1976

Evaluation of Numerical Methods for Elliptic Partial Differential Equations

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

Robert E. Lynch
Purdue University, rel@cs.purdue.edu

T. S. Papatheodorou

J. R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
76-204

EVALUATION OF NUMERICAL METHODS FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

E.N. Houstis, R.E. Lynch, T.S. Papatheodorou and J.R. Rice
Computer Science Department
Purdue University
West Lafayette, Indiana 47907

CSD TR 204
October, 1976

EVALUATION OF NUMERICAL METHODS FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

E.N. Houstis, R.E. Lynch, T.S. Papatheodorou and J.R. Rice
Computer Science Department
Purdue University

CSD TR 204
October, 1976

CONTENTS

ABSTRACT

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. 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.

EVALUATION OF NUMERICAL METHODS FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

I. 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 [1975]. This approach is a specific instance of the general framework presented by Rice [1976a]. 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. 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 briefly discussed in Sections II and III and a more detailed synopsis of them is given in Appendix 2.

A common weakness of previous efforts of this type is the lack of precision and information about the numerical methods. It is well known that 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 used 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" (60,000 words or less of memory needed).

4 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 accuracy. Accuracy is the maximum error divided by the size of the solution and is usually measured in decimal digits.

Conclusions:

1. There is normally a "cross-over 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.

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.

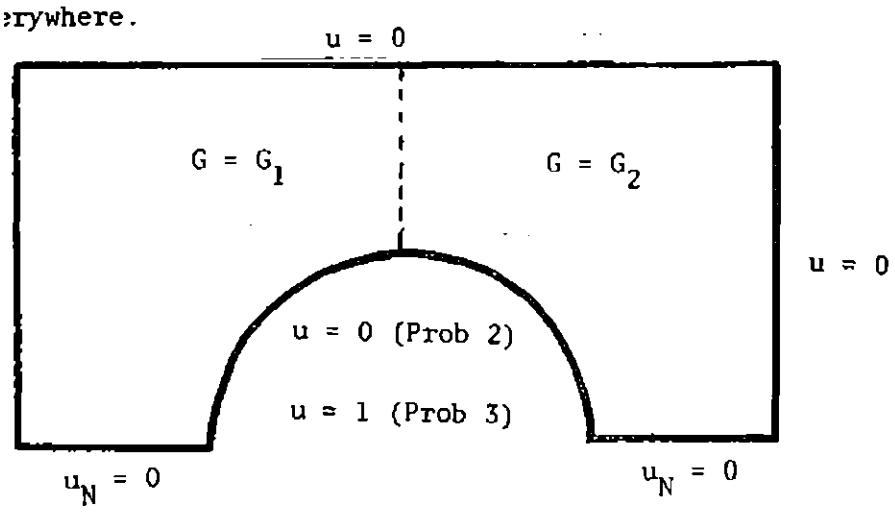
II. COMPARISON OF STANDARD FINITE DIFFERENCES AND COLLOCATION WITH HERMITE CUBICS.

II.1 The Numerical Methods and Problem Set. 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, Fix and Strang [1973] and Collatz [1966] 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.

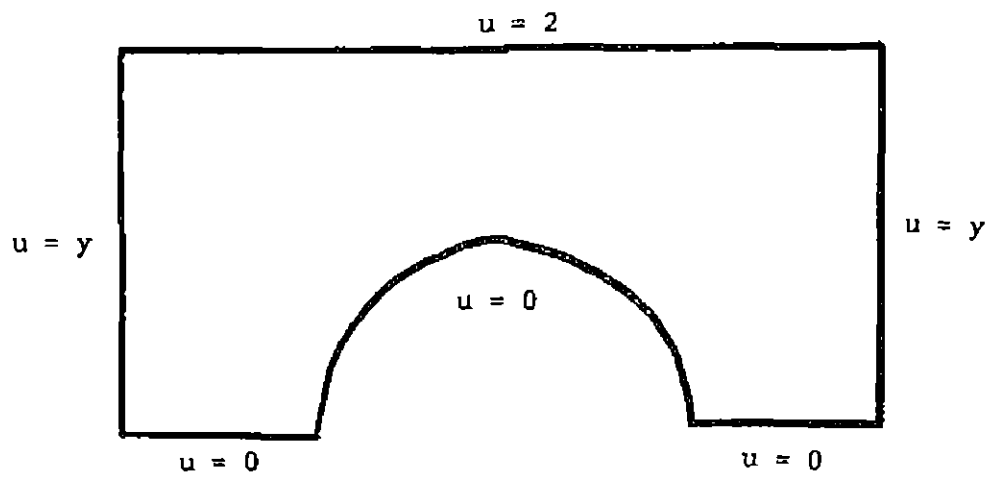
In simple situations with a uniform mesh length of h, the finite difference method is second order, $0(h^2)$ and collocation is fourth order, $0(h^4)$. Thus, asymptotically in these situations, as the accuracy increases, collocation becomes more efficient than standard finite differences. This suggests the existence of a cross-over point in the performance where collocation becomes more efficient. One of our objectives is to ascertain whether simple collocation applies to more general problems and to determine the expected location of the cross-over point. The operators, domains, boundary conditions and true solutions for the 17 problems we used are given in Table 1. The first 8 were previously considered by us in Houstis et al, [1975]. We give additional information about some of them:

Prob. 2/3. Torsion in a bimetal shaft, Ely and Zienkiewicz [1960]. The shear modulus G is a step function with $G_1/G_2 = 3$ (see Figure 1a). We have replaced the step by a short interval (length = 0.001) where a cubic polynomial blends the two values of G smoothly. We measure accuracy

e geometry and boundary conditions for problems 2, 3, 14 and 17.
oblem 16 uses the geometry of (c) with the boundary condition u = g
erywhere.



(b)



(c)

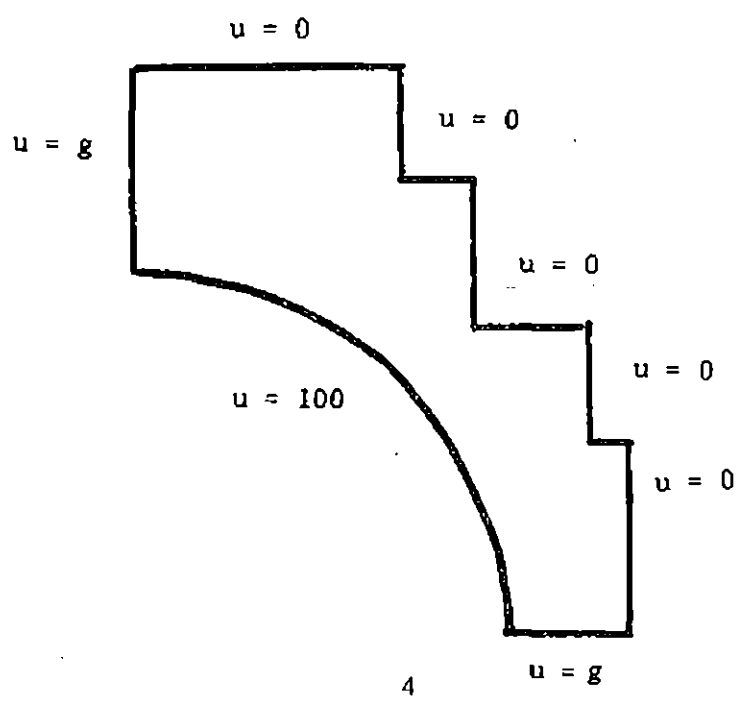


Table 1. The 17 problem space sample used in this paper. The letters f and g denote functions whose values are determined to make the problem have the specified true solution. The references are to papers where the problem or a closely related one has been considered.

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	[9]
2/3	$(\frac{1}{G} u_x)_x + (\frac{1}{G} u_y)_y = f$ with $f = -2\theta$ or 0 u is unknown ,	0.87 or 0.8	See Fig. 1a	See Fig. 1a	[7] [9]
4	$u_{xx} + u_{yy} = f$ $u = (e^x+e^y)/(1+xy)$,	7.6	Ellipse	u=g	[9]
5	$u_{xx} + u_{yy} = 0$ $u = \tan^{-1}(y/x)$,	2.6	Circle	u_N=g	[9]
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	$u-u_N=0$	[9]
7	$u_{xx} + u_{yy} = - 6xye^x e^y (xy + x + y - 3)$ $u = 3e^x e^y (x - x^2)(y - y^2)$,	0.58	Unit Square	u=0	[9] [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	[9]

Table 1 Continued

Problem	Partial Differential Equation Operator True Solution	Size of Solution	Domain	Boundary Conditions	References
9	$4u_{xx} + u_{yy} - 64\,u = f$ $u = 4(x^2-x)(\cos(2\pi y) - 1)$,	2.0	Unit Square	u=0	
10	$u_{xx} + u_{yy} - [100 + \cos(3\pi x)+\sin(2\pi y)]u = f$ $u = [5.4-\cos(4\pi x)]\sin(\pi x)(y^2-y)[5.4-\cos(4\pi y)]$ $*[1/(1+\phi^4)-1/2]$ $\phi = 4(x-.5)^2 + 4(y-.5)^2$,	3.2	Unit Square	u=0	[10]
11/12	$u_{xx} + u_{yy} - 100u = (\mu^2-100)\cosh y / \cosh \mu$ with μ=10 or 20 $u = \cosh 10x/\cosh 10 + \cosh \mu y/\cosh \mu$,	2.0	Unit Square	u=g	
13	$u_{xx} + u_{yy} = f$ $u = \phi(x) * \phi(y)$, see text ,	1.0	Unit Square	u=g	
14	$u_{xx} + u_{yy} = f$ $u = y[(x-2)^2+y^2-1]e^{-.0625x(x-4)(y-2)}/[(3+(x-2)^2)(3+y^2)]$,	2.0	See Fig. 1b	See Fig. 1b	[6]
15	$u_{xx} + u_{yy} = f$ $u = 10\ \phi(x)*\phi(y)$, $\phi(x)=e^{-100(x-.5)^2}\ (x^2-x)$,	0.6	Unit Square	u=0	
16	$u_{xx} + u_{yy} = 2e^{x+y}$ $u = e^{x+y}$	4.9	See Fig. 1c	u=g	[17]
17	$u_{xx} + u_{yy} = f$ see text and Appendix 4	100.0	See Fig. 1c	See Fig. 1c	[17]

here by comparing with a numerical solution we have computed which we believe is much more accurate than the ones considered in this paper.

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

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

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

Prob. 10. This is a version of a problem from stratospheric physics, see McDonald et al [1974].

Prob. 11/12. These problems are of boundary layer type; the square is centered at the origin and has side 2. Symmetry was not used.

Prob. 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

$$\phi(x) = \begin{cases} 1 & x \leq .35 \\ p(x) & .35 \leq x \leq .65 \\ 0 & .65 \leq x \end{cases}$$

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

Prob. 14. This problem is similar to that of steady flow past a sphere, Desai and Abel [1972]. The true solution satisfies the same boundary conditions and has the same shape as the solution of the physical problem.

Prob. 15. The solution has a sharp peak at the center of the square and it is very small for $(x-.5)^2+(y-.5) > .01$.

Prob. 16/17. This problem is derived from that of heat flow in the concrete shield of a nuclear reactor, see Zienkiewicz and Cheung [1965].

Problem 16 only has the geometry and operator of the real problem. The true solution of Problem 17 (see Appendix 4) 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 Prob. 6 are formally self-adjoint.

II.2 Results of the Comparisons. The data obtained are presented in two forms. In Appendix 1 we give a set of 17 graphs of 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 IV. We used a CDC 6500 whose long word length gives ample insulation from round-off errors in these calculations.

In Table 2 we tabulate the cross-over 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 non-rectangular 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 2 that the cross-over points range from 0 to 4 digits with 2 as a median value. One of the high cross-over 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 $\sqrt{N_F}/N_C$ is about 1. The value of N_C is small (from 1 to 6 with 3 as median) for all cases.

Table 2. Tabulation of the cross-over points for 17 problems. 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.

Problem	Digits = log(max error/solution size)	N_F Finite Difference	N_C Collocation	$\frac{\sqrt{N_F}}{N_C}$
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

Our results here differ in some cases from those published earlier, Houstis et al [1975]. 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 cross-over 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 3.

Table 3. Sample data on the breakdown of execution time between formation and solution of the linear equations.

		Time for linear system		Ratio of
		Formation	Solution	Formation/Total
Prob. 1.	Collocation, N=4	0.25 sec	0.46 sec	.54
	Finite Differences, N=10	0.25	0.56	.50
	Collocation, N=8	1.0	4.5	.22
	Finite Differences, N=17	0.9	3.6	.20
Prob. 10	Collocation, N=8	1.4	4.4	.24
	Finite Differences, N=17	1.2	3.4	.26

The solution of the matrix equation was always by Gauss elimination (frontal or profile version) and it is possible that iterative methods or nested dissection would be significantly more efficient. Indeed, this is known to be true for certain simple problems and finite differences. However, we are concerned with problems with some complexity (even though we included some simple examples in our sample) and there the theoretical relationship between iterative methods and Gauss elimination is unknown. Iterative methods also normally involve choosing one or more parameters and that could be very delicate for complex problems. Thus we must leave the question of the impact of using iterative methods on these problems as an open question for future research. The few comparisons that we are aware of have various defects that leaves the situation inconclusive in our minds.

II.3 Conclusions. A study of Table 2 and the graphs in 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 methods 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 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).

III. COMPARISON OF COLLOCATION, GALERKIN AND LEAST SQUARES.

III.1 The Methods. In all three of these methods we use Hermite cubic polynomials as approximations. More specific details are given in Appendix 2 but there are two facts worth noting here. 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 chosing 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 problems 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.

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.

III.2 Results of the Comparisons. The graphs given in Appendix 1 for these six problems show the data for all three methods. An examination of these graphs shows that there is rarely a significant difference between the Galerkin and Least Squares method. Table 4 gives a sample of some additional typical data for comparing the collocation and Galerkin methods.

One sees from Table 4 that collocation is always faster for equal accuracy. The advantage decreases as N increases and an operations count shows that eventually the Galerkin method is faster. This is because eventually most of the time is spent in solving the linear system and the Galerkin system is symmetric and hence can be solved twice as fast as the nonsymmetric collocation system. The timing data given in Table 4 is compatible with an operations count analysis for these two methods. One also sees for a fixed set of elements (grid) that collocation is sometimes much less accurate than Galerkin and never more accurate. However, the graphs show that the accuracy advantage of Galerkin never compensates for its speed disadvantage in these cases. One may compare accuracy from the graphs by noting that the last point plotted for each method has the same number of elements.

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.

III.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 to apply because the boundary collocation points must be selected carefully for complicated regions. See Appendix 3. Thus collocation

is the method of choice among these three for the class of problems represented here.

Table 4. Selected data comparing collocation and Galerkin for six problems. Times are given in seconds.

	Factors of			Time Break Down					
				Collocation			Galerkin		
Prob No	Speed Advantage for Coll.	Accuracy Advantage for Galer.	N	Matrix Formation	Matrix Sol.	Error	Matrix Formation	Matrix Sol.	Error
1	4 to 12	2 to 3	3	.137	.203	$5.6*10^{-3}$	2.15	.218	$2.4*10^{-3}$
			7	.792	2.961	$1.8*10^{-4}$	10.57	3.67	$1.0*10^{-4}$
7	3 to 6	2	4	.159	.477	$2.8*10^{-4}$	2.01	.538	$2.6*10^{-4}$
			8	.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	.081	.213	$1.6*10^{-4}$	1.12	.21	$6*10^{-5}$
			8	.633	4.33	$4.8*10^{-6}$	7.89	5.95	$2*10^{-6}$
9	3 to 7	1 to 4	2	.034	.053	$5*10^{-2}$	.566	.055	$1.5*10^{-2}$
			7	.489	2.88	$1.6*10^{-3}$	6.88	3.71	$8.6*10^{-4}$
10	6 to 15	1 to 2	2	.052	.055	$8.5*10^{-1}$	1.98	.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	.239	.482	$3.4*10^{-1}$	4.81	.54	$8*10^{-2}$
			8	.95	4.39	$8*10^{-2}$	18.8	5.82	$1.1*10^{-2}$

IV. THREE OBSERVATIONS.

IV.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 more complicated. The extra complexity (which is not great) of collocation partially stems from its greater flexibility. One manifestation of this is that 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 finitie differences where unequal mesh spacing reduces the order from second to first.

