1987

A Fast Grid Adaption Scheme for Elliptic Partial Differential Equations

Calvin J. Ribbens

Report Number:
87-678

A FAST GRID ADAPTION SCHEME FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

Calvin J. Ribbens*
Purdue University

CSD-TR 678
April 1987

Abstract

We describe the *Recursive Subdivision* (RS) grid adaption method—an efficient and effective adaptive grid scheme for two-dimensional, elliptic partial differential equations (PDEs). The RS method generates a new grid by recursively subdividing a rectangular domain. We use a heuristic approach which attempts to equidistribute a given density function over the domain. The resulting grid is used to generate an adaptive grid domain mapping, which may be applied to transform the PDE problem to another coordinate system. The PDE is then solved in the transformed coordinate system using a uniform grid. The RS method generates good adaptive grids at a small cost compared to the cost of the entire computation. We describe the algorithm in detail and demonstrate the effectiveness of our scheme on two realistic test problems.

1 Introduction

This paper describes the Recursive Subdivision (RS) method for generating adaptive grids for the numerical solution of partial differential equations. The problems we consider are two-dimensional, second-order, linear, elliptic boundary value problems. The general form of such a problem is

$$au_{xx} + 2bu_{xy} + cu_{yy} + du_x + eu_y + fu = g \quad \text{on } R, \tag{1}$$

$$pu_x + qu_y + ru = s \quad \text{on } \partial R, \tag{2}$$

*Supported in part by Air Force Office of Scientific Research grant AFOSR 84-0385 and National Science Foundation contract DMS-8301589.

where a, b, c, d, e, f, g, p, q, r and s are functions of x and y. We seek an approximation to the unknown function u which satisfies (1) in the two-dimensional region R and (2) on the boundary ∂R. In the present implementation of the RS method, R must be a rectangular domain.

The power of grid adaption for improving the accuracy of numerical approximations is well-known (see [3] and [19] for surveys). The most common approaches use grid motion or local grid refinement to modify the grid, basing adaption on information from a previous solution to the problem. In the case of time-dependent PDE problems, a natural strategy is to base the adaption on the approximate solution at the most recent time-step. Thus the grid moves with the solution through time. Computing new grid point locations, or identifying areas where refinement is needed, is often quite expensive. For example, Adjerid and Flaherty [1] solve a system of ordinary differential equations in order to determine the grid motion. Similarly, in the moving finite element method [8] the unknowns solved for at each time step include both the approximate solution to the PDE and the position of each grid point at the next time step. A system of simple elliptic PDEs may be solved to generate grids with certain adaptive characteristics (see [20]). In these approaches, the considerable added cost of adapting the grid is often worthwhile, since the solution at several time steps may thereby be improved.

For steady state PDE problems a solution at a previous time step is not available. Thus, traditional approaches to grid adaption would require that an initial solution be done using a non-adapted grid. This may nearly double the work needed to get the final, more accurate solution. One reasonable approach is to do the initial solution using a relatively coarse grid and possibly with a cheaper numerical method. In this way the cost of the initial solution is small compared to the cost of the entire computation. It is still not obvious however, that given the problem information from the initial solution, one can effectively adapt the grid, and do it without significant added cost. We find that such methods are possible. We describe an effective and inexpensive adaptive grid scheme which relies on *a posteriori* information; that is, information from a previous numerical solution of the PDE. See [12] for a description of methods which use *a priori* problem information to generate adaptive grids.

Our method adapts by moving points (rather than adding new points) in a rectangular *tensor product* grid—a grid consisting of the intersection of a set of grid lines in one coordinate direction with a set of lines in the other direction. Even after adaption, grids of this type are still logically tensor products. Besides simplifying the programming task, this class of grids seems to be the most promising with respect to parallelization (see [10]). We distinguish between the grid produced by the RS method, called the *mapping grid*, and the *discretization grid* which is used to discretize and solve the PDE. The mapping grid is used to construct an adaptive grid domain mapping (see [13]). This domain mapping is used to transform a PDE problem from the original domain R to a computational domain S. Solving the transformed PDE problem in Domain S using a

uniform discretization grid is equivalent to solving the original problem in Domain R using an adapted discretization grid. We represent the mapping from Domain S to Domain R by a function $F^{-1}(s,t)$ which takes a point (s,t) in S and maps it to the corresponding point (x,y) in R. Our method chooses F^{-1} so that it approximately maps a uniform grid in Domain S to the mapping grid in Domain R chosen by RS. We represent F^{-1} by a bicubic spline function whose coefficients are determined by least squares. See [13] for a complete description of how to represent adaptive grid domain mappings.

The purpose of this paper is to describe in detail the RS algorithm for generating adaptive mapping grids and to illustrate its performance. In Section 2 we describe the basic steps of the method and mention some method variations. Two extensive examples are given in Section 3. We conclude with some discussion of the results and of future work in Section 4.

