JOURNAL OF COMPUTATIONAL PHYSICS 27, 323–350 (1978)

Evaluation of Numerical Methods for Elliptic Partial Differential Equations

E. N. HOUSTIS, R. E. LYNCH, AND J. R. RICE

Computer Science Department, Purdue University, West Lafayette, Indiana 47907

AND

T. S. PAPATHEODOROU

Mathematics Department, Clarkson College of Technology, Potsdam, New York 13676

Received December 7, 1976; revised May 31, 1977

We systematically evaluate four methods for solving two-dimensional, linear elliptic partial differential equations on general domains. The four methods are: standard finite differences; collocation, Galerkin, and least squares using Hermite cubic piecewise polynomials. The collocation method is new in that it applies to general curved domains and we describe this aspect in detail. Our test set of 17 problems ranges from simple to moderately complex. The principal conclusion is that collocation is the most efficient method for general use. Standard finite differences is sometimes more efficient for very crude accuracy (where efficiency is not important anyway) but it is also sometimes enormously less efficient even for very modest accuracy. The accuracy of the Galerkin and least-squares methods is sometimes better than collocation, but the extra cost always negates this advantage for our problems.

1. STATEMENT OF THE PROBLEM AND PROCEDURES, CONCLUSIONS

Our approach to evaluating numerical methods for partial differential equations has already been outlined in Houstis *et al.* [11]. This approach is a specific instance of the general framework presented by Rice [15]. Briefly this approach is to first choose a sample set of problems from the domain of interest. The domain here is linear, second order elliptic partial differential equations which are somewhat "general." That is, they have various complications (variable coefficients, curved domains, reentrant corners, etc.) that are typical in applications and which prevent the straightforward use of specialized methods or theories. We specifically exclude such specialized methods from consideration even though they are applicable to some of the problems as we are concerned with evaluating methods with general applicability. One next selects some solution methods (four in this paper) and criteria of performance (accuracy achieved, execution time, and memory used) and finally one applies the methods to the sample set of problems while measuring the performance criteria.

The cost of solving partial differential equations forces a small sample set (17 problems here) and thus the reliability of the evaluations is not as high as we would like. Nevertheless, most of the phenomena observed here are quite consistent over the problem set which suggests that the probability of this being the result of chance is quite low.

One key to validity of an evaluation such as this is the precise definition of the problems, methods, and measures of performance. The sample problem set is presented in the next section. The numerical methods are discussed in Sections 2 and 3 along with a detailed synopsis of them. Our method of colloctaion for curved boundaries is new.

A common weakness of previous efforts of this type is the lack of precision and information about the numerical methods.

It is insufficient to simply state "Method X was used"; variations in the implementation of Method X affect the performance measures by factors of 2, 10, or 1000. We believe that we have implemented all the numerical methods in a way that gives close to maximum performance. We have particularly striven to be "fair" to each method and have not used special techniques (e.g.,assembly language code) for one in order to enhance its performance relative to the others.

We summarize our procedure and conclusions as follows:

Problem class. Second-order linear elliptic partial differential equations of general nature, i.e., some complication present in coefficients, domain, or solution.

Solution requirements. Moderate accuracy (1 to 3 digits correct) achievable "in core" with up to 300 unknowns (60,000 words or less of memory needed).

Four numerical methods. Standard finite differences; collocation, Galerkin and least squares using piecewise cubic polynomials (Hermite cubics).

Criteria of performance (efficiency). Execution time for a given relative accuracy (in the max norm).

Conclusions (in the limited context specified above). 1. There is normally a "crossover point" at low accuracy beyond which collocation is more efficient than standard finite differences. Even when finite differences is more efficient, it is by a small amount while collocation is sometimes dramatically more efficient than finite differences. Collocation is much superior for problems whose boundary conditions involved derivatives.

2. There is practically no difference at all between Galerkin and least squares in performance. They tend to be slightly more accurate than collocation but are very much less efficient because of the increased work to compute the coefficients in the matrix problem to be solved.

2. Comparison of Standard Finite Differences and Collocation with Hermite Cubics

2.1. *The Numerical Methods*

The first comparison made in this paper is between the standard finite difference method (5-point star) and collocation with Hermite cubics. See Appendix 2, Strang and Fix [17], and Collatz [5] for detailed information on these methods. Simply stated, in collocation the coefficients of the approximate solution are chosen to satisfy *exactly* the partial differential equation and boundary conditions at selected points.

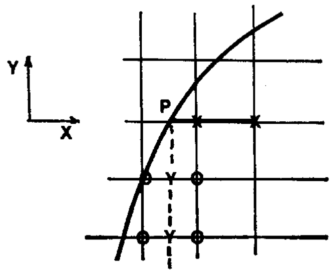
In simple situations with a uniform mesh length of h, the finite difference method is second order, $O(h^2)$ and collocation is fourth order, $O(h^4)$. Thus, asymptotically in these situations, as the accuracy increases, collocation becomes more efficient than standard finite differences. This suggests the definition of a *crossover point* in the performance, where collocation becomes more efficient. One of our objectives is to ascertain how collocation applies to more general problems and to determine the expected location of the crossover point.

Standard finite differences. This method has the following components.

(a) Grid: A rectangular grid is placed over the domain and all points in the domain or on its boundary are used. The grid is uniformly spaced except for Problems 16, 17, where the geometry made that undesirable.

(b) Approximation to the operator: The derivatives in the differential equation are replaced by simple central, 3-point finite difference approximations involving the grid points.

(c) Approximation to the boundary conditions: Derivatives in Neumann or



mixed boundary conditions are approximated as indicated by the diagram: x-derivative at P estimated from value at P and the two x-points; y-derivative at P estimated from value at P and the two y-points. The values at the y-points are found by linear interpolation from the o-points.

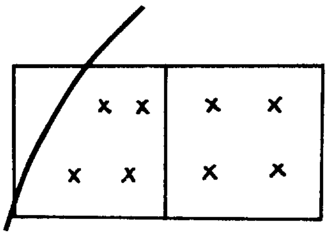
(d) Equation solution: The linear system is solved by Gauss elimination taking into account the zeros in the system (profile or frontal method).

Collocation. This method has the following components.

(a) Elements: A rectangular grid is placed over the domain. Rectangular elements whose center is not inside the domain are discarded. The grid is uniform except for Problems 16, 17.

(b) Approximation space: the Hermite bicubic polynomials.

(c) Approximation to the operator: The approximate solution satisfies the differential equation exactly at the four Gauss points of a rectangular element. For nonrectangular elements near the boundary the four Gauss points are projected inside the element as indicated by the diagram, where x = differential equation collocation points.



(d) Approximation to the boundary conditions: The boundary conditions are interpolated at a selected set of boundary points for either Dirichlet or Neumann boundary conditions. If the domain is a rectangle and the problem has Dirichlet conditions = 0 (Problems 1, 7, 8, 9, 10, and 15) then the Hermite bicubics are selected so as to automatically satisfy the boundary conditions and no boundary approximation equations are used. This is the same procedure as for the Galerkin and least-squares methods. The details on how the boundary collocation points are selected are given below.

