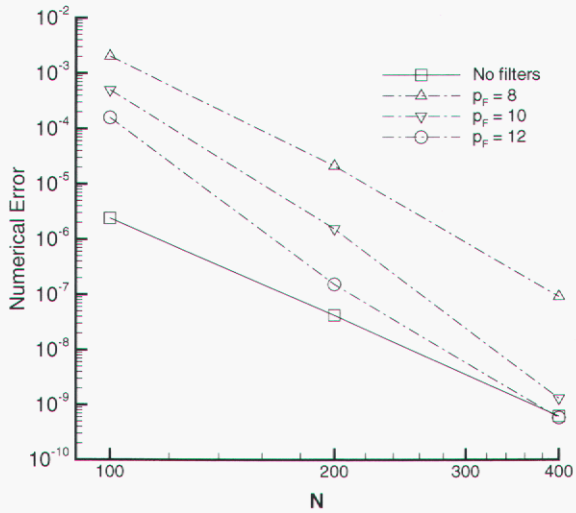

Figure 9. The RMS difference between numerical and exact solution for the FitzHugh-Nagumo equations solved on a $N \times N$ uniform mesh using $p_D = 6$ with filters of order $p_F = 8, 10$ and 12. Errors from runs without filtering are also plotted. $\triangle, \triangledown, \bigcirc$ are used to denote $p_F = 8, 10$ and 12 results; $\square$ is used for runs without filtering.

would be no need for filtering).

However, the choice of p_D in a scientific simulation will probably be made on grounds of accuracy and/or resolution requirements rather than the ability to show the theoretical convergence. Consequently, we redo the problems in Sec. 3.2.1 with $p_D = 6$, with the same filters and compare with the $p_D = 4$ runs. Δt is kept the same, but $p_I = 6$ is used, in keeping with the findings in [20]. Using $p_I = 8$ with $p_D = 4$ made no difference in the results. In Fig. 10 only errors on Level0 are plotted; errors on other levels are very similar, as seen for $(p_D)^{th}$ runs in Fig. 8. On a uniform 100×100 mesh, the filter errors dominate and there is little difference between the $p_D = 4$ and 6 runs. As levels of refinement are added, both the filter and discretization errors decrease, but at different rates. Even at one level of refinement, the discretization errors dominate for the $p_D = 4$ runs and the difference between the $(p_D = 4, p_F = 8)$ and $(p_D = 4, p_F = 10)$ are insignificant. Not surprisingly, a fourth-order convergence is seen. This is not the case in the $p_D = 6$ results; there is a significant difference in the results computed with $p_F = 8$ and 10, indicating that the filter errors are dominant. It is only at three levels of refinement that the results become comparable. However, except for the Level0 run, $p_D = 6$ errors are mostly smaller than $p_D = 4$ errors, thus recovering some of the advantages of using a high-order method.

