Advanced algorithms for computations on block-structured adaptively refined meshes

Jaideep Ray[1], **C. Kennedy**[2], **J. Steensland and H. Najm**

Sandia National Laboratories, Livermore, CA

E-mail: `jairay@ca.sandia.gov`

Abstract. Block-structured adaptively refined meshes are an efficient means of discretizing a domain characterized by a large spectrum of spatiotemporal scales. Further, they allow the use of simple data structures (multidimensional arrays) which considerably assist the task of using them in conjunction with sophisticated numerical algorithms. In this work, we show how such meshes may be used with high order (i.e. greater than 2nd order) discretization to achieve greater accuracies at significantly less computational expense, as compared to conventional second order approaches. Our study explores how these high order discretizations are coupled with high-order interpolations and filters to achieve high order convergence on such meshes. One of the side-effects of using high order discretizations is that one now obtains shallow grid hierarchies, which are easier to load balance. As a part of this work, we introduce the concept of bi-level (grid) partitioning and motivate, via an analytical model, how it holds the potential to significantly reduce load-imbalances while incurring a minimal communication cost.

1. Introduction

Essential to any computational method is the manner in which a continuous domain of interest is discretized into a mesh. Generally, if the computation exhibits structures far smaller than the domain size in sparsely dispersed clusters, an adaptive, locally refined meshing approach is adopted to minimize the size the mesh while ensuring that solution structures are properly resolved. One such adaptive meshing technique is block-structured adaptive mesh refinement (SAMR). The SAMR [1] method can be summarized as follows: a coarse Cartesian mesh is overlaid on a rectangular domain, and, based on a suitably defined error metric, the grid points which require further refinement are identified. These points are flagged and collated into *rectangular* child patches on which another, denser, Cartesian mesh is imposed. This is done recursively, leading to the formation of a Grid Hierarchy, GH. The refinement factor between the parent and the child mesh is usually kept constant for a given problem (2 for the purposes of this paper). The solution can be advanced in time using explicit or implicit time integration; we use the time-refinement technique described in [2]. The solution is advanced with a different timestep (as determined by stability constraints) on each level of the GH. Periodically the levels are "synchronized" by interpolating the accurate solutions from the finer to the coarser meshes. The sequence in which the different levels are integrated corresponds to the in-order traversal of a binary tree.

Since SAMR techniques always refine regions in rectangular patches, the multidimensional array is the obvious data structure of choice. Such a simple solution renders SAMR a very attractive option for multiscale simulations when the domain geometries are simple. However, there are certain practical

[1] Corresponding author

[2] Formerly at Sandia National Laboratories, Livermore, CA

Table 1. Coefficients for fourth and sixth order accurate derivative stencils. Third order skewed stencils used at a right boundary are also shown. Their left counterparts are obtained by reflection and an inversion of signs. In the first column, S denotes stencil, suffix E indicates a symmetric explicit stencil while U indicates an upwind/skewed one. L.O.T.E. stands for Leading Order Truncation Error.

S	c_L	b_L	a_L	Υ	a_R	b_R	c_R	L.O.T.E.
4E	0	1/12	-2/3	0	2/3	-1/12	0	-(1/30)ξ^5
6E	-1/60	3/20	-3/4	0	3/4	-3/20	1/60	+(i/140)ξ^7
3U	0	0	-2/6	-1/2	1	-1/6	0	-(i/12)ξ^4
3UU	0	0	0	-11/6	3	-3/2	1/3	+(i/4)ξ^6
4U	0	0	-1/4	-5/6	3/2	-1/2	1/12	+(i/20)ξ^5

difficulties. Time-refinement requires that fine mesh patches sub-cycle far more than coarse patches, and most of the time is spent computing (and communicating) at the finer levels. Further, sub-cycling is done in a recursive manner, leading to significant recursion overheads if the GH is deep. Interpolations from coarse-to-fine patches (and vice versa) also add a significant cost to SAMR approaches, especially in 3D. Sub-cycling also results in a peaked distribution of work load (as a function of space) which is difficult to partition in an equitable manner for domain decomposition based parallelization. There are two possible solutions: (a) avoid deep hierarchies by using high order ($> 2^{\text{nd}}$ order) discretizations to reduce the need for excessive refinement and (b) adopt a quasi-domain-decomposition based partitioning approach that achieves load-balance at the price of some communication cost. Both these approaches are investigated below.