(e) Equation solution: Same as for standard finite differences.

The most sensitive aspect of collocation is the placement of the boundary collocation points for nonrectangular domains. First, one must take care that these points are reasonably separated from interior collocation points. This is not difficult to do even in an automatic way, but the penalty for overlooking this point is an ill-conditioned computation with large errors.

One first overlays the region with a rectangular grid and discards the elements which intersect the domain slightly or not at all. Let S_b be the number of boundary sides of the resulting rectangular partition. Then the number of boundary collocation

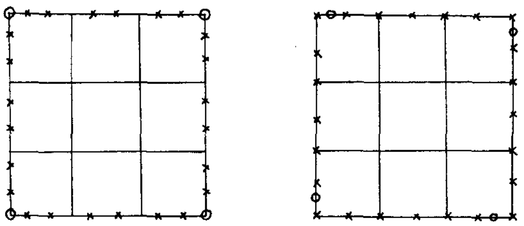


FIG. 1. Two schemes for distributing boundary collocation points. (Left: 2-point scheme; right: midpoint scheme.) The x's are the systematic collocation points and the 0's are the four extra ones.

points required is $2S_b + 4$. We use two basic schemes for distributing the boundary collocation points as illustrated by the diagrams for a simple rectangle (Fig. 1).

A theoretical analysis indicates that the 2-point scheme is superior for rectangular regions if the two points used are the Gauss points for each boundary segment. We compared the Gauss points with equally spaced points and found that the equally spaced points give slightly better accuracy for rectangular domains. We made numerous numerical experiments which confirmed that the 2-point scheme is superior for rectangular regions.

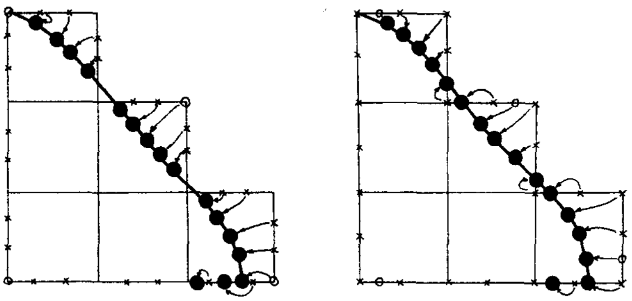


FIG. 2. The two schemes for a simple curved domain. (Left: 2-point scheme; right: midpoint scheme.) The lines show how the collocation points are placed on the edge of the rectangular partition and then mapped onto the portions of the boundary intersecting each rectangular element.

The extension of these two schemes to curved domains is illustrated in Fig. 2. The theoretical advantage of the 2-point scheme no longer holds for curved boundaries and our experiments confirm that it has no advantage over the midpoint scheme in this case. In fact it is, on the average, slightly less accurate. Furthermore, the midpoint scheme automatically gives collocation of the boundary conditions at any extremities of the domain (for example, for a piecewise rectangular boundary such as in Problems 16 and 17, see Fig. 5). *It is often essential that collocation of the boundary conditions be made at all exterior corners of the domain.* The midpoint scheme naturally provides this.

Our procedure is to use the 2-point scheme for boundaries which are straight (or nearly so) and parallel to a coordinate axis and to use the midpoint scheme

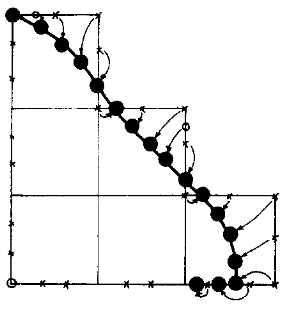


FIG. 3. The combination of the two schemes for a partially rectangular region. The mapping from the point on the rectangular edges to the curved boundary is indicated.

otherwise. The two schemes may be used together for a domain such as that shown above and we do this as shown in Fig. 3.

There seems to be no particularly advantageous method of distributing the four extra collocation points beyond putting them in elements with exterior corners and spreading them somewhat evenly around the boundary. We always map the midpoint-type collocation points to segments of the curved boundary which are interior to the rectangular partition. The points are placed uniformly on each such segment. At times this may leave rather large segments of a curved boundary "unused," but we have not found a reliable method of placing collocation points on the intermediate segments. We do place collocation points outside the rectangular partition for the 2-point scheme. An example is shown in Fig. 4 which illustrates these procedures.

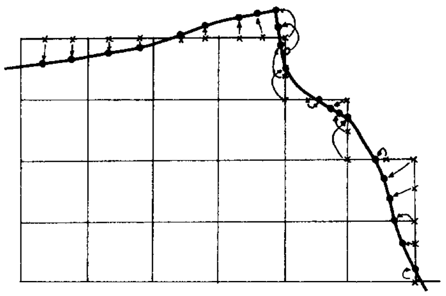


FIG. 4. Example which illustrates boundary collocation points for the 2-point scheme which are outside the rectangular partition and collocation for the midpoint scheme are inside. Collocation is not done on two large boundary segments.

2.2. *The Problem Set*

The operators, domains, boundary conditions, and true solutions for the 17 problems we used are given in Table I. The first eight were previously considered by us in Houstis *et al.* [11]. We give additional information about some of them:

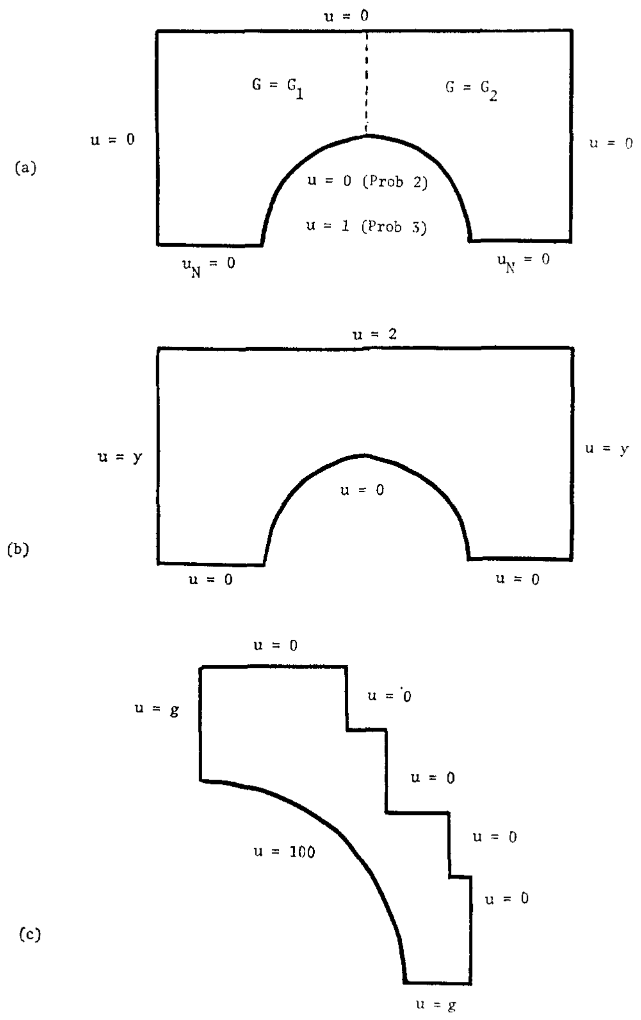


FIG. 5. The geometry and boundary conditions for problems 2, 3, 14, and 17. Problem 16 uses the geometry of (c) with the boundary condition $u = g$ everywhere.