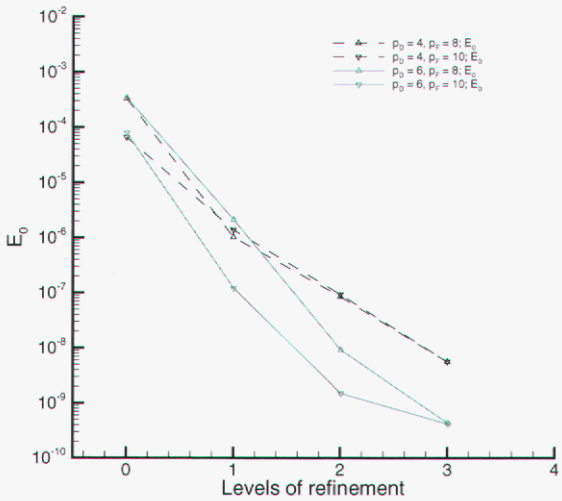

Figure 10. Comparison of the RMS difference between exact and numerical solutions of the FitzHugh-Nagumo equations when computed with $p_D = 4, p_I = 8$ and $p_D = 6, p_I = 8$ and filtered with filters of order $p_F = 8$ and 10. $\Delta t = 10^{-6}$ and the problem is integrated till $t = 4 \times 10^{-4}$. Results are plotted for a uniform mesh and meshes with 1 to 3 levels of refinement with a Level0 mesh of 100×100. $\triangle$ and $\triangledown$ indicate $p_F = 8$ and 10; dashed lines indicate $p_D = 4$ and solid ones $p_D = 6$. E_0 is the RMS error on Level0.

4 Conclusions

We have developed expressions for high-order discretizations, interpolations, and fi lters, in multiple dimensions, using a Fourier approach. This particular choice, as opposed to the traditional Taylor series approach, enables implementation of these high-order operations in higher dimensions with relative ease. We also provided a brief Mathematica [26] program that produces the necessary code for all these high-order operations. We demonstrated the implementation of these high-order constructions on three model problems in one and two-dimensions.

We also derived a formulation/rule for the requisite interpolation order for given discretization and derivative orders. We examined this rule empirically by pairing high-order stencils and interpolations on SAMR grid hierarchies and examining high-order convergence where analytical results were available. Achieving such a convergence requires care in the choosing of the fi lters and interpolants.

- For the simple boundary conditions used here time-instability near the boundaries was not observed, at least with the fourth and sixth order approaches investigated in this paper.
- Oscillations in the solution caused by high-order interpolants, due to insuffi cient resolution, was indeed a problem. As a part of the solution procedure, solutions are generated at all grid points on all levels, even though the solution in the overlaid areas is later discarded (during restrictions). During

this process, oscillatory interpolated values were generated by the high order interpolants, usually on Level1 and Level2, leading to the Runge phenomenon. Filters remove this problem but introduce dissipation errors, and consequently need to be chosen with care. In some cases they may dominate the discretization errors, depending on the grid resolution and discretization order. Since in block-structured AMR there will be patches/grids with low resolution, filters may turn out to be as important a factor in determining accuracy as the discretizations.

Combined with [20], this paper presents an alternative approach to the one developed in [3]. Our method, vis-a-vis [3], trades off one complexity versus another. While the high-order 1D interpolations in [3] are far simpler than our multi-dimensional ones, their application requires book-keeping viz. tracking (a) patch configurations and (b) tangential and normal directions to a coarse-fine boundary, whereas we use our interpolations uniformly. Also, unlike [3], we have no requirement in principle that the refinement ratio between levels be any particular value, although the practical complexities of deriving and implementing the high-order multidimensional interpolations have led us to use a fixed refinement factor of 2 in the present work. This also ensures that the difference in (resolvable) wavenumber spectra between successive levels of refinement are kept manageable. While this is not much of a problem in vertex centered grids where restrictions (interpolating the fine mesh solution to the coarse mesh) are mere injections, cell-centered grids require interpolations to be performed and care must be taken not to transfer high wavenumbers' energies to lower ones while interpolating. This is obviously simpler if the wavenumber spectra are not too different. However, the ramifications of the choice (2 versus 4) of the level-to-level refinement factor, within the context of high order AMR, needs a systematic study. Furthermore, as shown in Sec. 3, our interpolations have consistently provided high-order convergence even when the coarse-fine boundary was placed, on purpose, in a region of high gradient. While such an eventuality is not likely in an AMR simulation, the threshold-based refinement-derefinement criteria does not guarantee against it and the high-order interpolations serve as a fallback in borderline resolution cases.

