Computing 26, 91—105 (1981)

Computing
© by Springer-Verlag 1981

An Adaptive, Multi-Level Method for Elliptic Boundary Value Problems*

R. E. Bank and A. H. Sherman**, Austin, Texas

Received September 9, 1979; revised May 23, 1980

Abstract — Zusammenfassung

An Adaptive, Multi-Level Method for Elliptic Boundary Value Problems. Subroutine PLTMG is a Fortran program for solving self-adjoint elliptic boundary value problems in general regions of R^2. It is based on a piecewise linear triangle finite element method, an adaptive grid refinement procedure, and a multi-level iterative method to solve the resulting sets of linear equations. In this work we describe the method and present some numerical results and comparisons.

AMS (MOS) Subject Classifications (1970): 65N20.

Key words: Adaptive mesh refinement, multigrid methods, finite element methods.

Eine adaptive mehrstufige Methode für elliptische Randwertprobleme. Das Unterprogramm PLTMG ist ein FORTRAN-Programm zur Lösung selbstadjungierter elliptischer Randwertprobleme für beliebige Bereiche des R^2. Es basiert auf einer stückweise-linearen Finite-Element-Methode, einer adaptiven Gitterverfeinerungsmethode und einer mehrstufigen iterativen Methode zur Lösung des resultierenden Systems linearer Gleichungen. In dieser Arbeit wird die Methode beschrieben, und einige numerische Ergebnisse und Vergleiche werden dargelegt.

1. Introduction

Consider the model elliptic boundary value problem

$$\begin{aligned} L u &= -\nabla \cdot (a \nabla u) + b u = f && \text{in } \Omega \subset R^2 \\ u &= g_1 && \text{on } \partial\Omega_1 \\ \frac{\partial u}{\partial n} &= g_2 && \text{on } \partial\Omega_2 = \partial\Omega - \partial\Omega_1 \end{aligned} \tag{1.1}$$

* A version of this paper was presented as paper SPE 7683 at the 1979 Society of Petroleum Engineers of AIME Symposium on Reservoir Simulation held in Denver, Colorado, February 1—2, 1979. Work supported in part by the National Aeronautics and Space Administration under grant NSG-1632.

** The work of this author was supported in part by the U.S. Air Force Office of Scientific Research under contract F49620-77-C-0037.

0010-485X/81/0026/0091/$ 03.00

where the coefficient $a(x, y)$ $\big(b(x, y)\big)$ is positive (nonnegative) in Ω. In this work we discuss the performance of a program which solves (1.1) using a Rayleigh-Ritz-Galerkin method based on piecewise-linear triangular finite elements, a multi-level iterative scheme for solving the resulting matrix equations, and an adaptive grid refinement procedure. A more detailed discussion of the program appears in [14].

For expositional convenience, we assume $g_1 = g_2 = 0$, and that Ω is a polygon, although our FORTRAN subroutine PLTMG is designed for the more general equation (1.1).

In the Rayleigh-Ritz-Galerkin procedure [10, 11], we seek an approximate solution to the weak form of (1.1): Find $u \in \mathscr{H}_E^1(\Omega)$ satisfying

$$a(u, v) = (f, v) \quad \text{for all} \quad v \in \mathscr{H}_E^1(\Omega) \tag{1.2}$$

where

$$a(u, v) = \int_\Omega a \cdot \nabla u \cdot \nabla v + b\, u\, v \, dx,$$

and $(\cdot, \cdot)$ denotes the usual $L^2(\Omega)$ inner product. We use $\mathscr{H}_E^1(\Omega) \subseteq \mathscr{H}^1(\Omega)$ to denote the subspace of the usual Sobolev space $\mathscr{H}^1(\Omega)$ whose elements satisfy essential boundary conditions [10, 11]. Associated with the bilinear form $a(\cdot, \cdot)$ is the energy norm $|||u|||^2 = a(u, u)$.

Let $\mathscr{T}$ denote a triangulation of Ω, and let $\mathscr{M} \subset \mathscr{H}_E^1(\Omega)$ denote the N-dimensional space of C^0 piecewise-linear polynomials associated with $\mathscr{T}$. The finite element approximation of u in (1.2) is the function $\tilde{u} \in \mathscr{M}$ which satisfies

$$a(\tilde{u}, v) = (f, v) \quad \text{for all} \quad v \in \mathscr{M}. \tag{1.3}$$

Once a basis $\{\varphi_i\}_{i=1}^N$ for $\mathscr{M}$ has been selected (the nodal basis [10, 11] is usually chosen), (1.3) can be reformulated as a system of linear equations

$$A\,U = F \tag{1.4}$$

where $A_{ij} = a(\varphi_j, \varphi_i)$, and $F_i = (f, \varphi_i)$. Usually N is large and the stiffness matrix A is sparse.