Problem 2/3. Torsion in a bimetal shaft (Ely and Zienkiewicz [9]). The shear modulus G is a step function with $G_1/G_2 = 3$ (see Fig. 5a). We have replaced the step by a short interval (length = 1.E–5) where a cubic polynomial blends the two values of G smoothly. We measure accuracy here by comparing with a numerical solution we have computed which we believe is much more accurate than the ones considered in this paper.

Problem 4. The ellipse is centered at (0, 0) with major and minor axes of 2 and 1. By symmetry only the quarter of the elliptical region in the first quadrant was used in the computation.

TABLE I

The 17 Problem Space Sample Used in This Paper[a]

Problem	Partial differential equation operator true solution	Size of solution	Domain	Boundary conditions	References
1	$(e^{xy}u_x)_x + (e^{-xy}u_y)_y - \frac{u}{1+x+y} = f$ $u = e^{xy}\sin(\pi x)\sin(\pi y)$	1.3	Unit square	$u = 0$	[8]
2/3	$\left(\frac{1}{G}u_x\right)_x + \left(\frac{1}{G}u_y\right)_y = f$ with $f = -2$ or 0 u is unknown	0.87 or 0.8	See Fig. 5a	See Fig. 5a	[8]
4	$u_{xx} + u_{yy} = f$ $u = (e^x + e^y)/(1 + xy)$	7.6	Ellipse	$u = g$	[11]
5	$u_{xx} + u_{yy} = 0$ $u = \tan^{-1}(y/x)$	2.6	Circle	$u_N = g$	[5]
6	$u_{xx} + (1 + y^2)u_{yy} - u_x - (1 + y^2)u_y = f$ $u = e^{x+y} + (x^2 - x)^2\log(1 + y^2)$	7.4	Unit square	$u - u_N = 0$	[11]
7	$u_{xx} + u_{yy} = 6xye^xe^y(xy + x + y - 3)$ $u = 3e^xe^y(x - x^2)(y - y^2)$	0.58	Unit square	$u = 0$	[11,14]
8	$u_{xx} + u_{yy} = f$ $u = x^{5/2}y^{5/2} - xy^{5/2} - x^{5/2}y + xy$	0.1	Unit square	$u = 0$	[11]

9	$4u_{xx} + u_{yy} - 64u = f$ $u = 4(x^2 - x)(\cos(2\pi y) - 1)$	2.0	Unit square	$u = 0$	
10	$u_{xx} + u_{yy} - [100 + \cos(3\pi x) + \sin(2\pi y)]u = f$ $u = [5.4 - \cos(4\pi x)] \sin(\pi x)(y^2 - y)[5.4 - \cos(4\pi y)]$ $\times [1/(1 + \phi^4) - \frac{1}{2}]$ $\phi = 4(x - 0.5)^2 + 4(y - 0.5)^2$	3.2	Unit square	$u = 0$	[13]
11/12	$u_{xx} + u_{yy} - 100u = (\mu^2 - 100) \cosh y/\cosh \mu$ with $\mu = 10$ or 20 $u = \cosh 10x/\cosh 10 + \cosh \mu y/\cosh \mu$	2.0	Unit square	$u = g$	
13	$u_{xx} + u_{yy} = f$ $u = \phi(x) \times \phi(y)$ (see text)	1.0	Unit square	$u = g$	
14	$u_{xx} + u_{yy} = f$ $u = y[(x - 2)^2 + y^2 - 1]e^{-0.0625x(x-4)(y-2)}/[(3 + (x - 2)^2)(3 + y^2)]$	2.0	See Fig. 5b	See Fig. 5b	[6]
15	$u_{xx} + u_{yy} = f$ $u = 10\phi(x) \times \phi(y)$, $\phi(x) = e^{-100(x-0.5)^2}(x^2 - x)$	0.6	Unit square	$u = 0$	
16	$u_{xx} + u_{yy} = 2e^{x+y}$ $u = e^{x+y}$	4.9	See Fig. 5c	$u = g$	[19]
17	$u_{xx} + u_{yy} = f$ (see text and Appendix 2)	100.0	See Fig. 5c	See Fig. 5c	[19]

[a] The letters f and g denote functions whose values are determined to make the problem have the specified true solution. The references are to papers where the problem or a closely related one has been considered. u_N denotes the normal boundary derivative.

Problem 5. The circle has radius 0.5 and center at (0.5, 0.5). The solution is uniquely determined by imposing the additional condition $u(0.5, 0) = 1$.

Problem 8. The true solution has a discontinuity in the "2.5" derivative.

Problem 10. This is a version of a problem from stratospheric physics; see McDonald *et al.* [13].

Problem 11/12. These problems are of boundary layer type; the square is centered at the origin and has side 1. Symmetry was not used.

Problem 13. The product solution $\phi(x)\,\phi(y)$ has a steep slope (or wave front) along a right angle at the center of the domain. We have

$$\begin{aligned} \phi(x) &= 1, & x &\leqslant 0.35, \\ &= p(x), & 0.35 \leqslant x &\leqslant 0.65, \\ &= 0, & 0.65 &\leqslant x, \end{aligned}$$

where $p(x)$ is a quintic polynomial determined so that $\phi(x)$ has two continuous derivatives.

Problem 14. This problem is similar to that of steady flow past a sphere (Desai and Abel [6]). The true solution satisfies the same boundary conditions and has the same shape as the solution of the physical problem.

Problem 15. The solution has a sharp peak at the center of the square and it is very small for $(x - 0.5)^2 + (y - 0.5)^2 > 0.01$.

Problem 16/17. This problem is derived from that of heat flow in the concrete shield of a nuclear reactor (see Zienkiewicz and Cheung [19]). Problem 16 only has the geometry and operator of the real problem. The true solution of Problem 17 (see Appendix 2) is a complicated function which exhibits the same shape (including small singularities at the three reentrant corners) and satisfies the same boundary conditions (except along $x = 0$ and $y = 0$) as the solution of the physical problem.

Problems 1, 7, 8, 9, 13, and 15 are separable and all the operators except for those of Problem 6 are formally self-adjoint.

2.3. *Results of the Comparisons*

The data obtained are presented in two forms. In Appendix 1 we tabulate the accuracy achieved versus computer time used. For *both* methods the error is measured only at the nodes of the grid used. For most problems we have also measured the error at many more points in the domain and this sometimes gives a considerably different result. This is discussed in more detail in Section 4. We used a CDC 6500, whose long word length gives ample insulation from round-off errors in these calculations.