In conclusion, high-order methods, coupled with AMR, hold the potential to perform high-accuracy simulations economically. These methods, as presented in this paper, are quite general - discretizations, filters and interpolations of any order n may be generated using the expressions developed here. There are neither special cases nor assumptions and/or constructions predicated on a specific choice of stencils. Replacements of interpolations etc are effected simply by replacing subroutines. [20] presented a comparison of the computational work, i.e. the floating point operations required for the same problem with second and fourth-order discretizations. The difference between the two, even at modest accuracy requirements (0.1 - 1 %), is significant in favor of the fourth order approach. At tighter accuracies, the difference is startling. This is observed in [3] too. A degree of care needs to be exercised when choosing the interpolants and filters needed in such computations, and we have indicated some principles in this paper. We see that errors introduced by filters may become the determining factor as far as accuracy is concerned, but this will happen at high discretization orders.

References

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

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

[3] M. Barad and P. Colella. A fourth-order accurate local refi nement method for Poisson equation. *Journal of Computational Physics*, 209, 2005. Online edition;doi:10.1016/j.jcp.2005.02.027.

[4] L. Jameson. A wavelet-optimized, very high order adaptive grid and order numerical method. *SIAM Journal on Scientific Computing*, 19(6):1980–2013, 1998.

[5] L. Jameson and T. Miyama. Wavelet analysis and ocean modeling: Dynamically adaptive numerical method "WOFD-AHO. *Monthly Weather Review*, 128(5):1536–1548, 2000.

[6] L. Jameson. AMR vs High Order Schemes. *Journal of Scientific Computing*, 18(1), 2003.

[7] L. Jameson. High-order shcemes for RM and RT. Technical Report UCRL-ID-141537, Lawrence Livermore National Laboratories, Livermore, CA, 2000.

[8] I. Fatkullin and J. S. Hesthaven. Adaptive high-order fi nite difference method for nonlinear wave problems. *SIAM Journal on Scientific Computing*, 16(1):47–67, 2001.

[9] M. Holmstrom. Solving hyperbolic PDEs using interpolating wavelets. *SIAM Journal on Scientific Computing*, 21(2):405–420, 1999.

[10] L. N. Trefethen and J. A. C. Weideman. Two results on polynomial interpolation in equally spaced points. *Journal of Approximation Theory*, 65(3):247–260, 1991.

[11] J. Ray and L. Jameson. Estimation of shock induced vorticity on irregular gaseous interfaces: A wavelet-based approach. *Shock Waves : An International Journal*, 14(3):147–160, 2005.

[12] R. Samtaney. Suppression of the Richtmyer-Meshkov instability in the presence of a magnetic fi eld. *Physics of Fluids*, 15(8):L53–L56, 2003.

[13] M. H. Carpenter, D. Gottlieb, and S. Abarbanel. Stable and accurate boundary treatments for compact, high-order fi nite-difference schemes. *Applied Numerical Mathematics*, 12(1-3):55–87, 1993.

[14] P. Olsson. Summation by parts, projections, and stability. I. *Mathematics of Computations*, 64(211):1035–1065, 1995.

[15] P. Olsson. Summation by parts, projections, and stability. II. *Mathematics of Computations*, 64(212):1473–1493, 1995.

[16] B. Strand. Numerical studies of hyperbolic IBVP with high-order fi nite difference operators satisfying a summation by parts rule. *Applied Numerical Mathematics*, 26(4):497–521, 1998.

[17] M. Gunther, A. Kværno, and P. Rentrop. Multirate partitioned Runge-Kutta methods. *BIT*, 41:504–514, 2001.

[18] C. A. Kennedy and M. H. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44:139–181, 2003.

[19] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations I; Stiff and Differential Algebraic Equations*. Springer-Verlag, Berlin, 1996.

[20] S. Lefantzi, J. Ray, C. A. Kennedy, and H. N. Najm. A component-based toolkit for reacting flows with high order spatial discretizations on structured adaptively refi ned meshes. *Progress in Computational Fluid Dynamics: An International Journal*, 5(6):298–315, 2005.

[21] L. Jameson. AMR vs. high-order schemes, wavelets as a guide. Technical Report UCRL-ID-141537, Lawrence Livermore National Laboratory, Livermore, California, 2000.