Our multi-level solution procedure involves a sequence of nested triangulations $\mathscr{T}_j$, with $\mathscr{T}_{j+1}$ nested in $\mathscr{T}_j$, $j \geq 1$, and the corresponding N_j-dimensional subspaces, $\mathscr{M}_j$, of C^0 piecewise-linear polynomials. (By nested, we mean that each triangle of $\mathscr{T}_{j+1}$ intersects the interior of exactly one triangle of $\mathscr{T}_j$.) Corresponding to each subspace $\mathscr{M}_j$ is the problem P_j, the analog of (1.3): Find $u_j \in \mathscr{M}_j$ satisfying

$$a(u_j, v) = (f, v) \quad \text{for all} \quad v \in \mathscr{M}_j. \tag{1.5}$$

The multi-level method also requires the solution of problems P_j' of more general form: Find $x_j \in \mathscr{M}_j$ satisfying

$$a(x_j, v) = G(v) \quad \text{for all} \quad v \in \mathscr{M}_j \tag{1.6}$$

where $G(v)$ is a linear functional defined for $v \in \mathscr{M}_j$. Both (1.5) and (1.6) require the solution of a linear system involving a stiffness matrix like A.

The current interest in multi-level methods is primarily due to the fact that they are of optimal order in terms of computational complexity [3—5, 8—9, 15]. That is, under certain hypotheses on the continuous problem (1.1)—(1.2) and the triangulations $\mathscr{T}_j$, the work required to compute an approximation $\tilde{u}_j \in \mathscr{M}_j$ to $u_j \in \mathscr{M}_j$ of (1.5), which satisfies

$$|||\tilde{u}_j - u||| \leq C N_j^{-q} \tag{1.7}$$

is proportional to N_j. Here $C = C(u, a, b, \Omega, \mathscr{T}_1)$ and q is the "correct" exponent provided by the standard finite element analysis [10, 11]. Several of these theoretical hypotheses may not be satisfied by some problems which may be solved using PLTMG; nonetheless, as we will see later, the multi-level scheme appears empirically to work well even in these cases.

There are five major organizational blocks within PLTMG: subroutines GRID, MATRIX, RHS, MG, and ADAPT. GRID is used to construct a sequence of triangulations according to user specifications. ADAPT is used to construct triangulations using an adaptive refinement procedure based on the ideas of Babuška and Rheinboldt [1, 2]. In both cases, the user need only describe Ω using a minimal number of triangles. Our approach to the refinement problem is adapted from techniques used by Bank and Dupont in the solution of certain non-linear parabolic problems, in which the grid was periodically redefined in order to track the propagation of fronts and singularities. A brief description of our refinement procedure is given in Section 2.

MATRIX and RHS are used to assemble the matrices and right hand sides for problems (1.4). Aside from the interface with our data structures, these routines are basically standard, and we will not describe them there. MG is the driver for the multi-level iteration; some details of this part of our implementation are found in Section 3.

Since multi-level schemes employ a sequence of triangulations, they are ideally suited for use in conjunction with adaptive refinement procedures in which the computed solution $\tilde{u}_{j-1}$, corresponding to $\mathscr{T}_{j-1}$ is used to determine the refinement pattern for $\mathscr{T}_j$. Section 4 describes some aspects of subroutine ADAPT.

Finally, in Section 5 we present some numerical results, including examples with domain singularities and comparisons of the solution time required for our multi-level scheme with those of several other solution methods.

2. Grid Refinement

As noted in the Introduction, a major portion of PLTMG is devoted to grid refinement. In this section we describe some of the procedures used to refine the user-supplied triangulation $\mathscr{T}_0$ of Ω, while in Section 4, we discuss the adaptive refinement process.

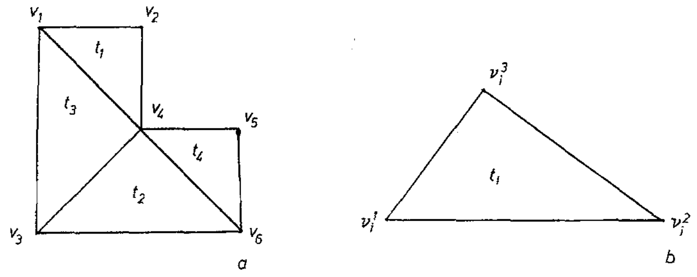

Fig. 2.1. *a* global notation, *b* local notation

Initially, the user of PLTMG supplies a coarse triangulation $\mathscr{T}_0$ of Ω consisting of a small number of triangles t_i, $1 \le i \le \max t_0$ (cf. Fig. 2.1 a). Each triangle t_i contains three vertices v_i^j, $1 \le j \le 3$, and three edges ε_i^j, $1 \le j \le 3$, with ε_i^j opposite v_i^j (cf. Fig. 2.1 b). It is convenient to assign global numbers to the vertices and edges in $\mathscr{T}_0$, denoted by v_k, $1 \le k \le \max v_0$, and e_k, $1 \le k \le \max e_0$, respectively. Thus for $1 \le i \le \max t_0$ and $1 \le j \le 3$, $v_i^j = v_k$ for some k, $1 \le k \le \max v_0$, and $\varepsilon_i^j = e_l$ for some l, $1 \le l \le \max e_0$. Throughout this paper, we will view local designations (e.g. v_i^j) and global designations (e.g. v_k) as interchangeable names for a unique entity, and we will use whichever designation makes more sense in context.

Each edge of a triangle t_i either is a boundary edge of Ω or is part of the perimeter of one or more other triangles in the grid. We define the neighbor of t_i across edge ε_i^j, denoted τ_i^j, as the smallest regular triangle with one edge which completely overlaps ε_i^j. (A regular triangle is one obtained by regular subdivision; see below.) If ε_i^j is a boundary edge, we define $\tau_i^j \le 0$ (the value depending on the boundary conditions). Note that the neighbor relation need not be symmetric and is time dependent.