2 The RS grid adaption method

The idea behind the RS grid adaption method is to construct an adapted mapping grid based on problem information over the entire domain. RS is a global method, as opposed to a local approach in which a grid point is influenced by neighboring points only. RS is also an *a posteriori* method in that problem information usually comes from a previous solution. The basic strategy used by the RS method in order to determine the points in the mapping grid is to approximately equidistribute the integral of a given *density* function over the domain. A good choice for the density function is not always obvious, but it clearly should be a function which gives some indication of where the problem is difficult. A density using residuals, gradients, etc., from the previous solution is a common choice. A density-equidistribution heuristic such as ours is not uncommon in adaptive grid numerical methods. Piecewise polynomial approximation of functions is often done with adaptive knot selection methods of this type. In the case of spline approximation, deBoor [5] gives an explicit density function which must be equidistributed for optimal knot selection. Many adaptive methods for PDEs use density-balancing schemes as well. Flaherty et al. [7] attempt to balance an estimate of the local discretization error in solving parabolic PDEs. The truncation error is also used by Berger and Oliger [4] to adapt grids for hyperbolic problems. The moving finite element method [8] uses a density function that is essentially the residual in the approximate solution. The residual is used in the density functions reported in [2] and [9] as well. The TWODEPEP system [17] requires the user to supply a density function which is used to to refine triangular elements.

Basic step. The heart of RS grid adaption is an element subdivision step. This fundamental step is repeated again and again, in a recursive manner, until the mapping grid is determined. We describe a canonical instance of this basic step. Consider a general quadrilateral element E with corners a, b, c and d (see Figure 1(a)). The task is to split E into four subelements by choosing

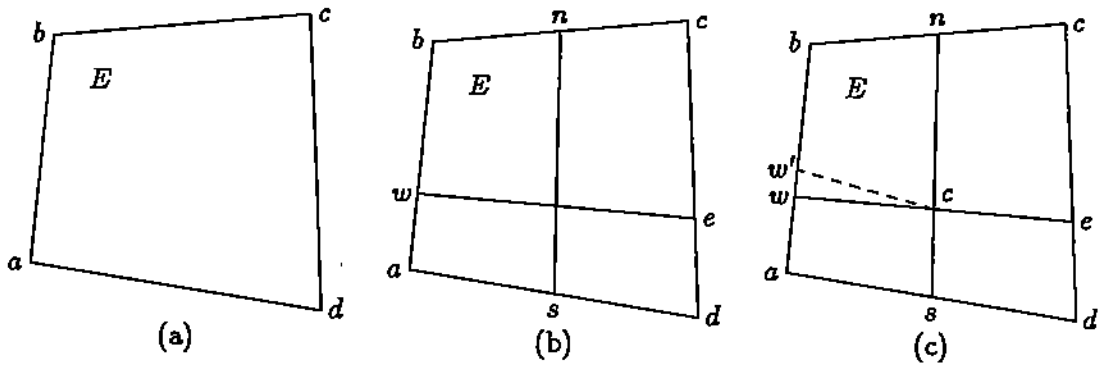

Figure 1: The basic element subdivision step of RS grid adaption.

five new grid points—one on each side of E and one in the interior. The points are chosen so as to approximately equidistribute the integral of the density function D over the four resulting subelements. We proceed in two main substeps:

1. Choose points n and s on the boundary of E (see Figure 1(b)) so that the integral of D is approximately equally distributed between the right and left half of E. Choose points e and w in a similar manner—attempting to balance the density between the top and bottom subelements of E.

2. Fix the point c where the two lines ns and ew intersect (see Figure 1(c)). Allow each of the new points n, s, e and w, to move along its respective side once, if doing so would improve the density distribution. For example, if the density in the northwest subelement is much larger than in the southwest, move w up proportionally.

The problem of selecting two points at once in substep 1 is posed so that a one-dimensional minimization technique may be used. When selecting points n and s we require the slope of the line ns to be the average of the slopes of the lines ab and cd; a similar constraint is imposed for the line we. We use a modified regula falsi algorithm (see [15, Chapter 8]) to find the line which approximately minimizes the difference in density between the left and right subelements. We impose constraints to prevent points from getting too close to one another.

The integral of the density function over a subelement is estimated by summing the density values at each of the points in a *density evaluation grid* which lie in the given subelement. The density evaluation grid is a uniform rectangular grid on the original domain R where we evaluate the density function during an initialization phase of the algorithm. By default the density evaluation grid is twice as fine as the mapping grid. The number of points in the density evaluation grid may be adjusted by a parameter *delta* if more accurate estimates of the integrals are desired. When estimating the integral over a subelement with this scheme, points near the subelement's

Fix the four corner points $(1,1),(m,1),(1,m),(m,m)$ (lower-left, lower-right, upper-left, upper-right respectively). Point (i,j) has coordinates (x_{ij}, y_{ij}).

Evaluate density function D on $n \times n$ grid, where $n = (m-1)/delta + 1$. Parameter *delta* has default value 0.5.

for $l = 1$ **to** *maxlevel* **do**

$k = 2^{maxlevel-l+1}$

for $i = 1$ **to** $m - k$ **step** k **do**

for $j = 1$ **to** $m - k$ **step** k **do**

if $l \leq maxdepth$ **then**

Set points $(i + k/2, j)$ and $(i + k/2, j + k)$ by regula falsi, subject to the constraint $x_{i,j} + dx \leq x_{i+k/2,j} \leq x_{i+k,j} - dx$, where $dx = (x_{i+k,j} - x_{i,j}) * gbound$. Parameter *gbound* has default value 0.25.

Set points $(i, j + k/2)$ and $(i + k, j + k/2)$ in a similar fashion.

Set point $(i + k/2, j + k/2)$ by finding intersection of two new lines.