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 Prob. 13 (wave front on a right angle) and Prob. 15 (sharp peak at center). We solved both of these problems with unequally spaced meshes and the resulting improvements are tabulated in Table 5. The unequally spaced meshes for these examples were chosen in what seemed a plausible way, but no systematic attempt was made to optimize the mesh.

Table 5. Illustration of the possible improvement in accuracy of the collocation method by using an unequally spaced mesh.

Case		ERROR Equally Spaced Mesh	Unequally Spaced Mesh
Prob 13 ,	N=6	$1.5*10^{-2}$	$1.8*10^{-3}$
	N=8	$7*10^{-2}$	$4.1*10^{-4}$
Prob 15	N=3	.57	.29
	N=6	.16	.06
	N=8	.08	.026

IV.2 Additional Accuracy at the Mesh Nodes for Collocation. For general collocation there is a phenomenon called super convergence, see deBoor and Swartz [1974] where the order of accuracy at the mesh nodes is higher than elsewhere. However, in theory this phenomenon does not occur when 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 Prob 7 and 15 for Prob 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 occured 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.

IV.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 as the theory plans 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.

V. 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, 1975] 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, 1976] and [Birkhoff and Fix,1971] 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. This point is commonly overlooked and invalidates some otherwise interesting comparison studies.

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 [1973] (there asymptotic rates are reliable guides for 3 example problems), in Birkhoff and Fix [1974] and in Swartz [1974] where several different order methods are compared.

Roache [1972] has a section entitled "Remarks on Evaluating Methods" (pp. 109-112) and he strongly favors simple, low order methods and describes the performance of higher order methods as "disappointing". He supports the conclusions with citations of 12 papers, half of which have no relevant material on the question of the performance or comparison of methods. Most of those papers which involve shock wave and turbulence computations suggest that low order methods are the best of the methods used. However,

we (and some of the authors) interpret these papers' results on smoother problems differently than Roache. One paper explicitly states that first order methods compare poorly and a third order method gives "striking" improvement in accuracy with no more computation for some shock wave problems [Burstein and Mirin, 1970]. A comparison of methods for weather prediction by [Grammeltvedt, 1969] suggests to us that fourth order methods may be superior, but Roache states the opposite. None of these papers attempts a controlled comparison of methods and thus no definitive conclusions can be reached from them.

Eason [1976] has a bibliography of 241 items relevant to the least squares method for partial differential equations. He tabulates the references in various ways including Table III. Comparisons where least-squares methods are superior in accuracy, convenience or computing speed and Table IV. Comparisons where least-squares methods produce equivalent or comparable results. Eason is a strong advocate of the least squares method which may explain why a table where least squares does worse is not included. For example, Table III has 26 entries for collocation and 14 for Galerkin. We have examined most of these references and they are, in general, one of two types. First, someone attempts to solve a problem, say, with collocation using 12 polynomial terms and with least squares using 8 trigonometric polynomial terms. The problem has an unknown solution so the actual accuracy is unknown. The author reports his subjective evaluation of the quality of the results obtained. Usually there is insufficient data about the calculation to attempt to reproduce the results. Note that the differences observed are primarily due to using polynomials versus trigonometric polynomials rather than using collocation versus least squares. The second type of paper is more systematic, but involves trivial problems in one way or another (i.e. either the problem is trivial or the method used

is trivial). For example, one sees solutions of three fairly simple problems by five methods which compute a quadratic polynomial approximation. Then general conclusions are stated. We did not locate any systematic and realistic evaluation of methods among these 40 references. Most papers do not even give conclusive evidence in the particular context of the problem they consider.

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 or no penalty in efficiency.

Leissa et al [1969] present a systematic study of the value of 9 methods for two plate bending problems: a simply supported elliptic plate and a square plate supported a 4 "random" points. In both cases the "exact" solution is a series expansion truncated at 48 terms, but the authors do not view this as just another numerical method which might give worse results than some of the other methods they apply. 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 9 methods considered were of limited flexibility and none could be applied to all 17 problems included in this study.

REFERENCES

1. Birkhoff, G. [1971], The Numerical Solution of Elliptic Equations, SIAM, Regional Conf. Ser. Appl. Math., 1, Philadelphia.

2. Birkhoff, G. and Fix, G. [1974], Higher Order Finite Element Methods, AEC and ONR Report, 33 pages.

3. de Boor, C. W. and Swartz, B. K. [1973], Collocation at Gaussian Points, SIAM J. Numer. Anal., 10, pp. 582-606.

4. Burstein, S. Z. and Mirin, A. A. [1970], Third Order Difference Methods for Hyperbolic Equations, J. Comp. Physics, 5, pp. 547-571.

5. Collatz, L. [1966], The Numerical Treatment of Differential Equations, 3rd Edition, Springer Verlag, New York.

6. Desai, C. S. and Abel, J. F. [1972], Introduction to the Finite Element Methods, Van Nostrand, New York, pp. 423-426.

7. Eason, E. D. [1976], A review of least-squares methods for solving partial differential equations. Int. J. Numer. Meth. Engin. 10 pp. 1021-1046.

8. Eisenstat, S. C. and M. H. Schultz [1973], Complexity of partial differential equations, in "Complexity of Sequential and Parallel Numerical Algorithms" (J. F. Traub, ed.) Academic Press, pp. 271-282.

9. Ely, J. F. and Zienkiewicz, O. C. [1960], Torsion of Compound Bars - a Relaxation Solution, Int. J. Mech. Sci., 1, pp. 356-365.

10. Grammeltvedt, A. [1969], A Survey of Finite-Difference Schemes for the Primitive Equations for a Barotropic Fluid, Monthly Weather Review, 97, pp. 384-403.

11. Houstis, E. N., Lynch, R. E., Papatheodorou, T. S. and Rice, J. R. [1975], Development, Evaluation and Selection of Methods for Elliptic Partial Differential Equations, Ann. Assoc. Inter. Calcul Analog., 11, pp. 98-103.

12. Leissa, A. W., Clausen, W. E., Hulbert, L. E. and Hopper, A. T. [1969], A comparison of approximate methods for the solution of plate bending problems. AIAA Journal, 7 pp. 920-928.

13. McDonald, B. E., Coffey, T. P., Ossakow, S. and Sudan, R. N. [1974], Preliminary Report of Numerical Simulation of Type 2 Irregularities in the Equatorial Electrojet, J. Geophysical Res., 79, pp. 2551-2554.

14. Rice, J. R. [1976], Algorithmic Progress in Solving Partial Differential Equations, SIGNUM Newsletter.

15. Rice, J. R. [1976a], The Algorithm Selection Problem, in Advances in Computers, Vol. 15 (Rubicoff and Yovits, eds.), Academic Press, New York.

16. Roache, P. J. [1972], Computational Fluid Dynamics, Hermosa Publishers, Albuquerque, N. Mexico.

17. Strang, G. and Fix, G. J. [1973], An Analysis of the Finite Element Methods, Prentice Hall, New York.

18. Swartz, B. K. [1974], The Construction and Comparison of Finite Difference Analogs of Some Finite Element Schemes, in Mathematical Aspects of Finite Elements in Partial Differential Equations (C. de Boor, ed.) Academic Press, New York, pp. 279-312.

19. Zienkiewicz, O. C. and Cheung, Y. K. [1965], Finite Elements in the Solution of Field Problems, The Engineer, 6, pp. 507-510.

APPENDIX ONE

GRAPHS OF THE COMPARISON DATA FOR 17 PROBLEMS

The data for the comparison of methods is plotted on log-log paper with accuracy achieved versus execution time. The accuracy is plotted as the actual error at the location of the maximum error. The execution time is in seconds on a CDC 6500. A consistent scheme of plotting is used for the four methods: solid for collocation, dots for finite differences, dashes for Galerkin and dot-dash for Least Squares. Occasionally, some extra curves are plotted which are identified by a special label.

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

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

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 A1 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.

Table A1. Measured slopes α to estimate the order of the methods from their actual performance.

Problem	Finite Diff.		Collocation		Galerkin		Problem	Finite Diff.		Collocation		Galerkin	
	α	4α	α	4α	α	4α		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	—	

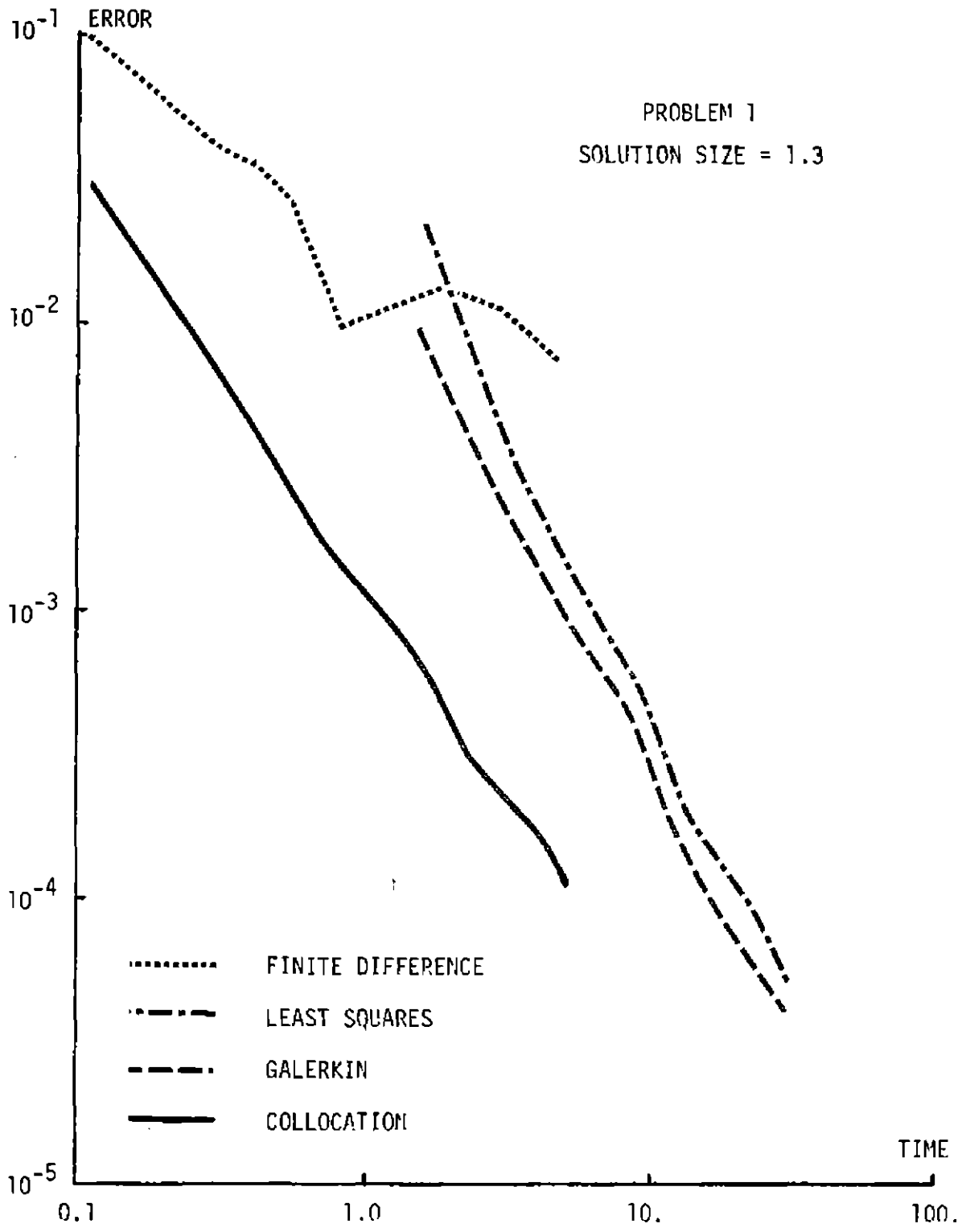
Figure A1. The data for Problems 1 to 4. Galerkin and Least Squares data is given for Problem 1. For Problem 4 we also plot the maximum error over the whole region to compare with that at the nodes.

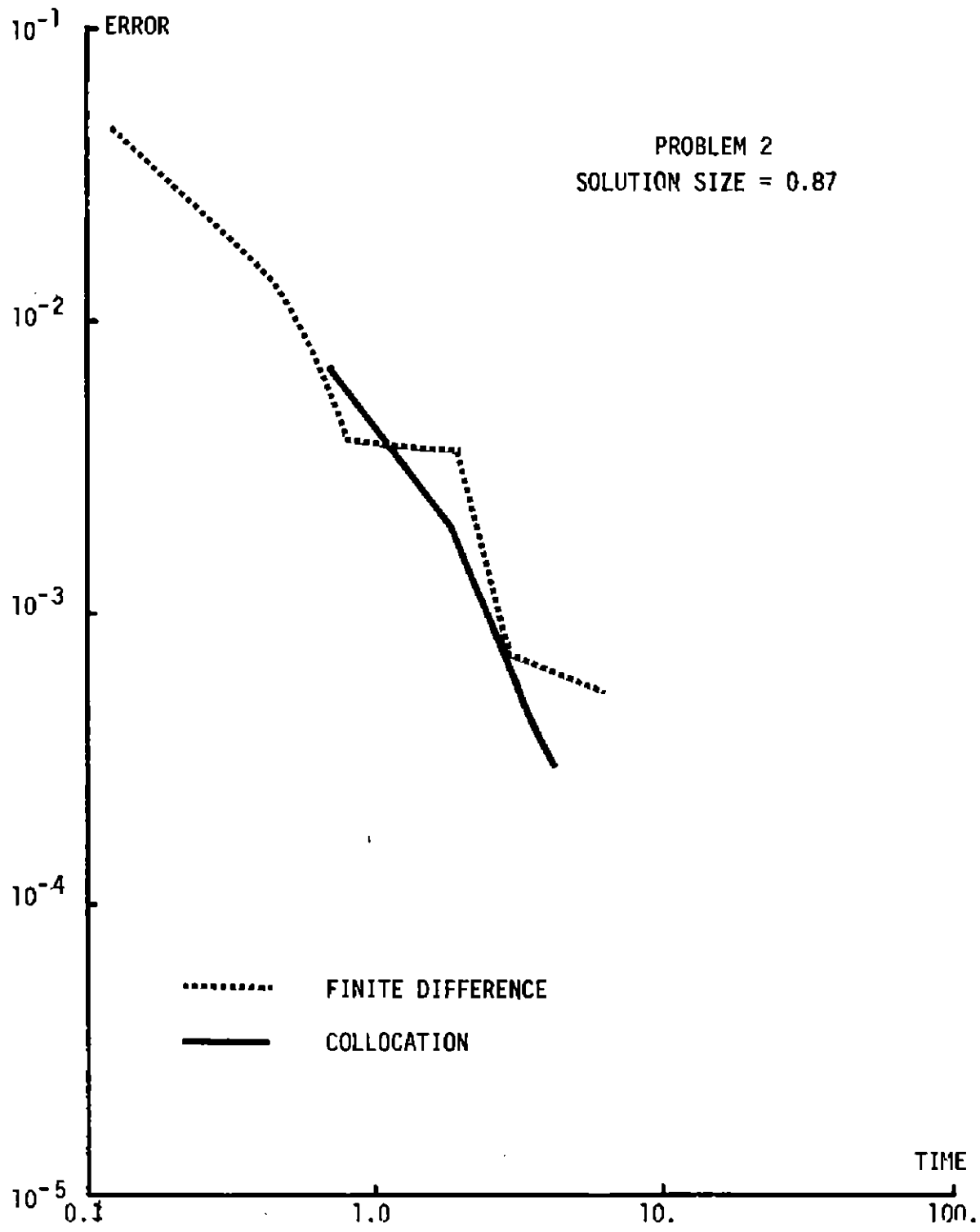
Figure A2. The data for Problems 5 to 8. The solution to Problem 8 has a mild singularity, which seems to affect the collocation solution more than Galerkin or Least Squares.

