N94-21470

MGGHAT: ELLIPTIC PDE SOFTWARE WITH ADAPTIVE REFINEMENT, MULTIGRID AND HIGH ORDER FINITE ELEMENTS

S6-64
197566
p- 10

William F. Mitchell
GE Advanced Tech Labs
Moorestown, NJ 08041

SUMMARY

MGGHAT (MultiGrid Galerkin Hierarchical Adaptive Triangles) is a program for the solution of linear second order elliptic partial differential equations in two dimensional polygonal domains. This program is now available for public use. It is a finite element method with linear, quadratic or cubic elements over triangles. The adaptive refinement via newest vertex bisection and the multigrid iteration are both based on a hierarchical basis formulation. Visualization is available at run time through an X Window display, and a posteriori through output files that can be used as GNUPLOT input. In this paper, we describe the methods used by MGGHAT, define the problem domain for which it is appropriate, illustrate use of the program, show numerical and graphical examples, and explain how to obtain the software.

INTRODUCTION

MGGHAT (MultiGrid Galerkin Hierarchical Adaptive Triangles) is a program for the solution of linear second order elliptic partial differential equations in two dimensional polygonal domains. It solves equations of the form:

$$(pu_x)_x + (qu_y)_y + ru = f \quad \text{in } \Omega$$

$$u = g \quad \text{on } \partial\Omega_1$$

$$\frac{\partial u}{\partial n} + cu = g \quad \text{on } \partial\Omega_2$$

where Ω is a polygonal domain in $\mathbf{R}^2$ and p, q, r, f, c, and g are functions of x and y, and n is the unit normal direction.

MGGHAT uses a finite element method with linear, quadratic or cubic elements over triangles. The adaptive refinement via newest vertex bisection and the multigrid iteration are both based on a hierarchical basis formulation. Visualization is available at run time through an X Window display, and for post-run analysis through output files that can be used as GNUPLOT input. The program is now available in the public domain through mgnet and netlib.

NUMERICAL METHOD

The numerical method used by MGGHAT is a finite element method with adaptive refinement of the grid and a multigrid solution of the equations. In this section we briefly describe the method used. More details of the method can be found in [1], and a full description and analysis in [2], which is contained in the MGGHAT software package.

Discretization

MGGHAT solves elliptic differential equations using the standard Galerkin finite element method. A triangular mesh is used over the 2D domain. The basis functions are C^1 continuous piecewise polynomials of any specified degree. Currently, the program only handles linear, quadratic and cubic polynomials, but can be modified to handle higher order polynomials by defining a quadrature rule of the appropriate accuracy.

Adaptive refinement

The program provides automatic adaptive refinement of the grid to ensure the highest accuracy for the number of nodes used. The refinement of triangles is performed using the newest vertex bisection method. This method divides pairs of triangles through the midpoint of their common edge, which is equivalent to enhancing the approximation space by one hierarchical basis function (in the linear case). The error estimate, used to determine which triangles should be divided, is based on an estimate of the coefficient of the new hierarchical basis function.

Solution

The equations are solved using a hierarchical basis multigrid method. The relaxation phase consists of red-black Gauss-Seidel iterations on the nodal basis equations. The number of iterations can be user specified, but usually a red phase before coarse grid correction and a red and black phase after coarse grid correction suffices for optimal convergence rates. The grid transfers are a natural consequence of the transformation between the nodal and hierarchical bases, and can be shown to lead to a method equivalent to the "Galerkin" multigrid method in simple cases.

MGGHAT SOFTWARE

MGGHAT is written in standard FORTRAN 77, and is callable as a subroutine. An example *main* program for MGGHAT is shown in Figure 1. The program has been tested on 3 computer configurations: 1) a Pyramid computer using the f77 compiler under a dual port of UNIX SysV Release 2.0, 2) a Sun workstation using the f77 compiler under SunOS 4.1.1, and 3) an i486 based PC using the f2c translator and gcc compiler under the Linux operating system. The program is easily installed with the makefile provided in the distribution, and requires only a FORTRAN compiler for the basic functionality. A C compiler is required for the UNIX dependent supplied timer routine (which can be replaced by the user). A C compiler and X Window libraries are required for the (optional) X Window graphics capability.

