PARALLEL ADAPTIVE MESH REFINEMENT FOR FOSLS-AMG

L. TANG[†]

Abstract. This paper develops new adaptive mesh refinement (AMR) strategies for first-order system least-squares (FOSLS) finite elements in conjunction with algebraic multigrid (AMG) methods in the context of nested iteration (NI). The goal is to reach a certain error tolerance with the least amount of computational cost and nearly uniform distribution of the error over all elements. To accomplish this, the refinement decisions at each refinement level are determined based on minimizing the "accuracy-per-computational-cost" (ACE) efficiency measure that take into account both error reduction and computational cost. The NI-FOSLS-AMG-ACE approach produces a sequence of refinement levels in which the error is equally distributed across elements on a relatively coarse grid. Once the solution is numerically resolved, refinement becomes nearly uniform. Accommodations of the ACE approach to massively distributed memory architectures involve a geometric binning strategy to reduce communication cost. Load balancing begins at very coarse levels. Elements and nodes are redistributed using parallel quadtree structures and a space filling curve. This automatically ameliorates load balancing issues at finer levels. Numerical results show that the NI-FOSLS-AMG-pACE approach is able to provide highly accurate approximations to rapidly varying solutions using a small number of work units. Excellent weak and strong scalability are demonstrated on 4,096 processors for problems with 15 million biquadratic elements.

1. Introduction. Adaptive Mesh Refinement (AMR) are being used extensively to approximate solutions of partial differential equations (PDEs) containing local features; see, e.g., [11, 14, 18, 9, 3]. The refinement process starts on a coarse grid, $\mathcal{T}_0$ (level = 0), and iteratively refines and approximates the PDE on levels $\ell = 1, 2, ...$ until the error satisfies a certain criterion. The goal is to construct a sequence of grids that converge to an optimal grid. A grid is called optimal if it is capable of approximating the solutions of given PDEs to certain accuracy with the least number of degree of freedom (DOF). In one dimension, it has been proved that this is accomplished by equally distributing the error over all elements; [11]. It is believed that this also holds for higher dimensions. Based on this premise, a simple method to mark elements for refinement was introduced by Babuška [11]: an element is marked for refinement if its local error is within a certain factor of the largest local error at that level. In [9], a more complicated algorithm was proposed.

ALGORITHM 1 (Threshold-based Marking). *Given a parameter $0 < \theta \leq 1$, construct a minimal subset $\hat{\mathcal{T}}$ of $\mathcal{T}$ such that the local error in $\hat{\mathcal{T}}$ is no less than θ times the global error and mark all elements in $\hat{\mathcal{T}}$ for refinement.*

The AMRs in [12] start with this approach, then further mark elements based on oscillation terms. With a proper choice of θ, one can establish the convergence as well as near optimality of the finest grid. However, the real computational cost was not addressed. Also, the proper choice of θ is different for various problems and unknowns *a priori*. We argue that the goal of AMR should take into account the real computational cost, meaning, the goal should be to achieve the optimal grid with the least amount of work. To do that would require the value of θ to be free to change on each level. A new approach, described in [8, 13], was developed to address this issue. The algorithm refines elements that minimize the 'accuracy-per-computational-cost'

[*]I would like to thank my advisor, Tom Manteuffel and Steve McCormick, for guidance. Thanks also to John Ruge, Marian Brezina, and Josh Nolting for their preliminary works on pFOSPACK.

[†]Department of Applied Mathematics, University of Colorado at Boulder, Boulder, CO 80309–0526 (tangl@colorado.edu).

efficiency factor (ACE). The ACE algorithm was applied to first-order system least-sqares (FOSLS) finite element methods in the conjunction with algebraic multigrid (AMG) in the context of nested iteration (NI). The NI-FOSLS-AMG approach yields measures that allows us to estimate the error reduction and computational cost, which can be used to make the refinement decisions. The results show that the approach is capable of effectively and efficiently detecting the solution's local features; see [1, 13].

