DUAL FUNCTIONS FOR A PARALLEL ADAPTIVE METHOD

RANDOLPH E. BANK[*] AND JEFFREY S. OVALL[†]

Abstract. In this paper, we investigate the effects of pollution error on the performance of the parallel adaptive finite element technique proposed by Bank and Holst in 2000. In particular, we consider whether the performance of the algorithm as it was originally proposed can be improved through the use of certain dual functions which give indication of global influences on subdomain errors.

Key words. Dual functions, Bank–Holst algorithm, parallel adaptive grid generation

AMS subject classifications. 65N50, 65N30, 65Y05

1. Introduction.

One of the most difficult aspects of adaptive computations in a parallel environment is the issue of load balancing. If an initial mesh is distributed fairly among a number of processors, a good error estimator coupled with adaptive refinement may quickly produce a very bad load imbalance among the processors. A number of static and dynamic load balancing approaches for unstructured meshes have been proposed in the literature [23, 17, 19, 20, 13, 27, 29]. Such approaches tend to be communication intensive, since they must first assess the current state of imbalance among the processors, and then perform a redistribution step to equalize the load.

The Bank/Holst adaptive meshing paradigm [9, 10, 8] presented a new way for addressing the issue of load balancing. Their scheme has the additional benefits of keeping communication costs low, and using existing sequential adaptive software such as PLTMG without the need for extensive investment in recoding for use in a parallel environment. The paradigm consists of three major steps.

Step 1: Load Balancing. We solve a small problem on a coarse mesh, and use a posteriori error estimates to partition the mesh into p subdomains, one for each processor. Each subregion has approximately the same error, although subregions may vary considerably in terms of numbers of elements or grid points.

Step 2: Adaptive Meshing. Each processor is provided the complete coarse mesh and instructed to sequentially solve the *entire* problem, with the stipulation that its adaptive refinement should be limited largely to its assigned subdomain. The target number of elements and grid points for each problem is the same. At the end of this step, the global mesh should be regularized so that the global mesh described in Step 3 will be conforming.

Step 3: Global Solution. A final mesh is computed using the union of the refined subdomains provided by each processor. A final solution is computed using a standard domain decomposition solver or parallel multigrid technique. For the experiments done in this paper, the final global iterative solve is performed by use of the domain decomposition solver described in [6].

[*]Department of Mathematics, University of California, San Diego, La Jolla, California 92093-0112. Email: rbank@ucsd.edu. The work of this author was supported by the National Science Foundation under contract DMS-0511766.

[†]Scientific Computation Group, Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany. Email: ovall@mis.mpg.de. The work of this author was supported by a research fellowship of the Max Planck Society.

Here the load balancing problem is reduced to the numerical solution of a small elliptic problem on a single processor, using a sequential adaptive solver, without requiring any modifications to the sequential solver. The small elliptic problem is used to produce a posteriori error estimates to predict future element densities in the mesh, which are then used partition the mesh according to equal error. The underlying assumption here is that roughly equal element densities implies roughly equal workload. We note that, although this assumption is seemingly fragile, the procedure does generally produce good workload balance in practice. It is important to emphasize that the adaptive mesh generation in Step 2 is done asynchronously and without communication. The only communication needed prior to the final global solve in Step 3 is the initial fan out of the coarse mesh at the end of Step 1, and the mesh regularization at the the end of Step 2. (The fan out could be avoided by having every processor solve the same course problem and compute the same load balance in Step 1.) By providing each processor with the entire problem, information about the global behavior of the the solution, which normally would require some communication between processors, is generated independently and without communication on each processor, albeit on a coarse mesh. In effect, extra coarse grid computations are substituted for the interprocessor communication that would otherwise be required. This is likely to be an advantageous trade-off in situations where communication costs are high relative to computation costs, a situation typical of small Beowulf clusters.

In early investigations of the algorithm, the issue of restricting refinement on a given processor largely to its own subdomain was addressed by multiplying a posteriori error estimates for elements outside the subdomain by 10^{-6}. This small change allowed sequential adaptive codes (like the PLTMG package used here) to work in parallel without additional changes. This scheme tricks the adaptive refinement routine into thinking errors outside the given subdomain are much smaller than they actually are, and hence not good candidates for refinement. Some refinement outside the given processors subdomain is still required for other reasons, e.g., the shape-regular grading of the mesh from the small elements inside the refined region to the larger elements in the coarse regions.

While initially derived as a simple way to implement and test the concept, the 10^{-6} trick proved to be surprisingly effective for a wide variety of elliptic PDEs. However, because of the important role the course grid plays in the paradigm, it is important to consider heuristics based on more solid mathematical foundations than the 10^{-6} trick to help determine an appropriate refinement of the coarse grid outside the subregion assigned to the processor. It is this point that is the main focus of the present work.

For example, the coarse grid provides information about the effect of singularities outside the given subdomain on the solution inside (so-called pollution effects). See [1, 16, 28] for some discussion of such pollution errors. If convection is present, good information about the solution along the upstream boundaries of the subregion is needed to form an accurate impression of the convective behavior within the subregion. Because of the continuous dependence of the solution of elliptic problems on data throughout the domain, the approximation quality in any given subdomain depends on the approximation quality in the rest of the domain.