2. High order discretizations on SAMR meshes

High order methods exploit solution characteristics (usually smoothness) to achieve accuracy at modest resolutions. In a SAMR context, where solutions are interpolated between coarse and fine meshes, they need to be paired with interpolants of an appropriate order. Also, since high order approaches have low numerical dissipation, unresolved wave numbers in the solution have to be explicitly filtered out before they corrupt the solution. Further, wave-numbers resolved by less than approximately six grid points often cause interpolants to fail [3].

All first-order derivatives in this work can be written as

$$f'_i = \frac{c_L f_{i-3}}{(\Delta x)} + \frac{b_L f_{i-2}}{(\Delta x)} + \frac{a_L f_{i-1}}{(\Delta x)} + \frac{\Upsilon f_i}{(\Delta x)} + \frac{a_R f_{i+1}}{(\Delta x)} + \frac{b_R f_{i+2}}{(\Delta x)} + \frac{c_R f_{i+3}}{(\Delta x)}. \quad (1)$$

where the coefficients for the 4th and 6th order discretizations (the stencils of interest for the purposes of this paper) are in Table 1. At domain boundaries, skewed operators of a lower order (3rd) are used (see in Table 1). Interpolations of data from a coarse parent to a halo of points around a fine child patch (*prolongation*) is used to allow the use of symmetric discretization operators on the fine patches. Periodically, data is also *restricted* from the fine patches to the coarse ones. Interpolations are done using a square (in 2D; cube in 3D) patch around the target interpolation point; at domain boundaries the interpolation squares are skewed. Details of the coefficients for discretization, filters and interpolations are in [4, 5]. We will denote the order of discretization, interpolation and filter by p_D, p_I and p_F respectively.

We solve a problem $U_t = D\nabla^2 U + AU(1-U)(U-\alpha)$ subject to the boundary condition $\nabla U \to 0$ as $\mathbf{x} \to \pm\infty$ on a unit square with a 2-level GH ($\{L_0, L_1\}$). This equation admits a traveling wave solution $U(\xi,t) = \frac{1}{2}(1+\tanh\frac{\xi}{2\varepsilon})$ where $\varepsilon = 2\sqrt{\frac{D}{A}}$, $\xi = x+y-0.5+st$ and $s = \sqrt{AD}(1-2\alpha)$. We