Our refinement algorithm is motivated by three constraints:

(i) The size of the smallest interior angle of any triangle should be bounded away from 0.

(ii) The transition between large and small triangles in the grid should be "smooth".

(iii) The user of PLTMG should be able to control the amount of refinement in various regions of Ω.

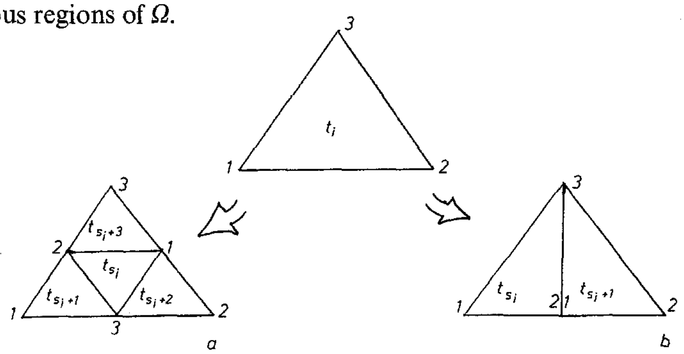

Fig. 2.2. *a* regular subdivision, *b* "green" subdivision. (Vertex labels refer to the superscript in v_i^j notation)

To ensure that (i) is met, we allow only two types of triangle subdivision: regular and "green"[1]. In regular subdivision (cf. Fig. 2.2 a) a triangle t_i is divided into four smaller triangles, denoted t_{s_i+j}, $0 \leq j \leq 3$, by joining the midpoints of its edges. Each of the four new triangles (called "sons of t_i") is similar to t_i (its "father"), so that regular subdivision never reduces the size of the interior angles.

In green subdivision a triangle t_i is divided into two smaller "green triangles", denoted t_{s_i} and t_{s_i+1}, by inserting a "green edge" joining a vertex v_i^j to the midpoint of the opposite edge ε_i^j (cf. Fig. 2.2 b). Green subdivision may reduce the size of the smallest interior angle, so repeated use could violate (i). Hence we only use it to "clean up" the grid by removing degenerate quadrilaterals which remain after all regular subdivision has been completed.

To meet objective (ii), and to insure that the "clean up" will involve only degenerate quadrilaterals, we divide a triangle t_i using the regular subdivision process whenever two neighbors have been divided once, or one neighbor has been divided twice (cf. Fig. 2.3).

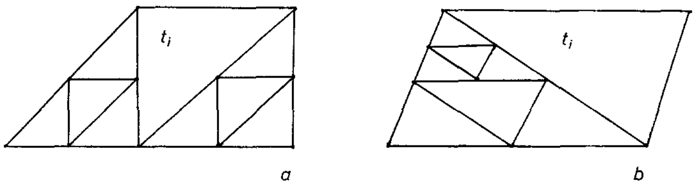

Fig. 2.3. *a* two neighbors divided once, *b* one neighbor divided twice. (Situations requiring regular subdivision of t_i)

Finally, to allow the user of PLTMG to guide the refinement process, we introduce the notions of triangle and vertex level numbers. The level number of a triangle t_i is an indication of the number of subdivisions required to obtain t_i from a triangle of $\mathcal{T}_0$, while the level number of a vertex v_k is an indication of the amount of refinement which should occur near v_k. These level numbers are used to control the refinement process in the sense that any triangle containing a vertex v_k must be regularly subdivided if its level is smaller than that of v_k.

The vertex levels for vertices in $\mathcal{T}_0$ are user-specified, and through them the user of PLTMG can cause different amounts of refinement to occur in different regions of Ω. The vertex level of a vertex v_k which is created at the midpoint of an edge ε_i^j during regular dubdivision is defined by a user-specified weighted average of the levels of the endpoints of ε_i^j. By changing the weighting parameter, the user can adjust the sharpness of the mesh grading near vertices of large level number.

To conclude this section we discuss the way that our refinement algorithm fits into the multi-level solution scheme. Applying the algorithm to the initial triangulation causes a sequence of regular triangle subdivision to occur, eventually

[1] The term "green" subdivision was used by Bank and Dupont and can be traced back to Donald Rose.

leading to a fully-refined grid. However, we can stop the process early by limiting the maximum allowable triangle level number and, after adding necessary green edges, obtain a partially refined triangulation suitable for use with the multi-level scheme. To obtain the necessary sequence of such triangulations, we simply make use of an increasing sequence of limits on the triangle level numbers. The result is an efficient procedure that generates a sequence of triangulations of Ω which are nested (except for certain green triangles) and which satisfy the other constraints requisite to their use with the multi-level solution scheme (cf. Bank and Dupont [4].

3. The Multi-Level Iteration

The term "multi-level iteration" can be applied to a diverse class of iterative methods (cf. [3—5, 8 – 9, 15]). In this section we describe the particular algorithm within this class which we have implemented in PLTMG.