Dual functions and goal related adaptivity have a long history in the context of adaptive meshing algorithms [4, 3, 2, 5, 21, 18, 14, 15, 22]. Usually the functional is some quantity of physical interest for a particular application (lift, drag, stress intensity factor, etc). Our idea here is to adapt this goal oriented adaptive meshing technology to the present situation. In Step 2 of the paradigm, we view each processor as having the "goal" of producing a good mesh in its subregion. Thus the goal-related functional in this case is not provided externally by the user, but rather is developed internally and independently by the software running on each processor. The dual functions developed on each processor may be quite different from one another, since the subregions assigned to each processor for refinement may require quite different refinement patterns for their respective coarse meshes in order to produce good fine meshes for their assigned subregions. In particular, for this heuristic, we develop functionals intended to express the influence of the solution outside a given processor's subdomain on the solution within. The dual solution, generated by solving a problem with the adjoint operator and given functional as data, should then reflect the influence of the coarse parts of the domain on the processor's own region. The adjoint problem is solved using the same mesh as the primal problem, i.e., fine in its own region and coarse elsewhere. Then values (or local norms) of the dual function are used in place of 10^{-6} to weight the a posteriori error estimates in the coarse subregions, which in turn are used in the adaptive refinement procedure on that processor.

We emphasize that the primary objective of Step 2 in the algorithm is for each processor to generate an well-adapted mesh in its subdomain as part of the global fine mesh. However, a secondary objective is for each processor to contribute to a reasonable initial guess in its subdomain of the solution of the global fine problem in Step 3 - this speeds up the convergence of the parallel solver. Clearly these two goals are related, but it is not always necessary to have a good approximate solution in order to generate a well-adapted mesh from the associated error estimates. Indeed, the ability to generate good meshes from relatively inaccurate solutions explains the success of many adaptive methods. However, if the solution quality in a given subdomain is relatively poor and is not improved by further refinement in that subdomain, then the quality of the initial guess for Step 3 will be diminished. This is another motivation for investigating the use of dual weighted error estimates, the topic of Section 3, which might better indicate global influences on subdomain error.

The rest of this manuscript is organized as follows. In Section 2, we establish notation and assumptions for this study. In Section 3, we develop the functionals and corresponding dual solutions suitable for our parallel adaptive meshing algorithm. Finally, in Section 4, we present some numerical examples.

2. Notation and Assumptions. Let $\Omega \subset \mathbb{R}^2$ be a bounded domain with Lipschitz boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, and define

$$\mathcal{H} \equiv \left\{ v \in H^1(\Omega) : v|_{\partial\Omega_D} = 0 \text{ in the trace sense} \right\}. \tag{2.1}$$

For simplicity, we will assume that $\partial\Omega$ is a polygon. The usual spaces $W_p^k(\Omega)$ and $H^k(\Omega) \equiv W_2^k(\Omega)$ are equipped with norms $\|\cdot\|_{k,p,\Omega}$ and $\|\cdot\|_{k,\Omega} \equiv \|\cdot\|_{k,2,\Omega}$ respectively. Let data functions $a : \bar{\Omega} \to \mathbb{R}^{2\times 2}$, $\mathbf{b} : \bar{\Omega} \to \mathbb{R}^2$, $c, f : \bar{\Omega} \to \mathbb{R}$ and $g : \partial\Omega_N \to \mathbb{R}$ be given. The primal problem is to find $u \in \mathcal{H}$ such that

$$B(u,v) = F(v) \text{ for all } v \in \mathcal{H}, \tag{2.2}$$

$$B(u,v) \equiv \int_\Omega a\nabla u \cdot \nabla v + (\mathbf{b}\cdot\nabla u + cu)v\,dx \tag{2.3}$$

$$F(v) \equiv \int_\Omega fv\,dx + \int_{\partial\Omega_N} gv\,dS. \tag{2.4}$$

We assume that the data functions are smooth, and that the matrix a is positive-definite, with smallest eigenvalue bounded below on Ω by some constant $\gamma > 0$. We make the following standard boundedness and coercivity assumptions for the bilinear form B and linear functional F: There exist constants $\alpha, \nu, \mu > 0$, such that, for all $v, w \in \mathcal{H}$,

$$\begin{aligned} |F(v)| &\leq \alpha \|v\|_{1,\Omega}, \\ |B(v,w)| &\leq \nu \|v\|_{1,\Omega} \|w\|_{1,\Omega}, \\ B(v,v) &\geq \mu \|v\|_{1,\Omega}^2. \end{aligned}$$

Let $\mathcal{T}_h$ denote a shape-regular triangulation of Ω with mesh size $h \in (0,1)$. Let $V_h \subset \mathcal{H}$ denote the space of continuous, piecewise-linear polynomials defined on $\mathcal{T}_h$, and $\bar{V}_h \subset \mathcal{H}$ denote the continuous, piecewise-quadratic polynomials. Let $u_\ell \in V_h$ and $u_q = \bar{V}_h$ denote piecewise linear and quadratic interpolants of u on $\mathcal{T}_h$. We make the following standard assumptions about their asymptotic approximation quality:

$$\|u - u_\ell\|_{k,\Omega} \lesssim h^{2-k} \|u\|_{2,\Omega}, \tag{2.5}$$

$$\|u - u_q\|_{k,\Omega} \lesssim h^{3-k} \|u\|_{3,\Omega}, \tag{2.6}$$

for $0 \leq k \leq 1$.

In this work we assume the existence of a gradient recovery operator $\mathcal{R}\nabla u_h \in V_h \times V_h$ possessing certain superconvergence properties. Let $u \in H^3(\Omega) \cap W^{2,\infty}(\Omega)$, and $u_h \in V_h$ be any approximation of u satisfying

$$\|u - u_h\|_{k,\Omega} \lesssim h^{2-k} |u|_{2,\Omega}, \quad k = 0, 1, \tag{2.7}$$

$$\|u - u_h\|_{0,\infty,\Omega} \lesssim h^2 |\log h| |u|_{2,\infty,\Omega}. \tag{2.8}$$

Then we assume

$$\|\nabla u - \mathcal{R}\nabla u_h\|_{0,\Omega} \lesssim h\delta \left(\|u\|_{3,\Omega} + |u|_{2,\infty,\Omega} \right), \tag{2.9}$$

where $\delta \to 0$ as $h \to 0$.