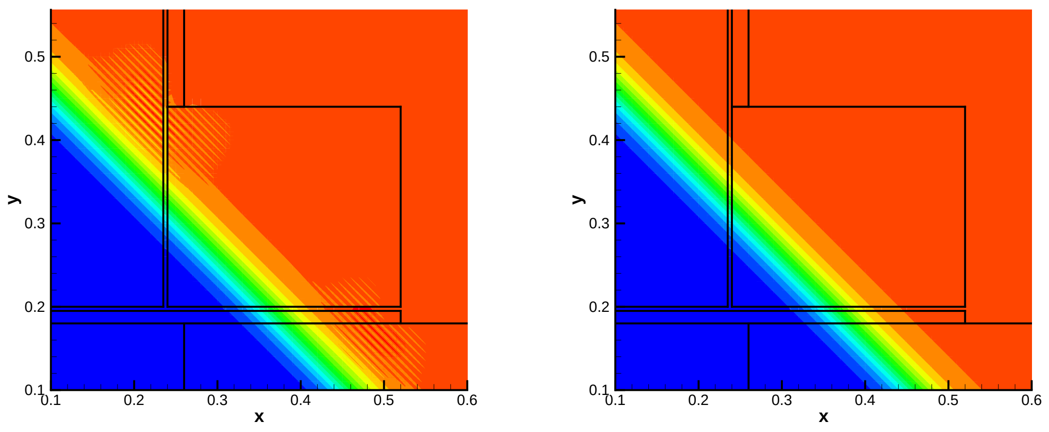

Figure 1. Results from a 2-level GH run with $p_D = 4, p_I = 6$ on a 100×100 $\{L_0\}$ mesh. Results from a zoom into $0.1 < x < 0.6, 0.1 < y < 0.55$ are shown at $t = 6 \times 10^{-4}$. The result on the left was computed without filtering. We see oscillatory Runge phenomena developing around patch edges and propagating inward. Applying a $p_F = 8$ filter at the beginning of each timestep removes the problem (frame on the right).

choose $D = 1.0, \alpha = 0.3$ and $\varepsilon = 0.02$. The problem is formulated along the lines of [6] without using special stencils at coarse–fine patch interface. Fourth and sixth order spatial discretizations are used in conjunction with Heun's method (RK-2) to advance the solution to $t = 10^{-3}$. Eighth order filters are used.

Fig. 1 shows the effect of filters; their absence causes the Runge phenomenon. The choice of p_D, p_I and p_F was guided by [4]. In Fig. 2 we see plot the convergence of error (w.r.t. the analytical result) on both L_0 and L_1 for three different L_0 resolutions. We see that in both cases the desired order of convergence (4^{th} and 6^{th} for the left and right figures respectively) are achieved on L_0 while L_1 shows a faster convergence since the errors there are dominated by the higher order interpolation errors. Thus the 1D results from [4] hold true in the present 2D implementation.

3. Bi-level partitioning of SAMR meshes

Consider a 2D domain distributed over a number of processors after being subjected to a purely domain-based decomposition. Let a sub-domain on a processor have N_0 grid points on the coarsest level. $\mathcal{G}_l$ refers to the set of all patches on level l. Let a fraction of this coarse level be refined into level 1 patches, contained in the set $\mathcal{G}_1$. Thus, the number of grid points in $\mathcal{G}_1$, (i.e., the load on $\mathcal{G}_1$) is $\mathcal{L}_1 = \sum_{i=0}^{M_1-1} \alpha_i^1 N_0 R$ where M_l is the number of patches in $\mathcal{G}_l$, R is the refinement factor between 2 successive levels (usually 2) and α_i^l is the fraction of the sub-domain that exists in patch i on level l. Patches on a level are indexed from 0 to $M_l - 1$. Let $\mathcal{G}_1$ be recursively refined into $\mathcal{G}_2, \mathcal{G}_3, \ldots$. Let L denote the index of the finest level. During a time-step, the compute load T_{comp} is $T_{comp} = t_{comp} N_0 \sum_{l=0}^{L} R^{2l} \sum_{i=0}^{M_l-1} \alpha_i^l$ where t_{comp} is the *computation time / load of a unit operation*. Let us assume that patches are roughly square and that the width of the halo of grid points kept around a patch to enable the use of symmetric stencils $\propto \sqrt{\alpha_i^l N_0 R^l}$. Since the *intra-level* communication follows a similar pattern as the computation, $T_{comm} = t_{comm} \sum_{l=0}^{L} \sum_{i=0}^{M_l-1} 4\beta_i^l \sqrt{\alpha_i^l N_0 R^l} R^l$ where t_{comm} is a *unit communication time* and β_i^l is the fraction of the perimeter of patch $G_{l,i}$ that abuts another patch on the same level but in a different sub-domain. At a given level, data is interpolated from child patches to the parent at the end of each time step. If t_{interp} is the unit interpolation cost, then the total time spent in interpolation, T_{interp}, is $T_{interp} = N_0 t_{interp} \sum_{l=0}^{L-1} R^{2l+1} \sum_{i=0}^{M_{l+1}-1} \alpha_i^{l+1}$. Note that T_{interp} in a purely domain-based partitioning does not incur a communication cost. Thus, the total time for an arbitrary processor to execute a time-step, T_{total}, is $T_{total} = T_{comp} + T_{interp} + T_{comm} + t_{wait}$ where t_{wait} is the wait induced by load-imbalance among processors.