Our algorithm is built upon what we shall call the k-level scheme for solving the problem P_k' introduced in Section 1. Assume that we have a nested sequence of triangulations $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ generated as described in Section 2, and let $b_k(\cdot,\cdot)$ be a symmetric bilinear form associated with $\mathcal{T}_k$. One step of the k-level scheme takes an initial approximation $z_0 \in \mathcal{M}_k$ to a final approximation $z_{m+1} \in \mathcal{M}_k$, where m is a parameter of the method which may be specified. This single step may be defined recursively as follows:

(i) For $l = 1, 2, \ldots, m$, solve the following for z_l:

$$b_k(z_l - z_{l-1}, \varphi) = G(\varphi) - a(z_{l-1}, \varphi) \quad \text{for all} \quad \varphi \in \mathcal{M}_k. \tag{3.1}$$

(ii) Find $\tilde{\delta} \in \mathcal{M}_{k-1}$, an approximation to $\delta \in M_{k-1}$, the solution of

$$a(\delta, \varphi) = G(\varphi) - a(z_m, \varphi) \equiv \bar{G}(\varphi) \quad \text{for all} \quad \varphi \in \mathcal{M}_{k-1}. \tag{3.2}$$

(iii) Set

$$z_{m+1} = z_m + \tilde{\delta}. \tag{3.3}$$

Iteration (3.1) is called a smoothing iteration; its purpose is to damp components of the error which oscillate on the order of the mesh size in $\mathcal{T}_k$. The bilinear form $b_k(\cdot,\cdot)$ can represent any iterative method [12, 13] for systems of linear equations which will do this, although under-relaxed Jacobi-like schemes are typically chosen in theoretical studies. In PLTMG, however, we have chosen to implement a symmetric Gauss-Seidel iteration [12, 13]. For other possibilities, see [5].

If the smoothing iteration is successful, then the error becomes less oscillatory; and we can anticipate that it can be well-approximated by an element in the space of smaller dimension $\mathcal{M}_{k-1}$. Thus in (3.2) we compute $\tilde{\delta}$, an approximate elliptic projection of the error into $\mathcal{M}_{k-1}$, using sparse Gaussian elimination for $k = 2$ or two iterations of the $k-1$ level scheme with initial approximation zero for $k > 2$.

In (3.3) we add the approximate error to z_m to obtain z_{m+1}. If $\mathcal{T}_k$ is nested in $\mathcal{T}_{k-1}$ (i.e. no triangles in $\mathcal{T}_k$ intersect more than one triangle in $\mathcal{T}_{k-1}$), then

$\mathcal{M}_{k-1} \subseteq \mathcal{M}_k$ so that both $\tilde{\delta}$ and z_{m+1} will be in $\mathcal{M}_k$. In practice, however, this may not quite hold because some triangles which are divided into two green triangles in $\mathcal{T}_{k-1}$ are regularly divided in $\mathcal{T}_k$. In this event, we simply replace $\tilde{\delta}$ in (3.3) by its interpolant into $\mathcal{M}_k$.

Using the k-level scheme (3.1)—(3.3) as an inner iteration, we can now build the overall multi-level iterative process aimed at solving the sequence of problems P_j, $1 \le j \le k$, introduced in Section 1. In this process we successively compute the approximate solutions $\tilde{u}_j$ of P_j, $1 \le j \le k$ as follows:

(i) For $j=1$, compute $\tilde{u}_1$ by solving (1.5) directly.

(ii) For $2 \le j \le k$, compute $\tilde{u}_j$ using r iterations of the j-level scheme applied to (1.5) with $\tilde{u}_{j-1}$ as the initial approximation. (Here r is a parameter that may be user-specified.)

As an example, the multi-level process for the case $k=3$ is summarized below, illustrating the relation between the parameters m and r:

(i) Solve directly for $\tilde{u}_1$.

(ii) Starting from $\tilde{u}_1$, carry out r iterations of (a)—(c):
  (a) m smoothing iterations at level 2,
  (b) direct solution at level 1,
  (c) update iterate at level 2.
  Take level 2 iterate as $\tilde{u}_2$.

(iii) Starting from $\tilde{u}_2$ carry out r iterations of (a)—(h):
  (a) m smoothing iterations at level 3,
  (b) m smoothing iterations at level 2,
  (c) direct solution at level 1,
  (d) update iterate at level 2,
  (e) m smoothing iterations at level 2,
  (f) direct solution at level 1,
  (g) update iterate at level 2,
  (h) update iterate at level 3.
  Take level 3 iterate as $\tilde{u}_3$.

In (iii), the smoothing iterations at level 3 correspond to the original problem (1.5) with $j=3$, those at level 2 correspond to the residual equations, and the direct solutions at level 1 correspond to the residual equations of the residual equations.

To implement the multi-level scheme in general, we need four basic pieces of software in addition to a straightforward driver:

(i) a module to carry out m smoothing iterations for any problem P_j', $j \ge 2$;

(ii) a module to compute the righthand side of (3.2) for any pair of spaces $(\mathcal{M}_{j-1}, \mathcal{M}_j)$, $j \ge 2$;

(iii) a module to carry out (3.3); and

(iv) a module to directly solve P_1'.

In PLTMG we have written our own subroutines for (i)—(iii) and made use of the Yale Sparse Matrix Package [7] for (iv).