Problem Definition

The differential equation, boundary conditions and domain are defined by user supplied subroutines. Figure 2 contains examples of these routines. The subroutine *pde* defines the equation by providing the value of the functions *p, q, r* and *f* at any point (*x,y*). Subroutine *bcond* contains the boundary conditions. The boundary is partitioned into a set of pieces in the initial triangulation. The piece containing the point (x,y) is passed to *bcond* through *ipiece*. *bcond* returns the functions c and g and sets *itype* to flag the boundary condition as Dirichlet or Mixed (including Neuman if c=0). If the true solution is known, the user can supply functions *true*, *truex*, and *truey* to obtain error calculations. The initial triangulation (coarse grid) is defined by the user in subroutine *inittr* (not shown).

Parameters

The user has control over the program through several parameters.
mxvert, mxtri, mxlev, mxnode and *mxtime*: maximum values for the number of vertices, triangles, refinement levels, nodes and execution time can be used as termination criteria.
tol: an error tolerance that can be used as a termination criterion.
outlev: controls the amount of printed output. Can be 0 for no output, 1 for summary at the end of execution, 2 for summary after each program phase, 3 for detailed information, and 4 and 5 for debugging level output. An extraction from a level 2 output is illustrated in Figure 3.
iorder: specifies the order (degree+1) of the piecewise polynomial basis functions.
nu1 and *nu2*: number of (half) red-black Gauss-Seidel iterations to perform before and after coarse grid correction, respectively.
ncyc: number of multigrid cycles to perform in each solution phase.
unifrm: a logical variable to indicate a uniform refinement should be used rather than adaptive refinement.

Graphics

Graphics support is provided in two forms: run time graphics on an X Window display, and output files suitable for input to GNUPLOT. The run time graphics use a small set of routines which call on the X Window graphics library. The user can expand this to support other graphics devices by writing equivalent routines (draw a point, draw a line, print some text, etc.) for the desired device. There are nine forms of run time graphics:

1) contour plot of computed solution with triangulation
2) contour plot of true solution with triangulation
3) contour plot of error with triangulation
4) color plot of computed solution
5) color plot of true solution
6) color plot of error
7) triangulation
8) graph of number of nodes vs. relative error in energy norm (or error estimate)
9) contour plot of both computed solution and true solution

Either one or two of these forms can be displayed during one run. When two are displayed, additional numerical information is printed on the display, including grid size information, norms of the error and error estimate, and execution time. Figure 4 contains an example of the run time graphic displays.

The user can select to save information in data files for later processing by GNUPLOT. These files contain the triangulation, computed and true solutions, and convergence data. Figures 5 and 6 contain plots generated by GNUPLOT.

OBTAINING MGGHAT

MGGHAT is now available in the public domain. It can be obtained either from mgnet or netlib.

mgnet

To obtain MGGHAT from mgnet (the multigrid network) ftp to casper.cs.yale.edu. Login as *anonymous* and use your email address as the password. Change to the mgghat directory by typing *cd mgnet/mgghat*. Then type *ls* to see what files are available, and *get filename* for each file you desire. To learn more about mgnet, also get the file *mgnet.README* from the *mgnet* directory.

netlib

MGGHAT can be obtained from netlib using ftp, the mail server, or *xnetlib*. For ftp retrieval, ftp to research.att.com and follow the anonymous login procedure described above. Look for MGGHAT in the directory netlib/pdes/mgghat. To obtain MGGHAT via email, send a message to netlib@ornl.gov, netlib@research.att.com, or one of the other netlib servers with the message *send index from pdes/mgghat*. To learn how to obtain materials from netlib through an X Window interface, send the message *send index from xnetlib* to one of the netlib mail servers. For more information on netlib, send the message *send index* to one of the netlib mail servers.

REFERENCES

1. Mitchell, W. F.: Optimal Multilevel Iterative Methods for Adaptive Grids. *SIAM J. Sci. Statist. Comput.*, vol. 13, 1992, pp. 146-167.

2. Mitchell, W. F.: Unified Multilevel Adaptive Finite Element Methods for Elliptic Problems. Ph.D. thesis, Technical report UIUCDCS-R-88-1436, Department of Computer Science, University of Illinois, Urbana, IL, 1988.

```
      program main
      include 'commons'     ! all parameters are passed through common
c
c set maximum allowed values based on dimensions
c
      mxvert = ndvert
      mxtri  = ndtri
      mxlev  = ndlev
      mxnode = ndnode
c
c set program parameters
c
      mxtime = 12.*60.*60.  ! maximum execution time in seconds
      ioutpt = 6            ! unit for printed output
      outlev = 2            ! amount (level) of printed output
      iorder = 2            ! polynomial order (linear in this case)
      nu1    = 1            ! number of relaxation iterations before
      nu2    = 2            ! and after coarse grid correction
      ncyc   = 1            ! number of multigrid cycles
      tol    = 0.001        ! error tolerance for termination
      mgfreq = 2.           ! how often to do multigrid cycle
      unifrm = .false.      ! flag for uniform/adaptive grid
      igrf1 = 0             ! run time graphics selections (no
      igrf2 = 0             ! graphics in this example)
      grf1st = 0.           ! a value for which a contour line is drawn
      grfsiz = .1           ! and the spacing between contours
      grffmn =  0.          ! bounds for determining the color
      grffmx =  2.          !  map for color contour plots
      gptri = 0             ! set to 1 to save triangulation for gnuplot
      gpsol = 0             ! set positive to save solution for gnuplot
      gpconv = 0            ! set to 1 to save convergence info for gnuplot
c
      call mgghat           ! invoke mgghat
      stop
      end
```