Typically, SAMR packages [7, 8] load-balance by moving an arbitrary patch $G_{l,i}$ from a processor with $t_{wait} = 0$ to another with maximum t_{wait} i.e., the least loaded one. This saves on compute τ^i_{comp} and interpolation τ^i_{interp} loads for the patch i but incurs the cost of *inter-level* communication, i.e., the cost of bringing back the interpolated data from the off-processor patch: $\alpha_i^l N_0 R^{2l-1} t_{comm}$. Let γ denote the fraction of this communication time that could not be overlapped with computation. Then, the load change on the sending processor

$$\Delta t_{send} = -\tau^i_{comp} - \tau^i_{interp} + \gamma\tau^i_{comm} = \alpha_i^l N_0 R^{2l-1}(\gamma t_{comm} - t_{interp} - R t_{comp}). \tag{2}$$

Thus, the requirement for $\Delta t_{send} < 0$ is $\gamma t_{comm} < t_{interp} + R t_{comp}$. This is rarely realized since t_{interp} and t_{comp} are far smaller than t_{comm} on todays fast processors. However, load-balance and (sub-linear) scalability are realized.

We now motivate an alternate approach. A way to render $\Delta t_{send} < 0$ is to increase the savings in compute and interpolation costs while keeping the communication overhead unchanged. We consider the case of moving all patches above level "q" off-processor in an effort to reduce T_{total}. Thus Δt_{send} is given by

$$\begin{aligned} \Delta t_{send} &= \sum_{\mathcal{G}_q} \tau^i_{comm} - \sum_{\mathcal{G}_m, m>q} (\tau^i_{comp} + \tau^i_{interp}) \\ &= \sum_{i=0}^{M_q-1} \alpha_i^q N_0 R^{2q} \gamma t_{comm} - \sum_{l=q+1}^{L} \sum_{i=0}^{M_l-1} \alpha_i^l N_0 R^{2l} t_{comp} - \sum_{l=q}^{L-1} \sum_{i=0}^{M_{l+1}-1} \alpha_i^{l+1} N_0 R^{2l+1} t_{interp}. \end{aligned} \tag{3}$$