[22] C. A. Kennedy and M. H. Carpenter. Several new numerial methods for compressible shear layer simulations. *Applied Numerical Mathematics*, 14:397–433, 1994.

[23] J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Wadsworth & Brooks/Cole Mathematics Series, Belmont, California, 1989.

[24] A. D. Polyanin and V. F. Zaitsev. *Handbook of Nonlinear Partial Differential Equations*. Chapman and Hall/CRC, Boca Raton, 2004.

[25] C. A. Kennedy and M. H. Carpenter. Comparison of several numerical methods for simulation of compressible shear layers. Technical Report 3484, National Aeronautical and Space Administration, Langley Research Center, Hampton, VA 23681-2199, 1997.

[26] S. Wolfram. *The Mathematica Book*. Wolfram Media, 1999.

[27] C. P. Fall, E. S. Marland, J. M. Wagner, and J. J. Tyson, editors. *Computational Cell Biology*, volume 20 of *Interdisciplinary Applied Mathematics, Mathematical Biology*. Springer, 2001.

A Stencil coeffi cient computation

Below is a Mathematica[26] program that may be used to generate a FORTRAN77 fi le containing all boundary and interior stencils discussed in this paper. On a workstation having dual 2.2GHz processors, fourth-, sixth, eighth-, and tenth-order stencils require approximately 1, 8, 29, and 189 seconds to compute. First, write individual stencils as functions.

```
 (* Second-Order *)
Psi2I[a_]:=Cos[a/2];
Psi2a[a_]:=(Cos[a]+I*Sin[a])*(Cos[a/2]+I*(-2*Sin[a/2]));

(* Fourth-Order *)
Psi4I[a_]:=Cos[a/2]*(-5+Cos[a])/4;
Psi4a[a_]:=(Cos[a]+I*Sin[a])*(5*Cos[a/2]+3*Cos[3*a/2]+
I*(-10*Sin[a/2]-2*Sin[3*a/2]))/8;
Psi4b[a_]:=Cos[2*a]+I*Sin[2*a])*(7*Cos[a/2]-15*Cos[3*a/2]+
I*(-28*Sin[a/2]+20*Sin[3*a/2]))/8;

(* Sixth-Order *)
Psi6I[a_]:=Cos[a/2]*(89-28*Cos[a]+3*Cos[2*a])/64;
Psi6a[a_]:=(Cos[a]+I*Sin[a])*(70*Cos[a/2]+63*Cos[3*a/2]-5*Cos[5*a/2]+
I*(-140*Sin[a/2]-42*Sin[3*a/2]+2*Sin[5*a/2]))/128;
Psi6b[a_]:=Cos[2*a]+I*Sin[2*a])*(42*Cos[a/2]-135*Cos[3*a/2]-35*Cos[5*a/2]+
I*(-168*Sin[a/2]+180*Sin[3*a/2]+28*Sin[5*a/2]))/128;
Psi6c[a_]:=(Cos[3*a]+I*Sin[3*a])*(198*Cos[a/2]-385*Cos[3*a/2]+315*Cos[5*a/2]+
I*(-1188*Sin[a/2]+770*Sin[3*a/2]-378*Sin[5*a/2]))/128;

(* Eighth-Order *)
Psi8I[a_]:=Cos[a/2]*(-762+299*Cos[a]-54*Cos[2*a]+5*Cos[3*a])/512;
Psi8a[a_]:=(Cos[a]+I*Sin[a])*(525*Cos[a/2]+567*Cos[3*a/2]-75*Cos[5*a/2]+