More computing resources are almost always desired to achieve the ever-increasing demands on solution resolution. For this reason, this paper focuses developing the NI-FOSLS-AMG-ACE approach on massively parallel distributed memory machines. Extending ACE algorithm in parallel utilizes geometric binning strategies. Elements are grouped into bins based on local error, then refinement decisions can be made based on treating each bin as an abstract element. We demonstrate that the parallel ACE algorithm produces results similar to the original serial algorithm, but greatly reduces the communication cost. Load balancing issues must be addressed to scale dynamical AMR in parallel. We start to balance the load on coarser grids, where the computation and communication are relatively cheap. At each refinement level, a space filling curve and parallel tree structures are used to redistribute nodes and elements in order to preserve locality of the new partition and, thus, reduce communication cost. We present results to demonstrate excellent strong and weak scalability.

This paper is organized as follows. In section 2, the basic concepts of the NI-FOSLS-AMG-ACE approach are described. Accommodation of the refinement strategies in parallel and details of the parallel AMR implementations are discussed in section 3. Next, in section 4, numerical efficiency and parallel scalability of the NI-FOSLS-AMG-pACE approach are demonstrated by test results. Finally, conclusions are formulated in section 5.

2. Preliminaries. In this section, we briefly describe the basic concepts behind the NI-FOSLS-AMG-ACE approach.

2.1. FOSLS Methodology. FOSLS is a finite element method that reformulates a PDE as a system of first-order equations and poses the problem as a minimization of a functional. To illustrate the basic concepts, consider a first-order system

$$\mathcal{L}_i \mathbf{u} = f_i, \qquad i = 1, 2, ..., M. \tag{2.1}$$

Assuming $f_i \in L^2(\Omega)$, consider the associated FOSLS functional given by

$$\mathcal{G}(\mathbf{u}, \mathbf{f}) = \sum_{i=1}^{M} ||\mathcal{L}_i \mathbf{u} - f_i||^2_{0,\Omega}, \tag{2.2}$$

where $||u||_{0,\Omega} = \sqrt{\int_\Omega |u|^2}$ is the L^2-norm. The minimization problem is

$$\mathbf{u} = \arg\min_{\mathbf{v} \in \mathcal{V}} \mathcal{G}(\mathbf{v}; \mathbf{f}). \tag{2.3}$$

Here, $\mathcal{V}$ is an appropriate Hilbert space, usually (equivalent to) a product of H^1 spaces. In many cases, under general regularity assumptions, $\mathcal{G}(\mathbf{u}; \mathbf{f})$ is "elliptic" with respect to the $\mathcal{V}$ norm; see, e.g., [6, 7]. That is, its homogeneous part, $\mathcal{G}(\mathbf{v}; \mathbf{0})$, is equivalent to the squared $\mathcal{V}$ norm:

$$c_1 \leq \frac{\mathcal{G}(\mathbf{v}; \mathbf{0})}{||\mathbf{v}||^2_{\mathcal{V}}} \leq c_2 \tag{2.4}$$

for some positive constants c_1 and c_2 and for every $\mathbf{v} \in \mathcal{V}$. This ellipticity enables the existence and uniqueness of the solution, $\mathbf{u}$. Let $\mathcal{V}^h \subset \mathcal{V}$ be a finite-dimensional subspace of $\mathcal{V}$. The discretization can be written as the minimization problem

$$\mathbf{u}^h = \arg \min_{\mathbf{v}^h \in \mathcal{V}^h} \mathcal{G}(\mathbf{v}^h; \mathbf{f}). \tag{2.5}$$

Well-posedness of (2.5) follows directly from the ellipticity.

The FOSLS functional provides a unique capability for adaptive refinement: an effective error indicator at no additional computation cost. For any element τ, define the local FOSLS functional,

$$\mathcal{G}_\tau(\mathbf{u}^h; \mathbf{f}) = \sum_{i=1}^{M} ||\mathcal{L}_i \mathbf{u}^h - f_i||^2_{0,\tau}. \tag{2.6}$$

Writing $\epsilon_\tau = \sqrt{\mathcal{G}_\tau(\mathbf{u}^h; \mathbf{f})}$, the ellipticity in (2.4) implies that

$$\frac{1}{c_2}\epsilon_\tau^2 = \frac{1}{c_2}\mathcal{G}_\tau(\mathbf{u}^h - \mathbf{u}; 0) \leq ||\mathbf{u}^h - \mathbf{u}||^2_{\mathcal{V},\tau} \tag{2.7}$$

and

$$||\mathbf{u}^h - \mathbf{u}||^2_{\mathcal{V}} \leq \frac{1}{c_1}\mathcal{G}(\mathbf{u}^h - \mathbf{u}; 0) = \frac{1}{c_1}\sum_{\tau \in \mathcal{T}} \epsilon_\tau^2. \tag{2.8}$$

An error estimator, ϵ_τ, that satisfies an inequality of type (2.7) is called locally sharp. It implies that if ϵ_τ is large, then the error is large within that element. In the literature, an inequality of type (2.8) is called a reliability bound; see [18]. Note that a small sum of local estimators, ϵ_τ, implies a small global error.