Move points $(i + k/2, j)$, $(i + k, j + k/2)$, $(i + k/2, j + k)$ and $(i, j + k/2)$ if doing so improves the density distribution.

Adjust points $(i, j + k/2)$ and $(i + k/2, j)$ if neighboring elements have already chosen values for these points.

else

Set points $(i + k/2, j)$, $(i + k, j + k/2)$, $(i + k/2, j + k)$, $(i, j + k/2)$ and $(i + k/2, j + k/2)$ by bisecting the current element in both directions.

Figure 2: Simplified version of RS algorithm. This version assumes an $m \times m$ mapping grid is to be generated.

boundary are weighted so that the function appears continuous to the minimization routine. The approximate integration is heavily optimized; it takes advantage of as much previous work as possible in evaluating the sums. This is a fast but inexact way to estimate the integrals, and we find that it is satisfactory for most problems.

Complete algorithm. The complete RS algorithm consists of an initial evaluation of the density function at points in the density evaluation grid, followed by the application of the basic step—first to the entire domain, and then to smaller and smaller subelements, until a mapping grid of the desired dimension is defined. We give a simplified psuedocode description of the complete algorithm in Figure 2. The algorithm proceeds through a series of *maxlevel* levels, where

$$maxlevel = \log_2(\max(m_1, m_2) - 1),$$

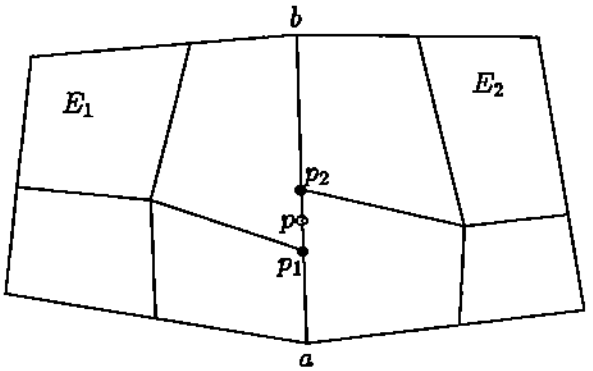

Figure 3: Coordination of element subdivision across interfaces. If the basic step on E_1 selects p_1 and the basic step on E_2 selects p_2, then the midpoint p is taken.

and m_1 and m_2 are the dimensions of the mapping grid to be constructed. In the present implementation each dimension of the $m_1 \times m_2$ mapping grid must by one greater than a power of two. This simplifies the logic of the recursive nature of the algorithm; it is not an essential constraint on the idea of the algorithm. In general, at level l, for $l = 1, 2, \ldots, \mathit{maxlevel}$, we perform the basic step on each of 4^{l-1} subelements. For example (if $m_1 = m_2$), at level 1 we subdivide a single element (the entire domain S) into four subelements. At level 2 each of these four subelements is in turn divided in four, and so forth. If $m_1 \neq m_2$ we divide elements into two subelements instead of four, until the number of levels left in each direction is equal. In our implementation there is a parameter *maxdepth* which indicates the maximum number of levels in which to apply the basic step. If more mapping grid points are needed after *maxdepth* levels, we simply continue by bisecting existing subelements. We find that this is a good approach, since the extra effort needed to select grid points carefully at the last levels, when the elements are relatively small, is often unwarranted. The first few levels are far and away the most important determiners of the shape of the final grid. In fact, this is one of the fundamental ideas behind the RS method—information from the whole domain strongly influences the position of each point.

After each application of the basic step, some of the new grid points may need to be adjusted so that grid lines are continuous across element boundaries. Figure 3 illustrates such a case with two neighboring elements E_1 and E_2. Points p_1 and p_2 are two candidates for the same grid point. The conflict is resolved by simply taking their midpoint along the line ab for the new point p.

Method variations. We have implemented three variations of the RS method. The simplest is essentially the one described above. The other two variations use more expensive quadrature methods to estimate the integral of the density function D over a subelement. In these method variations, at levels $2, \ldots, \mathit{maxlevel}$ the subelement on either side of a proposed grid line is divided into two triangles. A seven point, fifth degree quadrature formula from Stroud [18] is used on each

triangle. Thus, each evaluation of the minimization function requires $2 * 2 * 7 = 28$ evaluations of D, or some estimate of D. The only difference between the two more expensive variations is that one actually evaluates the density function D everywhere, while the other uses a bilinear interpolant of values on the density evaluation grid computed during the initialization phase. If the given density function is expensive to evaluate, the interpolation approach is considerably cheaper. Interpolation also serves to smooth the density function, which is often important since a badly behaved density function may cause severe difficulties for the minimization routine. For these more expensive variations of the RS method it proves worthwhile to use more quadrature points at level 1, when the half-elements are comparatively large and the choice of satisfactory grid lines is crucial to the overall success of the method. Consequently, at the first level each half-element is split into eight triangles. In this case $2*8*7 = 112$ function evaluations are needed.