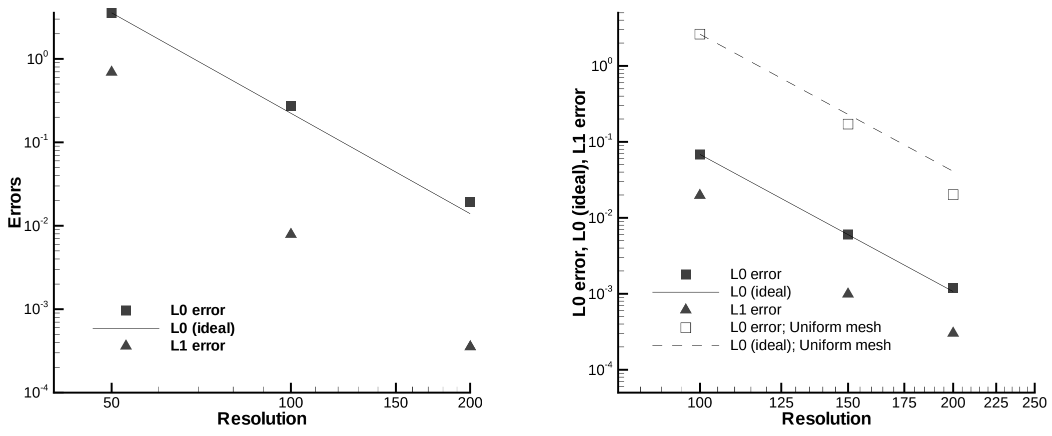

Figure 2. Convergence of the error with $p_D = 4, P_I = 6$ (left) and $p_D = 6, P_I = 8$ (right). The L_0 mesh was discretized at 3 different resolutions. The numerical error (with respect to the analytical solution) is plotted with symbols; the ideal convergence (fourth or sixth) is plotted as a line. We see that fourth (sixth) order convergence is seen on L_0 in the left (right) figures. L_1 convergence is dominated by interpolation errors which converge as $p_I > p_D$. We use $\Delta t = 10^{-5}$ and 0.25×10^{-5} for the $p_D = 4$ and $p_D = 6$ runs respectively. In the $p_D = 6$ results, we plot the convergence of the error from a uniform mesh run as a reference for the convergence slope.

Since q only takes integral values between 1 and L, the above expression can be evaluated repeated to seach for the largest q for which $\Delta t_{send} < 0$. This thus enables one to partition a GH partially, in "units" of bi-levels, thus providing an extra parameter in the partitioning problem. We are currently evaluating partitioners based on bi-level / partial partitioning, particularly with the view of quantifying the sensitivity of Δt_{send} with respect to q in realistic problems.

4. Conclusions

SAMR approaches for computational science, though attractive, pose unusual challenges in the context of parallel efficiency, scalability and load-balancing. In some cases, as was done in Sec. 2, techniques common in uniform mesh and/or unstructured mesh computations may be adapted to SAMR meshes. In conjunction with [4], it is clear that high order convergence can be realized on SAMR meshes provided that discretizations, interpolants and filters are chosen appropriately. This opens up the possibility of achieving similar accuracies and efficiencies on SAMR meshes that high order numerical schemes have delivered for uniform meshes. They also hold the possibility of rendering many of the issues that arise from deeply refined grids irrelevant by making them unnecessary. In case deep refinements are unavoidable, robust load-partitioning methods that optimize a single metric (e.g. load balance) may be sub-optimal, as was shown in Sec. 3. Such cases require specialized approaches, which, in the adaptive context, usually require that the metric being optimized itself be dynamic. Such an adaptive approach to load-partitioning requires that a GH be characterized w.r.t. its parallel efficiency and "partitionability" so that an appropriate partitioning technique can be chosen. Preliminary efforts and their results [9] are encouraging.Coupling them with more sophisticated partial-partitionings as motivated in Sec. 3 are under way.

Acknowledgments

This work was supported by the US Department of Energy (DOE), Office of Basic Energy Sciences (BES), SCIDAC Computational Chemistry Program; and by the DOE BES Division of Chemical Sciences, Geosciences, and Biosciences. J. Steensland was supported under Sandia's ASC (Advanced Simulation and Computing) program. Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000.

References

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

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

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

[4] Sophia Lefantzi, Jaideep Ray, Christopher A. Kennedy, and Habib N. Najm. A component-based toolkit for reacting flows with high order spatial discretizations on structured adaptively refined meshes. *Progress in Computational Fluid Dynamics*, 5:298–315, 2005.

[5] C. A. Kennedy and M. H. Carpenter. Several new numerial methods for compressible shear layer simulations. *Appl. Num. Math.*, 14:397–433, 1994.

[6] P. J. J. Ferket and A. A. Reusken. A finite difference discretization method for elliptic problems on composite grids. *Computing*, 56:343–369, 1996.

[7] CHOMBO, 2003. http://seesar.lbl.gov/anag/chombo/, NERSC, ANAG of Lawrence Berkeley National Lab, CA, USA.

[8] Scott Kohn. SAMRAI homepage, structured adaptive mesh refinement applications infrastructure, 1999. http://www.llnl.gov/CASC/SAMRAI/.

[9] Johan Steensland and Jaideep Ray. A partitioner-centric model for samr partitioning trade-off optimization: Part i. *International Journal of High Performance Computing Applications*, 19:1–14, 2005.