2.2. Algebraic Solver. Nested iteration (NI), or full-multigrid [4] (FMG) as it is called in the multigrid context, involves starting the solution process on a relatively coarse grid, where the computational cost is relatively cheap. The solution on the coarse grid is used as an initial guess for the problem on the next finer grid. Since the objective on each grid is to minimize the FOSLS functional, the coarse-grid solution should provide a good starting guess. On each refinement level, solving discrete minimization problem (2.5) involves fast iterative solvers applied to the matrix equations. If the FOSLS functional is equivalent to a product H^1 norm, then there exists an optimal multilevel solution algorithm [19, 7]. Experience shows that, in this context, AMG also yields an approximate solution to the discrete equations associated with quasi-uniform grids in optimal time with convergence factor, ρ, bounded uniformly below 1, independent by mesh size h. AMG methods, together with the NI strategy and local refinement, provide a powerful approach for approximating solutions of PDEs. Numerical and theoretical results confirm that the overall cost of such a scheme resides predominantly in the cost of the finest-level processing. The total cost is usually cheaper than solving the problem directly on the finest grid, which generally is not even known in advance. Details can be found in [1, 2, 16].

2.3. Efficiency-based Adaptive Refinement for NI-FOSLS-AMG. Given a potential refinement strategy, one needs to estimate

- the reduction in the functional norm of the error and
- the computational cost of solving the resulting system of equations.

2.3.1. FOSLS Approximation Heuristics. If element τ_i with polynomial degree p is split in two in each dimension, then we have 2^d new elements $\tau_{i,1}, ..., \tau_{i,2^d}$ in $\mathbb{R}^d$. Theory developed in [13, 1] shows that, under rather mild hypotheses, we can estimate the local functional after refinement as

$$\epsilon^2_{\tau_{i,j}} \approx \left(\frac{1}{2^{2p+d}}\right) \epsilon^2_{\tau_i} \tag{2.9}$$

and

$$\sum_{j=1}^{2^d} \epsilon^2_{\tau_{i,j}} \approx \frac{1}{2^{2p}} \epsilon^2_{\tau_i}. \tag{2.10}$$

2.3.2. Estimates for NI-AMG. The algorithm starts by ordering the elements at level ℓ so that

$$\epsilon^2_1 \geq \epsilon^2_2 \geq ... \geq \epsilon^2_{N_\ell}. \tag{2.11}$$

The goal is to ensure that elements that contain large local error are refined first. Let $r \in [0, 1]$ be the fraction of elements to be refined. Define $E_\ell(r)$, the associated fraction of the functional on the elements to be refined; that is,

$$E_\ell(r) = \frac{\sum_{i \leq rN_\ell} \epsilon^2_i}{\sum_{i=1}^{N_\ell} \epsilon^2_i}. \tag{2.12}$$

The algorithm allows an element to be refined more than once at each level. Let $r_i \in [0, 1]$ be the fraction of elements to be refined i times at level ℓ. Let m be the maximum refinements allowed on a single level. Writing $\mathbf{r} = (r_1, r_2, ..., r_m)$ and combining (2.12) and (2.10), we estimate the functional reduction as a function of $\mathbf{r}$:

$$\gamma_\ell(\mathbf{r}) = 1 - E_\ell(r_1) + \sum_{k=1}^{m-1} \frac{1}{2^{2kp}} \left[E_\ell(r_{k+1}) - E_\ell(r_k)\right] + \frac{1}{2^{2mp}} E_\ell(r_m). \tag{2.13}$$