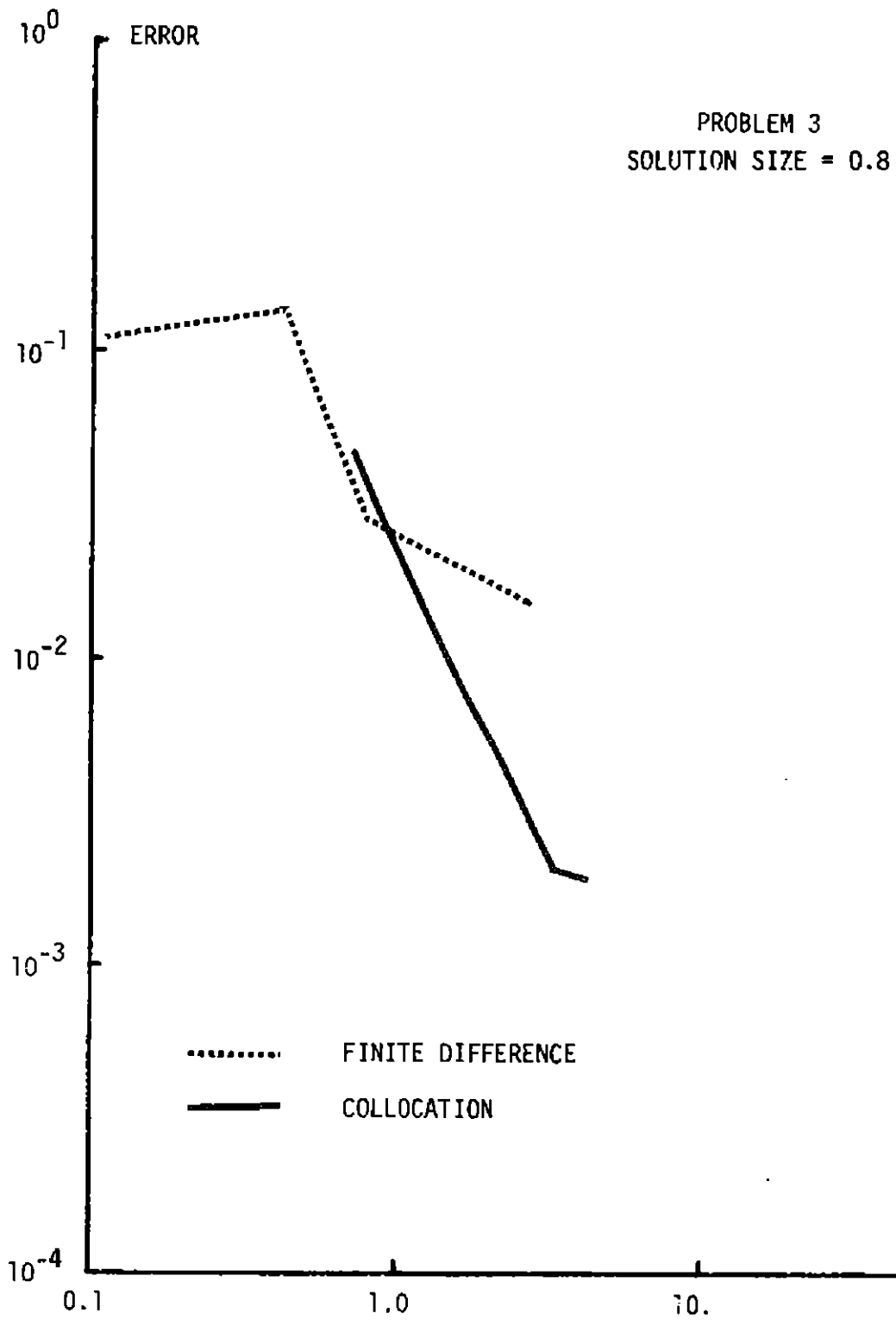
Figure A3. The data for Problems 9 to 12. Galerkin and Least Squares show erratic behavior for Problem 10. The "boundary layer" of Problem 12 adversely affects both methods of solution.

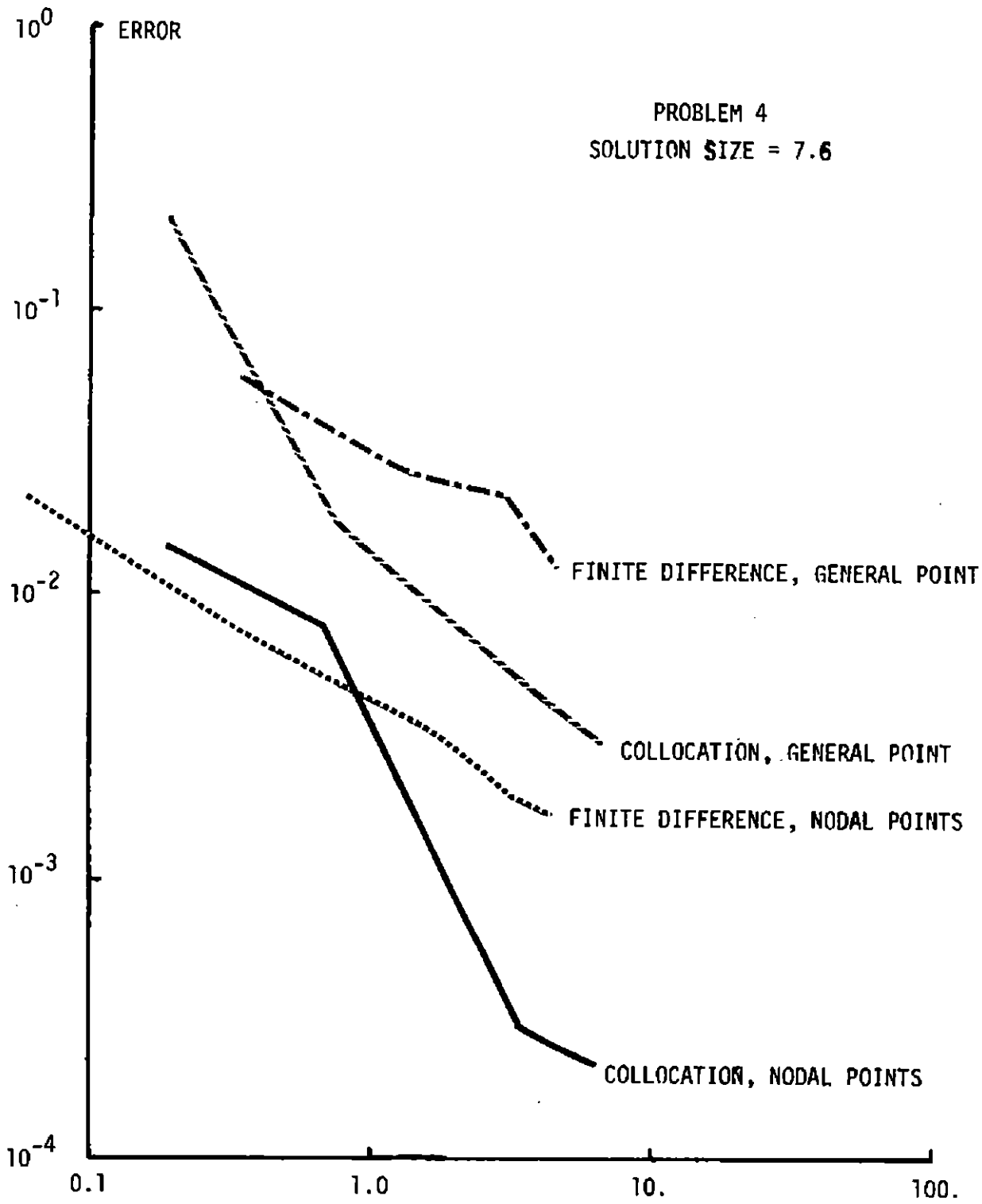
Figure A4. The data for Problems 13 to 15. The effect of collocation with a non-uniform mesh for the wave front on a right angle (Problem 13) and for an isolated sharp peak (Problem 15) is seen. The erratic behavior of collocation with a uniform mesh for Problem 13 seems to be due to the chance relationship between the mesh and the wave front.

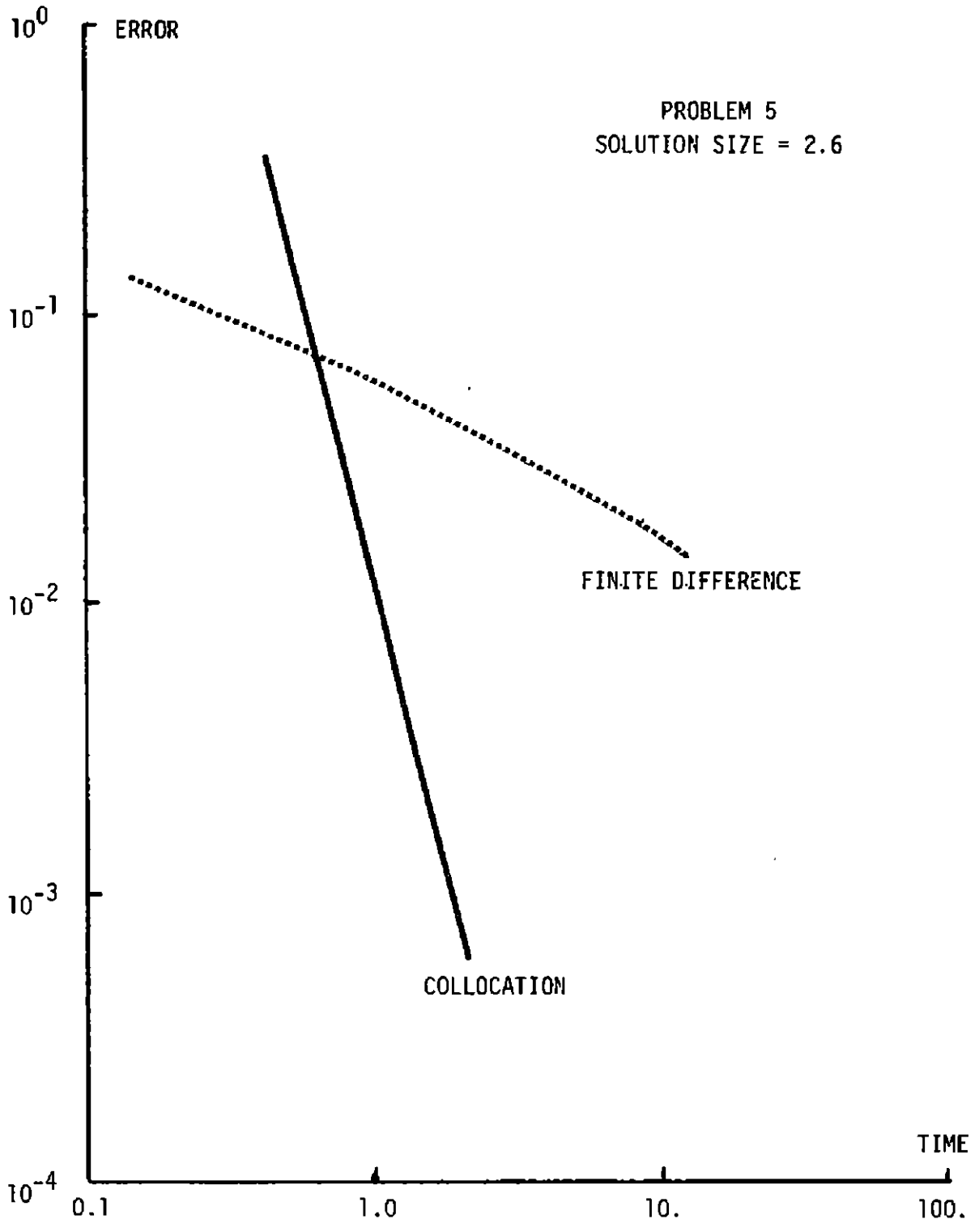
Figure A5. The data for Problems 16 and 17 with the complicated geometry of Figure 1(c). The complex geometry does not adversely affect Problem 16 where surprising accuracy is obtained. The singularities and complex geometry also do not seem to adversely affect Problem 17 (recall that the true solution is of size 100.) where the geometry forced non-uniform meshes for both collocation and finite differences.

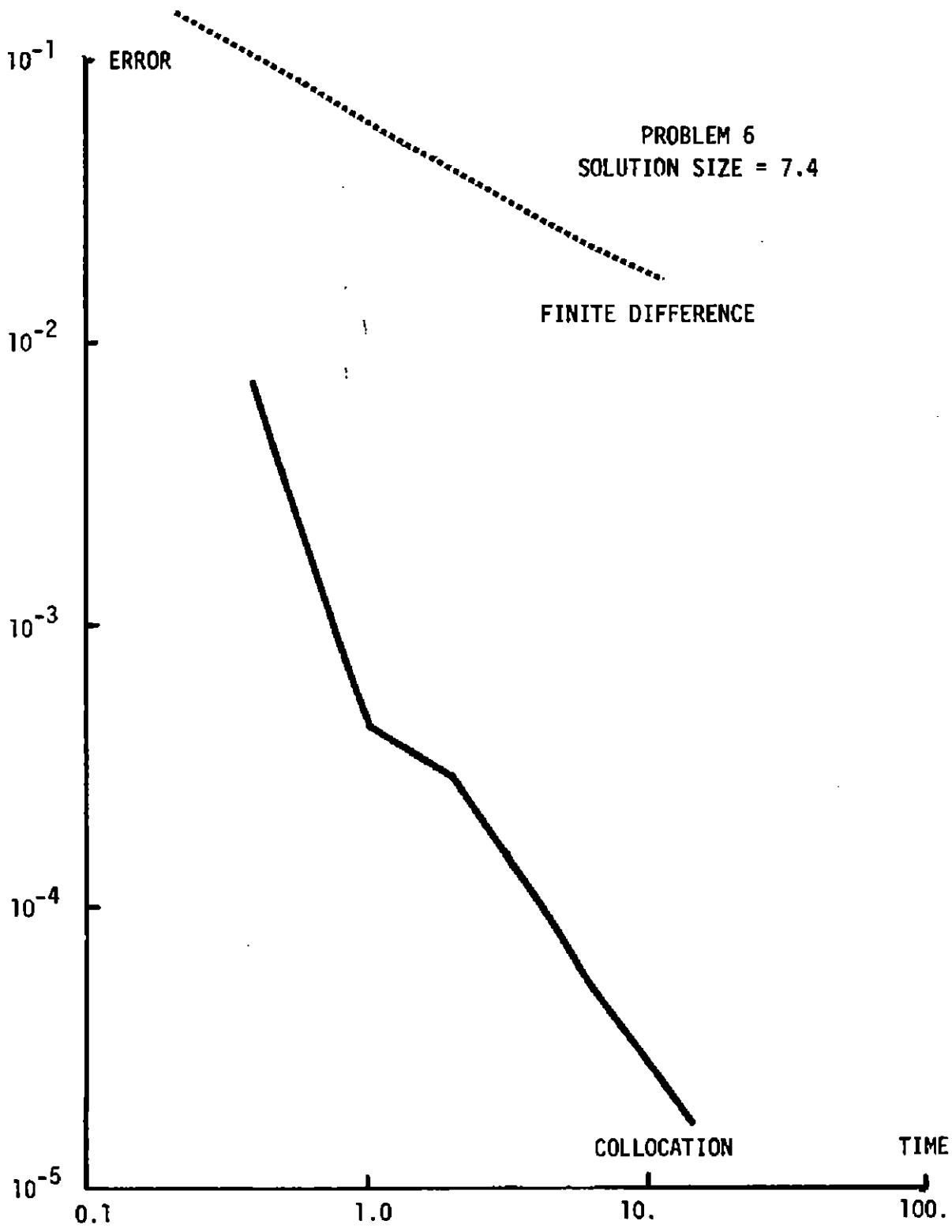

ERROR
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
PROBLEM 1
SOLUTION SIZE = 1.3
FINITE DIFFERENCE
LEAST SQUARES
GALERKIN
COLLOCATION
TIME
0.1
1.0
10.
100.