7*Cos[7*a/2]+I*(-1050*Sin[a/2]-378*Sin[3*a/2]+30*Sin[5*a/2]-
2*Sin[7*a/2]))/1024;
Psi8b[a_]:=Cos[2*a]+I*Sin[2*a])*
(231*Cos[a/2]-891*Cos[3*a/2]-385*Cos[5*a/2]+21*Cos[7*a/2]+
I*(-924*Sin[a/2]+1188*Sin[3*a/2]+308*Sin[5*a/2]-12*Sin[7*a/2]))/1024;
Psi8c[a_]:=(Cos[3*a]+I*Sin[3*a])*(429*Cos[a/2]-1001*Cos[3*a/2]+1365*Cos[5*a/2]+
231*Cos[7*a/2]+I*(-2574*Sin[a/2]+2002*Sin[3*a/2]-1638*Sin[5*a/2]-
198*Sin[7*a/2]))/1024;
Psi8d[a_]:=Cos[4*a]+I*Sin[4*a])*(3575*Cos[a/2]-7371*Cos[3*a/2]+5775*Cos[5*a/2]- 3003*Cos[7*a/2]+I
3432*Sin[7*a/2]))/1024;

(* Tenth-Order *)
Psi10I[a_]:=Cos[a/2]*(25609-11528*Cos[a]+2708*Cos[2*a]-440*Cos[3*a]+
35*Cos[4*a])/16384;
```

```
Psi10a[a_]:=(Cos[a]+I*Sin[a])*(16170*Cos[a/2]+19404*Cos[3*a/2]-
3300*Cos[5*a/2]+539*Cos[7*a/2]-45*Cos[9*a/2]+I*(-32340*Sin[a/2]-
12936*Sin[3*a/2]+1320*Sin[5*a/2]-154*Sin[7*a/2]+10*Sin[9*a/2]))/32768;
Psi10b[a_]:=-(Cos[2*a]+I*Sin[2*a])*(6006*Cos[a/2]-25740*Cos[3*a/2]-
14300*Cos[5*a/2]+1365*Cos[7*a/2]-99*Cos[9*a/2]+I*(-24024*Sin[a/2]+
34320*Sin[3*a/2]+11440*Sin[5*a/2]-780*Sin[7*a/2]+44*Sin[9*a/2]))/32768;
Psi10c[a_]:=(Cos[3*a]+I*Sin[3*a])*(7722*Cos[a/2]-20020*Cos[3*a/2]+
35100*Cos[5*a/2]+10395*Cos[7*a/2]-429*Cos[9*a/2]+I*(-46332*Sin[a/2]+
40040*Sin[3*a/2]-42120*Sin[5*a/2]-8910*Sin[7*a/2]+286*Sin[9*a/2]))/32768;
Psi10d[a_]:=-(Cos[4*a]+I*Sin[4*a])*(24310*Cos[a/2]-55692*Cos[3*a/2]+
56100*Cos[5*a/2]-51051*Cos[7*a/2]-6435*Cos[9*a/2]+I*(-194480*Sin[a/2]+
148512*Sin[3*a/2]-89760*Sin[5*a/2]+58344*Sin[7*a/2]+5720*Sin[9*a/2]))/32768;
Psi10e[a_]:=(Cos[5*a]+I*Sin[5*a])*(293930*Cos[a/2]-639540*Cos[3*a/2]+
554268*Cos[5*a/2]-285285*Cos[7*a/2]+109395*Cos[9*a/2]+I*(-2939300*Sin[a/2]+
2131800*Sin[3*a/2]-1108536*Sin[5*a/2]+407550*Sin[7*a/2]-121550*Sin[9*a/2]))/32768;
```

Next, set desired parameters. This is the only section of the code that needs to be modified by the user. In the following example, we wish to compute a fourth-order stencil for a point shifted from the center by zero grid points in x, $\Psi_I^{4(x)}$, one in y, $\Psi_{\mathrm{a}}^{4(y)}$, and two in z, $\Psi_{\mathrm{b}}^{4(z)}$, or $\Psi_{I,\mathrm{a,b}}^{4(xyz)}$.

```
 (* Specify stencil *)
order = 4;
Psi = Expand[ TrigToExp[ Psi4I[x]*Psi4a[y]*Psi4b[z] ] ];
ix = 0; iy = 1; iz = 2;
```