For the work required to achieve this reduction, the anticipated increase in DOF is easily computed

$$N_{\ell+1} \simeq \eta_\ell(\mathbf{r}) N_\ell, \tag{2.14}$$

where

$$\eta_\ell(\mathbf{r}) = 1 - r_1 + \sum_{k=1}^{m-1} 2^{kn}(r_{k+1} - r_k) + 2^{md} r_m. \tag{2.15}$$

Assume that, at each level, AMG V-cycles are applied until the discretization error, measured by the FOSLS functional norm $\sqrt{\mathcal{G}_\ell(\mathbf{u}^h; \mathbf{f})}$, is resolved. Then, the anticipated number of V-cycles, $\kappa_{\ell+1}(\mathbf{r})$, is determined by

$$\rho^{\kappa_{\ell+1}} = \sqrt{\gamma_\ell(\mathbf{r}))}. \tag{2.16}$$

Now, the work on the next level, $\ell + 1$, is given by

$$\mathcal{W}_{\ell+1}(\mathbf{r}) = [c_s + c_v \kappa_{\ell+1}(\mathbf{r})] \times N_{\ell+1} = [c_s + c_v \kappa_{\ell+1}(\mathbf{r})] \times \eta_\ell(\mathbf{r}) \times N_\ell, \tag{2.17}$$

where c_s and c_v represent the respective set-up cost and cost factor for a V-cycle.

2.3.3. ACE algorithm. The ACE algorithm is based on minimizing the effective error reduction at each refinement level.

ALGORITHM 2 (ACE). *At level ℓ, order the elements so that*

$$\epsilon_1^2 \geq \epsilon_2^2 \geq ... \geq \epsilon_{N_\ell}^2.$$

Allow m-multiple refinements, e.g., $m = 1, 2, 3$. Let $\mathbf{r} = (r_1, r_2, ..., r_m)$ with $0 \leq r_m \leq ... \leq r_1 \leq 1$. Find

$$\mathbf{r}_{opt} = arg\min_{\mathbf{r}} \gamma_\ell(\mathbf{r})^{\frac{1}{W_{\ell+1}(\mathbf{r})}}. \tag{2.18}$$

Then, refine the first $\lceil r_i N_\ell \rceil$ elements i times, $i = 1, 2, ..., m$.

The efficiency and effectiveness of the ACE approach were demonstrated in [1, 13].

3. Parallel Implementation.

3.1. Parallel Efficiency-based Marking Strategies. The main difficulty to parallelize the ACE algorithm is the global sort of local functional values. The usual way to deal with this problem is binning strategy. Refining elements that contains similar local functionals would result in similar error reduction per cost, it is reasonable to group elements according to local functionals into bins, and treat each bin as an abstract element in the ACE marking scheme. Two major aspects are considered to devise the binning strategy.

- It should produce results similar to that without using bins.
- It should greatly reduce communication cost.

A binning strategy can be seen to be equivalent to applying a piecewise linear interpolation, $I_h E(r)$, to $E(r)$, (see Fig 3.1). Now, the choice of a binning strategy becomes the question of how best to interpolate $E(r)$ with as few nodes as possible. When error distribution is close to uniform, all binning strategies would result in nearly uniform refinement. We care about the case when large error concentrates in a small fraction of elements. A good piecewise linear interpolation to such a function requires more resolution near zero, where the derivative is large. The geometric binning strategy is considered for this purpose; see Fig (3.1). Let $\epsilon^2_{\max}$ be the maximum local functional value among all elements. Choosing $0 < q < 1$, the strategy creates bins such that the first bin has elements with local functional values in the range of $[\epsilon^2_{\max}, q\epsilon^2_{\max}]$, the second bin has elements with local functional values in $[q\epsilon^2_{\max}, q^2\epsilon^2_{\max}]$, etc.