One other difference among the three method variations is the minimization technique used to choose the new points. As mentioned above, we use a modified regula falsi algorithm in the simplest method variation. In this case the underlying function is continuous and relatively smooth. When the more expensive variations are used however, the underlying density function often appears nonsmooth and even discontinuous to the minimization algorithm. Consequently, a modified golden section algorithm is used for these two cases. It performs well, even if the density function is discontinuous. For either minimization method, we use relatively large stopping criteria, since the notion of an absolutely optimal location for a grid point is so poorly understood anyway. The RS method constructs good adapted mapping grids even though the density function is only approximately equidistributed. An exact equidistribution would be too expensive for our purposes. For similar reasons, we find that the simplest method variation described above is more than satisfactory for most problems. It is rarely worth the extra work to estimate the integral of the density function with great accuracy, since this does not seem to guarantee a significantly better grid. We illustrate the performance of the simple variation of the RS method in the examples of the next section.

3 Examples

Example 1. We illustrate the performance of the RS method on a problem with two boundary layers. Problem 9 in the population of elliptic PDEs given in [14] is defined by

$$\begin{aligned} u_{xx} + u_{yy} - 100u &= f(x,y) \qquad \text{on } R = [0,1] \times [0,1], \\ u &= g(x,y) \qquad \text{on } \partial R. \end{aligned}$$

The functions f and g are chosen so that the true solution u is known:

$$u(x,y) = \frac{1}{2}\left(\frac{\cosh 10x}{\cosh 10} + \frac{\cosh \alpha y}{\cosh \alpha}\right).$$

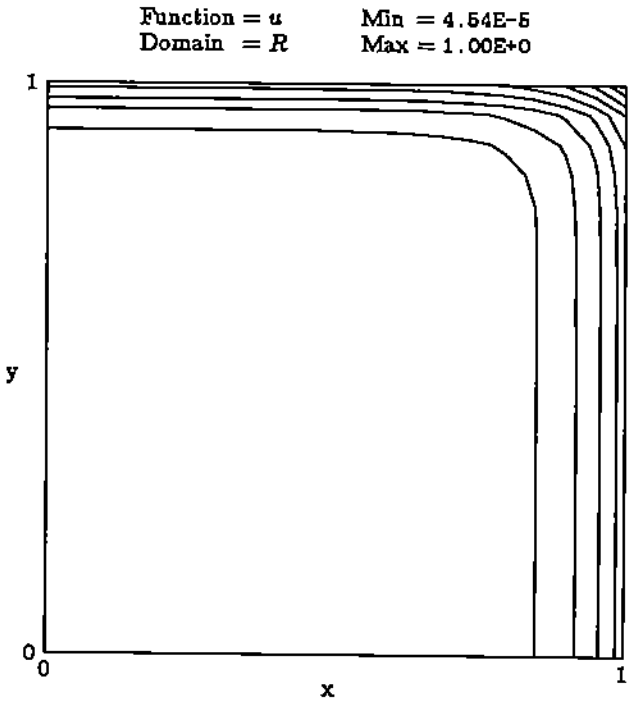

Figure 4: True solution for Example 1.

This problem is parameterized by a constant α which determines the strength of a boundary layer along the top side of the unit square. We set $\alpha = 20$. Figure 4 shows that the true solution u has a steep boundary layer along the line $y = 1$, and a less severe boundary layer along the line $x = 1$.

We solve this problem using the RS method as it is implemented in Multidomain ELLPACK [10]. Multidomain ELLPACK is an extension of the ELLPACK system for numerically solving elliptic PDEs (see [16]). All of the computation reported below is done in double precision on a Ridge 32 running ROS 3.3, with the FORTRAN compiler f77. In order to get initial information about the problem we use a 5×5 uniform grid and solve the PDE using ELLPACK modules INTERIOR COLLOCATION (collocation with Hermite bicubic basis functions) and BAND GE (band Gauss elimination with scaled partial pivoting). For a density function we select

$$D(x,y) = \sqrt{U_{xx}^2 + U_{yy}^2},$$

where U is the initial approximate solution. Intuitively, the justification for this choice of D is that a problem tends to be difficult where the second derivatives of the solution are large. Dwyer, et al. [6] use essentially this density function to adapt grid points in one space dimension. Figure 5 indicates that this function may indeed be a good predictor of the difficult areas of Example 1.

Applying RS with this choice for density function D, we get the mapping grid shown in Figure 6. The default value for each of the other RS parameters is used. The time taken by RS to adapt

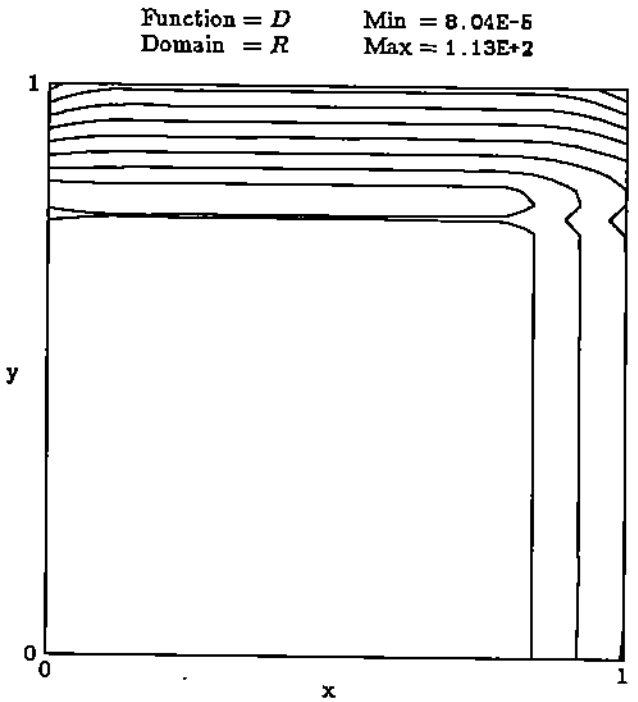

Figure 5: Density function $D = \sqrt{\nabla U}$ for Example 1 based on a solution using a 5×5 discretization grid.