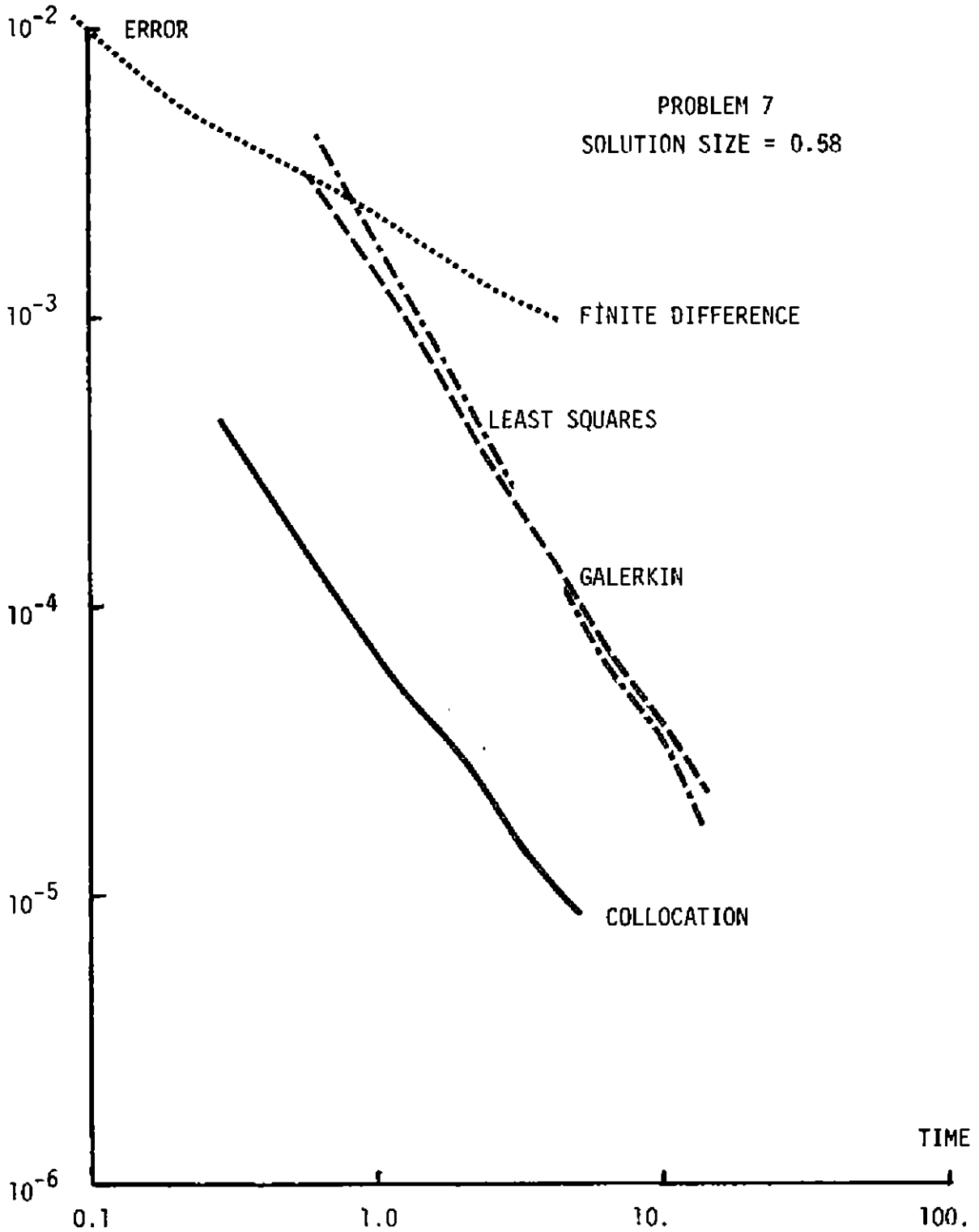

ERROR
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
PROBLEM 2
SOLUTION SIZE = 0.87
FINITE DIFFERENCE
COLLOCATION
TIME
0.1
1.0
10.
100.

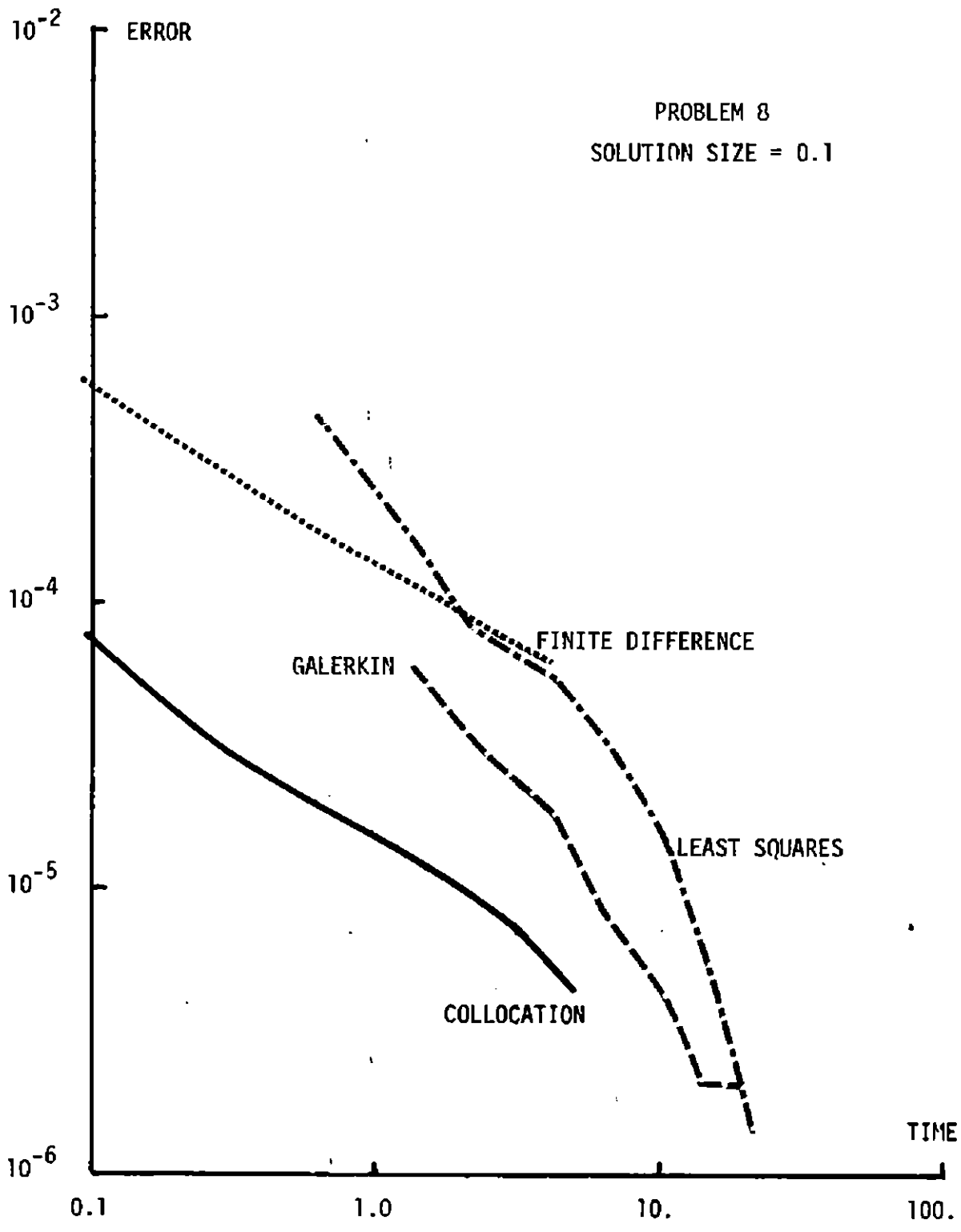

10^0
ERROR
PROBLEM 3
SOLUTION SIZE = 0.8
10^{-1}
10^{-2}
10^{-3}
FINITE DIFFERENCE
COLLOCATION
10^{-4}
TIME
0.1
1.0
10.
100.

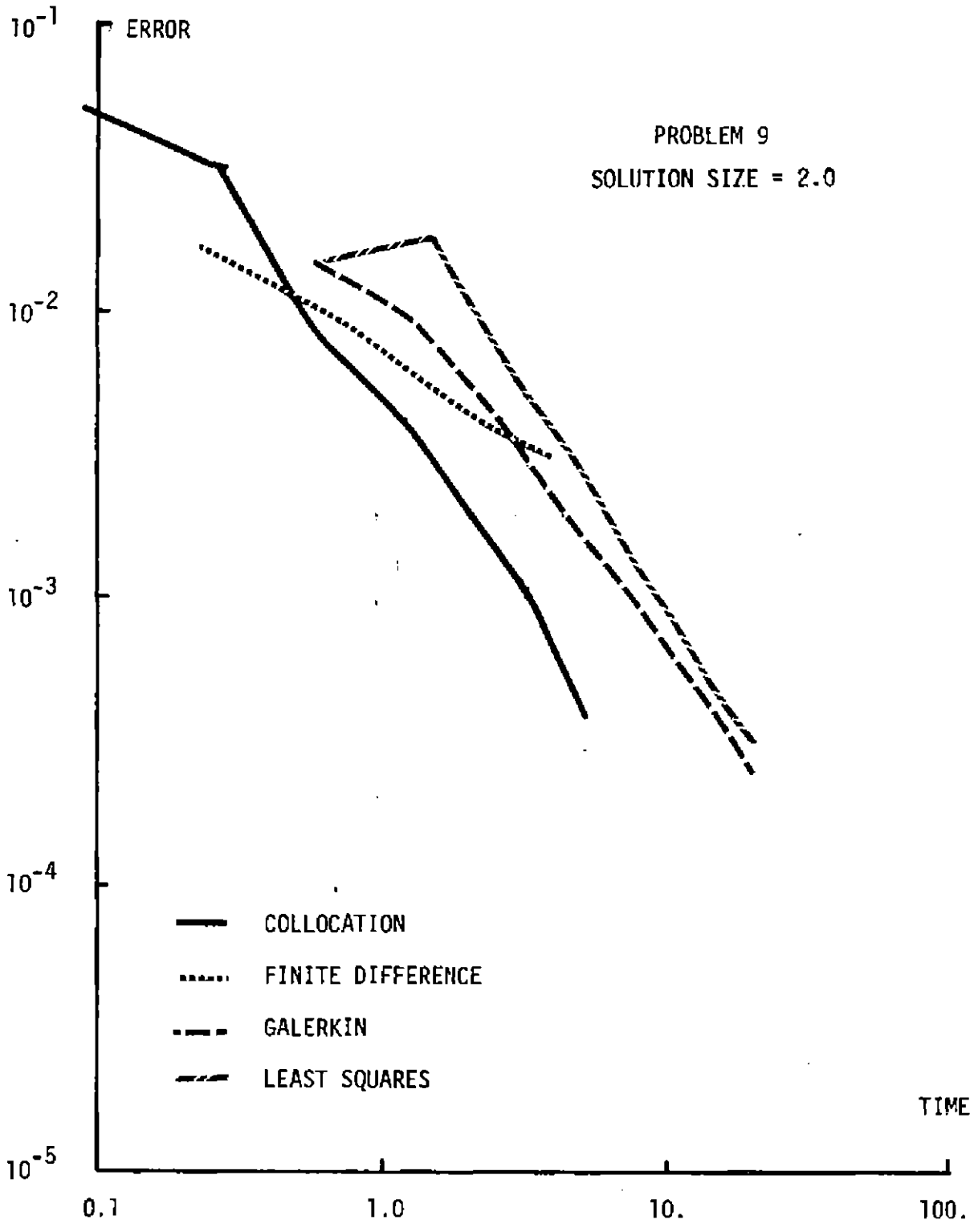

10^0
ERROR
PROBLEM 4
SOLUTION SIZE = 7.6
10^{-1}
10^{-2}
10^{-3}
10^{-4}
FINITE DIFFERENCE, GENERAL POINT
COLLOCATION, GENERAL POINT
FINITE DIFFERENCE, NODAL POINTS
COLLOCATION, NODAL POINTS
0.1
1.0
10.
100.

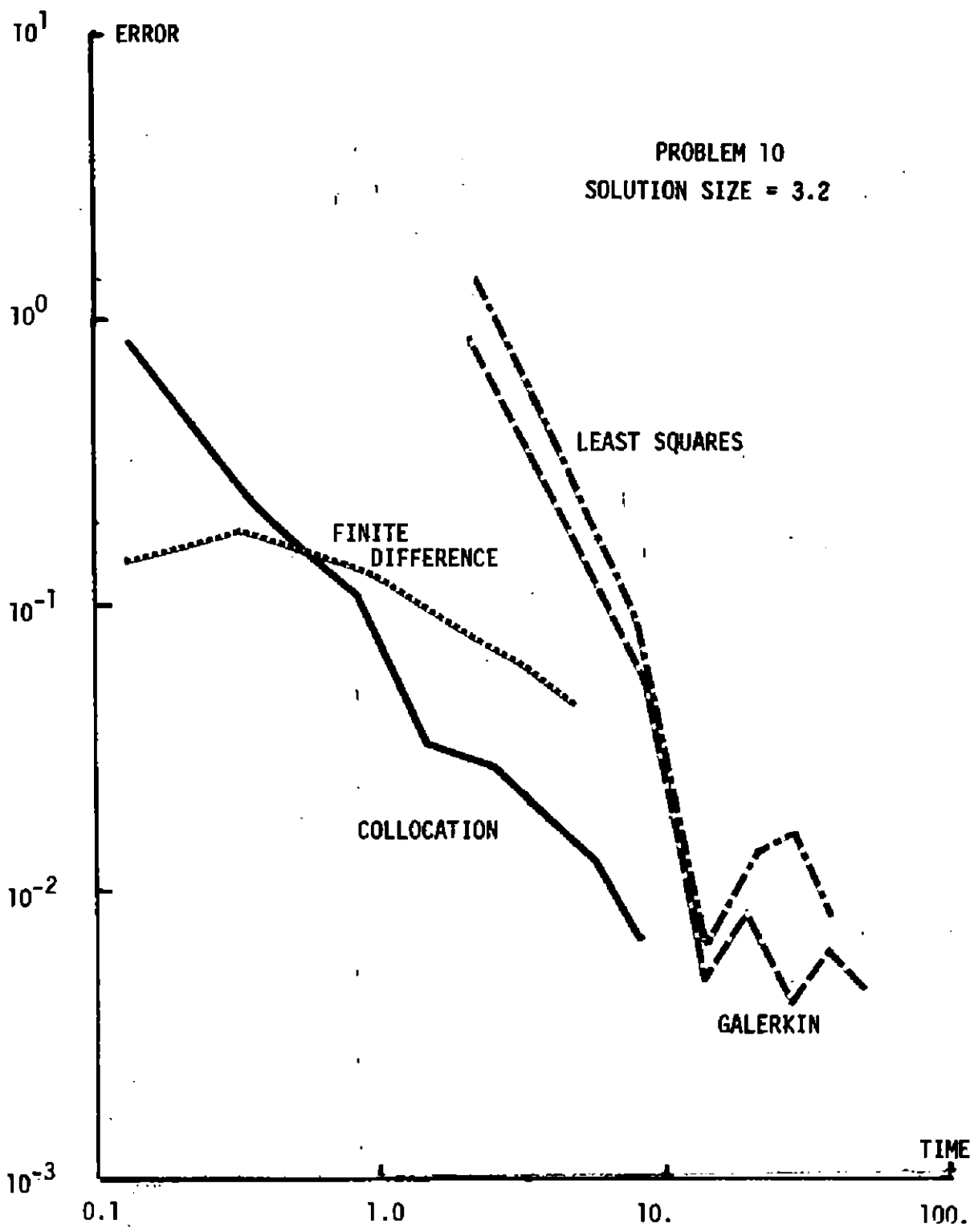

10^0
ERROR
PROBLEM 5
SOLUTION SIZE = 2.6
10^{-1}
FINITE DIFFERENCE
10^{-2}
10^{-3}
COLLOCATION
TIME
10^{-4}
0.1
1.0
10.
100.