Some recovery procedures that satisfy bounds like (2.9) are described in [11, 12]. It is important to note that u_h needs only to satisfy the required approximation properties in order for (2.9) or Lemma (2.1) to apply; in particular, there is no requirement that u_h be an approximate solution to (2.2)-(2.4). That stated, for the rest of this work we will use $u_h \in V_h$ to denote the finite element solution of the variational problem

$$B(u_h, v) = F(v) \text{ for all } v \in V_h. \tag{2.10}$$

For Tables and Figures, we will use the abbreviation $e_h = u - u_h$ to denote the error in the finite element approximation.

Although the details of $\mathcal{R}$ are not needed in the analysis here, for completeness we briefly summarize the scheme analyzed in [12]. In this case

$$\mathcal{R}\nabla u_h = S^m Q_h \nabla u_h,$$

where Q_h is the operator corresponding to L^2 projection onto V_h, and $S : V_h \to V_h$ is a smoothing operator. In words, the discontinuous, piecewise constant gradient ∇u_h is projected into the space of continuous piecewise linear polynomials, and then smoothed, using a multigrid-like smoothing operator. Typically $m = 1$ or $m = 2$ smoothing steps are used.

Given a superconvergent gradient approximation, one can form $\mathcal{R}\nabla u_h - \nabla u_h$ to approximate the error $\nabla u - \nabla u_h$. In many cases we can expect such an estimate to be asymptotically exact.

LEMMA 2.1. *Suppose that $u \in H^3(\Omega) \cap W^{2,\infty}(\Omega)$ and $u_h \in V_h$ satisfies (2.7)-(2.8), and*

$$\|\nabla u - \nabla u_h\|_{0,\Omega} \geq c_0 h$$

for some $c_0 > 0$. Then

$$\lim_{h\to 0} \frac{\|(\mathcal{R} - I)\nabla u_h\|_{0,\Omega}}{\|\nabla u - \nabla u_h\|_{0,\Omega}} = 1.$$

Proof. It holds that

$$\left| \frac{\|(\mathcal{R} - I)\nabla u_h\|_{0,\Omega}}{\|\nabla u - \nabla u_h\|_{0,\Omega}} - 1 \right| \leq \frac{\|\mathcal{R}\nabla u_h - \nabla u\|_{0,\Omega}}{\|\nabla u - \nabla u_h\|_{0,\Omega}} \lesssim \delta\ ,$$

and $\delta \to 0$ as $h \to 0$. □

The local error indicators $\|(\mathcal{R} - I)\nabla u_h\|_{0,\tau}$ form the basis of the standard adaptive refinement algorithm for PLTMG[7], the PDE software used for the experiments in this work.

3. Duality and Global Influence on Local Error.

In many applications, it is not the PDE solution u which is of primary interest, but rather some functional $G(u)$, where $G \in \mathcal{H}^*$. The examples often given in the literature are related to physical quantities of interest which can be computed from u, such as the lift or drag on a wing of an airplane during cruising conditions (see, for example, [25]). The use of appropriate dual (adjoint) problems for the accurate approximation of functional quantities and for driving adaptive algorithms (goal-oriented adaptive refinement) has been an active area of research for some time [21, 22, 25, 26, 24]. The sort of functionals that are of interest to us in the context of the Bank-Holst Paradigm are those where we can use the dual solution to extract information concerning outside influences on the solution u within a given subdomain. We include some of the basic theory below for completeness before presenting the dual error estimator used in this work.

Given a functional G, we define the related dual problem

$$B^*(\omega, v) = G(v) \text{ for all } v \in \mathcal{H}, \tag{3.1}$$

where $B^*(w, v) \equiv B(v, w)$. The key relation between the solutions of the primal and dual problems is that

$$G(u - u_h) = B^*(\omega, u - u_h) \tag{3.2}$$
$$= B(u - u_h, \omega) \tag{3.3}$$
$$= F(\omega) - B(u_h, \omega). \tag{3.4}$$

In this context, ω provides a weighting of the global error (3.2, 3.3) or of the global residual (3.4) that indicates the influence of global approximation errors on the subdomain error $G(u - u_h)$.

Because we cannot generally compute ω, we must approximate $G(u - u_h)$ using one of the expressions (3.2, 3.3, 3.4). For the computations in this paper, we use the Dual Error Estimate Weighting technique (DEW) introduced in [24]. In order to make this paper more self-contained, we briefly present the pertinent details below.

Let $\omega_h \in V_h$ be the piecewise-linear finite element solution of the dual problem,

$$B^*(\omega_h, v) = G(v) \text{ for all } v \in V_h. \tag{3.5}$$

Noting that

$$B(u-u_h,\omega) = B(u-u_h,\omega-\omega_h) = \int_\Omega a\nabla(u-u_h)\cdot\nabla(\omega-\omega_h)\,dx + \mathcal{O}(h^3),$$

we define the approximator $\mathcal{G}_h \approx G(u-u_h)$ by

$$\mathcal{G}_h = \int_\Omega a(\mathcal{R}-I)\nabla u_h \cdot (\mathcal{R}-I)\nabla\omega_h\,dx. \tag{3.6}$$

We bound the approximation error, $|G(u-u_h)-\mathcal{G}_h|$, in the following theorem:

Theorem 3.1. *If $u,\omega \in H^3(\Omega)\cap W^{2,\infty}(\Omega)$, and the finite element solutions $u_h,\omega_h \in V_h$ satisfy (2.7)-(2.8), then*

$$|G(u-u_h)-\mathcal{G}_h| \lesssim h^2\delta\left(\|u\|_{3,\Omega}+|u|_{2,\infty,\Omega}\right)\left(\|\omega\|_{3,\Omega}+|\omega|_{2,\infty,\Omega}\right), \tag{3.7}$$

where $\delta \to 0$ as $h \to 0$

Proof. We have, by the triangle and Cauchy-Schwarz inequalities,

$$\begin{aligned}|G(u-u_h)-\mathcal{G}_h| \lesssim{}& \|\nabla u - \mathcal{R}\nabla u_h\|_{0,\Omega}\|\nabla(\omega-\omega_h)\|_{0,\Omega}\\ &+ \|(\mathcal{R}-I)\nabla u_h\|_{0,\Omega}\|\nabla\omega-\mathcal{R}\nabla\omega_h\|_{0,\Omega}+\mathcal{O}(h^3).\end{aligned}$$