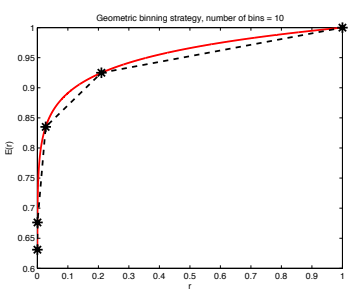


FIG. 3.1. *Geometric binning strategy: local error concentrates in the first a few elements.*

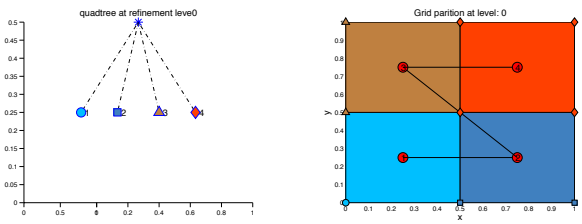
Recall that the local functional reduction by a single h-refinement of an order p element in $\mathbb{R}^d$ is approximated by

$$\epsilon^2_{i,j} \approx \frac{1}{2^{2p+d}}\epsilon_i^2, \qquad \text{for } j = 1, 2, ..., 2^d. \tag{3.1}$$

We argue that one can choose $q = \frac{1}{2^{2p+d}}$ so that if an element in the i^{th} bin is refined, and the $(i+1)^{\text{st}}$ bin is not refined, then children elements from refining the i^{th} bin will land in the $(i+1)^{\text{st}}$ bin. Doing this results in almost all elements are assigned to the first one or two bins on finer levels, which leads to equal distribution of error.

3.2. Quadtree-based Parallel Adaptive Mesh Refinement. Space filling curves and quadtree (octree in 3D) structures are being used to scale dynamical AMR to thousands of processors; see [15, 17]. These methods usually employ quadrilateral or hexahedra elements. Refining the mesh is equivalent to refining the associated tree (forest). The Pre-order traversal of the refined tree generates a Lebesgue space filling curve (or Morton ordering of the elements), which is used for load balancing. Examples of quadtree-based AMR implementations are Octor [17] and ALPS [5].

To illustrate the basic ideas of quadtree-based parallel AMR methods, consider an example of one element that covers the unit square. A single h-refinement results in 4 child elements. The parent-children correspondence is represented by the quadtree in Fig. 3.2(a). A pre-order traversal gives the SFC that connects the four children elements. If the grid is partitioned to 4 processors, one can partition the curve into four equal segment, and assign the i^{th} segment to the i-processor (see Fig. 3.2). Now,



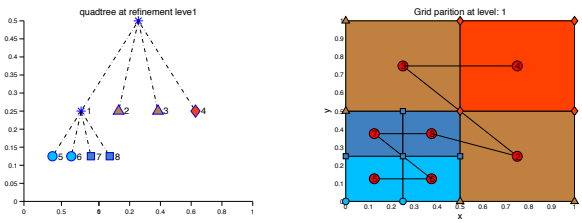
(a) Quadtree at the coarsest level (b) Initial grid partition

FIG. 3.2. *Initial grid partition based on space filling curve.*

suppose element #1 is refined, which gives four children, $\#5, \#6, \#7, \#8$. The tree is expanded (or refined). The pre-order traversal gives the following order

$$\#5 \to \#6 \to \#7 \to \#8 \to \#2 \to \#3 \to \#4,$$

which is the Lebesgue curve that connects active elements in the refined mesh. Equal



(a) Quadtree after local refinement (b) Grid partition after local refinement

FIG. 3.3. *Grid partition after refinement.*

partition of the curve yields the new partition of elements

$$(\#5, \#6) \to \text{ proc } \#0 \qquad (\#7, \#8) \to \text{ proc } \#1$$
$$(\#2, \#3) \to \text{ proc } \#2 \qquad (\#4) \to \text{ proc } \#3$$

Next, nodes are assigned to processors according to the new partitioning. At last, ghost elements and nodes are loaded, which completes the load balancing; see Fig (3.3).

4. Performance Study. The test problem is the Poisson equation on the unit square, $\Omega = (0,1) \times (0,1)$,

$$\begin{cases} -\Delta p = f(x,y) & \text{in } \Omega, \\ p = g & \text{on } \partial\Omega. \end{cases} \tag{4.1}$$

H^1-ellipticity of the FOSLS formulation is shown in [7].