TABLE II

Tabulation of the Crossover Points for 17 Problems[a]

Problem	Digits (log(max error/solution size))	N_F Finite difference	N_C Collocation	$N_F^{1/2}/N_C$
1	1.8	5	2	1.12
2	3.0	13	4	0.90
3	1.5	12	3	1.15
4	3.0	12	4	0.87
5	1.9	6	2	1.22
6	0	1	1	1.00
7	1.8	5	1	2.23
8	4.0	5	2	1.12
9	3.0	9	4	0.75
10	1.1	8	3	0.94
11	2.2	13	6	0.60
12	1.3	9	4	0.75
13	1.3	15	5	0.77
14	3.6	17	5	0.82
15	1.2	15	4	0.97
16	4.1	16	4	1.00
17	1.8	20	6	0.75

[a] The accuracy (in digits) and numbers N_F and N_C of grid lines is given for the comparison of Standard Finite Difference and Collocation with Hermite Cubics.

In Table II we tabulate the crossover points for all 17 problems. This is expressed both in terms of accuracy measured in digits as log(max error/solution size) and the number N of subdivisions in each variable. For the nonrectangular regions we give an approximate "equivalent" value of N which would give about the same number of unknowns, if the region were rectangular.

We see from Table II that the crossover points range from 0 to 4 digits with 2 as a median value. One of the high crossover points comes from Problem 16, where high accuracy is obtained by very coarse meshes. Let N_F and N_C denote the values of N at the cross over point for finite differences and collocation, respectively. There is a fairly consistent pattern in the relationship of the values of N_F and N_C, namely, $N_F^{1/2}/N_C$ is about 1. The value of N_C is small (from 1 to 6 with 3 as median) for all cases.

Our results here differ in some cases from those published earlier (Houstis *et al.* [11]). The efficiency of both programs has been improved but their relative efficiency has not changed much. In our earlier paper we measured the error at many points over the entire domain (bilinear interpolation was used to extend the finite difference solutions). The few noticeable differences from the earlier data are due to this change in error measurement. We also previously gave data on memory usage as well as execution time. We have omitted memory data here as the crossover points for memory are somewhat the same as for execution time (this is true also for the new problems introduced in this paper).

We timed separately the formation and the solution of the linear equations. Both finite differences and collocation are very similar in the breakdown of execution time, as seen in Table III. The solution of the matrix equation was by profile Gauss elimination.

TABLE III

Sample Data on the Breakdown of Execution Time between Formation and Solution of the Linear Equations

	Time for linear system (sec)		Ratio of
	Formation	Solution	formation/total
Problem 1			
Collocation, $N = 4$	0.25	0.46	0.54
Finite differences, $N = 10$	0.25	0.56	0.50
Collocation, $N = 8$	1.0	4.5	0.22
Finite differences, $N = 17$	0.9	3.6	0.20
Problem 10			
Collocation, $N = 8$	1.4	4.4	0.24
Finite Differences, $N = 17$	1.2	3.4	0.26

2.4. *Conclusions*

A study of Table II and Appendix 1 shows that collocation becomes more efficient than standard finite differences at rather low accuracies and/or small values for N. Furthermore, when finite differences are more efficient, it is by a small margin, whereas collocation is often dramatically more efficient than finite differences. These results cover a reasonably broad range of two-dimensional linear elliptic problems and show that there is no reason from the point of view of efficiency to use the standard finite difference method for this class of problems.

It is also relevant to note that in practical problems one must almost always compute solutions to higher accuracy than is actually required. That is to say, the only reliable ways to be certain that one has an error of, say, 5 % (or less) involve computing a solution accurate to 1 % or better. This is especially the case for low accuracy requirements (e.g., 1–10 % error). Collocation is particularly superior for Problems 5 and 6, which involve derivative boundary conditions.

3. Comparison of Collocation, Galerkin, and Least Squares

3.1. *The Methods*

In all three of these methods we use Hermite cubic polynomials as approximations. More specific details are given below but there are two facts worth noting first. First, both the Galerkin and least-squares methods involve the evaluation of integrals and these have been estimated by using 9-point quadrature in each grid rectangle based on the tensor product of the 3-point Gauss rule. All the information from the equation must be evaluated at 9 points; this compares with 4 points needed for collocation in each element (grid rectangle).

Second, the Galerkin and least-squares methods were implemented only for the case where the boundary conditions can be exactly satisfied by choosing the Hermite cubic basis appropriately. This restriction makes them intrinsically less flexible and should give them an advantage over collocation whenever they are applicable. To offset this advantage we used the same Hermite cubic basis for collocation on those prəblems where all three methods are compared. In complex problems it can be very difficult (and tedious) to modify the original problem into one where the boundary conditions can be satisfied exactly by piecewise cubic polynomials.

Ritz–Galerkin and least squares. The components of these methods are:

(a) Elements: Same as for collocation.

(b) Approximation space: Same as for collocation.

(c) Approximation to the operator: In each element E of the partition we have the Galerkin equation

$$\sum_{i=1}^{16} \alpha_i \iint \{pD_xB_iD_xB_j + qD_yB_iD_yB_j + rB_iB_j\}\, dx\, dy = \iint_E fB_j\, dx\, dy,$$

where the operator L and the true solution U^* are defined by

$$LU^* = (pU_x{}^*)_x + (qU_y{}^*)_y + rU^* = f,$$

and

D_x, D_y = differentiation operators,
$B_i(x, y)$, $B_j(x, y)$ = the i and j elements of the Hermite bicubic basis,
α_i = coefficient of B_i in the approximate solution
(the index i refers to one element only).

The least-squares equation in each element is

$$\sum_{i=1}^{16} \alpha_i \iint_E L(B_i) \cdot L(B_j)\, dx\, dy = \iint_E fL(B_j)\, dx\, dy.$$

rule for rectangles (only rectangular domains were used with these methods).

(d) Approximation to the boundary conditions: The boundary conditions were exactly satisfied by the Hermite cubic basis for all problems (1, 7, 8, 9, 10, and 15) attempted with these methods.

(e) Equation solution: The local equations are assembled (by the direct stiffness method) to form the global matrix. This equaiton is solved by Cholesky decomposition for band matrices (profile method).

There are only six problems (1, 7, 8, 9, 10, and 15) where Galerkin and least squares could be applied, but the results are so consistent that this number seems sufficient to draw general conclusions.

3.2. *Results of the Comparisons*

Appendix 1 has data for these six problems and for all three methods. We see that there is rarely a significant difference between the Galerkin and least-squares methods. Table IV gives a sample of some additional typical data for comparing the collocation and Galerkin methods.