Alternatively, we could compute the stencil $\Psi_{\mathrm{b,c,d}}^{8(xyz)}$ by writing

---

```
 (* Specify stencil *)
order = 8;
Psi = Expand[ TrigToExp[ Psi8b[x]*Psi8c[y]*Psi8d[z] ] ];
ix = 2; iy = 3; iz = 4;
```

Now the actual stencil coefficients are computed and printed to the screen in FORTRAN77 format.

```
 (* Compute stencil *)
leftx=-(order-1)+2*ix;
```

```
lefty=-(order-1)+2*iy;
leftz=-(order-1)+2*iz;
psi = Expand[ Factor[ Normal[ Series[ Psi,{x,0,order},{y,0,order},{z,0,order}]]]];
Print[" psi = ", psi ];
psilistx = CoefficientList[ psi - 1,x] /.  {y -> 0, z -> 0};
psilisty = CoefficientList[ psi - 1,y] /.  {x -> 0, z -> 0};
psilistz = CoefficientList[ psi - 1,z] /.  {x -> 0, y -> 0};
actualorder = Sum[ (psilistx[[i]]+psilisty[[i]]+psilistz[[i]]), {i,1,order-1}];

Do[
Print["Do loop counter = ", k];
Do[
Do[ bnd[i,j,k] = ExpToTrig[ Expand[
Psi*Exp[-I*(leftx+2*(i-1))*x/2]*Exp[-I*(lefty+2*(j-1))*y/2]*Exp[-I*(leftz+2*(k-1))*z/2]]];
,{i,1,order}];
x = 3; y = 17; z = 9973;
Do[ bnd[i,j,k] = Part[ bnd[i,j,k], 1 ];, {i,1,order}];
,{j,1,order}]; ,{k,1,order}];

denominator = Part[ {8,2^(12),2^(24),2^(33),2^(48),2^(58)}, order/2];
letters = {I,a,b,c,d,e,f};
Do[ bnd[i,j,k] = Factor[ denominator*bnd[i,j,k] ];,
{i,1,order},{j,1,order},{k,1,order}];
Do[ Print[" bnd",order,Part[letters,ix+1],Part[letters,iy+1],Part[letters,iz+1],
"(",i,",",j,",",k,") = ", bnd[i,j,k],".d0/",denominator,".d0" ];,
{i,1,order},{j,1,order},{k,1,order}];
If[ actualorder == 0, Print["Stencil is correct"], Print["Stencil is wrong"]];
```