Figure 1. Sample main program.

```
      subroutine pde(x,y,p,q,r,f)
      real x,y,p,q,r,f
c
c return the values of the pde coefficents at (x,y)
c
c   -( p(x,y) * u  )  -( q(x,y) * u  ) + r(x,y) * u = f(x,y)
c
      p=1.
      q=1.
      r = 0.
      f=-20.*(x**3 + y**3))
      return
      end
c
      subroutine bcond(x,y,ipiece,c,g,itype)
      real x,y,c,g
      integer ipiece,itype
c
c returns boundary condition coefficients at (x,y)
c
c   u  + c(x,y)*u = g(x,y)   or   u = g(x,y)
c    n
c In this example, the b.c. is Dirichlet on piece 1, and 0 Neuman on piece 2
c
      if (ipiece.eq.1) then
         itype = 1
         c = 0.
         g = true(x,y)
      else
         itype = 2
         c = 0.
         g = 0.
      endif
      return
      end
c
      real function true(x,y)            ! true solution of the pde
      real x,y
      true = x**5 + y**5
      return
      end
```

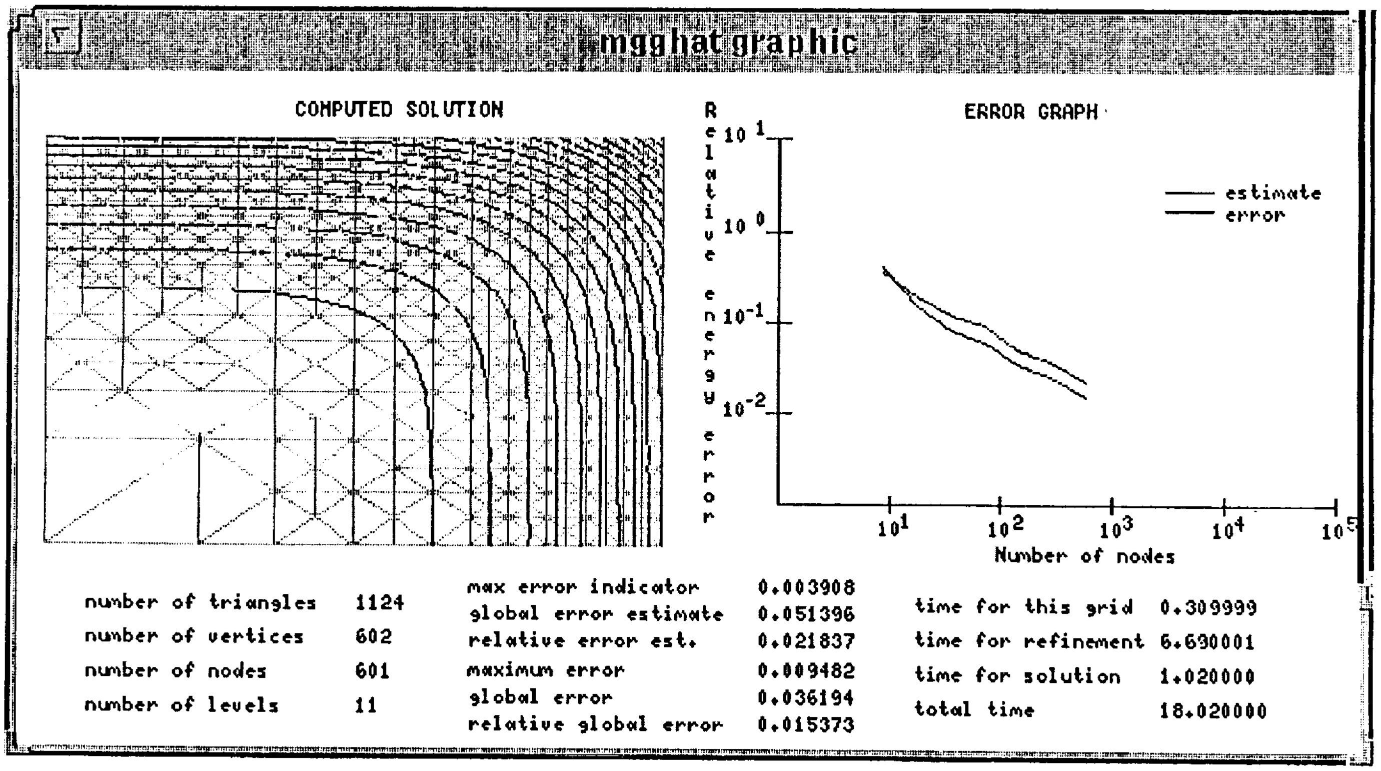
Figure 2. Examples of subroutines to define the problem.

```
MULTIGRID GALERKIN HIERARCHICAL ADAPTIVE TRIANGLES (MGGHAT)

Version 0.9 (March 1993)

input parameters:
output level            2
polynomial order          2
number of cycles          1
relaxes before cgc       1
relaxes after cgc       2
multigrid frequency    2.00
error tolerance     0.0E+00
refinement          adaptive

begin initialization
initializations complete
time for initialization          .00

begin refinement
refinement complete

number of vertices     18
number of nodes         18
number of triangles    22
number of levels        3
time for refinement (this grid)          .02
time for refinement (all grids)          .02

begin solution
solution complete

norms of error:
max norm at vertices     1.20466471E-01
max norm at nodes         1.20466471E-01
max norm at quad pts     2.12660193E-01
continuous energy norm  3.30431342E-01
relative energy norm     1.49259701E-01

time for solution (this grid)          .01
time for solution (all grids)          .01
```