One sees from Table IV that collocation is always faster for equal accuracy. The advantage decreases as N increases. Note that each collocation equation has 16 nonzero terms while there are 36 nonzero coefficients for a Galerkin equation. The band widths of the coefficient matrices are essentially the same, but the collocation matrix has a much "narrower" profile on the average because the band width is variable. On the other hand, the Galerkin coefficient matrix is symmetric which allows for a 50% savings in the computation (this symmetry disappears for non-self-adjoint equations). Overall, we feel that the collocation equations can be solved more rapidly even for very large values of N. This view is supported by the data in Table IV. One also sees for a fixed set of elements (grid) that collocation is sometimes less accurate than Galerkin and never more accurate. However, the accuracy advantage of Galerkin never compensates for its speed disadvantage in these cases.

Note that Problem 10 involves fairly complicated functions in the differential operator and that this has a large negative effect for the Galerkin and least-squares methods.

3.3. *Conclusions*

We see that collocation is a more general method and that it is also more efficient than Galerkin or least squares. Collocation is more delicate for complicated regions but this is even more the case for Galerkin and, probably, least squares. Thus collocation is the method of choice among these three for the class of problems represented here.

TABLE IV

Selected Data Comparing Collocation and Galerkin for Six Problems

	Factors of			Time breakdown (sec)					
				Collocation			Galerkin		
Problem	Speed advantage for Collocation	Accuracy advantage for Galerkin	N	Matrix formation	Matrix solution	Error	Matrix formation	Matrix solution	Error
1	4 to 12	2 to 3	3	0.137	0.203	5.6×10^{-3}	2.15	0.218	2.4×10^{-3}
			7	0.792	2.961	1.8×10^{-4}	10.57	3.67	1.0×10^{-4}
7	3 to 6	2	4	0.159	0.477	2.8×10^{-4}	2.01	0.538	2.6×10^{-4}
			8	0.645	4.45	1.7×10^{-5}	8 1	5.85	1.7×10^{-5}
8	2.5 to 8	2.5 to 4	3	0.081	0.213	1.6×10^{-4}	1.12	0.21	6×10^{-5}
			8	0.633	4.33	4.8×10^{-6}	7.89	5.95	2×10^{-6}
9	3 to 7	1 to 4	2	0.034	0.053	5×10^{-2}	0.566	0.055	1.5×10^{-2}
			7	0.489	2.88	1.6×10^{-3}	6.88	3.71	8.6×10^{-4}
10	6 to 15	1 to 2	2	0.052	0.055	8.5×10^{-1}	1.98	0.059	8.6×10^{-1}
			9	1.71	6.66	7×10^{-3}	40.0	9.15	4×10^{-3}
15	5 to 10	1 to 7	4	0.239	0.483	3.4×10^{-1}	4.81	0.54	8×10^{-2}
			8	0.95	4.39	8×10^{-2}	18.8	5.82	1.1×10^{-2}

4. Some Observations

4.1. *Unequal Mesh Spacing for Collocation*

There are two disadvantages to collocation compared to standard finite differences: (1) It is not well known, (2) Its implementation is slightly more complicated, primarily because of the code for the basis functions. The extra complexity (which is not great) of collocation is compensated by its greater flexibility. For example, unequal mesh spacings can be used with no extra difficulty, no loss in accuracy, and a negligible increase in computation. By no loss of accuracy we mean that collocation remains a fourth-order method as contrasted to standard finite differences where unequal mesh spacing reduces the order from second to first. Similarly, boundary conditions involving derivatives cause only minor changes in the computation even for curved domains.

In fact, unequal mesh spacing can dramatically increase the accuracy of collocation solutions and often one can see (with little trouble) a reasonable mesh to use. Several examples of this occur among the 17 problems considered here, including Problem 13 (wave front on a right angle) and Problem 15 (sharp peak at center). We solved both of these problems with unequally spaced meshes and the resulting improvements are tabulated in Table V. The unequally spaced meshes for these examples were chosen in what seemed a plausible way, but no attempt was made to optimize the mesh.

TABLE V

Illustration of the Possible Improvement in Accuracy of the Collocation Method by Using an Unequally Spaced Mesh

	ERROR	
Case	Equally spaced mesh	Unequally spaced mesh
Problem 13		
$N = 6$	1.5×10^{-2}	1.8×10^{-3}
$N = 8$	7×10^{-2}	4.1×10^{-4}
Problem 15		
$N = 3$	0.57	0.29
$N = 6$	0.16	0.06
$N = 8$	0.08	0.026

4.2. *Additional Accuracy at the Mesh Nodes for Collocation*

For general collocation there is a phenomenon called superconvergence (see deBoor and Swartz [3]) where the order of accuracy at the mesh nodes is higher than elsewhere. However, in theory this phenomenon does not occur in using cubic polynomials. Nevertheless, we observed substantially improved accuracy at the nodes for some problems while there was none for some others. For two problems there was a constant increase in the accuracy at the nodes: a factor of 4 for Problem 7, and 15 for Problem 4. In some other problems (e.g., 8, 10, 11, and 13) there was a more erratic factor of increase, but it exceeded 4 in some case of each of these problems. No such phenomenon occurred for the least-squares or Galerkin methods.

There is a plausible explanation of this as follows: The nature of the theoretical error term for collocation is different at the mesh nodes than that at other points, but the use of cubic polynomials results in the same order of accuracy for both cases. However, for some problems the coefficient of the principal error term at the nodes might be significantly smaller than that of the general error term. This could account for the phenomenon that we observe.

4.3. *Dependence of Accuracy on the Nature of the Operator as well as the Solution*

It is obvious that the difficulty of obtaining a numerical solution of a partial differential equation depends on the nature of the differential operator as well as the nature of its solution. This fact may be overlooked, since theory places heavy emphasis on the nature of the solution. The effect of the operator, however, can be quite significant. For example, compare the widely varying results that are obtained for Problems 6, 7, and 16 whose solutions are nearly the same. On the other hand, Problems 1, 7, and 9 have very similar results, as one would guess from the fact that the differential operators and boundary conditions are similar in nature and all three have very well-behaved solutions. We have considered several sets of different problems which all have the same solution and have seen a very wide range of difficulty in obtaining the same function from problems with different operators.

4.4. *Comparison with Previous Work*

There has been little effort on systematic comparisons of different methods for solving partial differential equations; our previous paper (Houstis *et al.* [11]) was one of the first. There have been a number of abstract comparisons based on asymptotic rates of convergence and asymptotic operation counts for the solution of linear systems of equations. See Rice [14] and Birkhoff and Fix [2] for a large number of examples of this analysis and references to earlier work. Experience has shown that operation counts are reliable for estimating the efficiency of solving linear systems of equations. For iterative methods one must take extreme care to *terminate the iteration at a level compatible with the discretization error of the method.*

The usefulness of asymptotic rates of convergence as guides to the efficiency of numerical methods for elliptic problems is still open to question. Specifically, it is not known how reliable these rates are as guides for the moderate accuracy requirements of typical applications. Discussions of this question is given in the last section of Strang and Fix [17] (their asymptotic rates are reliable guides for three example problems), in Birkhoff and Fix [2], and in Swartz [18] where several different order methods are compared.