B Interior Stencils for 3D interpolants

Stencil	2nd-order	4th-order	6th-order	8th- order	10th-order
aaa	1/8	$729/2^{12}$	$3375000/2^{24}$	$1838265625/2^{33}$	$62523502209000/2^{48}$
baa	0	$-81/2^{12}$	$-562500/2^{24}$	$-367653125/2^{33}$	$-13894111602000/2^{48}$
bba	0	$9/2^{12}$	$93750/2^{24}$	$73530625/2^{33}$	$3087580356000/2^{48}$
bbb	0	$-1/2^{12}$	$-15625/2^{24}$	$-14706125/2^{33}$	$-686128968000/2^{48}$
caa	0	0	$67500/2^{24}$	$73530625/2^{33}$	$3572771554800/2^{48}$
cba	0	0	$-11250/2^{24}$	$-14706125/2^{33}$	$-793949234400/2^{48}$
cbb	0	0	$1875/2^{24}$	$2941225/2^{33}$	$176433163200/2^{48}$
cca	0	0	$1350/2^{24}$	$2941225/2^{33}$	$204158374560/2^{48}$
ccb	0	0	$-225/2^{24}$	$-588245/2^{33}$	$-45368527680/2^{48}$
ccc	0	0	$27/2^{24}$	$117649/2^{33}$	$11666192832/2^{48}$
daa	0	0	0	$-7503125/2^{33}$	$-637994920500/2^{48}$
dba	0	0	0	$1500625/2^{33}$	$141776649000/2^{48}$
dbb	0	0	0	$-300125/2^{33}$	$-31505922000/2^{48}$
dca	0	0	0	$-300125/2^{33}$	$-36456852600/2^{48}$
dcb	0	0	0	$60025/2^{33}$	$8101522800/2^{48}$
dcc	0	0	0	$-12005/2^{33}$	$-2083248720/2^{48}$
dda	0	0	0	$30625/2^{33}$	$6510152250/2^{48}$
ddb	0	0	0	$-6125/2^{33}$	$-1446700500/2^{48}$
ddc	0	0	0	$1225/2^{33}$	$372008700/2^{48}$
ddd	0	0	0	$-125/2^{33}$	$-66430125/2^{48}$
eaa	0	0	0	0	$55135363500/2^{48}$
eba	0	0	0	0	$-12252303000/2^{48}$
ebb	0	0	0	0	$2722734000/2^{48}$
eca	0	0	0	0	$3150592200/2^{48}$
ecb	0	0	0	0	$-700131600/2^{48}$
ecc	0	0	0	0	$180033840/2^{48}$
eda	0	0	0	0	$-562605750/2^{48}$
edb	0	0	0	0	$125023500/2^{48}$
edc	0	0	0	0	$-32148900/2^{48}$
edd	0	0	0	0	$5740875/2^{48}$
eea	0	0	0	0	$48620250/2^{48}$
eeb	0	0	0	0	$-10804500/2^{48}$
eec	0	0	0	0	$2778300/2^{48}$
eed	0	0	0	0	$-496125/2^{48}$
eee	0	0	0	0	$42875/2^{48}$
$\psi_{N,N,N}\xi^N\eta^N\zeta^N$	$-(1/8)\xi^2$	$-(3/128)\xi^4$	$-(5/1024)\xi^6$	$-(35/32768)\xi^8$	$-(63/262144)\xi^{10}$
	$-(1/8)\eta^2$	$-(3/128)\eta^4$	$-(5/1024)\eta^6$	$-(35/32768)\eta^8$	$-(63/262144)\eta^{10}$
	$-(1/8)\zeta^2$	$-(3/128)\zeta^4$	$-(5/1024)\zeta^6$	$-(35/32768)\zeta^8$	$-(63/262144)\zeta^{10}$

Table B.1. Coefficients of three-dimensional, vertex-centered interpolants in terms of ξ, η, and ζ.

C Rules for pairing a derivative and an interpolation stencil

In this section we determine a rule to pair a spatial derivative stencils of order P with an interpolation of order Q given that the higher spatial derivative that needs to be evaluated is R.

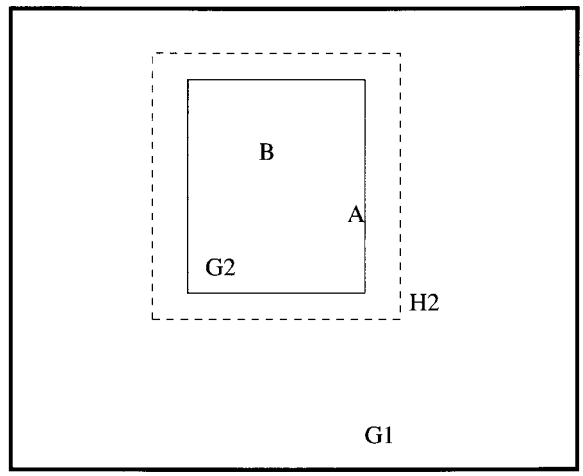

Figure C.1. Two overlaid patches G1 and G2. G2 is finer and has a halo H2 around it to facilitate the evaluation of discrete derivatives via stencils. Points A and B lie on the finer G2 patch.