We have the bounds

$$\begin{aligned}\|\nabla u-\mathcal{R}\nabla u_h\|_{0,\Omega} &\lesssim h\delta\left(\|u\|_{3,\Omega}+|u|_{2,\infty,\Omega}\right),\\ \|(\mathcal{R}-I)\nabla u_h\|_{0,\Omega} &\lesssim h\left(\|u\|_{3,\Omega}+|u|_{2,\infty,\Omega}\right),\\ \|\nabla\omega-\mathcal{R}\nabla\omega_h\|_{0,\Omega} &\lesssim h\delta\left(\|\omega\|_{3,\Omega}+|\omega|_{2,\infty,\Omega}\right),\\ \|\nabla(\omega-\omega_h)\|_{0,\Omega} &\lesssim h\|\omega\|_{2,\Omega}.\end{aligned}$$

Combining these four estimates completes the proof. □

We have the following immediate Corollary:

Corollary 3.2. *Under the assumptions of Theorem 3.1, if there exists some constant $c_0 > 0$ for which $|G(u-u_h)| \geq c_0h^2$, then $\mathcal{G}_h$ is an asymptotically exact estimator of $G(u-u_h)$,*

$$\frac{\mathcal{G}_h}{G(u-u_h)} \to 1.$$

Proof. It holds that

$$\begin{aligned}\left|1-\frac{\mathcal{G}_h}{G(u-u_h)}\right| &= \left|\frac{G(u-u_h)-\mathcal{G}_h}{G(u-u_h)}\right|\\ &\lesssim \delta\left(\|u\|_{3,\Omega}+|u|_{2,\infty,\Omega}\right)\left(\|\omega\|_{3,\Omega}+|\omega|_{2,\infty,\Omega}\right),\end{aligned}$$

which implies the conclusion. □

We motivate our local error indicators for duality-based adaptive refinement, where the objective is to reduce $|G(u-u_h)|$, by observing that

$$\begin{aligned}|\mathcal{G}_h| &= \left|\sum_{\tau\in\mathcal{T}}\int_\tau a(\mathcal{R}-I)\nabla u_h\cdot(\mathcal{R}-I)\nabla\omega_h\,dx\right|\\ &\le \sum_{\tau\in\mathcal{T}}\left|\int_\tau a(\mathcal{R}-I)\nabla u_h\cdot(\mathcal{R}-I)\nabla\omega_h\,dx\right|\\ &\le \sum_{\tau\in\mathcal{T}}\|a(\mathcal{R}-I)\nabla\omega_h\|_{0,\tau}\|(\mathcal{R}-I)\nabla u_h\|_{0,\tau}.\end{aligned}$$

So we use the local indicators

$$\|a(\mathcal{R}-I)\nabla\omega_h\|_{0,\tau}\|(\mathcal{R}-I)\nabla u_h\|_{0,\tau} \tag{3.8}$$

for adaptive refinement in this context. The local error estimates $\|(\mathcal{R}-I)\nabla u_h\|_{0,\tau}$ for the primal problem are the basis for the standard adaptive refinement procedure in PLTMG, so the quantities $\|a(\mathcal{R}-I)\nabla\omega_h\|_{0,\tau}$ can be thought of as dual weights which indicate global influences on the functional error.

We describe below the choice of functional which we will later use in our parallel experiments. Suppose that we are given a subdomain $\Omega_s\subset\Omega$. For purposes of practical implementation, we assume that Ω_s has a polygonal boundary and that all triangulations of Ω line up with $\partial\Omega_s$ perfectly. We have in mind subdomains Ω_s of the sort generated during the Load Balancing phase of the Bank/Holst Algorithm. We define our dual functional by

$$G(v)=\int_{\Omega_s} fv - a\nabla u_h\cdot\nabla v-(\mathbf{b}\cdot\nabla u_h+cu_h)v\,dx+\int_{\partial\Omega_s} a\nabla u_h\cdot\mathbf{n}v\,ds. \tag{3.9}$$

If the function u is a classical solution of the PDE in question, then we have

$$\begin{aligned}G(u-u_h)=\int_{\Omega_s} & a\nabla(u-u_h)\cdot\nabla(u-u_h)+(\mathbf{b}\cdot\nabla(u-u_h)+\\ & c(u-u_h))(u-u_h)\,dx-\int_{\partial\Omega_s}a\nabla(u-u_h)\cdot\mathbf{n}(u-u_h)\,ds\\ &=\int_{\Omega_s}a\nabla(u-u_h)\cdot\nabla(u-u_h)\,dx+\mathcal{O}(h^3|\log h|).\end{aligned}$$

We point out that the definition of G depends on the approximate solution u_h.

It is difficult to give *a priori* predictions of the smoothness of the dual solution ω. Therefore, we do not know that the assumptions of Theorem 3.1 generally apply. In other words, it is difficult to assess the effectivity of the approximation

$$\mathcal{G}_h\approx\int_{\Omega_s}a\nabla(u-u_h)\cdot\nabla(u-u_h)\,dx \tag{3.10}$$

apart from experimentation. We provide empirical evidence of the effectivity of the dual approximator in the following example. The example, henceforth called the **Simple Problem**, has primal problem:

$$\begin{aligned}-\Delta u&=\sin(2\pi y)(2+4\pi^2x(1-x)) && \text{in } \Omega,\\ u&=0 && \text{on } \partial\Omega,\end{aligned}$$

where Ω is the unit square $(0,1)\times(0,1)$. The primal solution is $u = x(1-x)\sin(2\pi y)$. For the dual problem, we choose the subdomain Ω_s with $x > y$ and $x > 0$. The approximate dual solution is pictured in Figure 3.1 on a mesh with approximately 16,000 vertices.