Roache [16] has a section entitled "Remarks on Evaluating Methods" (pp. 109–112) which cites 12 papers. He favors simple, low-order methods but none of these papers attempts a controlled comparison of methods.

Eason [7] has a bibliography of 241 items relevant to the least-squares method. He cites 26 papers where collocation and 14 papers where Galerkin are compared with least squares. We did not locate any systematic and realistic evaluation of methods among these 40 references.

If there is any consistent pattern in the results, it would be that authors find that the collocation of boundary conditions is delicate. Many find that least-squares approximations to the boundary conditions give better results, primarily because they do not use good boundary collocation points. This does suggest that collocation of the differential equation combined with least squares for the boundary conditions would give a more robust numerical method with little penalty in efficiency. We are [illegible]

plate bending problems: A simply supported elliptic plate and a square plate supported at four "random" points. The nine methods are compared on the basis of 11 criteria, e.g., "suitability for programming," "applicability to general regions," "ease in learning." Efficiency and accuracy were not included directly as criteria and apparently were not systematically measured. It is important to note that all of the nine methods considered were of limited flexibility and none could be applied to all 17 problems included in this study.

APPENDIX 1: The Comparison Data for 17 Problems

The data for the comparison of methods are tabulated in Table AIII with the grid spacing "N" versus the accuracy achieved and execution time. The *error* given is the actual error at the location of the maximum error. The execution *time* is in seconds on a CDC 6500. The "N" given is N_F or N_C (see Section 2.3). If the grid is square, this is the number of grid lines in one direction; otherwise it is the number of grid lines in a square which would have approximately the same number of unknowns.

The results are presented more compactly in two smaller tables. Table AI presents ratios of computation required to achieve a desired accuracy. The number of significant digits is tabulated versus the 17 problems and the entries are the ratios of computer time for collocation over time for finite differences. The ratios are estimated from the data in Table AIII and are quite rough. Many blanks exist because only very limited extrapolation was used. The pattern is quite clear: Collocation and finite differences start out at about the same efficiency and then collocation becomes more efficient, usually quite rapidly and by a large factor.

One may crudely estimate the "time order" α of these methods by measuring the slopes of the curves of error vs time when plotted on log–log paper. The order α estimated is for the relationship

$$\text{Error} = O(\text{Time}^{-\alpha}).$$

TABLE AI

Ratios of Computation Time for Collocation over Finite Differences for Specified Accuracy Measured in Number of Current Significant Digits

	Digits					
Problem	1	2	3	4	5	6
1	1	1/5	1/90			
2		1	1			
3	1	1/15				
4			1.6/1	1/10		
5	4/1	1/8	1/150			
6		1	1/120			
7		1.5/1	1/60			
8		1	1/30	1/50		
9		1.5/1	1/4	1/75		
10	2/1	1/7	1/40			
11		1.5/1	1/6			
12		1/2				
13		1				
14			1	1/5		
15		2/1	3/1	1/8		
16				10/1	1	1/4
17	1/1.2	1/1.2				

TABLE A II

Measured Slopes α to Estimate the Order of the Methods from Their Actual Performance

	Finite diff.		Collocation		Galerkin			Finite diff.		Collocation		Galerkin	
Problem	α	4α	α	4α	α	4α	Problem	α	4α	α	4α	α	4α
1	0.65	2.6	1.44	5.8	1.9	7.6	9	0.58	2.3	1.5	6.0	1.4	5.7
2	1.13	4.5	2.4	9.6	—		10	0.53	2.1	1.15	4.6	?	
3	0.94	3.8	1.7	6.8	—		11	0.54	2.2	1.06	4.2	—	
4	0.59	2.4	1.37	5.5	—		12	0.38	1.5	0.68	2.7	—	
5	0.47	1.9	4.0	16.0	—		13	0.67	2.7	?		—	
6	0.55	2.2	1.46	5.8	—		14	0.73	2.9	1.5	6.0	—	
7	0.61	2.4	1.39	5.6	2.0	6.2	15	0.85	3.4	1.19	4.8	1.2	4.8
8	0.58	2.3	0.67	2.7	1.5	6.1	16	1.44	5.8	2.34	9.4	—	
							17	1.05	4.2	1.05	4.2	—	

TABLE A III

The Data for the Comparisons[a]

				Problem 1				
F.D.								
N_F	5	7	10	13	15	17		
$10^4 \times$ Error	373.0	189.0	92.7	55.5	41.6	32.3		
Time	0.109	0.286	0.812	1.843	2.932	4.532		
Col.								
N_C	2	3	4	5	6	7	8	9
$10^4 \times$ Error	318.0	56.4	17.9	8.5	3.1	1.8	1.1	
Time	0.110	0.340	0.714	1.328	2.301	3.753	5.521	
Gal.								
$10^4 \times$ Error	96.5	24.5	9.71	5.02	2.11	1.03	0.69	0.404
Time	1.56	2.9	4.85	7.6	11.24	15.82	21.84	29.05
L.S.								
$10^4 \times$ Error	222.0	41.2	14.3	6.78	2.58	1.45	0.92	0.533
Time	1.624	3.091	5.189	8.239	12.03	16.97	23.13	30.51
				Problem 2				
F.D.								
N_F		9	12	15	16	19		
$10^4 \times$ Error	469.0	145.0	41.3	36.8	7.35	5.7		
Time	0.121	0.424	0.78	1.94	2.73	5.7		
Col.								
N_C	8	10	13	14				
$10^4 \times$ Error	70.8	21.7	4.82	3.14				
Time	0.7	1.71	3.33	4.03				
				Problem 3				
F.D.								
N_F	6	8	12	15	17			
$10^2 \times$ Error	11.2	13.9	3.24	1.88	1.53			
Time	0.11	0.412	0.77	1.92	2.71			
Col.								
N_C	8	10	13	14				
$10^2 \times$ Error	4.87	0.772	0.223	0.196				
Time	0.72	1.69	3.39	4.17				

[a] The abbreviations F.D., Col., Gal., and L.S. are used for finite differences, collocation, Galerkin, and least squares, respectively.