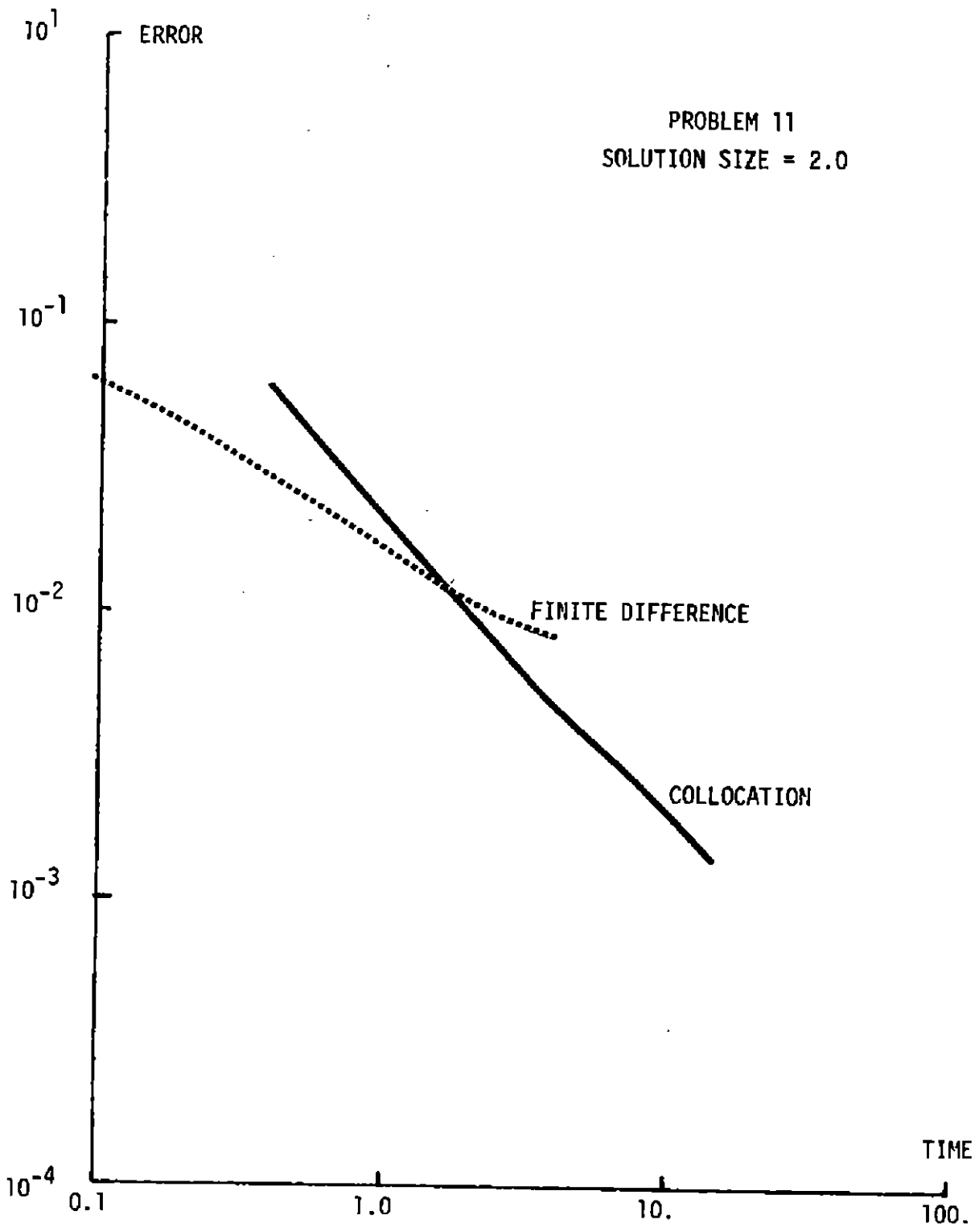

ERROR
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
PROBLEM 6
SOLUTION SIZE = 7.4
FINITE DIFFERENCE
COLLOCATION
TIME
0.1
1.0
10.
100.

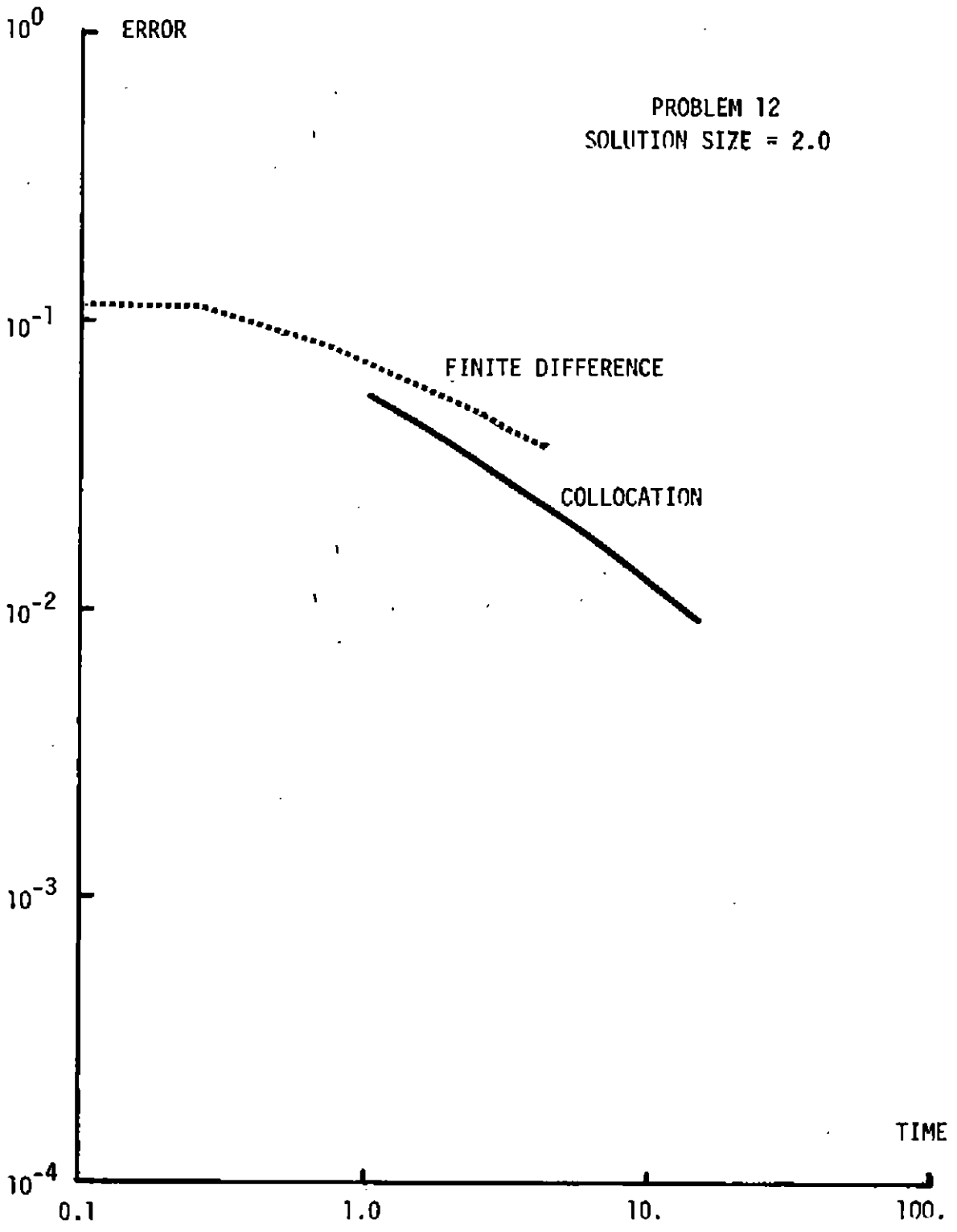

10^{-2}
ERROR
PROBLEM 7
SOLUTION SIZE = 0.58
10^{-3}
FINITE DIFFERENCE
LEAST SQUARES
GALERKIN
10^{-4}
10^{-5}
COLLOCATION
TIME
10^{-6}
0.1
1.0
10.
100.

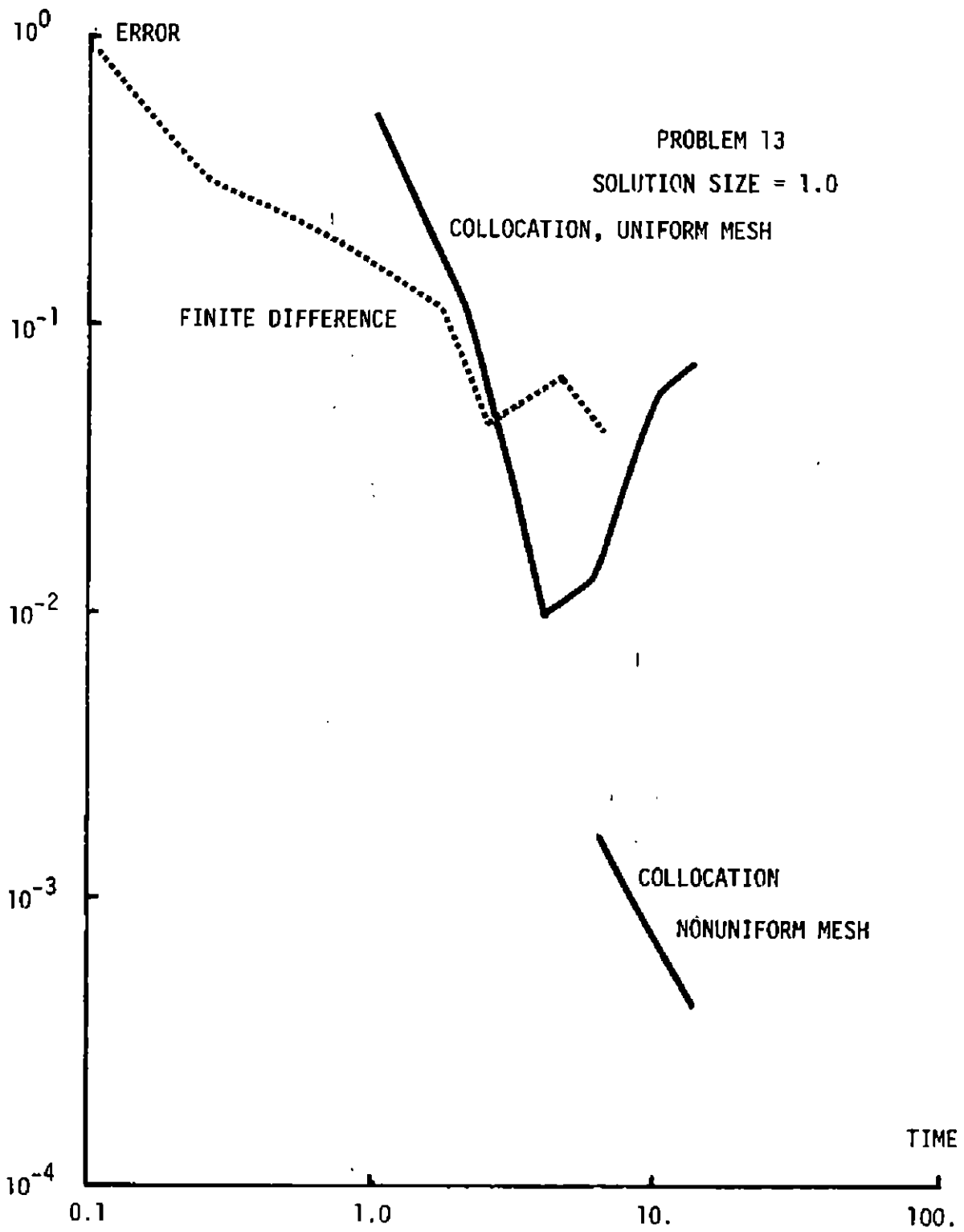

ERROR
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
PROBLEM 8
SOLUTION SIZE = 0.1
FINITE DIFFERENCE
GALERKIN
LEAST SQUARES
COLLOCATION
TIME
0.1
1.0
10.
100.

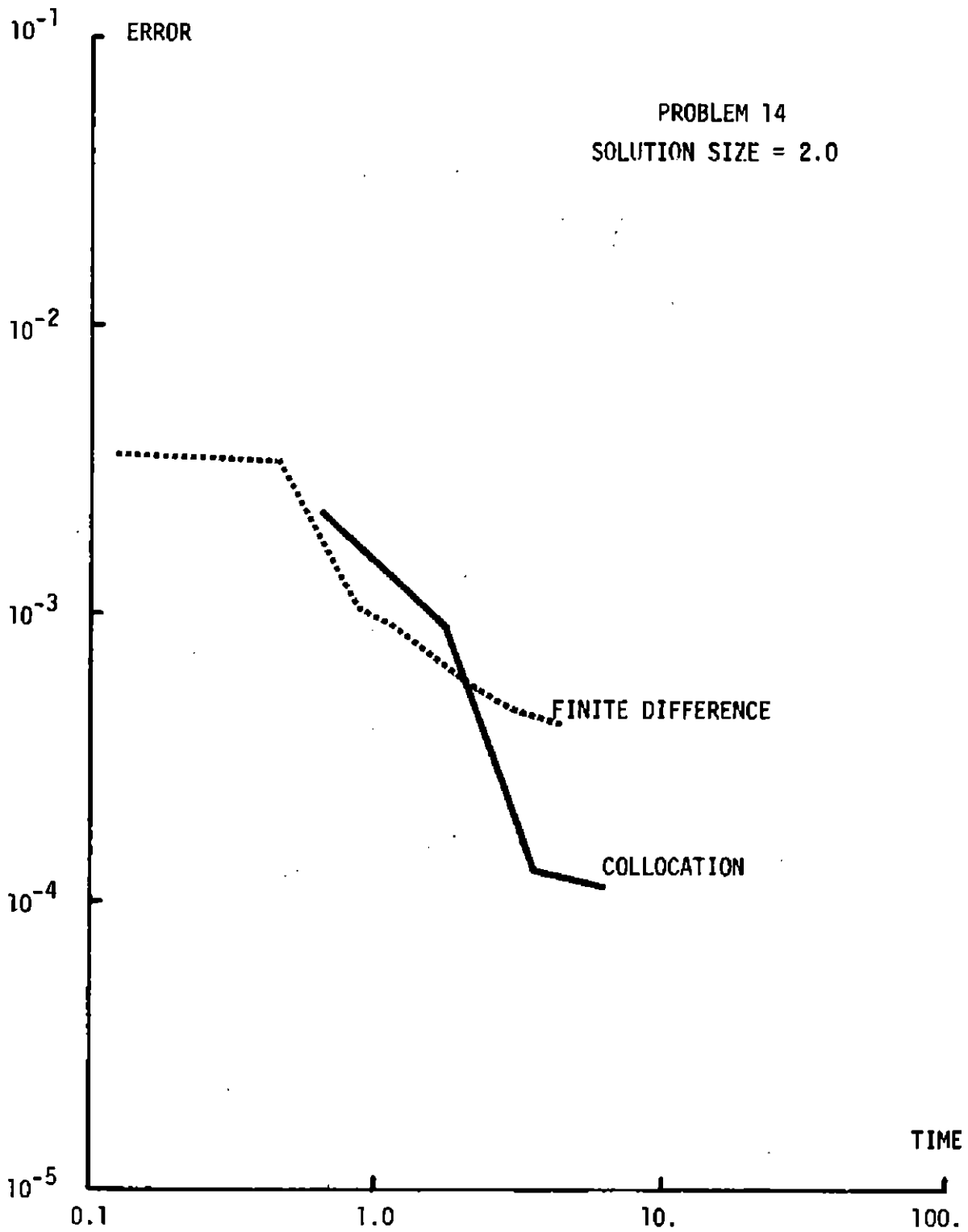

ERROR
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
PROBLEM 9
SOLUTION SIZE = 2.0
COLLOCATION
FINITE DIFFERENCE
GALERKIN
LEAST SQUARES
TIME
0.1
1.0
10.
100.