4. Adaptive Refinement

In the multi-level scheme described in the previous section, no information about the level j grid $\mathcal{T}_j$ is required until the approximation $\tilde{u}_{j-1}$ has been computed. Although we have assumed thus far that all grids are known in advance, it is clear that we may, in fact, use $\tilde{u}_{j-1}$ to adaptively determine $\mathcal{T}_j$.

Our adaptive refinement procedure is based on the ideas of Babuška and Rheinboldt [1, 2]. Let $t_i \in \mathcal{T}_{j-1}$ have diameter h_{t_i}, and let

$$|||z|||_{t_i}^2 = \int_{t_i} a\,|\nabla z|^2 + b\,z^2\,dx \tag{4.1}$$

denote the energy norm associated with t_i. Following [1, 2] we estimate the error, in energy, in t_i using

$$|||\tilde{u}_{j-1} - u|||_{t_i}^2 = C_1(a, t_i)\,h_{t_i}^2 \int_{t_i} (L\tilde{u}_{j-1} - f)^2\,dx + C_2(a, t_i)\,h_{t_i} \int_{\partial t_i} J^2\,dx \tag{4.2}$$

where J is the jump in normal derivative in $\tilde{u}_{j-1}$ across ∂t_i. C_1 and C_2 are computable constants which depend on the coefficient function $a(x, y)$ and the geometry of t_i (but are independent of the size of t_i). The factors of h_{t_i} are required to make the homogeneity of the right and left hand sides of (4.2) consistent. Formula (4.2) is modified slightly for triangles with one or more boundary edges. The right hand side of (4.2) is thus computable for each $t_i \in \mathcal{T}_{j-1}$ using only $\tilde{u}_{j-1}$ and the partial differential equation.

We now describe the adaptive computation of $\mathcal{T}_j$ from $\mathcal{T}_{j-1}$. Let $e_{\max}$ denote the largest estimated error in any triangle in $\mathcal{T}_{j-1}$, and let $t_{\max} \in \mathcal{T}_{j-1}$ be a triangle with estimated error $e_{\max}$. Let e_f denote the estimated error in the father of $t_{\max}$, assuming it is known from a previous step of the adaptive procedure. If the local rate of convergence behaves like $C h_{t_{\max}}^q$ for some constants C and q, then the regular refinement of $t_{\max}$ will result in estimated errors of approximate size $e_s = e_{\max}^2/e_f$ in the sons of $t_{\max}$. Therefore, to construct $\mathcal{T}_j$ from $\mathcal{T}_{j-1}$, we divide only those triangles in $\mathcal{T}_{j-1}$ which have estimated errors larger than the threshold value $e_{\max}^2/e_f$. (If e_f is not available, we assume $q=1$, and set the threshold to $e_{\max}/2$.)

To save both time and space, we have actually implemented the adaptive scheme in PLTMG with two modifications:

(i) A triangle t_i may be in both $\mathcal{T}_{j-1}$ and $\mathcal{T}_j$ if its error estimate is less than the threshold value. For such a triangle we do not recompute the error using $\tilde{u}_j$, but instead use the estimate computed previously using $\tilde{u}_{j-1}$.

(ii) If there is a severe singularity, it is possible that only a few triangles near that singularity will have errors larger than the threshold and be refined. If this were to occur in several consecutive refinements, the result would be a sequence of subspaces $\mathcal{M}_j$ of slowly increasing dimension, requiring extra storage for unnecessary matrices, and extra work for the multi-level iteration. To avoid this, we have incorporated a "level compression" feature into PLTMG. If sufficiently few vertices have been added in an adaptive refinement step, we discard the old triangulation $\mathscr{T}_{j-1}$ in favor of the newly created one. In this fashion, we generate a sequence of subspaces $\mathscr{M}_j$ whose dimensions approximately satisfy

$$2N_{j-1} \leq N_j \leq 4N_{j-1},$$

insuring a geometric increase in dimension.

We do not know theoretically what effect level compression has on the rate of convergence. Current convergence proofs explicitly or implicitly assume a sequence of quasi-uniform grids that is unlikely to be produced in situations where level compression is employed [3—4, 8—9, 15]. In practice, however, the procedure seems to have the desirable effects of reducing both work and storage, since it reduces the number of triangulations (and associated algebraic problems) that must be stored and used. Moreover, we have observed only a modest reduction in the observed rate of convergence.

5. Numerical Results

In this section we examine two aspects of the performance of PLTMG: the effectiveness of the adaptive procedure in dealing with singularities due to $\partial\Omega$ and the computation time required to approximately solve the discrete problem for $\tilde{u}_k$ when the triangulations $\mathscr{T}_1, \mathscr{T}_2, \ldots, \mathscr{T}_k$ and the corresponding matrices are already available. In the former case, we compare results obtained using PLTMG with those expected from the theory; while in the latter case, we compare PLTMG with alternative methods for solving the discrete equations (1.4).