Table continued

TABLE A III—*Continued*

			Problem 4				
F.D.							
N_F	4	8	12	16	17		
$10^3 \times$ Error	23.1	7.47	3.62	2.07	1.64		
Time	0.06	0.341	1.33	2.97	4.32		
Col.							
N_C	5	7	12	15			
$10^3 \times$ Error	14.2	7.8	0.328	0.22			
Time	0.191	0.68	3.34	6.23			
			Problem 5				
F.D.							
N_F	4	8	16	18			
$10^3 \times$ Error	132.7	59.81	19.19	14.13			
Time	0.144	0.992	8.554	12.69			
Col.							
N_C	2	3	4				
$10^3 \times$ Error	345.0	9.67	5.66				
Time	0.422	1.02	2.138				
			Problem 6				
F.D.							
N_F	5	7	10	13	15	17	
$10^4 \times$ Error	1470.	842.0	447.0	276.0	211.0	167.0	
Time	0.205	0.583	1.768	4.194	6.827	10.83	
Col.							
N_C	2	3	4	5	6	7	8
$10^4 \times$ Error	71.5	4.35	2.81	1.1	0.542	0.290	0.169
Time	0.394	1.018	2.018	3.841	6.190	9.906	14.971
			Problem 7				
F.D.							
N_F	5	7	10	13	15	17	
$10^4 \times$ Error	110.0	51.9	27.8	16.3	12.2	9.65	
Time	0.09	0.239	0.724	1.658	2.675	4.128	

Table continued

TABLE A III—*Continued*

				Problem 7 (*continued*)				
Col.								
N_C	2	3	4	5	6	7	8	
$10^4 \times$ Error		4.48	1.35	0.5	0.279	0.149	0.0863	
Time		0.286	0.638	1.220	2.109	3.337	5.097	
Gal.								
$10^4 \times$ Error	30.5	9.02	2.99	1.31	0.693	0.391	0.229	
Time	0.563	1.346	2.545	4.253	6.578	9.685	13.95	
L.S.								
$10^4 \times$ Error	40.84	9.46	2.73	1.15	0.570	0.304	0.178	
Time	0.623	1.447	2.754	4.590	6.976	10.39	14.56	
				Problem 8				
F.D.								
N_F	5	7	10	13	15	17		
$10^4 \times$ Error	5.9	3.19	1.68	1.04	0.797	0.631		
Time	0.092	0.240	0.721	1.662	2.662	4.119		
Col.								
N_C	2	3	4	5	6	7	8	9
$10^4 \times$ Error	0.75	0.32	0.20	0.14	0.0969	0.0710	0.0540	
Time	0.096	0.293	0.605	1.196	2.058	3.314	4.965	
Gal.								
$10^4 \times$ Error		0.598	0.292	0.185	0.0882	0.0395	0.0205	0.0205
Time		1.335	2.51	4.24	6.57	10.70	13.84	19.01
L.S.								
$10^4 \times$ Error	4.67	1.497	0.840	0.512	0.2813	0.1420	0.0555	0.0144
Time	6.15	1.454	2.731	4.572	7.043	10.36	14.64	20.15
				Problem 9				
F.D.								
N_F	7	10	13	15	17			
$10^4 \times$ Error	169.	93.9	53.6	40.6	31.8			
Time	0.234	0.717	1.58	2.57	3.902			
Col.								
N_C	2	3	4	5	6	7	8	9
$10^4 \times$ Error	515.0	305.0	78.9	42.1	19.8	10.4	3.96	
Time	0.089	0.275	0.620	1.203	2.082	3.368	5.051	

Table continued

TABLE A III—*Continued*

				Problem 9 (*continued*)				
Gal.								
$10^4 \times$ Error	149.0	92.9	40.5	19.7	11.6	6.19	4.25	2.49
Time	0.598	1.378	2.627	4.374	6.840	10.24	14.28	19.60
L.S.								
$10^4 \times$ Error	148.0	182.0	59.2	31.8	14.7	8.62	4.64	3.19
Time	0.621	1.483	2.806	4.760	7.230	10.59	14.98	20.30
				Problem 10				
F.D.								
N_F	5	7	10	13	15	17		
$10^2 \times$ Error	24.5	28.1	12.9	7.57	6.14	4.56		
Time	0.128	0.328	0.924	2.052	3.113	4.685		
Col.								
N_C	2	3	4	5	6	7	8	9
$10^2 \times$ Error	85.0	21.0	11.0	3.3	2.7		1.25	0.688
Time	0.139	0.381	0.800	1.44	2.49		5.773	8.379
Gal.								
$10^2 \times$ Error	81.0		0.540	0.478	0.840	0.313	0.660	0.450
Time	2.036		8.398	13.54	19.81	27.66	37.22	49.13
L.S.								
$10^2 \times$ Error	132.0	22.2	8.60	0.654	1.39	1.59	0.804	
Time	2.039	4.668	7.933	13.78	20.20	28.22	37.79	
				Problem 11				
F.D.								
N_F	5	7	10	13	15	17		
$10^2 \times$ Error	6.27	4.06	2.07	1.28	0.955	0.805		
Time	0.097	0.238	0.708	1.62	2.55	3.92		
Col.								
N_C	2	4	5	6	7	8		
$10^2 \times$ Error	6.0	0.917	0.523	0.321	0.207	0.14		
Time	0.398	2.02	3.72	6.20	9.71	14.54		

Table continued

TABLE A III—*Continued*

Problem 12							
F.D.							
N_F	6	8	11	14	16	18	
$10^2 \times$ Error	11.31	11.19	8.28	5.68	4.43	3.48	
Time	0.109	0.261	0.777	1.716	2.796	4.181	
Col.							
N_C	3	4	5	6	8		
$10^2 \times$ Error	5.55	3.73	2.53	1.76	0.917		
Time	1.00	2.03	3.71	6.27	14.88		
Problem 13							
F.D.							
N_F	5	7	10	13	15	17	19
$10^2 \times$ Error	89.1	32.5	19.7	11.7	4.71	6.48	4.35
Time	0.106	0.261	0.767	1.73	2.75	4.36	6.31
Col.							
N_F	3	4	5	6	7	8	
$10^2 \times$ Error	53.4	11.3	0.99	1.51	5.99	7.03	
$10^2 \times$ Error[b]				0.177		0.0413	
Time	1.	2.082	3.8	6.297	9.916	15.03	
Problem 14							
F.D.							
N_F	6	8	11	13	15	16	18
$10^4 \times$ Error	36.9	35.3	10.4	9.26	5.96	4.98	4.30
Time	0.123	0.447	0.860	1.128	2.051	2.851	4.515
Col.							
N_C	7	10	13	15			
$10^4 \times$ Error	23.7	9.31	2.31	1.14			
Time	0.654	1.789	3.546	6.393			
Problem 15							
F.D.							
N_F	5	7	10	13	15	17	19
$10^2 \times$ Error	125.0	45.8	25.6	5.38	4.87	4.20	3.59
Time	0.112	0.278	0.80	1.84	2.92	4.42	6.70

[b] Collocation, nonuniform mesh.