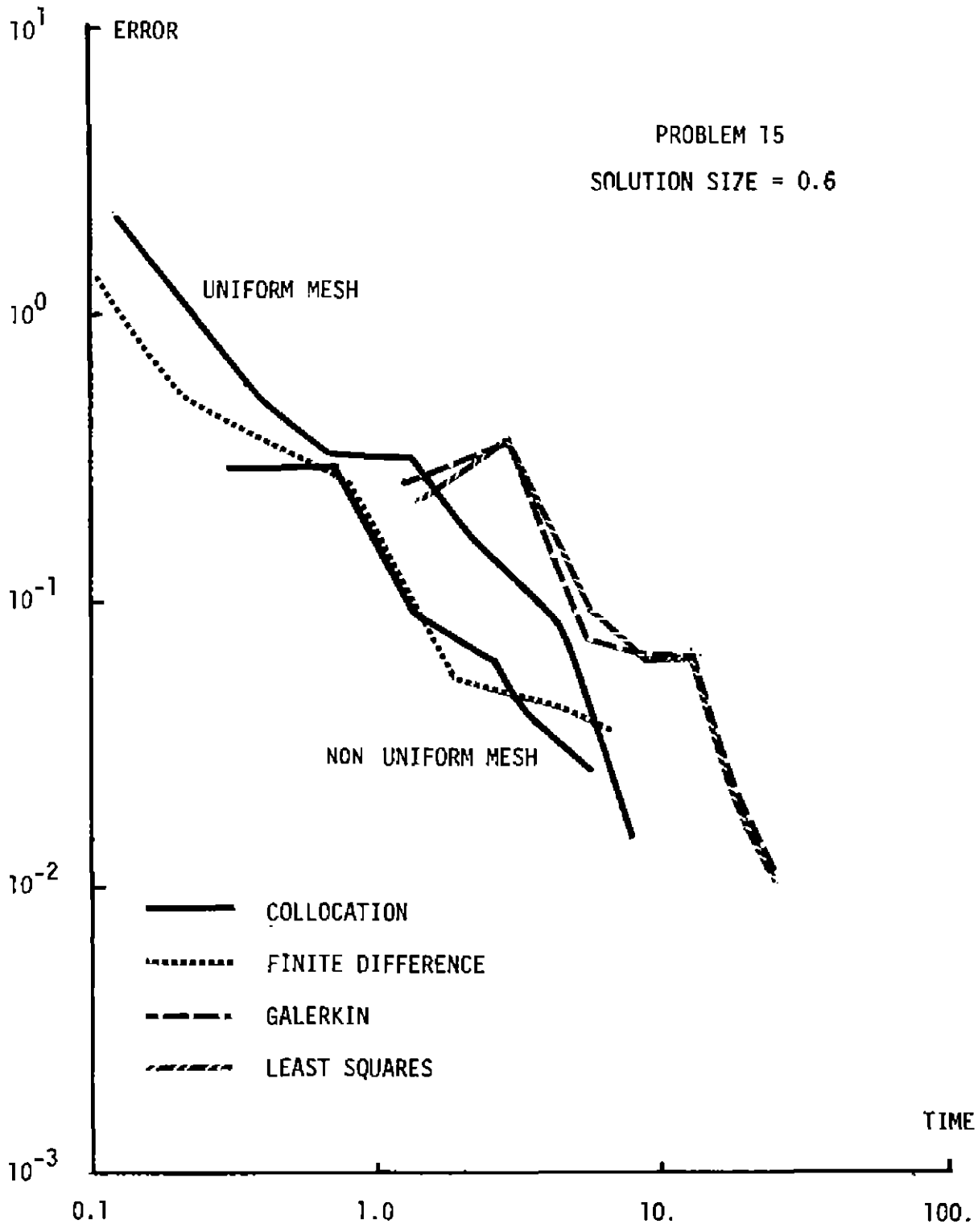

ERROR
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
PROBLEM 10
SOLUTION SIZE = 3.2
LEAST SQUARES
FINITE
DIFFERENCE
COLLOCATION
GALERKIN
TIME
0.1
1.0
10.
100.

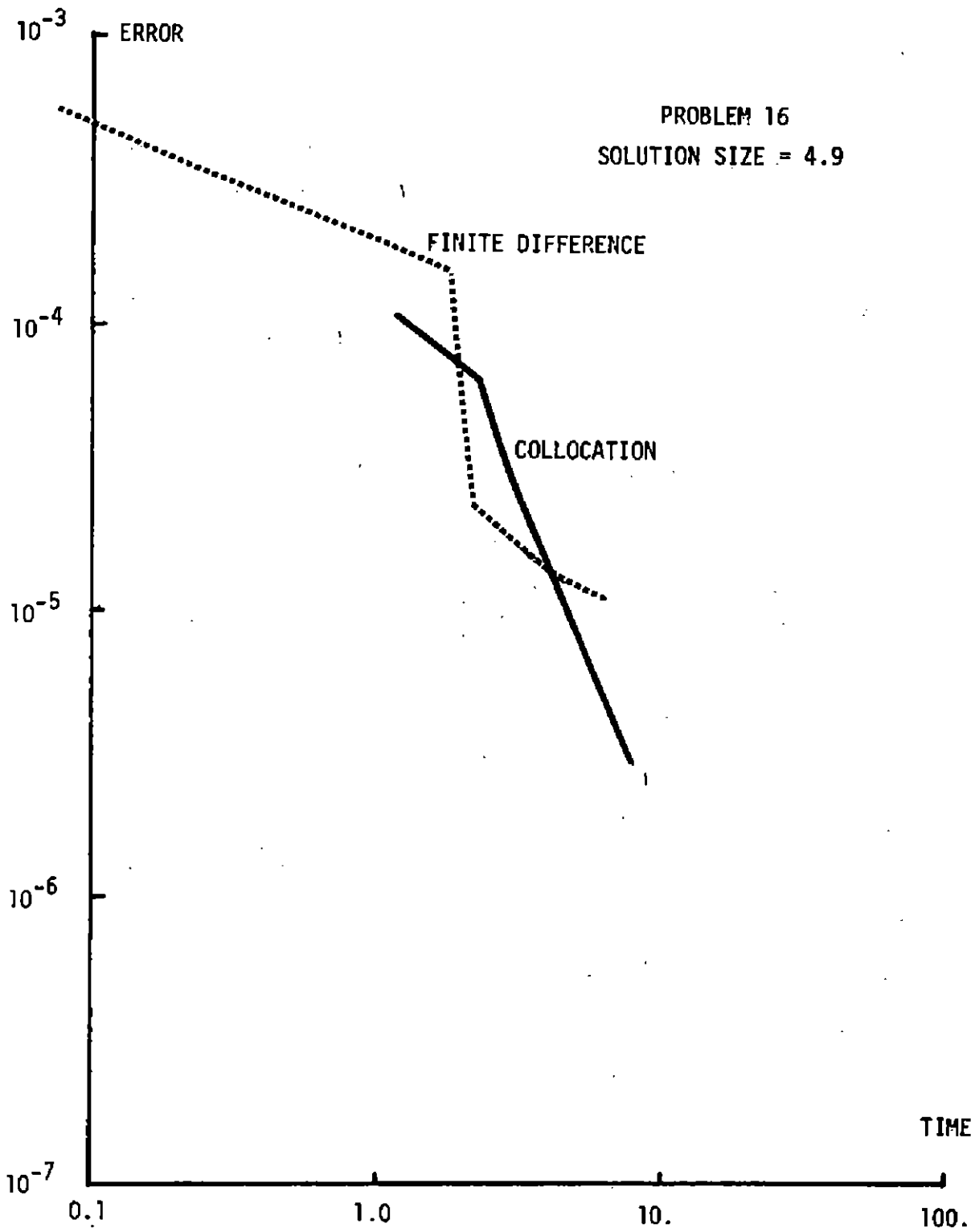

ERROR
PROBLEM 11
SOLUTION SIZE = 2.0
10^{1}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
FINITE DIFFERENCE
COLLOCATION
TIME
0.1
1.0
10.
100.

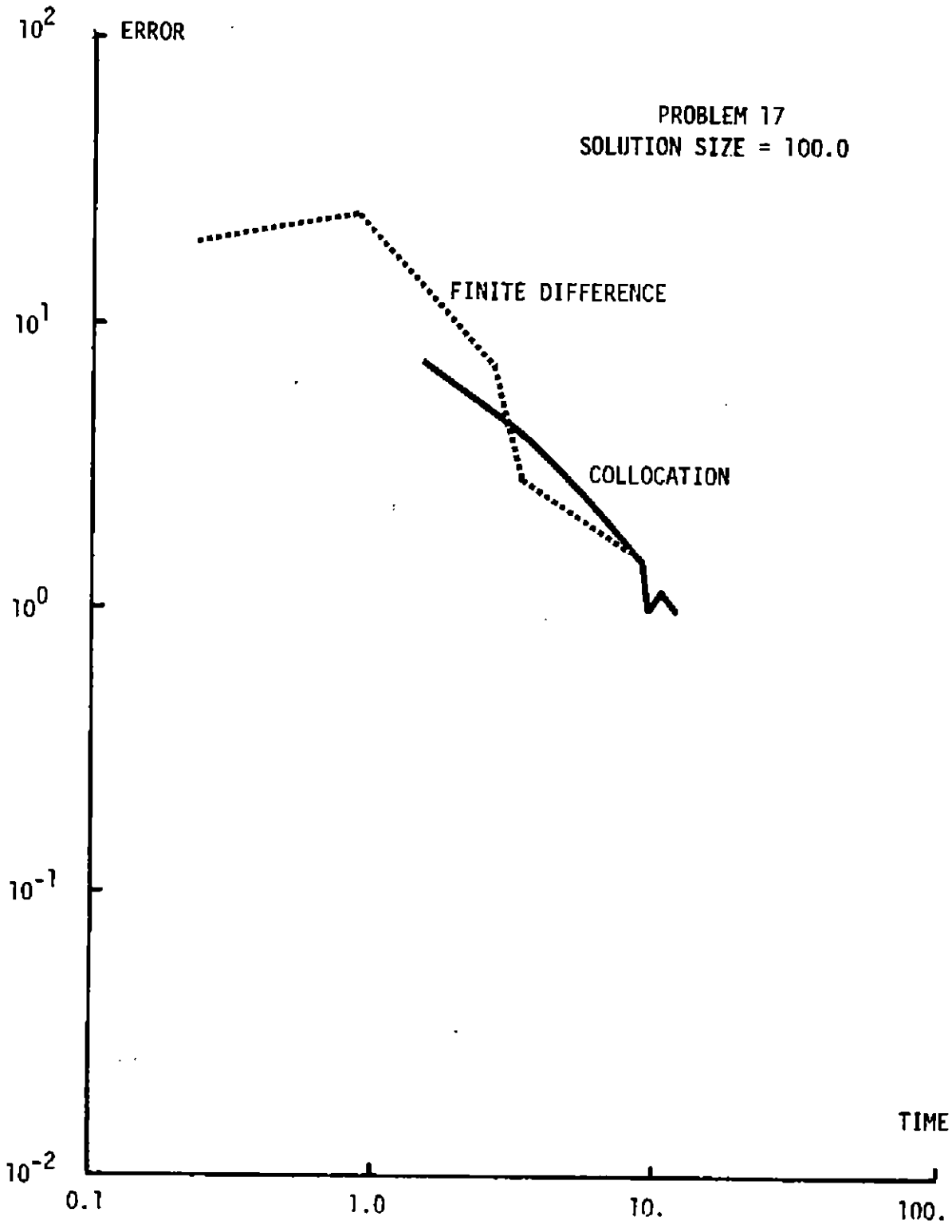

ERROR
PROBLEM 12
SOLUTION SIZE = 2.0
FINITE DIFFERENCE
COLLOCATION
TIME
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0.1
1.0
10.
100.


10^0
ERROR
PROBLEM 13
SOLUTION SIZE = 1.0
COLLOCATION, UNIFORM MESH
10^{-1}
FINITE DIFFERENCE
10^{-2}
10^{-3}
COLLOCATION
NONUNIFORM MESH
TIME
10^{-4}
0.1
1.0
10.
100.


ERROR
PROBLEM 14
SOLUTION SIZE = 2.0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
FINITE DIFFERENCE
COLLOCATION
TIME
0.1
1.0
10.
100.


ERROR
PROBLEM 15
SOLUTION SIZE = 0.6
UNIFORM MESH
NON UNIFORM MESH
COLLOCATION
FINITE DIFFERENCE
GALERKIN
LEAST SQUARES
TIME
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
0.1
1.0
10.
100.


10^{-3}
ERROR
PROBLEM 16
SOLUTION SIZE = 4.9
FINITE DIFFERENCE
10^{-4}
COLLOCATION
10^{-5}
10^{-6}
10^{-7}
TIME
0.1
1.0
10.
100.


ERROR
10^2
10^1
10^0
10^{-1}
10^{-2}
PROBLEM 17
SOLUTION SIZE = 100.0
FINITE DIFFERENCE
COLLOCATION
TIME
0.1
1.0
10.
100.

APPENDIX TWO

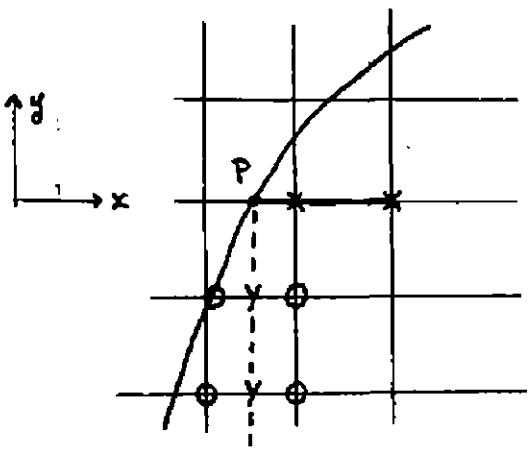
SYNOPSIS OF THE NUMERICAL METHODS

1. 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 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 2 x-points.

y-derivative at P estimated from value at P and the 2 y-points.

The values at the y-points are found by linear interpolation from the 0-points.

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

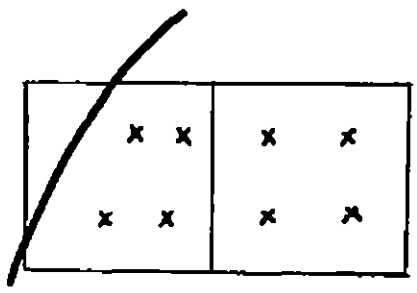
2. Collocation. This method has the following components.

(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 unless noted except for Problems 16, 17.

(b) Approximation space: the Hermite bicubics defined at the end of this appendix.

(c) Approximation to the operator: The approximate solution satisfies the differential equation exactly at the four Gauss point of a rectangular element. For non-rectangular elements near the boundary the four Gauss points are

projected inside the element as indicated by the diagram.



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, Neumann or Mixed 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. See Appendix 3 for details on how the boundary collocation points are selected.

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

3. 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 equations

$$\sum_{i=1}^{16} \alpha_i \iint_E \{p\, D_x B_i\, D_x B_j + q\, D_y B_i\, D_y B_j + r\, B_i\, B_j\} dxdy = \iint_E f\, B_j\, dxdy$$

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

$$LU^* = (p\, U_x^*)_x + (q\, U_y^*)_y + rU^* = f$$

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

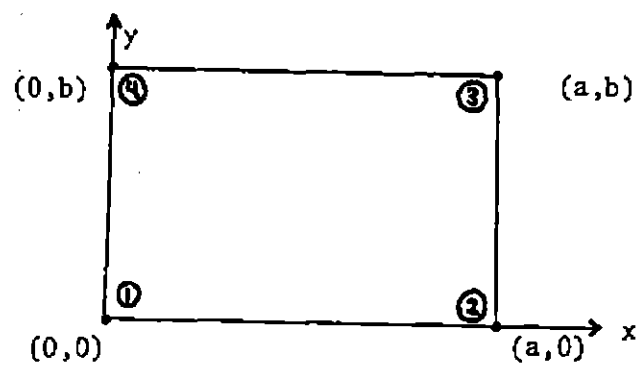
$$\sum_{i=1}^{16} \alpha_i \iint_E L(B_i) \cdot L(B_j)dxdy = \iint_F f\, L(B_j)dxdy$$

The integrals in these equations are approximated by the 9-point Gauss quadrature rule for rectangles (only rectangular domains were used with these methods).

(d) Approximation to the boundary conditions: the boundary condition 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 equation is solved by Gauss elimination for positive definite matrices.

4. <u>The Rectangular Bicubic Hermite Element</u>. The situation is shown in the diagram



$s = x/a$ and $0 \leq s \leq 1$

$t = y/b$ and $0 \leq t \leq 1$

The numerical labels on the corners are used later to index the points.

We use 8 one dimensional functions to construct the 16 basis functions for the rectangle:

$$B_{x1} = 1-3s^2+2s^3 \qquad B_{y1} = 1-3t^2+2t^3$$
$$B_{x2} = s^2(3-2s) \qquad B_{y2} = t^2(3-2t)$$
$$B_{x3} = as(s-1)^2 \qquad B_{y3} = bt(t-1)^2$$
$$B_{x4} = as^2(s-1) \qquad B_{y4} = bt^2(t-1)$$

Then $u^*(x,y)$ is approximated in each rectangle by

$$\begin{aligned} u(x,y) = {} & B_{x1} B_{y1} U_1 + B_{x2} B_{y1} U_2 + B_{x2} B_{y2} U_3 + B_{x1} B_{y2} U_4 \\ & + B_{x3} B_{y1} \sigma_{x1} + B_{x4} B_{y1} \sigma_{x2} + B_{x4} B_{y2} \sigma_{x3} + B_{x3} B_{y2} \sigma_{x4} \\ & + B_{x1} B_{y3} \sigma_{y1} + B_{x2} B_{y3} \sigma_{y2} + B_{x2} B_{y4} \sigma_{y3} + B_{x1} B_{y4} \sigma_{y4} \\ & + B_{x3} B_{y3} \tau_{xy1} + B_{x4} B_{y3} \tau_{xy2} + B_{x4} B_{y4} \tau_{xy3} + B_{x3} B_{y4} \tau_{xy4} \end{aligned}$$

where u_i = value at the point i

σ_{xi}, σ_{yi} = x and y derivatives at the point i

τ_{xyi} = xy (cross) derivative at the point i.