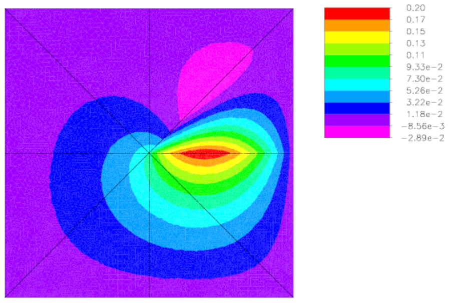

FIG. 3.1. *A contour plot of the approximate dual solution on a mesh with 16036 vertices.*

In Table 3.1 we see the results of several stages dual-weighted functional error estimation for the Simple Problem. We use N for the number of unknowns, e_h for the finite element error $u - u_h$, $\mathcal{G}_h$ as our dual approximator and EFF as the effectivity in approximation. The final row of data corresponds to the dual function pictured in Figure 3.1. We see that the initial effectivity is exceptionally good, decreases after the first refinement, and then begins to improve again upon successive refinements - all the while remaining in a useful range.

N	$\lvert e_h\rvert^2_{1,\Omega_s}$	$\mathcal{G}_h$	EFF
244	1.97e-3	1.93e-3	0.98
996	3.36e-4	2.18e-4	0.65
4004	7.35e-5	5.78e-5	0.79
16036	1.41e-5	1.18e-5	0.84

TABLE 3.1
The subdomain error $|e_h|^2_{1,\Omega_s}$, the dual approximation $\mathcal{G}_h$ and its effectivity.

4. Parallel Experiments.

During Step 2 of the Bank/Holst algorithm as presented in [9, 10], the adaptive refinement done on processor k is driven by the weighted error estimates

$$\eta_\tau = w_\tau \|(\mathcal{R} - I)\nabla u_h\|_{0,\tau} \text{ with } w_\tau = \begin{cases} 1 & \text{if } \tau \in \mathcal{T}_k \\ 10^{-6} & \text{otherwise} \end{cases},$$

where $\mathcal{T}_k$ is the portion of the triangulation coinciding with the subdomain Ω_k assigned to processor k. In this section we compare that weighting scheme with dual weighting:

$$\eta_\tau = w_\tau \|(\mathcal{R} - I)\nabla u_h\|_{0,\tau} \text{ with } w_\tau = \begin{cases} 1 & \text{if } \tau \in \mathcal{T}_k \\ \|a(\mathcal{R} - I)\nabla \omega_h^k\|_{0,\tau} & \text{otherwise} \end{cases},$$

where ω_h^k is the solution of the dual problem presented in the previous section associated with the subdomain Ω_k assigned to processor k.

This choice of weighting scheme deserves further explanation. In particular, why not use the dual weights in $\mathcal{T}_k$ as well? We recall that the goal of Step 2 of the algorithm is to produce a well-adapted, globally fine mesh and initial guess for use in the final global iterative solve. In other words, Step 2 should provide a good set-up for Step 3. Evidence that the dual weighting scheme has provided a better set-up than the original approach would include reduced solve times and better load balance during Step 3. Implicit in the original, 10^{-6}, version of Step 2 is the assumption that only coarse information outside Ω_k is needed for processor k to make a good contribution to the final global problem - so any defects due to essentially ignoring finer-scale pollution errors will be adequately addressed in Step 3, with negligible negative effects on its performance. In contrast, refinement on processor k which uses the dual weights both inside and outside $\mathcal{T}_k$ is acting as though there will be no Step 3 at all - as if the ultimate quality of the solution within its assigned subdomain is completely dependent upon what can be accomplished in Step 2. What one could reasonably expect at the end of Step 2 in this case is a mesh which is not significantly finer in its "fine" region than in its "coarse" region, with the effect that many degrees of freedom are discarded when the final global problem which is constructed from the fine portions on each processor. The weighting scheme used for our experiments is one of many potential compromises between these two extremes, which aims to use duality to better address finer-scale pollution effects while still recognizing that the independent adaptive refinement phase is not the final phase of the algorithm.

We ran a series of experiments on two problems using PLTMG on a cluster of Dual AMD Opteron 250s with 2.4 GHz CPU clockrate, 4GB RAM and a 2x Gigabit Ethernet Interconnect. The first problem is the Simple Problem introduced in the previous section. The second problem, which we call the **D-C-R Problem** (diffusion-convection-reaction), is given by

$$-2(u_{xx} + u_{ux} + u_{yy}) + u_x + u_y + 3u = 2e^x(5\sin 2y - \cos 2y)$$

in $(0,1)\times(0,1)$, with Dirichlet conditions on the top and bottom of the square and Neumann conditions on the left and right chosen to match the solution $u = e^x \sin 2y$.

For Step 1 of the algorithm a coarse problem of size N_C is solved, and error estimates are used to partition the domain into p subdomains - one for each processor. During Step 2 of the algorithm, error estimates weighted by either the 10^{-6} or dual schemes mentioned above are used for adaptive refinement, until a problem having the target size N_T is reached on each processor. We expect that most of the refinement done in Step 2 will be in the interior of the assigned subdomains on each processor; so the final global problem size N should be close to, though smaller than, $p(N_T - N_C) + N_C$. We use the domain decomposition algorithm of Bank and Lu [6] for the final global solve in Step 3.