Table continued

TABLE A III—*Continued*

Problem 15 (*continued*)

Col.								
N_C	2	3	4	5	6	7	8	9
$10^2 \times$ Error	230.0	57.1	33.8	32.0	15.9	10.3	8.16	1.49
Time	0.126	0.359	0.722	1.337	2.258	3.534	5.338	7.93
$10^2 \times$ Error[b]		29.0	30.0	9.1	6.16	3.8	2.65	
Time[b]		0.325	0.700	1.323	2.262	3.588	5.518	
Gal.								
$10^2 \times$ Error	26.3	36.7	7.58	6.65	6.35	2.03	1.13	
Time	1.246	2.879	5.348	8.622	12.93	18.21	24.59	
L.S.								
$10^2 \times$ Error	22.3	36.3	9.14	6.11	6.43	1.80	1.08	
Time	1.329	2.992	5.540	8.932	13.24	18.94	25.56	

Problem 16

F.D.					
N_F	6	12	14	17	19
$10^5 \times$ Error	57.07	14.90	2.38	1.37	1.10
Time	0.081	1.779	2.176	4.145	6.153

Col.				
N_C	8	10	11	16
$10^5 \times$ Error	10.67	6.41	2.60	0.29
Time	1.176	2.377	3.027	7.871

Problem 17

F.D.					
N_F	6	9	12	14	19
Error	19.39	24.46	7.39	2.85	1.60
Time	0.233	0.854	2.702	3.265	7.899

Col.						
N_C	8	11	15	16	17	18
Error	7.46	3.90	1.48	0.98	1.13	0.97
Time	1.588	3.585	8.793	9.153	10.05	11.58

If one assumes that most of the computer time is spent in solving the linear systems, then one would have

$$\text{Error} = O(N^{-4\alpha}).$$

This assumption is clearly not satisfied here. In Table AII we present our estimates of α and 4α. We see that there is some correlation with the simple model which gives $4\alpha = 2$ for finite differences and $4\alpha = 4$ for the Hermite cubic method. There are also some very wide deviations from this.

APPENDIX 2: The Solution of Problem 17

To describe the exact solution u of Problem 17, we set

$$u(x, y) = 100g(x, y, \theta, 0, 0)/g(x, y, a, b, c),$$

where, by construction, the numerator on the right is zero on the stair-step outer boundary of the domain (see Fig. 5). The numerator is the product of $(x - 1)$, $(y - 1)$, and three factors of the form $r_i^{2/3} \sin(3(\theta_i + \pi/2)/2)$, where r_i is the distance between (x, y) and the reentrant corner (x_i, y_i), $i = 1, 2, 3$. The denominator is a modification of the numerator, which is positive in a region containing the boundary of the heat shield and which is equal to the numerator along the circular part of the boundary. Note that this function has the correct singularities at the reentrant corners. Specifically:

$$g(x, y, a, b, c) = [(x - 1)(y - 1) + aC(x, y)] \prod_{i=1}^{3} T(x, y, x_i, y_i, b, c),$$

$$C(x, y) = (x^2 + y^2 - 0.64)^2,$$

$$\begin{aligned} T(x, y, x_i, y_i, b, c) &= R(x, y, x_i, y_i, b)\, S(x, y, x_i, y_i, c), \\ R(x, y, x_i, y_i, b) &= [(x - s_i)^2 + (y - y_i)^2 + bC(x, y)]^{1/3}, \\ S(x, y, X_i, y_i, c) &= \sin(2[\arctan([y - y_i]/[x - x_i]) + \pi/2]/3) + cC(x, y) \\ &\qquad \text{with branch cut along } y - y_i = x - x_i,\ x_i < x. \end{aligned}$$

After some experimentation, we found that $a = -0.5$, $b = 0.1$, $c = 7.0$ gives a solution u which is similar to the solution one expects for the temperature in the heat shield.

Remark about the evaluation of u and $f = \nabla^2 u$. In our first attempt at the construction of a suitable u, we used a symbolic differentiator to obtain function subroutines to evaluate u_{xx} and u_{yy}. The resulting programs for u_{xx} and u_{yy} are very complicated. We note that u, u_{xx}, u_{yy} can each be evaluated by successive calls to a number of very simple subroutines. Each of these evaluates V, V_{xx}, V_{yy}, where V is a product $V = WZ$. Schematically the program is (where WX denotes W_x, etc.)

$$\begin{aligned} W &= \cdots, \\ WX &= \cdots, \\ WXX &= \cdots, \\ Z &= \cdots, \\ ZX &= \cdots, \\ ZXX &= \cdots, \\ V &= W \times Z, \\ VZ &= WX \times Z + W \times ZX, \\ VXX &= WXX \times Z + 2.0 \times WX \times ZX + W \times ZXX, \end{aligned}$$

and similarly for the y-derivatives. The values of V, VX, VXX, VY, VYY are stored in a common block for use by subsequent routines. In most cases, statements like the first six above: $W = \cdots$, $\cdots$ $ZXX = \cdots$, do not appear since the values are already computed by previously called subroutines. The program can be quickly written and debugged.

References

1. G. Birkhoff, "The Numerical Solution of Elliptic Equations," SIAM Regional Conf. Ser. Appl. Math. Vol. 1, SIAM, Philadelphia, 1971.
2. G. Birkhoff and G. Fix, (1974), "Higher Order Finite Element Methods," AEC and ONR Report.
3. C. W. de Boor and B. K. Swartz, *SIAM J. Numer. Anal.* **10** (1973), 582–606.
4. S. Z. Burstein and A. A. Mirin, *J. Computational Phys.* **5** (1970), 547–571.
5. L. Collatz, "The Numerical Treatment of Differential Equations," 3rd ed., Springer–Verlag, New York, 1966.
6. C. S. Desai and J. F. Abel, "Introduction to the Finite Element Methods," pp. 423–426, Van Nostrand, New York, 1972.
7. E. D. Eason, *Int. J. Numer. Meth. Eng.* **10** (1976), 1021–1046.
8. S. C. Eisenstat and M. H. Schultz, *in* "Complexity of Sequential and Parallel Numerical Algorithms" (J. F. Traub, Ed.), pp. 271–282, Academic Press, New York, 1973.
9. J. F. Ely and O. C. Zienkiewicz, *Int. J. Mech. Sci.* **1** (1960), 356–365.
10. A. Grammeltvedt, *Mon. Weather Rev.* **97** (1969), 384–403.
11. E. N. Houstis, R. E. Lynch, T. S. Papatheodorou, and J. R. Rice, *Ann. Assoc. Inter. Calcul. Analog*, **11** (1975), 98–103.
12. A. W. Leissa, W. E. Clausen, L. E. Hulbert and A. T. Hopper, *AIAA J.* **7** (1969), 920–928.
13. B. E. McDonald, T. P. Coffey, S. Ossakow, and R. N. Sudan, *J. Geophy. Res.* **79** (1974), 2551–2554.
14. J. R. Rice, SIGNUM Newsletter, 1976.
15. J. R. Rice, *in* "Advances in Computers" (M. Rubinoff and M. C. Yovits, Eds.), Vol. 15, Academic Press, New York, 1976.

16. P. J. ROACHE, "Computational Fluid Dynamics," Hermosa, Albuquerque, N. Mex. 1972.
17. G. STRANG AND G. J. FIX, "An Analysis of the Finite Element Method," Prentice-Hall, New York, 1973.
18. B. K. SWARTZ, *in* "Mathematical Aspects of Finite Elements in Partial Differential Equations" (C. de Boor, Ed.), pp. 279–312, Academic Press, New York, 1974.
19. O. C. ZIENKIEWICZ AND Y. K. CHEUNG, *Engineer* **6** (1965), 507–510.