To study the behavior of the adaptive procedure, we consider a sequence of five problems defined on portions of the unit circle C centered at the origin. In particular, for $\alpha = 1 + k/4$, $0 \leq k \leq 4$, we consider the problem

$$\begin{aligned} -\Delta u_\alpha &= 0 \quad \text{in } \Omega_\alpha \\ u_\alpha &= r^{1/\alpha} \sin(\theta/\alpha) \quad \text{on } \partial\Omega_\alpha{}^2 \end{aligned} \tag{5.1}$$

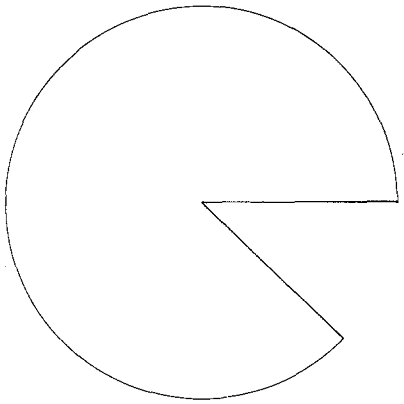

Fig. 5.1. Ω_α for $\alpha = 1.75$

[2] For expositional convenience, we use polar coordinates to represent the boundary conditions.

where Ω_α is C with a wedge of central angle $(2-\alpha)\pi$ removed (see Fig. 5.1). When $\alpha=1$, Ω_α is a semi-circle with no singularity at the origin; when $\alpha=2$, Ω_α has a crack at $0\le x\le 1$, $y=0$. The exact solution to (5.1), $u_\alpha=r^{1/\alpha}\sin(\theta/\alpha)$, displays the principal part of the singularity (if any), due to the corner or crack at the origin [11].

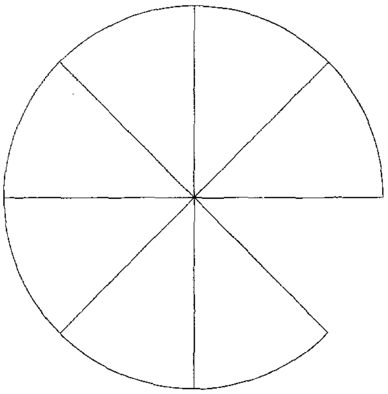

Fig. 5.2. The initial triangulation $\mathscr{T}_0$ for $\alpha=1.75$

For each α we solved (5.1) twice using a four-level scheme with $\mathscr{T}_1=\mathscr{T}_0$, where $\mathscr{T}_0$ consisted of 4α triangles formed by two radii of C and a circular arc of $\pi/4$ radians joining them along $\partial\Omega_\alpha$, (see Fig. 5.2). In the first case, $\mathscr{T}_j$, $2\le j\le 4$, was constructed by regular refinement of each triangle in $\mathscr{T}_{j-1}$. In the second case, triangulations were generated by the adaptive procedure using level compression. For $\alpha>1$, the smallest triangles in the adaptive case were always located near the origin and were significantly smaller in size than the smallest triangles in the uniform case. Fig. 5.3 shows the finest triangulation for both cases for $\alpha=1.75$.

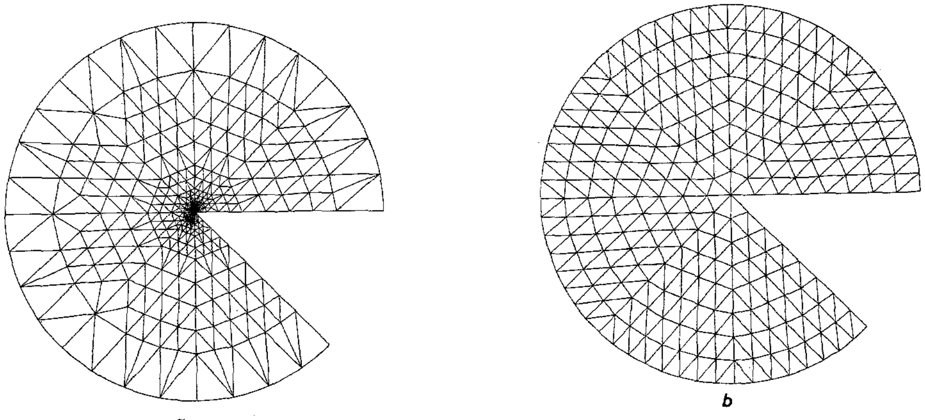

Fig. 5.3. *a* $\mathscr{T}_4$ for the adaptive refinement case, *b* $\mathscr{T}_4$ for the uniform refinement case

Table 5.1. *Convergence of $u_{\alpha,j}$ to u_α*

α	Type	$j=1$		$j=2$		$j=3$		$j=4$		c	q
		N_j	Digits	N_j	Digits	N_j	Digits	N_j	Digits		
1.00	Uniform	6	12.66	15	12.66	45	12.66	153	12.65	—	—
	Adaptive	6	12.63	31	12.63	131	12.61	376	12.59	—	—
1.25	Uniform	7	.75	18	.97	55	1.19	189	1.42	.42	.92
	Adaptive	7	.76	25	1.04	99	1.39	272	1.67	.56	1.16
1.50	Uniform	8	.49	21	.69	65	.88	225	1.07	.71	.79
	Adaptive	8	.50	30	.78	90	1.04	217	1.30	1.06	1.11
1.75	Uniform	9	.35	24	.53	75	.70	261	.87	.93	.70
	Adaptive	9	.35	27	.51	119	.82	265	1.16	1.60	1.07
2.00	Uniform	10	.25	27	.42	85	.57	297	.72	1.10	.62
	Adaptive	10	.26	32	.44	78	.67	239	.89	1.68	.93