We take $N_C = 1000$ for Step 1 and refine until we reach a target problem size of $N_T = 50\,000,\ 100\,000,\ 150\,000,\ 200\,000,\ 250\,000$. This is done on $p = 4,\ 8,\ 16$ processors. The data for these experiments is collected in Tables 4.1 and 4.2 , and are organized into blocks of six pieces of information in the following form:

$$\boxed{\begin{array}{ccc} N & |e_h|_{1,\Omega} & \|e_h\|_{0,\Omega} \\ T_{fin} & T_{tot} & T_{dif} \end{array}} \tag{4.1}$$

where N is the final global problem size, $e_h = u - u_h$ is the error in the finite element solution, and T_{fin}, T_{tot} and T_{dif} are the average processor time to perform the domain decomposition solve, the average processor time for the entire algorithm and the difference between the slowest and fastest processor times for the entire algorithm respectively. For greater ease in reading the numerical results, we also include the legend (4.1) at the bottom of Tables 4.1 and 4.2.

The initial guess for the global domain decomposition solve is constructed from the fine portion of the solution on each of the processors. It is, therefore, multivalued (hence discontinuous) along the interfaces between subdomains. One of the objectives of the domain decomposition algorithm, which is based on a mortar finite element formulation, is to move toward continuity at these interfaces upon convergence. The errors are computed as

$$|e_h|_{1,\Omega}^2 = \sum_{k=1}^{p} |e_h|_{1,\Omega_k}^2 \quad \text{and} \quad \|e_h\|_{0,\Omega}^2 = \sum_{k=1}^{p} \|e_h\|_{0,\Omega_k}^2 .$$

This is a slight abuse of notation for $|e_h|_{1,\Omega}$ because e_h may remain discontinuous at the interfaces upon convergence, so $|e_h|_{1,\Omega}$ is actually infinite. We include $\|e_h\|_{0,\Omega}$ in the data primarily because, if the measure $|e_h|_{1,\Omega}$ is nearly the same for both weighting schemes then a smaller measure $\|e_h\|_{0,\Omega}$ for one of the schemes probably indicates smaller jump-discontinuities at the interfaces.

4.1. The Simple Problem in Parallel. Looking at Table 4.1, we first point out that the global fine problem size N is approximately the same for both weighting schemes in each run of the program. The problem size is slightly smaller for the dual method than it is for the 10^{-6} method, which is expected because the dual weighting on a given processor is likely to cause more refinement outside of the region assigned to that processor. We also note that the values $|e_h|_{1,\Omega}$ are essentially identical in each case, but the values for $\|e_h\|_{0,\Omega}$ tend to be better for the dual method - sometimes dramatically better. As mentioned above, this is indication that the final computed solution is more nearly continuous for the dual method than for the 10^{-6} method.

We see the greatest difference between the two methods in terms of the execution times for the algorithm. The dual method generally tends to lead to better times T_{fin}, T_{tot} for the final solve and the algorithm in total, as well as in the measure of load balance/imbalance T_{dif}. The most dramatic case is $p = 4$, $N_T = 250\,000$, where T_{fin} for the dual method is less than half of its counterpart! Because the problem sizes are roughly equivalent for both methods, the better values for T_{fin} indicate that using the dual method during Step 2 of the algorithm provides a better set-up for Step 3. In fact, comparing the difference $T_{tot} - T_{fin}$ for both methods shows that the time to reach Step 3 is essentially the same, so the only real difference is how well each method sets up this final global solve in Step 2.

We make one final point before moving to the next problem. As the number of processors increased, the difference between the solve times for the two methods decreased. However, we also point out that the tendency is for the difference in execution times to increase in favor of the dual method. The only real violation of this tendency is the case $p = 16$, $N_T = 200\,000$, where the dual method is unusually fast.

4.2. The Diffusion-Convection-Reaction (D-C-R) Problem in Parallel. We first note in Table 4.2 that, in contrast to the behavior for the Simple Problem, here we see a noticeable difference in the final problem size generated by the two methods, with the dual method producing smaller problems for the global parallel solve. This is due to the fact that the dual weighting on each processor causes more refinement to be done outside of its assigned subdomain than the 10^{-6} weighting during Step 2, so the "fine" subdomains on each processor tend to be less fine for the

Simple 4	10^{-6}			Dual		
50000	194621	3.97e-3	2.89e-6	193436	3.98e-3	2.90e-6
	9.86	17.10	0.72	9.88	17.56	0.67
100000	393298	2.87e-3	1.52e-6	390749	2.87e-3	1.53e-6
	32.09	55.54	2.94	24.70	50.08	0.78
150000	592464	2.32e-3	1.01e-6	588500	2.33e-3	1.01e-6
	55.06	88.34	2.01	42.42	78.31	5.49
200000	791793	1.98e-3	1.06e-6	785629	1.99e-3	7.31e-7
	131.57	175.51	10.20	71.10	117.55	3.80
250000	990954	1.78e-3	9.71e-7	982117	1.79e-3	5.85e-7
	178.84	234.18	22.63	81.95	140.01	4.61

Simple 8	10^{-6}			Dual		
50000	385830	2.90e-3	1.58e-6	382321	2.92e-3	1.59e-6
	12.17	19.75	2.28	10.34	18.24	1.68
100000	782601	2.04e-3	2.57e-6	775796	2.05e-3	7.77e-7
	55.62	79.91	14.13	30.07	55.94	5.96
150000	1180258	1.70e-3	1.80e-6	1170255	1.71e-3	5.41e-7
	91.33	125.70	16.32	56.18	91.93	9.84
200000	1577903	1.44e-3	2.72e-6	1563607	1.44e-3	3.90e-7
	134.90	179.76	31.34	93.18	139.95	12.02
250000	1975825	1.23e-3	1.67e-6	1955312	1.23e-3	2.99e-7
	182.19	238.68	38.76	118.18	176.58	22.07