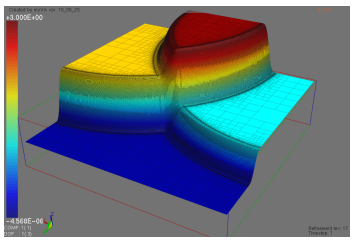
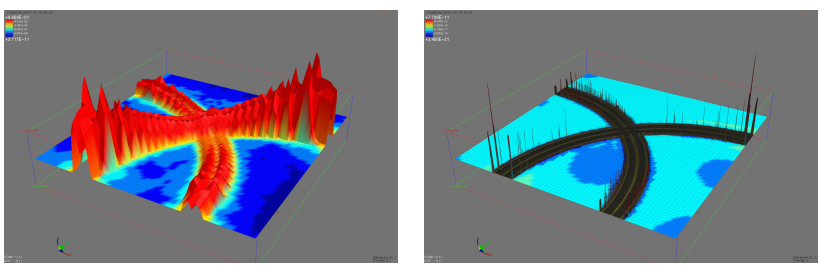


FIG. 4.1. *Exact solution*

The right-hand side, f, and boundary data, g, are chosen such that the exact solution, p, is given by Fig. 4.1. All tests are done on Frost, a four-rack Blue Gene/L system. A V(1,1)-BoomerAMG[10] cycle with symmetric Gauss-Seidel smoother is used as preconditioner for the conjugate gradient method. We measure all the work in terms of work units (WU) on the finest grid. For more test results, see [16].

4.1. Numerical Performance. Numerical results presented in this section are from a run on $1,024$ processors. Local functional distribution and grid alignment at different refinement levels are plotted in Fig. 4.2. Various values with respect to each refinement level are tabulated in Table. 4.1. We make a few observations. First, the



(a) Functional distribution at level 1 (b) Functional distribution at level 9

FIG. 4.2. *Poisson with steep gradients: locally-refined mesh and functional distribution.*

pACE algorithm refines a large fraction of elements at coarser levels when grids are too coarse to resolve the solution. Later, at the intermediate levels, refinements are concentrated in a small fraction of the elements where large errors are detected. After that, refinement becomes nearly uniform since the error is fairly equal-distributed. Efficiency of the algorithm can be reflected by the total work units. Another observation is that refinement on the finest grid is almost uniform refinement, which requires zero load balancing.

To illustrate the efficiency of the NI-FOSLS-AMG-pACE approach, we take the finest grid resulting from pACE, set up the FOSLS discrete problem, and solve it using AMG with zero initial guess. The results in Table. 4.2 show that the NI-FOSLS-AMG-pACE method requires only 4.6% of the work of solving the problem directly on the same finest grid.

To verify that the geometric binning strategy results in near equal-distribution of error on finer levels, the binning results are presented in Fig 4.3. It is clear that almost all elements on the finest level land in the first two bins.

ℓ	r_1	$E(r_1)$	N_ℓ	$\mathcal{G}_\ell$	$ncyc$	σ	ρ
1	1.00	1.00	4,096	2.43e+02	4	1.42	0.21
2	0.24	1.00	7,036	1.71e+01	4	1.46	0.21
3	0.50	1.00	17,944	1.33e+00	4	1.51	0.23
4	0.53	0.96	47,245	1.72e-01	4	1.52	0.23
5	0.90	1.00	176,224	1.18e-02	4	1.48	0.22
6	0.13	0.72	248,068	3.85e-03	4	1.49	0.23
7	0.93	1.00	944,353	2.50e-04	4	1.46	0.20
8	0.98	1.00	3,735,118	1.61e-05	4	1.44	0.19
9	0.99	1.00	14,814,997	1.03e-06	4	1.44	0.19

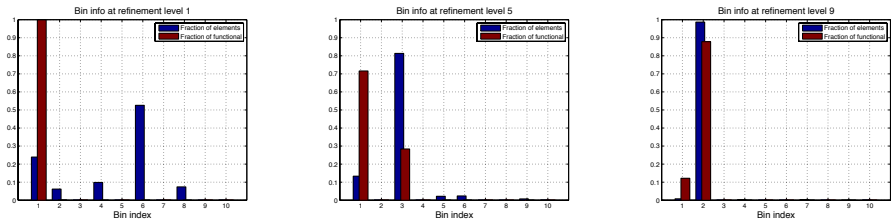
TABLE 4.1

NI-FOSLS-AMG-pACE for steep gradients: relative setup cost $C_s \approx 19.80$, setup 77.79 WU, solve 15.59 WU, overall runtime 74.59 sec. Here, σ is the AMG grid complexity.