Let $\phi^{(R)}$ be the Rth derivative of ϕ in the domain $\mathcal{A}$, the patch G2 in Fig. C.1. Let $D_P^{(R)}$ be a discrete operator (a stencil) that, when applied at a point in $\mathcal{A}$ evaluates the derivative with an error proportional to $(\Delta x)^P$. We will analyze only in the x-direction where points are indexed using i; the results will carry over to higher dimensional space.

Applying this at point B in Fig. C.1 poses no problems,

$$\begin{aligned} \phi_B^{(R)} &= \frac{D_P^{(R)}(\phi_B)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P) \\ &= \frac{D_P^{(R)}(\ldots,\phi_{i-1},\phi_i,\phi_{i+1},\ldots)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P) \end{aligned} \tag{32}$$

as long as the stencil has enough points on the left and right. At A however, the stencil "spills" over into the coarser G1 patch.

Typically, in order not to use skewed stencils, one keeps a halo H2 of points around G2, where data is interpolated from G1 with an interpolant of order Q. Let these interpolated ϕ be called $\tilde{\phi}$ i.e

$$\tilde{\phi} = \phi + \varepsilon_2(\Delta x^Q)$$

Thus we evaluate $\phi_A^{(R)}$ as

$$\begin{aligned}
\phi_A^{(R)} &= \frac{D_P^{(R)}(\phi_A)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P) \\
&= \frac{D_P^{(R)}(\ldots,\phi_{i-1},\phi_i,\tilde{\phi}_{i+1},\ldots)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P) \\
&= \frac{D_P^{(R)}(\ldots,\phi_{i-1},\phi_i,\phi_{i+1},\ldots)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P) + \varepsilon_3(\Delta x^{Q-R})
\end{aligned} \tag{33}$$

That $\varepsilon_1(\Delta x^P)$ and $\varepsilon_3(\Delta x^{Q-R})$ be of the same order requires

$$Q = P + R. \tag{34}$$

This is suffi cient to allow P^h order convergence of $\phi^{(R)}$. Note that ε_1 exists over $\mathcal{A}$ while ε_3 exists only along $\partial\mathcal{A}$, its boundary, a domain of dimension one less than $\mathcal{A}$.

However, the requirement above can be relaxed. Often, errors are measured, integrated and analyzed in a certain area. This was also the case in this paper. Thus,

$$\begin{aligned}
E &= \int_{\mathcal{A}} \varepsilon_1(\Delta x^P)dA + \int_{\partial\mathcal{A}} \varepsilon_3(\Delta x^{Q-R})ds \\
&= E_1(\Delta x^{P-2}) + E_3(\Delta x^{Q-R-1})
\end{aligned} \tag{35}$$

if $\mathcal{A}$ is 2D. If $\mathcal{A}$ is 3D, the right hand side is $E_1(\Delta x^{P-3}) + E_3(\Delta x^{Q-R-2})$. In order that E_1 and E_3 have the same convergence rates,

$$Q = P + R - 1. \tag{36}$$

Strictly speaking, Eq. 36 will ensure P^{th} order convergence of $\phi^{(R)}$ in a global sense if $\partial\mathcal{A}$ is one grid point in thickness. For our high-order discretizations, this is not the case; interpolations are done in a thin *region* of half the stencil width. Thus it is unclear whether Eq. 34 or Eq. 36 is preferable, but Eq. 34 should be suffi cient.

DISTRIBUTION:

6 M9159
Jaideep Ray, 08964

1 M9056
Habib Najm, 08351

3 MS 9018
Central Technical Files, 8940-1

1 MS 0899
Technical Library, 9616

1 MS 9021
Classifi cation Offi ce, 8511, for
Technical Library, MS 0899, 9616
DOE/OSTI via URL