Simple 16	10^{-6}			Dual		
50000	768628	2.05e-3	7.77e-7	758557	2.06e-3	7.93e-7
	16.20	23.75	5.43	14.14	21.97	3.70
100000	1560383	1.47e-3	1.37e-6	1543297	1.47e-3	9.78e-7
	57.67	81.41	27.47	57.95	83.50	20.92
150000	2355278	1.19e-3	2.11e-6	2330699	1.19e-3	7.99e-7
	105.32	139.32	50.59	98.42	133.74	35.39
200000	3150792	1.02e-3	2.10e-6	3115681	1.02e-3	1.98e-7
	149.24	194.36	72.32	93.81	139.90	25.62
250000	3946044	9.14e-4	2.81e-6	3898144	9.16e-4	7.2e-7
	184.29	239.57	55.61	169.58	226.63	51.94

N	$\|e_h\|_{1,\Omega}$	$\\|e_h\\|_{0,\Omega}$
T_{fin}	T_{tot}	T_{dif}

TABLE 4.1
Data for the Simple Problem.

dual method. In Figure 4.1 we see a contour plot of the dual solution and a plot of the element sizes during a stage of Step 2, corresponding to $p = 16$, $N_T = 100\,000$. We see that the dual solution is more pronounced in the direction of the convection, and that the refinement is also heavier in this direction.

Because the final global problem sizes for the two methods are noticeably different,

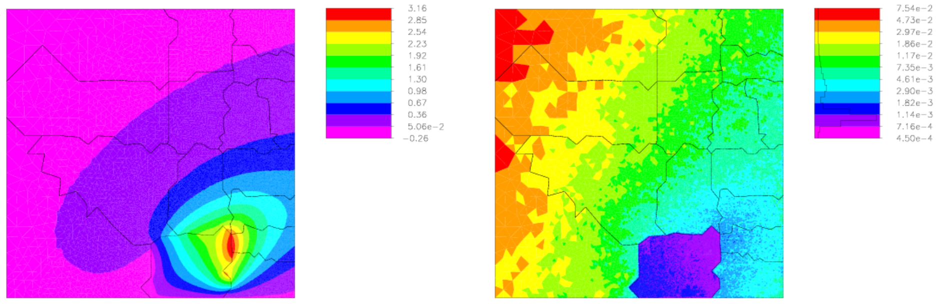

FIG. 4.1. *A contour plot of the approximate dual solution (left) and a plot of the element sizes.*

it is not surprising that the final errors are also different. In this case $|e_h|_{1,\Omega}$ tends to be slightly smaller for the 10^{-6} method, but $\|e_h\|_{0,\Omega}$ tends to be smaller for the dual method - often by an order of magnitude! To compare the gradient errors on more equal footing we consider the ratios $|e_h|_{1,\Omega}/\sqrt{N}$, because it is expected that $|e_h|_{1,\Omega} \sim h$ and $N \sim h^2$. The ratios $|e_h|_{1,\Omega}/\sqrt{N}$ are similar in each case for both methods, with the 10^{-6} method having slightly smaller values, but the dual method catching up as N_T and/or p is increased. We emphasize that the smaller values of the function error $\|e_h\|_{0,\Omega}$ for the dual method indicate a final solution which is more nearly continuous - as it should be.

Concerning the timing comparisons, the situation is similar to that for the Simple Problem. We see that the dual method is uniformly better in terms of time for the domain decomposition solve T_{fin} and total time for the algorithm T_{tot}, and tends to be better in terms of load balance T_{dif} as well. The differences between T_{fin} can be dramatic in some cases, with the value for $p = 4$, $N_T = 250\,000$ for the dual method being less than half of its counterpart!

REFERENCES

[1] I. Babuška, F. Ihlenburg, T. Strouboulis, and S. K. Gangaraj, *A posteriori error estimation for finite element solutions of Helmholtz' equation. II. Estimation of the pollution error*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 3883–3900.

[2] I. Babuška and A. Miller, *The post-processing approach in the finite element method. Part 1. Calculation of displacements, stresses and other higher derivatives of the displacements*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 1085–1109.

[3] ———, *The post-processing approach in the finite element method. Part 2. The calculation of stress intensity factors*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 1111–1129.

[4] ———, *The post-processing approach in the finite element method. Part 3. A posteriori error estimates and adaptive mesh selection*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 2311–2324.

[5] I. Babuška and T. Strouboulis, *The Finite Element Method and its Reliability*, Numerical Mathematics and Scientific Computation, Oxford University Press, Oxford, 2001.

[6] R. Bank and S. Lu, *A domain decomposition solver for a parallel adaptive meshing paradigm*, SIAM J. Sci. Comput., (submitted).

[7] R. E. Bank, *Pltmg: A software package for solving elliptic partial differential equations, users' guide 9.0*, tech. report, University of California, San Diego, 2004.

[8] ———, *Some variants of the Bank-Holst parallel adaptive meshing paradigm*, Computing and Visualization in Science, (accepted).

[9] R. E. Bank and M. Holst, *A new paradigm for parallel adaptive meshing algorithms*, SIAM J. Sci. Comput., 22 (2000), pp. 1411–1443 (electronic).