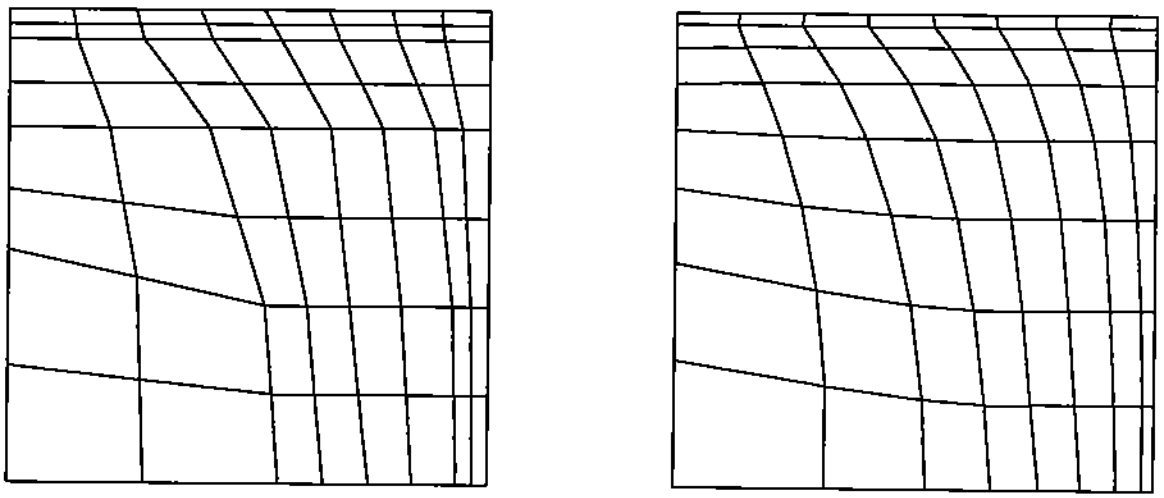

Figure 6: Mapping grid (left) and smoothed mapping grid (right) for Example 1.

the mapping grid and determine the coefficients of the mapping function F^{-1} is 1.2 seconds. The distribution of points in the mapping grid is not smooth, reflecting the non-uniform distribution of the density function and the fact that the RS method does not go to great lengths to generate a smooth grid. The second grid in Figure 6 shows the smoothing effect of the least squares approximation used to determine F^{-1}. This grid is the image of a 9×9 uniform grid in the transformed domain S, under the mapping F^{-1}. We re-solve the PDE using the domain mapping constructed by RS. Table 1 gives the data for several grid sizes. We list both time in seconds (total of discretization and linear system solution time) and error (estimated by taking the maximum absolute error over a fixed 40×40 grid in R) for solutions using a uniform grid on Domain R and using the adaptive mapping generated by RS to solve the transformed PDE in Domain S. In Figure 7 we give a log-log plot of error versus time for the two approaches. Using the adaptive grid domain mapping generated by the RS method reduces the error in the solution by over a factor of 10 for a given grid size. The time needed to solve the problem for a given grid size is slightly greater for the transformed PDE than for the original. This is to be expected since the coefficients of the transformed equation are more complicated. The extra cost is not that significant, however. For example, compare the time for a typical sequence:

$$0.93 \text{ (solve)} + 1.20 \text{ (adapt)} + 308.18 \text{ (re-solve)} = 310.31 \text{ seconds},$$

with the time using a 25×25 grid in the original domain (299.23 seconds). The RS method reduces the error by a factor of nearly 40 with an increase of less than 4% in time.

Example 2. We illustrate the performance of the RS method on a problem with two areas of difficulty—a boundary layer and a nearly singular point. Consider the elliptic equation

$$(1+x^2)u_{xx} + u_{yy} - \frac{1}{1+x}u_x - u = f$$

Table 1: Data for Example 1.

Grid	Solution in R		Solution in S	
	Time	Error	Time	Error
5×5	0.93	8.61E-2	1.34	7.38E-3
7×7	2.81	3.32E-2	3.65	1.49E-3
9×9	6.79	1.49E-2	8.06	6.43E-4
13×13	25.87	4.30E-3	28.47	1.19E-4
17×17	69.46	1.72E-3	73.91	2.43E-5
25×25	299.23	4.02E-4	308.18	1.01E-5