Figure 3. Sample level 2 output.

```
begin error indicators
error indicators and estimates complete

maximum error indicator  1.99157119E-01
error estimate           4.55372840E-01
effectivity index        1.37811637E+00
relative error estimate  1.96938977E-01
relative effect index    1.31943834E+00

time for error estimates (this grid)        .00
time for error estimates (all grids)        .00

time for this refinement/solution step        .03
total time so far                         .03

final solution complete

maximum error at vertices  7.39555359E-02
maximum error at nodes      7.39555359E-02
maximum error at quad pts  1.25789344E-01
continuous energy norm     2.70688415E-01
maximum error indicator    1.41879827E-01
error estimate             4.30013269E-01
effectivity index          1.58859134E+00
relative energy norm        1.87541485E-01
relative effect index      1.58822513E+00

number of vertices        32
number of nodes            32
number of triangles       45
number of levels           5

time for initializations        .00
time for refinement              .08
time for solution               .02
time for error estimates         .00
total time                    .10

termination due to achieving maximum nodes
execution sucessful
```

Figure 3. Sample level 2 output (continued).



```
number of triangles   1124   max error indicator      0.003908
                             global error estimate    0.051396   time for this grid   0.309999
number of vertices    602    relative error est.      0.021837   time for refinement  6.690001
number of nodes       601    maximum error            0.009482   time for solution    1.020000
number of levels      11     global error             0.036194   total time           18.020000
                             relative global error    0.015373
```

Figure 4. Sample run time graphics.