Method	Setup Cost	$ncyc$	Solve Cost	Total Work
NI-FOSLS-AMG-pACE	77.79	4	15.59	92.79
FOSLS-AMG	57.02	11	31.68	88.70

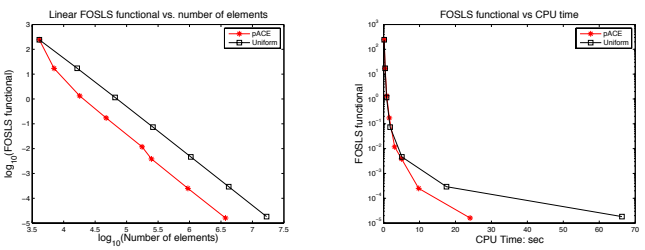
TABLE 4.2

Comparison of NI-FOSLS-AMG-pACE and applying FOSLS-AMG directly to the finest grid.



(a) Binning results at level 1 (b) Binning results at level 5 (c) Binning results at level 9

FIG. 4.3. *Geometric binning results from refinement level 1 to refinement level 5.*



(a) Functional vs. Number of elements (b) Functional vs CPU time

FIG. 4.4. *Comparison between AMR and uniform refinement running on 1,024 processors.*

We conclude our discussion of the numerical performance by comparing pACE and uniform refinement; see Fig 4.4. The pACE approach is able to reach the same error with only 20% of the elements that were required by uniform refinement. The overall run time of pACE is only 30% of uniform refinement.

4.2. Parallel Performance Study.

4.2.1. Strong Scaling. Speedups are given in Fig 4.5 for four different test problems. The smallest problem has 0.27 million elements, two intermediate problems

have 1.06 million and 3.74 million elements, and the largest problem has 10 million elements. Results show that strong scaling speedups are close to optimal.

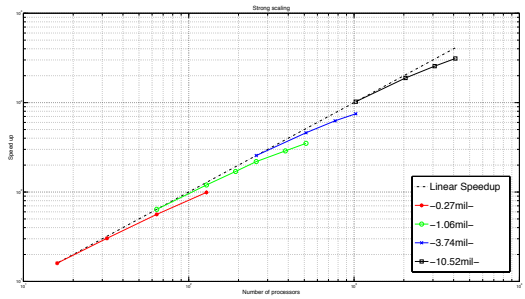
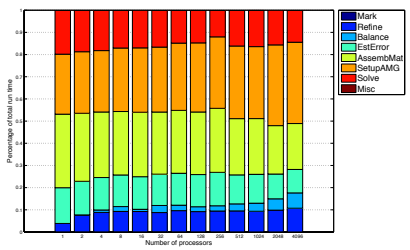
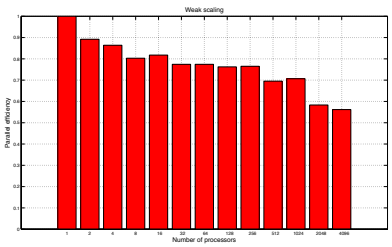


FIG. 4.5. *Strong scalability.*

4.2.2. Weak Scaling. Figure 4.6(a) and 4.6(b) provide evidence of the weak scalability. Figure 4.6(a) plots the breakdown of the overall runtime into two major categories: numerical PDE solves and AMR routines with problem size being roughly a constant per processor. The cost of all AMR routines are controlled within 20% of the overall runtime from 2 to 4,096 processors, which is comparable to 4 AMG-CG iterations per refinement level. Fig. 4.6(b) is the parallel efficiency measure by the total work units per processor per total run time for a given number of processors, normalized by the total work units per total run time for single processor. The weak scaling parallel efficiency remains above 50% from $np = 1, ..., 4,096$.



(a) Breakdown of total run time into different components related to numerical PDE routines (green, yellow, orange, and red) and AMR routines (light and dark blue)



(b) Parallel Efficiency

FIG. 4.6. *Weak scalability.*

5. Conclusions. Efficiency-based refinement algorithms for the FOSLS finite element method with algebraic multigrid solver in the context of nested iteration (NI-FOSLS-AMG) are developed. The algorithm choose elements to refine based on minimizing the "accuracy-per-computational-cost" efficiency factor (ACE). The modifications of the ACE algorithm designed for parallel computers employ geometric binning strategies that group elements into bins based on the local errors. Tests demonstrate that the parallel ACE algorithm based on geometric binning keeps the nice numerical properties of their serial counterparts and is capable of greatly reducing

communication. Load balancing starts at very coarser grids. Elements and nodes are redistributed using a space filling curve and parallel tree structures at each refinement level. The SFC preserves locality of each grid partition and, thus, reduces communication cost. Numerical tests show that the NI-FOSLS-AMG-pACE approach tends to equally distribute local errors on finer levels, where near uniform refinements are used. Excellent strong and weak scalability are demonstrated up to $4,096$ processors for problems with 25 million biquadratic elements.