[10] R. E. Bank and M. J. Holst, *A new paradigm for parallel adaptive meshing algorithms*, SIAM

D-C-R 4	10^{-6}			Dual		
50000	194496	0.18	2.89e-4	161018	0.20	3.52e-4
	12.90	21.16	0.51	10.33	18.70	1.33
100000	392900	0.13	1.99e-4	343059	0.14	1.72e-4
	54.70	80.31	9.83	28.92	56.10	2.32
150000	592156	0.10	1.03e-4	524186	0.11	1.14e-4
	81.38	117.28	10.93	49.47	87.03	6.78
200000	791366	8.77e-2	2.13e-4	698025	9.36e-2	8.20e-5
	152.19	199.64	35.22	70.67	118.84	5.86
250000	990514	7.75e-2	3.85e-4	866089	8.44e-2	6.65e-5
	204.86	262.93	26.58	98.93	158.10	27.38

D-C-R 8	10^{-6}			Dual		
50000	386063	0.13	2.72e-4	312943	0.14	1.82e-4
	25.20	33.49	4.37	12.77	21.04	1.72
100000	782568	9.08e-2	3.56e-4	674527	0.10	9.13e-5
	58.73	84.22	13.83	31.27	58.32	4.56
150000	1180401	7.32e-2	4.95e-4	1029826	7.84e-2	5.71e-5
	102.49	137.94	28.36	58.60	96.20	10.19
200000	1578249	6.28e-2	5.35e-4	1367799	6.75e-2	5.13e-5
	149.35	195.34	40.41	135.70	183.13	7.87
250000	1976366	5.63e-2	5.74e-4	1703877	6.13e-2	5.99e-5
	210.02	267.00	64.16	178.57	236.46	36.54

D-C-R 16	10^{-6}			Dual		
50000	768158	9.10e-2	8.83e-5	565141	0.11	1.05e-4
	18.13	26.26	5.46	11.23	19.54	4.60
100000	1559940	6.43e-2	3.85e-4	1257098	7.17e-2	4.68e-5
	64.26	89.62	19.13	40.28	67.15	8.81
150000	2354749	5.27e-2	4.29e-4	1924354	5.83e-2	4.58e-5
	110.96	146.35	36.09	97.34	134.64	28.79
200000	3150111	4.49e-2	6.05e-4	2563542	5.04e-2	3.99e-5
	165.74	211.21	65.21	143.11	191.21	29.61
250000	3945421	3.97e-2	7.65e-4	3195221	4.51e-2	3.27e-5
	224.25	279.77	97.58	193.13	252.28	47.88

N	$\lvert e_h\rvert_{1,\Omega}$	$\lVert e_h\rVert_{0,\Omega}$
T_{fin}	T_{tot}	T_{dif}

Table 4.2
Data for the Diffusion-Convection-Reaction (D-C-R) Problem.

Review, 45 (2003), pp. 291–323.

[11] R. E. Bank and J. Xu, *Asymptotically exact a posteriori error estimators. I. Grids with superconvergence*, SIAM J. Numer. Anal., 41 (2003), pp. 2294–2312 (electronic).

[12] ———, *Asymptotically exact a posteriori error estimators. II. General unstructured grids*, SIAM J. Numer. Anal., 41 (2003), pp. 2313–2332 (electronic).

[13] M. W. Beall and M. S. Shephard, *A general topology-based mesh data structure*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 1573–1596.

[14] R. Becker and R. Rannacher, *A feed-back approach to error control in finite element methods: basic analysis and examples*, East-West J. Numer. Math., 4 (1996), pp. 237–264.

[15] R. Becker and R. Rannacher, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numer., 10 (2001), pp. 1–102.

[16] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, vol. 15 of Texts in Applied Mathematics, Springer-Verlag, New York, 1994.

[17] H. L. deCougny, K. D. Devine, J. E. Flaherty, R. M. Loy, C. Ozturan, and M. S. Shephard, *Load balancing for the parallel adaptive solution of partial differential equations*, Appl. Num. Math., 16 (1994), pp. 157–182.

[18] D. J. Estep, M. J. Holst, and M. Larson, *Generalized green's functions and the effective domain of influence*, SIAM J. Sci. Comput., 26 (2005), pp. 1314–1339.

[19] J. E. Flaherty, R. M. Loy, C. Ozturan, M. S. Shephard, B. K. Szymanski, J. D. Teresco, and L. H. Ziantz, *Parallel structures and dynamic load balancing for adaptive finite element computation*, Appl. Num. Math., 26 (1998), pp. 241–263.

[20] G. Fox, R. Williams, and P. Messina, *Parallel Computing Works!*, Morgan-Kaufmann, San Francisco, 1994.

[21] M. Giles and E. Süli, *Adjoint methods for pdes: a posteriori error analysis and postprocessing by duality*, in Acta Numerica, Vol. 11, Cambridge University Press, 2002, pp. 145–236.

[22] V. Heuveline and R. Rannacher, *Duality-based adaptivity in the hp-finite element method*, J. Numer. Math., 11 (2003), pp. 95–113.

[23] S. Kohn, J. Weare, M. E. Ong, and S. B. Baden, *Software abstractions and computational issues in parallel structured adaptive mesh methods for electronic structure calculations*, in Workshop on Structured Adaptive Mesh Refinement Grid Methods, vol. 117 of Institute for Mathematics and Its Applications, University of Minnesota, New York, 2000, Springer Verlag, pp. 75–95.

[24] J. S. Ovall, *Asymptotically exact functional error estimators based on superconvergent gradient recovery*, Numer. Math., 102 (2006), pp. 543–558.

[25] N. A. Pierce and M. B. Giles, *Adjoint recovery of superconvergent functionals from PDE approximations*, SIAM Rev., 42 (2000), pp. 247–264 (electronic).

[26] S. Prudhomme and J. T. Oden, *On goal-oriented error estimation for elliptic problems: application to control of pointwise errors*, Comput. Methods Appl. Mech. Engrg., 1-4 (1999), pp. 313–331.

[27] P. M. Selwood, M. Berzins, and P. M. Dew, *3D parallel mesh adaptivity : Data structures and algorithms*, in Parallel Processing for Scientific Computing, Philadelphia, 1997, SIAM.

[28] L. B. Wahlbin, *Local behavior in finite element methods*, in Handbook of numerical analysis, Vol. II, Handb. Numer. Anal., II, North-Holland, Amsterdam, 1991, pp. 353–522.

[29] C. Walshaw and M. Berzins, *Dynamic load balancing for pde solvers on adaptive unstructured meshes*, Concurrency: Practice and Experience, 7 (1995), pp. 17–28.