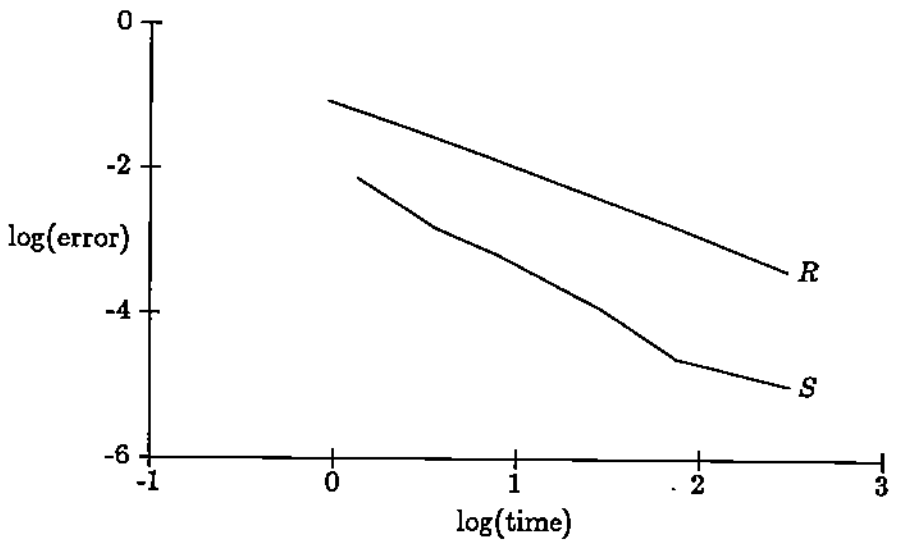

Figure 7: Log-log plot of error versus time taken to solve Example 1 in Domain R and Domain S.

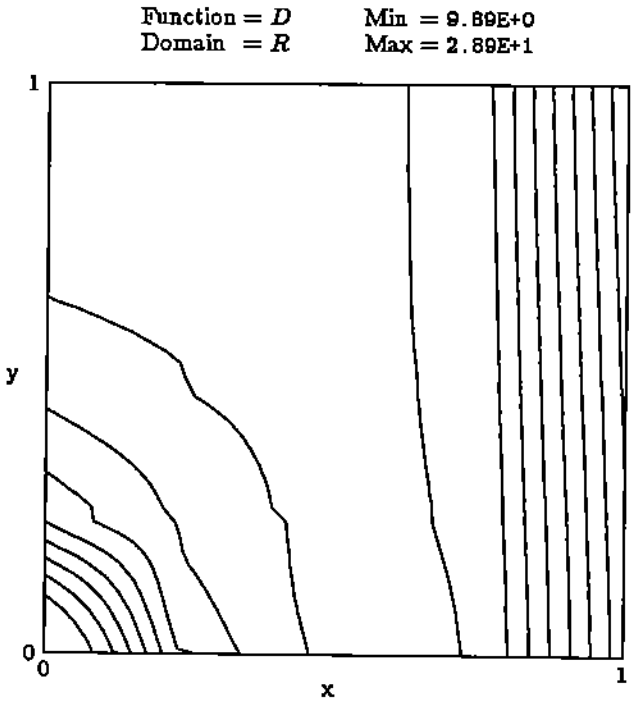

Figure 8: Density function $D = \sqrt{\nabla U}$ for Example 2 based on a solution using a 5×5 discretization grid.

on the unit square. The function f and Dirichlet boundary conditions are chosen as in [11] so that the true solution is

$$u(x,y) = smooth(x,y) + blayer(x,y) + singpt(x,y),$$

where

$$\begin{aligned} smooth(x,y) &= (\cos y + \sin(x-y))\,(1+\sin(x/2))\,, \\ blayer(x,y) &= e^{-5(1-x)}, \\ singpt(x,y) &= 5\left((x+.01)^2 + (y+.01)^2\right)^{.75}. \end{aligned}$$

This problem has a boundary layer along the right side and a near singularity at the lower left corner of Domain R. Proceeding in the usual way, we first solve the problem using a coarse uniform grid. ELLPACK modules INTERIOR COLLOCATION and BAND GE with a 5×5 uniform grid are used to compute an initial solution. We use the same density function $D = \sqrt{\nabla U}$ (see Figure 8). The mapping grid shown in Figure 9 is generated and the corresponding mapping function F^{-1} is constructed by RS in 1 second. Results from solving the original and transformed PDE with several different grid sizes are listed in Table 2 and plotted in Figure 10. The adaptive grid domain

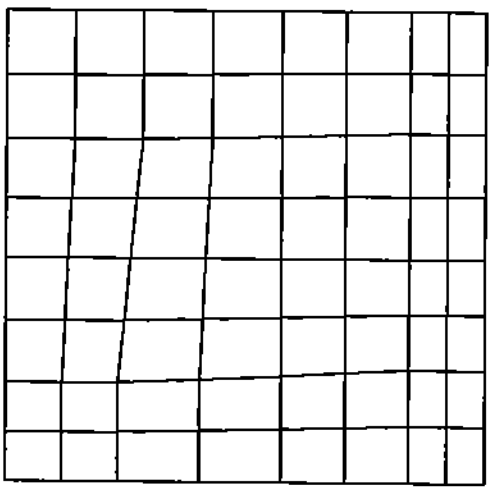

Figure 9: Mapping grid generated by RS for Example 2.

Table 2: Data for Example 2.