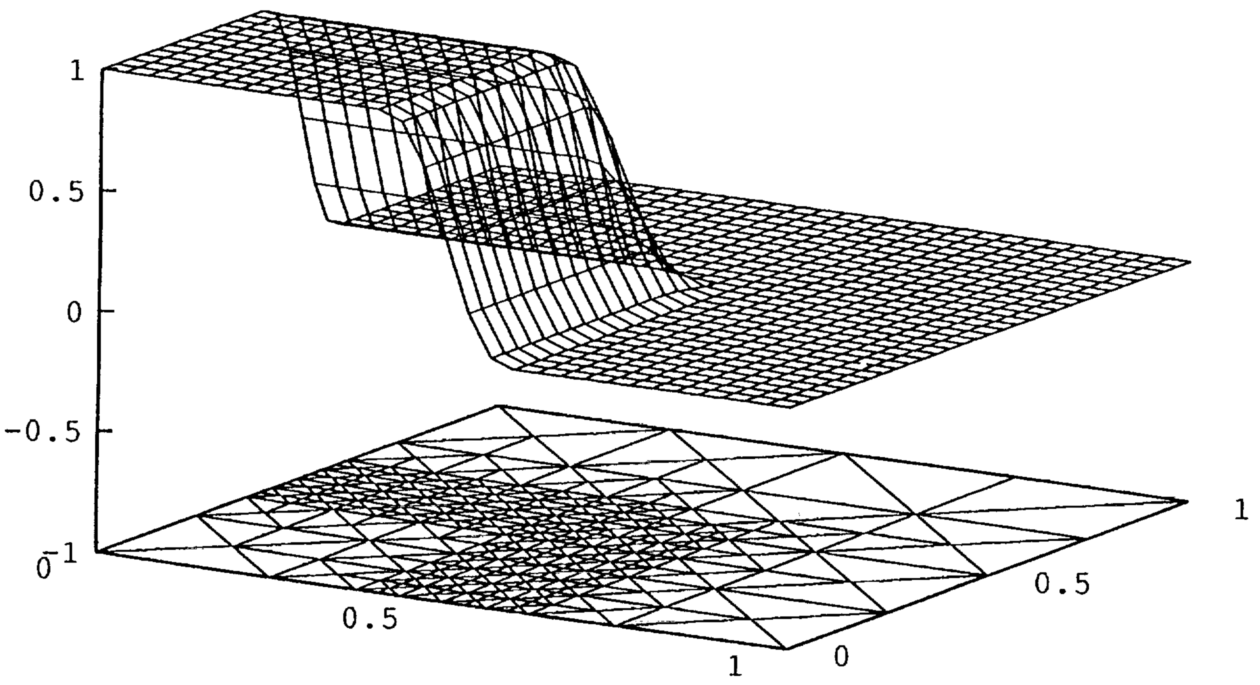


Figure 5. gnuplot plot of triangulation and solution.

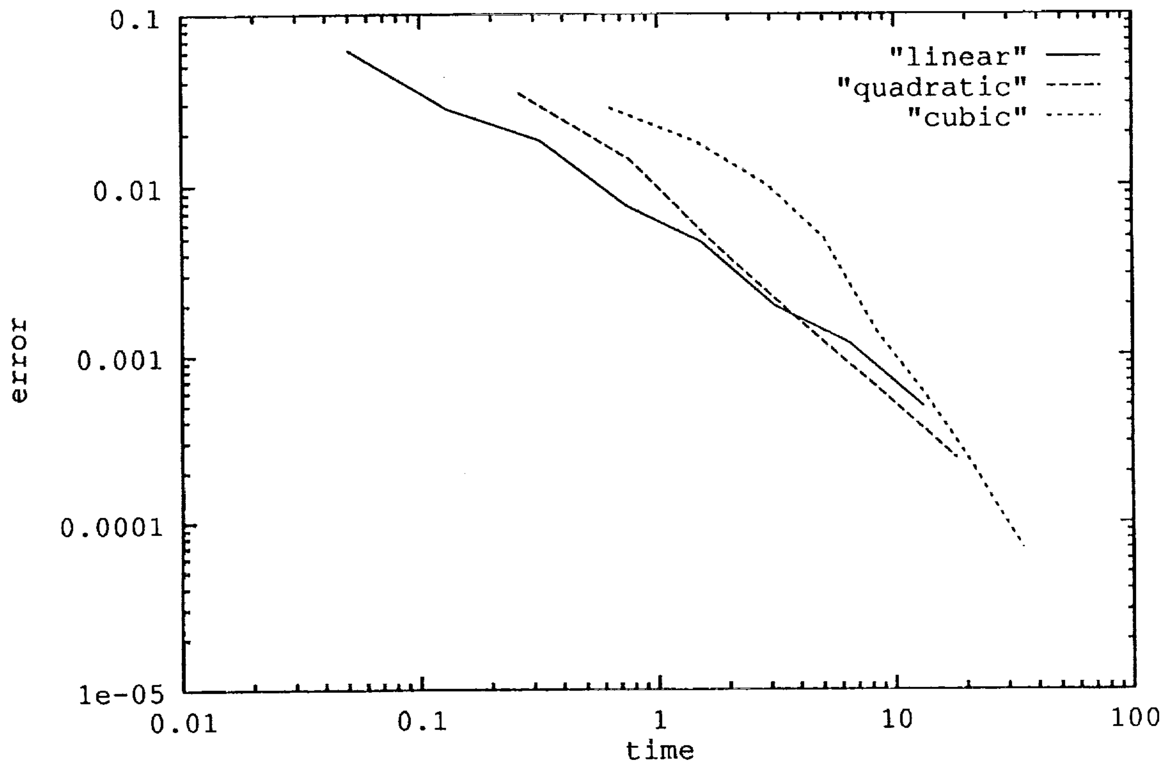


Figure 6. Convergence plot for 3 runs using gnuplot.

C-2