REFERENCES

[1] J. H. Adler, T. A. Manteuffel, S. F. McCormick, J. W. Nolting, J. W. Ruge, and L. Tang. Efficiency-based adaptive local refinement for first-order system least-squares formulations. *SIAM J. Sci. Comp. (SISC)*, 33(1):1–24, 2011.

[2] J. H. Adler, T. A. Manteuffel, S. F. McCormick, J. W. Ruge, and G. D. Sanders. Nested iteration and first-order system least squares for incompressible, resistive magnetohydrodynamics. *SIAM J. Sci. Comp. (SISC)*, 32(3):1506–1526, 2010.

[3] R. E. Bank and M. J. Holst. A new paradigm for parallel adaptive meshing algorithms. *SIAM Review*, 45:291–323, 2003.

[4] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, Philadephia, PA, 2nd edition, 2000.

[5] C. Burstedde, O. Ghattas, M. Gurnis, G. Stadler, E. Tan, T. Tu, L. C. Wilcox, and S. Zhong. Scalable adaptive mantle convection simulation on petascale supercomputers. In *Proceedings of ACM/IEEE SC '08*, 2008.

[6] Z. Cai, R. Lazarov, T. A. Manteuffel, and S. F. McCormick. First-order system least squares for second-order partial differential equations. i. *SIAM J. Numer. Anal.*, 31:1785–1799, 1994.

[7] Z. Cai, T. A. Manteulffel, and S. F. McCormick. First-order system least squares for second-order partial differential equations. ii. *SIAM J. Numer. Anal*, 34:425–454, 1997.

[8] H. De Sterck, T. A. Manteuffel, S. F. McCormick, J. W. Nolting, J. W. Ruge, and L. Tang. Efficiency-based h- and hp-refinement strategies for finite element methods. *Num. Lin. Alg. Appl.*, 15:89–114, 2008.

[9] W. Dörfler. A convergent adaptive algorithm for poisson's equation. *SIAM J. Numer. Anal.*, 33:1106–1124, 1996.

[10] R. Falgout and U. Yang. hypre: a library of high performance preconditioners. In *Computational Science - ICCS 2002 Part III*, volume 2331, pages 632–641. Springer-Verlag, 2002.

[11] W. Gui and I. Babuška. The h, p, and hp versions of the finite element method in 1 dimension, parts i, ii, iii. *Numerische Mathematik*, 49:577–683, 1986.

[12] P. Morin, R. H. Nochetto, and K. G. Siebert. Data oscillation and convergence of adaptive fem. *SIAM J. Numer. Anal.*, 38:466–488, 2000.

[13] J. W. Nolting. *Efficiency-based Local Adaptive Refinement for FOSLS Finite Elements*. PhD thesis, University of Colorado, Applied Mathematics Department, 2008.

[14] U. Rüede. *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, volume 13 of *Frontiers in Applied Mathematics*. SIAM, Philadephia, PA, 1993.

[15] H. Sundar, R. S. Sampath, and G. Biros. Bottom-up construction and 2:1 balance refinement of linear octrees in parallel. *SIAM J. Sci. Comp. (SISC)*, 30:2675–2708, 2008.

[16] L. Tang. *Parallel Efficiency-based Adaptive Local Refinement*. PhD thesis, University of Colorado, Applied Mathematics Department, 2010.

[17] T. Tu, D. O'Hallaron, and O. Ghattas. Scalable parallel octree meshing for terascale applications. In *Proceedings of ACM/IEEE SC '05*, 2005.

[18] R. Verfürth. *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*. Wiley-Teubner, Chichester, 1995.

[19] X. Zhang. Multilevel schwarz methods. *Numer. Math*, 63:521–539, 1992.