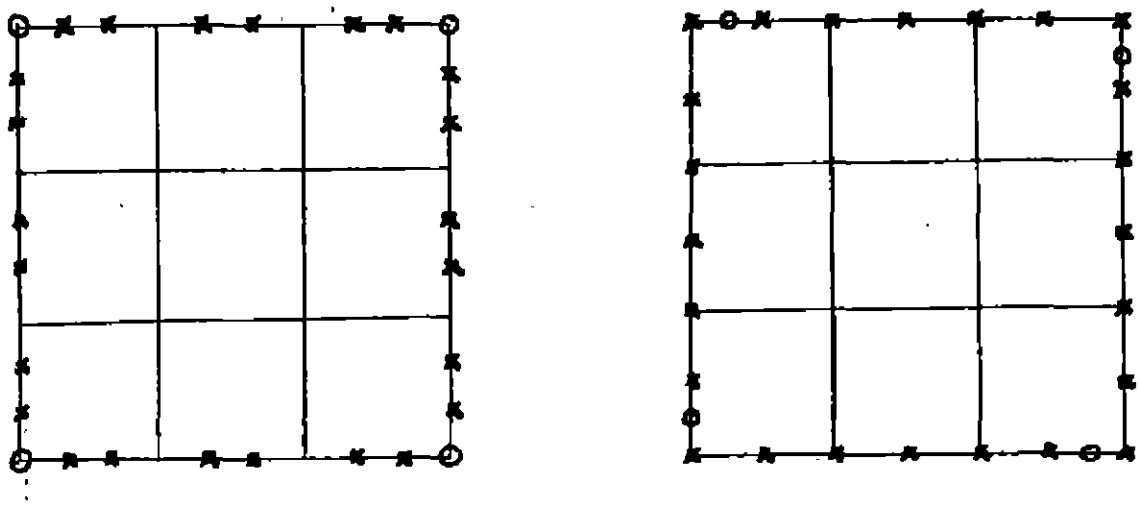
The 16 functions in the above equation are the ones denoted by $B_i(x,y)$ earlier in the Galerkin and Least Squares equations, e.g. $B_1(x,y) = B_{x1} B_{y1}$.

APPENDIX THREE

THE INTERPOLATION OF BOUNDARY CONDITIONS FOR COLLOCATION

The most sensitive aspect of collocation is the placement of the boundary collocation points for non-rectangular domains. First, one must take care that these points are reasonably separated from the points in the interior where one collocates with the differential operator. 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 points required is $2S_b + 4$. We use two basic schemes for distributing the boundary collocation points as illustrated by the diagrams below for a simple rectangle:



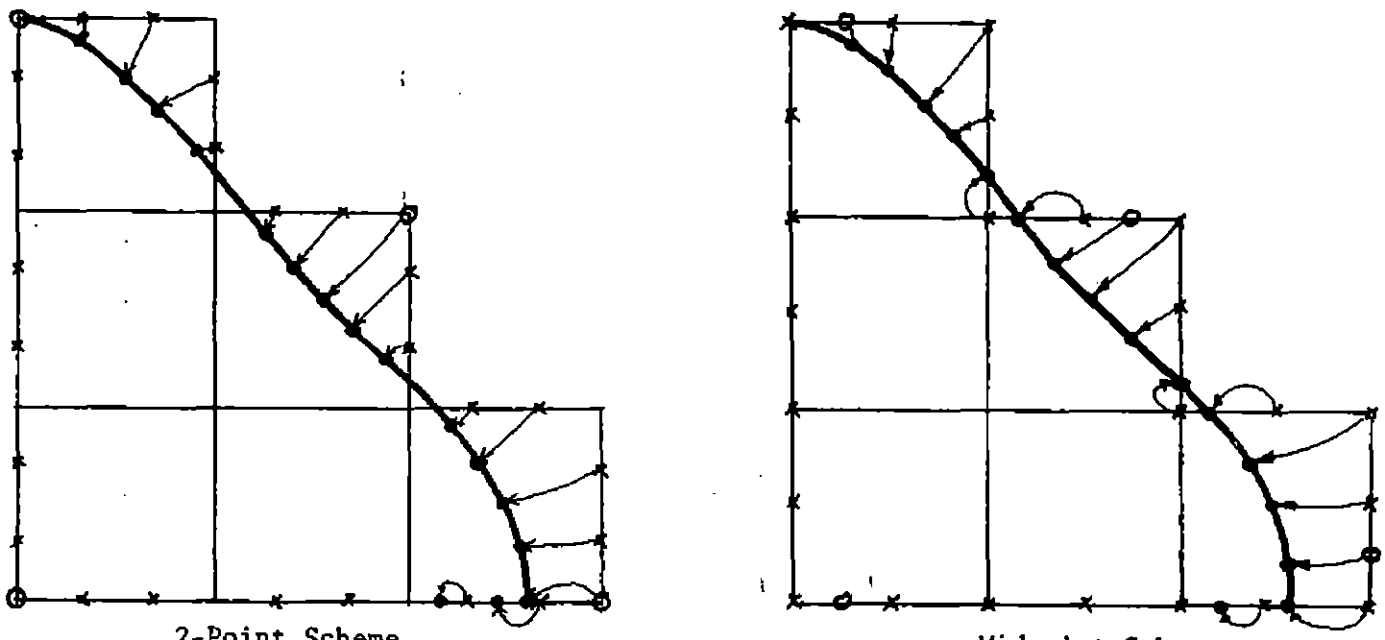
2-Point Scheme

Midpoint Scheme

Figure A6. Two schemes for distributing boundary collocation points. The x's are the systematic collocation points and the 0's are the four extra ones.

A theoretical analysis shows that the 2-point scheme is superior for rectangular regions provided the two points are taken to be the Gauss points for each boundary segment. We compared using the Gauss points with equally spaced points and found the equally spaced points give slightly better accuracy and they are slightly easier to use. We made numerous numerical experiments which confirmed that the 2-point scheme is superior for rectangular regions.

The extension of these two schemes to curved domains is illustrated in Figure A7.



2-Point Scheme

Midpoint Scheme

Figure A7. The two schemes for a simple curved domain. 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 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 Figure 1). It is often essential that collocation of the boundary conditions be made at all exterior corners of the domain.

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 otherwise. The two schemes may be used together for a domain such as shown above and we do this as shown in Figure A8.

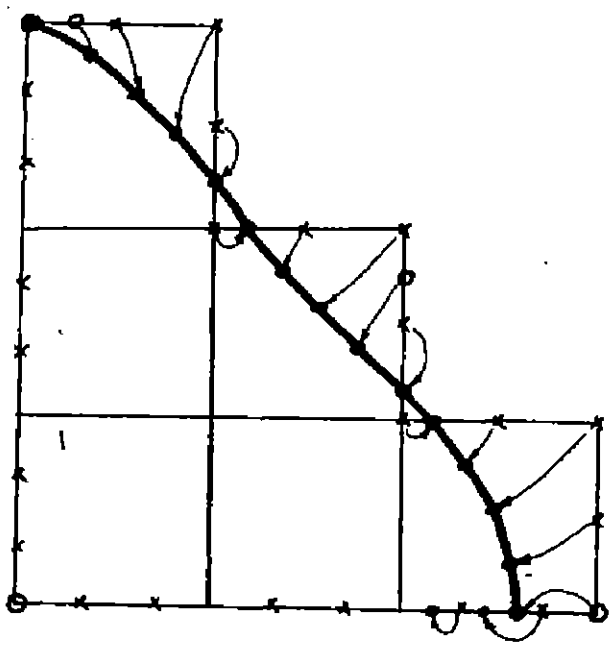


Figure A8. The combination of the two schemes for a partially rectangular region. The mapping from the point on the rectangular edges to the curved boundary is indicated.

There seems to be no particularly advantageous method to distribute the 4 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 to place collocation points on the intermediate segments. We do place collocation

outside the rectangular partition for the 2-point scheme. An example is shown in Figure A9 which illustrates these procedures.

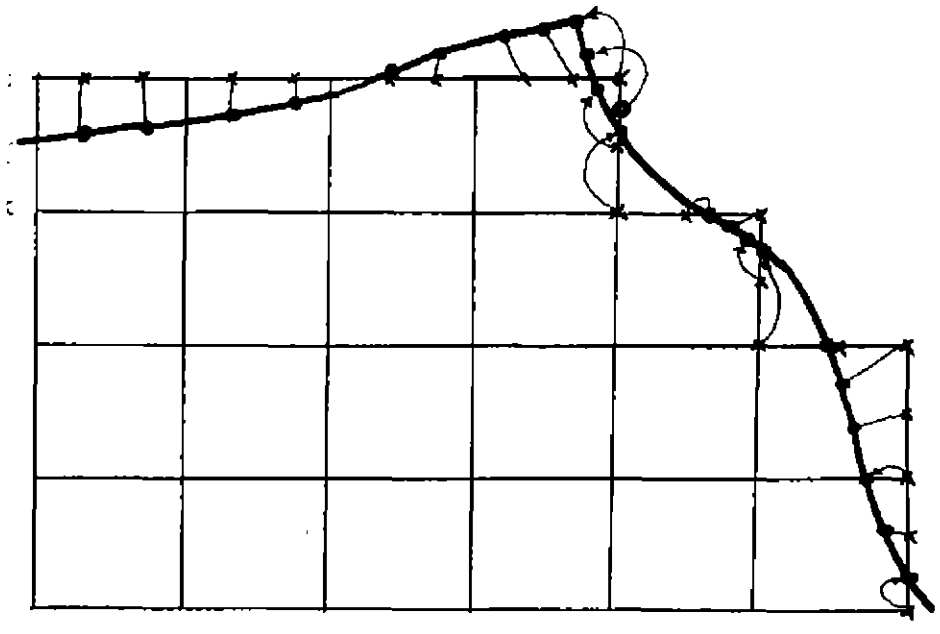


Figure A9. Example which illustrates boundary collocation points for the 2-point scheme which are outside the rectangular partition and collocation for the midpoint scheme are inside. Collocation is not done on two large boundary segments.

APPENDIX FOUR

THE SOLUTION OF PROBLEM 17 AND FUNCTIONS INVOLVED IN THE OTHER PROBLEMS

We describe the exact solution u of Problem 17 for the reactor heat shields $\nabla^2 u = f$.

We set

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

where, by construction, the numerator on the right is zero on the stair-step outer boundary of the domain (see Figure 1). The numerator is the product of $(x-1)$, $(y-1)$, and three factor 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:

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

$$C(x,y) = (x^2 + y^2 - .64)^2$$

$$T(x,y,x_i,y_i,b,c) = R(x,y,x_i,y_i,b)\ S(x,y,x_i,y_i,c)$$

$$R(x,y,x_i,y_i,b) = [(x-s_i)^2 + (y-y_i)^2 + b\ C(x,y)]^{1/3}$$

$$S(x,y,X_i,y_i,c) = \sin(2[\arctan([y-y_i]/[x-x_i]) + \pi/2]/3) + c\ C(x,y)$$

with branch cut along $y-y_i = x-x_i$, $x_i < x$

After some experimentation, we found that $a = -.5$, $b = .1$, $c = 7$. gives a solution u which is similar to that 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 somewhat simplier function [which later proved to be unsuitable because it had zeros in the interior of the region]. A Fortran program was written for the evaluation of u and it was processed by a symbolic differentiator to obtain function subroutines to evaluate u_{xx} and u_{yy}. The resulting programs for u, u_{xx}, u_{yy} were more complicated and much longer than the one we eventually wrote for our more complicated function. 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:

```
W   =  ...
WX  =  ...
WXX =  ...
Z   =  ...
ZX  =  ...
ZXX =  ...
V   =  W*Z
VX  =  WX*Z + W*ZX
VXX =  WXX*Z + 2.*WX*ZX + W*ZXX
```