In Table 5.1, we indicate the rates of convergence of the sequences $u_{\alpha,j}$ to u_α for each α where

$$\text{Digits} = -\log_{10}\{|||u_{\alpha,j}-u_\alpha|||/|||u_\alpha|||\}. \tag{5.2}$$

(The C^0-piecewise-quadratic interpolant of u_α with respect to the finest grid was used for purposes of computing the error.) Using this data, we did a least squares fit of the data to an error bound of the form

$$\frac{|||u_{j,\alpha}-u_\alpha|||}{|||u_\alpha|||} \cong C N_j^{-q/2} \tag{5.3}$$

to obtain C and q. Because of the singularity in u_α, the optimal value of q as $N\to\infty$ for uniform grids is $q=1/\alpha$ [1, 11]; the best possible rate of convergence for piecewise linear finite elements in general is $q=1$, a rate which was essentially achieved by the adaptive procedure.

We next did a convergence study to determine, for each α, the rate of convergence of $\tilde{u}_{\alpha,4}$ to $u_{\alpha,4}$ as a function of m and r, where

$$\text{Digits} = -\log_{10}\{|||\tilde{u}_{\alpha,4}-u_{\alpha,4}|||/|||u_{\alpha,4}|||\}. \tag{5.4}$$

Table 5.2. *Convergence of $\tilde{u}_{\alpha,4}$ to $u_{\alpha,4}$ for $\alpha=1.75$*

r	Uniform Grids					Adaptive Grids				
	m					m				
	1	2	3	4	5	1	2	3	4	5
1	1.49	1.87	2.04	2.15	2.23	1.06	1.22	1.31	1.38	1.45
2	2.26	2.87	3.16	3.35	3.50	1.30	1.50	1.66	1.80	1.94
3	2.91	3.80	4.22	4.52	4.75	1.44	1.71	1.95	2.17	2.37
4	3.54	4.71	5.28	5.68	5.99	1.56	1.92	2.23	2.52	2.79
5	4.15	5.62	6.33	6.85	7.23	1.68	2.12	2.51	2.87	3.21

The results for $\alpha=1.75$ appear in Table 5.2; those for other values of α are similar.

For the case of quasi-uniform grids, Bank and Dupont [4] have shown that the energy norm of the error is reduced by a factor of at least $(Cm^{-\gamma})^r$, where C and γ are positive constants independent of j, but which depend on the geometry and sizes of triangles in $\mathcal{T}_1$ and the partial differential equation. Technically, the adaptive case can be forced into this framework as well, since we are dealing with a fixed, finite number of multi-grid levels. The slower rate of convergence to $u_{\alpha,4}$ for the adaptive grids appears to be due to a combination of the presence of triangles of more widely varying sizes in the adaptive grids (leading to matrix elements of more widely varying magnitude), the use of level compression which accentuates this variance, and the ordering of the vertices. Note, however, that in all cases, for the choice $m=r=1$, $\tilde{u}_{\alpha,4}$ already has essentially all the significant digits which $u_{\alpha,4}$ has as an approximation to the solution of the partial differential equation.

In terms of computational effort, it can be shown that the work essentially depends linearly on m and r. As m becomes large, we derive less benefit from the additional smoothing iterations done on the coarser grids. If the bound $(Cm^{-\gamma})^r$ is *sharp*, one can show the optimal choice of m over a large class of interesting problems is bounded by a small number, e.g. 3—4.

To get an idea of the speed of the multi-level solution process, we now turn to a more standard partial differential equation:

$$\begin{aligned} -\Delta u &= -1 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega \end{aligned} \tag{5.5}$$

where Ω is the unit square $0<x<1$, $0<y<1$. We discretized Ω using five uniform refinements of an initial triangulation $\mathcal{T}_0$ consisting of eight triangles and nine vertices (see Fig. 5.4) and obtained a discrete system (1.4) of order $N=1089$.

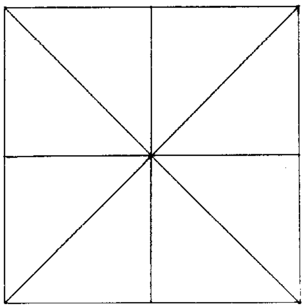
Fig. 5.4. The initial triangulation $\mathcal{T}_0$

We solved the discrete problem with our multi-level solution procedure and six other solution methods. For the multi-level procedure, we used a four-level scheme with $\mathcal{T}_1$ chosen as a uniform refinement of $\mathcal{T}_0$. We used the Yale Sparse Matrix Package [7] to reorder the linear equations and solve them with sparse symmetric Gaussian elimination. Finally, we used subroutines from ITPACK [6] for five different iterative methods (Jacobi with Conjugate Gradient acceleration

(J-CG), Jacobi with Chebychev acceleration (J-SI), Successive over-relatation (SOR), Symmetric-SOR with Conjugate Gradient acceleration (SSOR-CG), and SSOR with Chebyshev acceleration (SSOR-SI)). Each of these methods chooses zero as a starting guess and selects its parameters adaptively.