Grid	Solution in R		Solution in S	
	Time	Error	Time	Error
5×5	1.17	1.10E-2	1.62	7.25E-3
7×7	3.31	5.32E-3	4.14	1.92E-3
9×9	7.63	3.16E-3	8.97	1.11E-3
13×13	27.61	1.35E-3	30.74	5.36E-4
17×17	72.47	7.78E-4	77.98	2.71E-4
25×25	306.47	2.19E-4	317.42	3.89E-5

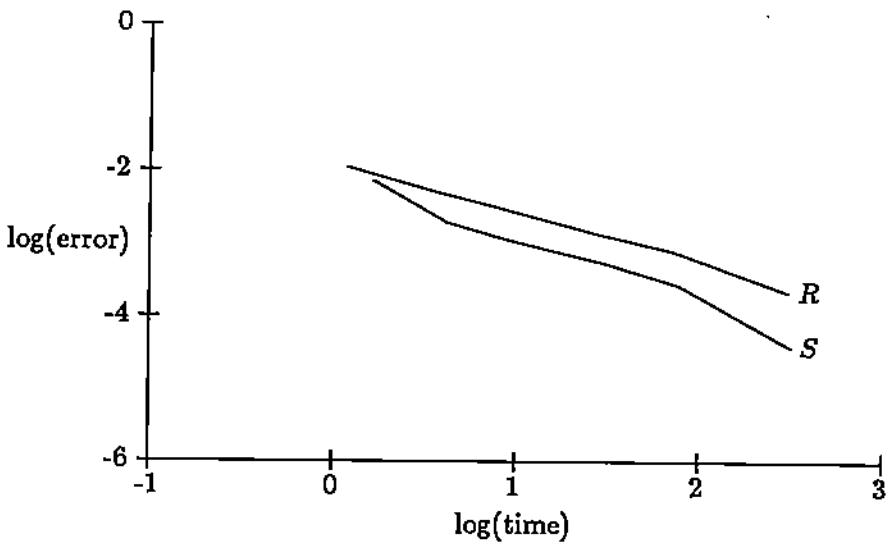

Figure 10: Log-log plot of error versus time taken to solve Example 2 in Domain R and Domain S.

mapping generated by RS again appears to be very effective. The total time to compute a solution using the finest adapted grid is

$$1.17 + 1.00 + 317.42 = 319.59 \text{ seconds.}$$

This compares favorably with the time using the non-adapted 25×25 grid (306.47 seconds). The error is reduced by a factor of 5.6 with an increase of about 4% in time. The plots in Figure 11 show how the error in the solution is spread out by solving the problem using the RS adaptive grid mapping.

4 Discussion

The purpose of the examples in the previous section is to demonstrate that the RS adaption method can quickly generate good adaptive grid domain mappings for difficult problems. The added costs introduced by the adaption process are more than compensated for by increased accuracy. These extra costs are not too significant compared to the cost of the entire computation, especially for fine grids. The time needed by the RS method to construct the mapping is quite small. The extra time to solve the transformed PDE is significant for coarse grids, but for finer grids its relative importance shrinks. This is not surprising since the time to solve the linear system is independent of whether adaption is used or not, and linear system solution time dominates as the grid is refined.

For problems with simple elliptic operators like Example 1, there are faster numerical methods than the ones we selected. These methods do not apply to the general operators that result from applying adaptive mappings. For this reason, our approach may not be as attractive for problems with simple operators.

We have not yet attempted to take advantage of the potential for parallelism which we believe exists within our framework. We are implementing grid adaption with domain mappings so that the regularity of uniform tensor product grids may still be exploited. The chances for significant parallelism seem good. This must be investigated next, since it is here that we expect our approach to have an advantage over local grid refinement methods.

In terms of the RS method itself, there are several issues which are still to be addressed. The choice of a density function is one aspect that is not well-understood. We have experimented with several density functions besides the one used in the examples in this paper. There are problems on which virtually any reasonable choice will yield a satisfactory mapping; on other problems the performance of the method is sensitive to the choice of density function. We plan to investigate this question further. The current implementation of the RS method has a few other parameters which must be specified. While the default values of these parameters are usually satisfactory, we would like to reduce further the likelihood of a user having to choose parameter values by trial and