Two types of timing comparisons are of some interest: first, the time required to accurately solve the discrete system of linear equations as a problem in numerical linear algebra, and, second, the time required to compute an approximate solution which yields an $0(N^{-1/2})$ accurate approximation to u, the solution of (5.5), as measured in the energy norm in the finest mesh. For Gaussian elimination, of course, the times will be the same in both cases; but the other methods should be substantially faster in the second case. For our study, we demanded six digits of accuracy in the first case and two digits of accuracy in the second, both measured in the energy norm. (The two digits in the second case is actually too much accuracy for $N=1089$, but it is unlikely that one would accept less than this in practice.) All times are in seconds on a CDC-6600 using the MNF Fortran compiler.

Table 5.3. *Times for solving the discrete equations with the Yale Sparse Matrix Package (Accuracy of approximately* 12 *digits)*

Domain Preprocessing	Minimum Degree Ordering	3.899
	Symbolic Factorization	.590
	Total	4.489
Numeric Processing	Reordering	.208
	Factorization	2.295
	Solution	.280
	Total	2.783
Total		7.272

Table 5.4. *Times for solving the discrete equations with iterative methods*

Method	"Full" Accuracy		$0(N^{-1/2})$ Accuracy	
	Digits	Time	Digits	Time
J-CG	5.98	6.79	2.65	4.13
J-SI	6.09	13.35	2.40	6.87
SOR	6.10	8.57	2.11	4.15
SSOR-CG	6.33	14.75	2.02	3.95
SSOR-SI	6.06	12.55	2.31	6.11
PLTMG ($r=5$, $m=2$)	6.75	3.44	—	—
PLTMG ($r=1$, $m=1$)	—	—	2.10	0.49

Our results are summarized in Tables 5.3 and 5.4. For accurately solving the discrete equations, it appears that Gaussian elimination is somewhat faster than the multi-level scheme if one discounts the cost associated with reordering the system of equations and performing other grid-dependent pre-processing[3]. The other iterative methods are much slower[4].

On the other hand, when one only wishes to solve the discrete equations up to $0(N^{-1/2})$ accuracy, the multi-level scheme and the other iterative methods look much better than Gaussian elimination. The reason for this, of course, is that it is not possible to make use of the lower accuracy requirements to reduce the costs of a direct method as one can with an iterative method. It is interesting to note that, in this situation, the multi-level scheme is still superior to any of the other methods that we tried for this problem.

References

[1] Babuška, I., Rheinboldt, W. C.: Error estimates for adaptive finite element computations. SIAM J. Numer. Anal. *15*, 736—754 (1978).

[2] Babuška, I., Rheinboldt, W. C.: Analysis of optimal finite-element meshes in R^1. Math. of Comp. *33*, 435—464 (1979).

[3] Bakhvalov, N. S.: On the convergence of a relaxation method with natural constraints on the elliptic operator. Zh. Vŷchisl. Mat. mat. *6*, 861—885 (1966).

[4] Bank, R. E., Dupont, T.: An optimal order process for solving finite element equations. (Revised January 1978), submitted.

[5] Brandt, A.: Multi-level adaptive solutions to boundary value problems. Math. of Comp. *31*, 333—390 (1977).

[6] Grimes, R. G., Kincaid, D. R., MacGregor, W. I., Young, D. M.: Adaptive iterative algorithms using symmetric sparse storage. Report CNA 139, Center for Numerical Analysis, University of Texas at Austin, August, 1978.

[7] Eisenstat, S. C., Gursky, M. C., Schultz, M. H., Sherman, A. H.: Yale sparse matrix package I: The symmetric codes. Research Report No. 112, Dept. of Computer Science, Yale University, 1977.

[8] Hackbusch, W.: On the convergence of multi-grid iterations. Beiträge Numer. Math. *9* (to appear).

[9] Nicolaides, R. A.: On the l^2 convergence of an algorithm for solving finite element equations. Math. of Comp. *31*, 892—906 (1977).

[10] Oden, J. T., Reddy, J. N.: An introduction to the mathematical theory of finite elements. J. Wiley 1976.

[3] This may be a reasonable thing to do if, as often happens with nonlinear and time-dependent problems, there are several problems to be solved on a fixed grid.

[4] For problems in two dimensions, several of the iterative methods will eventually become more efficient than Gaussian elimination (for large enough problems). For problems as simple as our model problem, the crossover point in terms of multiplicative operations does not occur until N is at least 12,500 [16]. For more complicated problems, the required size of N may be expected to be even larger. It is true, of course, that iterative methods may enjoy a substantial storage advantage, even for smaller problems.

[11] Strang, G., Fix, G.: An analysis of the finite element method. Prentice-Hall 1973.
[12] Varga, R. S.: Matrix iterative analysis. Prentice-Hall 1962.
[13] Young, D.: Iterative solution of large linear systems. Academic Press 1971.
[14] Bank, R. E., Sherman, A. H.: Algorithmic aspects of the multi-level solution of finite element equations. Report CNA 144, Center for Numerical Analysis, University of Texas at Austin, October 1978.
[15] Astrakhantsev, G. P.: An iterative method of solving elliptic net problems. Zh. Vȳchisl. Mat. mat. *11*, 439—448 (1971).
[16] Chandra, R.: Conjugate gradient methods for partial differential equations. Ph. D. Dissertation, Yale University, 1978.

Prof. A. H. Sherman
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, U. S. A.

Prof. R. E. Bank
Department of Mathematics
University of Texas at Austin
Austin, TX 78712, U. S. A.