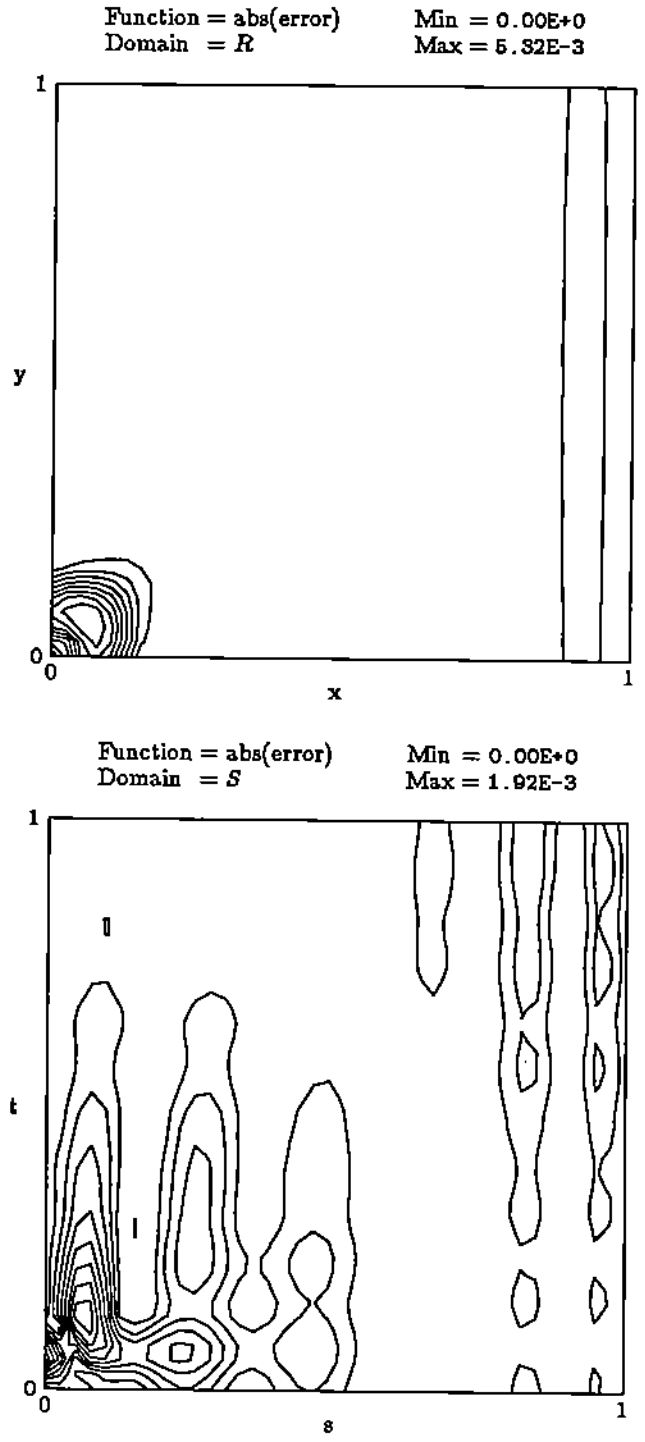

Figure 11: Error from solving the original PDE (top) and solving the transformed PDE (bottom) of Example 2, using a 7×7 discretization grid in both cases.

error. We plan to continue testing the RS method on a wide variety of difficult problems, to see if improvements can be made.

Finally, we are considering the possibility of applying a RS-like strategy at a different level of the numerical solution process. It may be that such a strategy is a good approach to partitioning a problem among many processors in a parallel computer. The present RS method is used to simply construct an adapted mapping grid. But we believe that RS could be used to partition a problem among processing elements (PEs) in such a way that each PE would have a computationally similar piece of the problem. We are planning to pursue this strategy in the near future.

References

[1] S. Adjerid and J. E. Flaherty, "Adaptive finite element methods for parabolic systems in one and two space dimensions", Technical Report 86-20, Rensselaer Polytechnic Institute, 1986.

[2] I. Babushka and W. C. Rheinboldt, "Error estimates for adaptive finite element computations", *SIAM J. Numer. Anal.*, 15(1978), pp 736–754.

[3] I. Babushka, J. E. Flaherty and J. Chandra (eds.), *Adaptive Computational Methods for PDEs*, SIAM, Philadelphia, 1983.

[4] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic PDEs", Technical Report NA-83-02, Stanford University, 1983.

[5] C. deBoor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[6] H. A. Dwyer, M. D. Smooke and R. J. Kee, "Adaptive gridding for finite difference solutions to heat and mass transfer problems", in *Numerical Grid Generation* (J. F. Thompson, ed.), North-Holland, New York, 1982, pp 339–356.

[7] J. E. Flaherty, M. Coyle, R. Ludwig and S. F. Davis, "Adaptive finite element methods for parabolic partial differential equations", in *Adaptive Computational Methods for PDEs* (I. Babushka, J. E. Flaherty and J. Chandra, eds.), SIAM, Philadelphia, 1983, pp 144–164.

[8] K. Miller and R. N. Miller, "Moving finite elements I", *SIAM Jnl. Num. Anal.*, 18(1981), pp 1019–1032.

[9] W. C. Rheinboldt, "On a theory of mesh-refinement processes", *SIAM J. Num. Anal.*, 17(1980), pp 766–778.

[10] C. J. Ribbens, "Domain mappings: a tool for the development of vector algorithms for numerical solutions of partial differential equations", Ph. D. Thesis, Department of Computer Sciences, Purdue University, 1986.

[11] C. J. Ribbens and J. R. Rice, "Realistic pde solutions for non-rectangular domains", Purdue University, Computer Science Department Report CSD-TR 639, 1986.

[12] C. J. Ribbens, "A priori grid adaption strategies for elliptic pdes", Purdue University, Computer Science Department Report CSD-TR 667, 1987.

[13] C. J. Ribbens, "A computational framework for constructing adaptive grid domain mappings", Purdue University, Computer Science Department Report CSD-TR 673, 1987.

[14] J. R. Rice, E. N. Houstis and W. R. Dyksen, "A population of linear, second order, elliptic partial differential equations on rectangular domains", *Math Comp.*, 36(1981), pp 475–484.

[15] J. R. Rice, *Numerical Methods, Software, and Analysis*, McGraw-Hill, New York, 1983.

[16] J. R. Rice and R. F. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, 1985.

[17] G. Sewell, "A finite element program with automatic user controlled mesh grading", in *Advances in Computer Methods for Partial Differential equations III* (R. Vichnevetsky and R.S. Stepleman, eds.), IMACS, New Brunswick, N. J., 1979, pp 8–10.

[18] A. H. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[19] J. F. Thompson, "A survey of dynamically-adaptive grids in the numerical solution of partial differential equations", AIAA Technical Report 84-1606, 1984.

[20] J. F. Thompson, Z. U. A. Warsi and C. W. Mastin, *Numerical Grid Generation*, North Holland, New York, 1985.