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 = ..., ... ZXX = ... , do not appear since the values are already computed by previously called subroutines. The program is quickly written and debugged.

```
***** PROBLEM  1  DATA *****
      FUNCTION COEF(X,Y,J)
      Z = EXP(X*Y)
      RZ = 1./Z
      GO TO (101,102,103,104,105),J
101   COEF = Z
      RETURN
102   COEF = RZ
      RETURN
103   COEF = Y * Z
      RETURN
104   COEF = -X * RZ
      RETURN
105   COEF = -1./(1. + X + Y)
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO (101,102) , J
101   PI = 3.14159265358979
      Z = EXP(X*Y)
      RZ = 1. / Z
      PIX = PI*X
      PIY = PI*Y
      PIZ = PI*Z
      SINX = SIN(PIX)
      SINY = SIN(PIY)
      TRUE = Z*SINX*SINY
      TEMP = PI*PI*TRUE
      XTRUE = X*TRUE
      YTRUE = Y*TRUE
      FX = PIZ*COS(PIX)*SINY
      FY = PIZ*COS(PIY)*SINX
       DXTR = YTRUE + FX
      DYTR = XTRUE + FY
      DDXTR = Y*YTRUE - TEMP + 2.*Y*FX
      DDYTR = X*XTRUE - TEMP + 2.*X*FY
      F = Z*DDXTR+RZ*DDYTR+Y*Z*DXTR-X*RZ*DYTR-TRUE/(1.+X+Y)
      RETURN
102   F = 0.
      RETURN
      END
      FUNCTION TRUE(X,Y)
      PI = 3.14159265358979
      TRUE = EXP(X*Y)*SIN(PI*X)*SIN(PI*Y)
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   BCOEF = 1.
      RETURN
102   BCOEF = 0.
      RETURN
103   BCOEF = 0.
      RETURN
      END
```

```
***** PROBLEM  2  DATA *****
      FUNCTION F(X,Y,J)
      GO TO (101,102),J
101   F = 0.
      RETURN
102   IF(X.EQ.0..OR.X.EQ.1.) GO TO 1
      IF(Y.EQ.0..OR.Y.EQ..5) GO TO 1
      F = 1.
      RETURN
1     F = 0.
      RETURN
      END
      FUNCTION COEF(X,Y,J)
      GO TO (1,2,3,4,5),J
1     COEF = G(X,Y)
      RETURN
2     COEF = G(X,Y)
      RETURN
3     COEF = 0.
      RETURN
4     COEF = 0.
      RETURN
5     COEF = 0.
      RETURN
      END
      FUNCTION G(X,Y)
      E = .00001
      X1 = .5-E
      X2 = .5 + E
      DX = X2 - X1
      IF( X .LE. X1 ) GO TO 1
      IF( X .GE. X2 ) GO TO 2
      POL = 3.-6.*(X-X1)**2/(DX*DX)+4.*(X-X1)**3/(DX**3)
      G = 1./POL
      RETURN
1     G = 1./3.
      RETURN
2     G = 1.
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   IF(X .GT. 0. .AND. X .LT. .25) GO TO 1
      IF(X .GT. .75 .AND. X .LT. 1.) GO TO 1
      BCOEF = 1.
      RETURN
1     IF(Y .EQ. 0.) GO TO 2
      BCOEF = 1.
      RETURN
2     BCOEF = 0.
      RETURN
102   BCOEF = 0.
      RETURN
103   IF(X.GT.0. .AND. X.LT..25)GO TO 11
      IF(X.GT..75 .AND. X.LT.1.) GO TO 11
      BCOEF = 0.
      RETURN
11    IF(Y .EQ. 0. ) GO TO 22
      BCOEF = 0.
      RETURN
22    BCOEF = 1.
      RETURN
      END
```

```
      ***** PROBLEM  3  DATA *****
            FUNCTION F(X,Y,J)
            GO TO (101,102),J
101         F = -20.
            RETURN
102         F = 0.
            RETURN
            END
            FUNCTION COEF(X,Y,J)
            GO TO (1,2,3,4,5),J
1           COEF = G(X,Y)
            RETURN
2           COEF = G(X,Y)
            RETURN
3           COEF = 0.
            RETURN
4           COEF = 0.
            RETURN
5           COEF = 0.
            RETURN
            END
            FUNCTION G(X,Y)
            E = .00001
            X1 = .5-E
            X2 = .5 + E
            DX = X2 - X1
            IF( X .LE. X1 ) GO TO 1
            IF( X .GE. X2 ) GO TO 2
            POL = 3.-6.*(X-X1)**2/(DX*DX)+4.*(X-X1)**3/(DX**3)
            G = 1./POL
            RETURN
1           G = 1./3.
            RETURN
2           G = 1.
            RETURN
            END
            FUNCTION BCOEF(X,Y,J)
            GO TO (101,102,103),J
101         IF(X .GT. 0. .AND. X .LT. .25) GO TO 1
            IF(X .GT. .75 .AND. X .LT. 1.) GO TO 1
            BCOEF = 1.
            RETURN
1           IF(Y .EQ. 0.) GO TO 2
            BCOEF = 1.
            RETURN
2           BCOEF = 0.
            RETURN
102         BCOEF = 0.
            RETURN
103         IF(X.GT.0. .AND. X.LT..25)GO TO 11
            IF(X.GT..75 .AND. X.LT.1.) GO TO 11
            BCOEF = 0.
            RETURN
11          IF(Y .EQ. 0. ) GO TO 22
            BCOEF = 0.
            RETURN
22          BCOEF = 1.
            RETURN
            END
```

```
***** PROBLEM  4  DATA *****
      FUNCTION TRUE(X,Y)
      TRUE = (EXP(X) + EXP(Y))/(1. + X*Y)
      RETURN
      END
      FUNCTION DXTRUE(X,Y)
      Z = 1./(1.+X*Y)
      DXTRUE = EXP(X)*Z-TRUE(X,Y)*Y*Z
      RETURN
      END
      FUNCTION DYTRUE(X,Y)
      Z = 1./(1.+X*Y)
      DYTRUE = EXP(Y)*Z -TRUE(X,Y)*X*Z
      RETURN
      END
      FUNCTION DXYTR(X,Y)
      Z = 1./(1.+X*Y)
      DXYTR = -(EXP(X)*X+EXP(Y)*Y)*Z*Z
     $ -TRUE(X,Y)*Z+2.*TRUE(X,Y)*X*Y*Z**2
      RETURN
       END
      FUNCTION F(X,Y,J)
      EX = EXP(X)
      EY = EXP(Y)
      Z = 1./(1. + X*Y)
      GO TO (1,2) , J
1     F = (EX + EY - 2.*Z*(Y*EX+X*EY-Z*(EX+EY)*(X*X+Y*Y)))*Z
      RETURN
2     F = (EX+EY)*Z
      RETURN
      END
      FUNCTION COEF(X,Y,J)
      GO TO (1,2,3,4,5),J
1     COEF = 1.
      RETURN
2     COEF = 1.
       RETURN
3     COEF = 0.
      RETURN
4     COEF = 0.
      RETURN
5     COEF = 0.
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (1,2,3),J
1     BCOEF = 1.
      RETURN
2     BCOEF = 0.
      RETURN
3     BCOEF = 0.
      RETURN
      END
```

```
***** PROBLEM  5  DATA *****
      FUNCTION TRUE(X,Y)
      TRUE = ATAN(Y/X) + 1.
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO (101,102) , J
101   F = 0.
      RETURN
102   IF(X.EQ..5 .AND. Y.EQ.0.)  GO TO 1
      F = - (Y - X)/(X  + Y - .25)
      RETURN
1     F = -1.
      RETURN
      END
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF = 1.
      RETURN
102   COEF = 1.
      RETURN
103   COEF = 0.
      RETURN
104   COEF = 0.
      RETURN
105   COEF = 0.
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (1,2,3) ,J
1     IF(X.EQ..5 .AND. Y.EQ.0.) GOTO 11
      BCOEF = 0.
      RETURN
11    BCOEF = -1.
      RETURN
2     IF(X.EQ..5 .AND. Y.EQ.0.)  GO TO 22
      BCOEF = Y/.5 - 1.
      RETURN
3     IF( X.EQ. .5 .AND. Y.EQ.0.) GO TO 33
      BCOEF = X/.5 -1.
      RETURN
33    BCOEF = 0.
      RETURN
      END
```

```
***** PROBLEM  6  DATA *****
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF = 1.
      RETURN
102   COEF = 1.+Y*Y
      RETURN
103   COEF = -1.
      RETURN
104   COEF = -(1.+Y*Y)
      RETURN
105   COEF = 0.
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO (101,102) , J
101   F = (-4.*X**4+18.*X**2-14.*X+2.)*ALOG(1.+Y*Y)-
     $     2.*((X*X-X)**2)*(Y*Y+Y**3+Y-1.)/(1.+Y*Y)
      RETURN
102   IF(X.EQ.0. .OR. Y.EQ.0.) GO TO 1
      F = (ALOG(2.)-1.)*(X*X-X)**2
      RETURN
1     F = 2.*EXP(X+Y)
      RETURN
      END
      FUNCTION TRUE(X,Y)
      TRUE = EXP(X+Y)+((X*X-X)**2)*ALOG(1.+Y*Y)
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   BCOEF = 1.
      RETURN
102   IF(X.EQ.0.) GO TO 1
      IF(X.EQ.1.) GO TO 2
      BCOEF = 0.
      RETURN
1     BCOEF = 1.
      RETURN
2     BCOEF = -1.
      RETURN
103   IF(Y.EQ.0.) GO TO 11
       IF(Y.EQ.1.) GO TO 12
      BCOEF = 0.
      RETURN
11    BCOEF = 1.
      RETURN
12    BCOEF = -1.
      RETURN
      END
```

```
***** PROBLEM  7  DATA *****
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF = 1.
      RETURN
102   COEF = 1.
      RETURN
103   COEF = 0.
      RETURN
104   COEF = 0.
      RETURN
105   COEF = 0.
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO (101,102) , J
101   F = 6.*X*Y*EXP(X)*EXP(Y)*(X*Y+X+Y-3.)
      RETURN
102   F = 0.
      RETURN
      END
      FUNCTION TRUE(X,Y)
      TRUE = 3.*EXP(X)*EXP(Y)*(X-1.)*X*(Y-1.)*Y
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   BCOEF = 1.
      RETURN
102   BCOEF = 0.
      RETURN
103   BCOEF = 0.
      RETURN
      END
```

```
***** PROBLEM  8  DATA *****
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF = 1.
      RETURN
102   COEF = 1.
      RETURN
103   COEF = 0.
      RETURN
104   COEF = 0.
      RETURN
105   COEF = 0.
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO (101,102) , J
101   XR = SQRT(X)
      YR = SQRT(Y)
      F = 3.75 * (XR * YR * (X**X + Y**Y) - XR * Y - X * YR)
      RETURN
102   F = 0.
      RETURN
      END
      FUNCTION TRUE(X,Y)
      XR = SQRT(X)
      YR = SQRT(Y)
      TRUE = XR**X**X**YR**Y**Y - X**YR**Y**Y -XR**X**X**Y + X**Y
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   BCOEF = 1.
      RETURN
102   BCOEF = 0.
      RETURN
103   BCOEF = 0.
      RETURN
      END
```

```
*****  PROBLEM  9  DATA  *****
      FUNCTION TRUE(X,Y)
      PI = 3.14159265358979
      TRUE = 4.*(X*X-X)*(COS(2.*PI*Y)-1.)
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO (101,102),J
101   PI = 3.14159265358979
      F = (32. + (256.+16.*PI*PI)*(X-X*X))*
     $    COS(2.*PI*Y)+256.*(X*X-X) -32.
      RETURN
102   F = 0.
      RETURN
      END
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF = 4.
      RETURN
102   COEF = 1.
      RETURN
103   COEF = 0.
      RETURN
104   COEF = 0.
      RETURN
105   COEF = -64.
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   BCOEF = 1.
      RETURN
102   BCOEF = 0.
      RETURN
103   BCOEF = 0.
      RETURN
      END
```

```
***** PROBLEM 10  DATA *****
      FUNCTION TRUE(X,Y)
      PI=3.141592653589793
      FOURP=4.*PI
      FPX=FOURP*X
      FPY=FOURP*Y
      CX=COS(FPX)
      CY=COS(FPY)
      F1=-CX+5.4
      F2=-CY+5.4
      F3= (X-.5)*(X-.5)+(Y-.5)*(Y-.5)
      F32=16.*F3*F3
      F34=F32*F32
      Z=1./(1.+F34)
      F4=Z-.5
      SPX=SIN(PI*X)
      GOFY=Y*Y-Y
      TRUE=F1*SPX*GOFY*F2*F4
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO (101,102),J
  101 PI=3.141592653589793
      FOURP=4.*PI
      FPX=FOURP*X
      FPY=FOURP*Y
      SXTPSQ=FOURP*FOURP
      SX=SIN(FPX)
      SY=SIN(FPY)
      CX=COS(FPX)
      CY=COS(FPY)
      F1=-CX+5.4
      F2=-CY+5.4
      DXF1=FOURP*SX
      DYF2=FOURP*SY
      DDXF1=SXTPSQ*CX
      DDYF2=SXTPSQ*CY
      F3= (X-.5)*(X-.5)+(Y-.5)*(Y-.5)
      F32=16.*F3*F3
      F33= F32*4.*F3
      F34=F32*F32
      Z=1./(1.+F34)
      F4=Z-.5
      DXF3= 2.*(X-.5)
      DYF3= 2.*(Y-.5)
      DDF3=2.
      ZZ=Z*Z
      W=F32*ZZ
      W32=F33*ZZ
      W6=W32*F33*Z
      DF4=-16.*W32
      DXF4=DF4*DXF3
      DYF4=DF4*DYF3
      A1=-192.*W
      A2=-16.*W32*DDF3
      A3=512.*W6
      DXF3S=DXF3*DXF3
      DYF3S=DYF3*DYF3
      DDXF4=(A1+A3)*DXF3S+A2
      DDYF4=(A1+A3)*DYF3S+A2
      SPX=SIN(PI*X)
      PICPX=PI*COS(PI*X)
      GOFY=Y*Y-Y
      DGOFY=2.*Y-1.
      UXX=DDXF1*SPX*GOFY*F2*F4
     *+DXF1*PICPX*GOFY*F2*F4
     *+DXF1*SPX*GOFY*DXF4*F2
     *+DXF1*PICPX*GOFY*F2*F4
     *   -F1*PI*PI*SPX*GOFY*F2*F4
     *   +F1*PICPX*GOFY*F2*DXF4
     *+DXF1*SPX*GOFY*F2*DXF4
     *+F1*PICPX*GOFY*F2*DXF4
```

```
     *+F1*SPX*GOFY*F2*DDXF4
      UYY=F1*SPX*DGOFY*DYF2*F4
     *+F1*SPX*2.*F2*F4
     *+F1*SPX*DGOFY*F2*DYF4
     *+F1*SPX*DGOFY*DYF2*F4
     *+F1*SPX*GOFY*DDYF2*F4
     *+F1*SPX*GOFY*DYF2*DYF4
     *+F1*SPX*DGOFY*F2*DYF4
     *+F1*SPX*GOFY*DYF2*DYF4
     *+F1*SPX*GOFY*F2*DDYF4
      A=100.+COS(2.*PI*X)+SIN(3.*PI*Y)
      A = - A
      U=TRUE(X,Y)
      F=UXX+UYY+A*U
      RETURN
102   F=0.
      RETURN
       END
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF=1.
      RETURN
102   COEF=1.
      RETURN
103   COEF=0.
      RETURN
104   COEF=0.
      RETURN
105   PI=3.141592653589793
      COEF=100.+COS(2.*PI*X)+SIN(3.*PI*Y)
      COEF = -COEF
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (1,2,3),J
1     BCOEF=1.
      RETURN
2     BCOEF=0.
      RETURN
3     BCOEF=0.
      RETURN
      END
```

```
***** PROBLEM  11  DATA *****
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF = 1.
      RETURN
102   COEF = 1.
      RETURN
103   COEF = 0.
      RETURN
104   COEF = 0.
      RETURN
105   COEF = -100.
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO (101,102) , J
101   F = 0.
      RETURN
102   F = TRUE(X,Y)
      RETURN
      END
      FUNCTION TRUE(X,Y)
      TRUE = (COSH(10.*X)+COSH(10.*Y))/COSH(10.)
      RETURN
      END
      FUNCTION COSH(X)
      COSH = (EXP(X)+EXP(-X))/2.
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   BCOEF = 1.
      RETURN
102   BCOEF = 0.
      RETURN
103   BCOEF = 0.
      RETURN
      END
```

```
***** PROBLEM  12  DATA *****
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF = 1.
      RETURN
102   COEF = 1.
      RETURN
103   COEF = 0.
      RETURN
104   COEF = 0.
      RETURN
105   COEF = -100.
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO (101,102) , J
101   F = 300.*COSH(20.*Y)/COSH(20.)
      RETURN
102   F = TRUE(X,Y)
      RETURN
      END
      FUNCTION TRUE(X,Y)
      TRUE = COSH(10.*X)/COSH(10.)+COSH(20.*Y)/COSH(20.)
      RETURN
      END
      FUNCTION COSH(X)
      COSH = (EXP(X)+EXP(-X))/2.
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   BCOEF = 1.
      RETURN
102   BCOEF = 0.
      RETURN
103   BCOEF = 0.
      RETURN
      END
```

```
      ***** PROBLEM  13  DATA *****
            FUNCTION COEF(X,Y,J)
            GO TO (101,102,103,104,105),J
      101   COEF = 1.
            RETURN
      102   COEF = 1.
            RETURN
      103   COEF = 0.
            RETURN
      104   COEF = 0.
            RETURN
      105   COEF = 0.
            RETURN
            END
            FUNCTION F(X,Y,J)
            GO TO (101,102) , J
      101   F = D2P(X)*P(Y) + P(X)*D2P(Y)
            RETURN
      102   F = TRUE(X,Y)
            RETURN
            END
            FUNCTION TRUE(X,Y)
            TRUE = P(X)*P(Y)
            RETURN
            END
            FUNCTION BCOEF(X,Y,J)
            GO TO (101,102,103),J
      101   BCOEF = 1.
            RETURN
      102   BCOEF = 0.
            RETURN
      103   BCOEF = 0.
            RETURN
            END
            FUNCTION P(X)
            A = 1.
            B = 0.
            E = .15
            X1 = .5 - E
            X2 = .5 + E
            IF(X .LT. X1) GO TO 1
            IF(X .GT. X2) GO TO 2
            DPHI = B - A
            DX = X2 - X1
            P = A + DPHI*(X-X1)**3/(DX**3)-3.*DPHI*(X-X1)**3*(X-X2)
           $ /DX**4 + 6.*DPHI*(X-X1)**3*(X-X2)**2/DX**5
            RETURN
        1   P = A
            RETURN
        2   P = B
            RETURN
            END
            FUNCTION D2P(X)
            A = 1.
            B = 0.
            E = .15
            X1 = .5 - E
            X2 = .5 + E
            IF(X .LT. X1) GO TO 1
            IF(X .GT. X2) GO TO 1
            DPHI = B - A
            DX = X2 - X1
            C3 = DPHI/DX**3
            C4 = -3.*DPHI/DX**4
            C5 = 6.*DPHI/DX**5
            D2P = 6.*C3*(X-X1)+6.*C4*(X-X1)*(X-X2)+
           $      6.*C4*(X-X1)**2+6.*C5*(X-X1)*(X-X2)**2+
           $      12.*C5*(X-X1)**2*(X-X2)
           $   +      2.*C5*(X-X1)**3
            RETURN
        1   D2P = 0.
            RETURN
            END
```

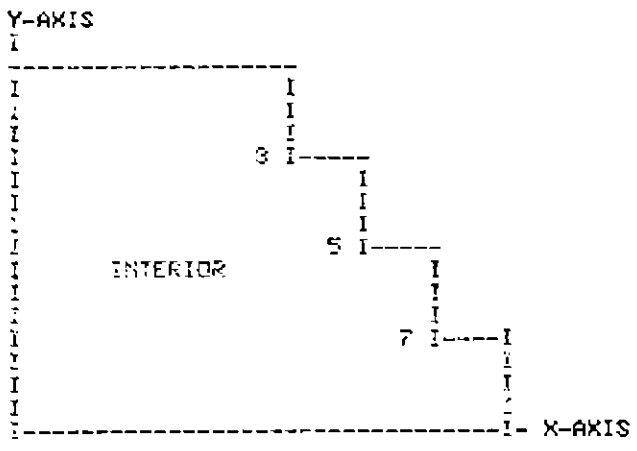
```
*****  PROBLEM  14  DATA  *****
      FUNCTION TRUE(T,S)
      E = .0625
      X = 4.*T
      Y = 4.*S
      T1 = 7.*Y*((X-2.)**2+Y*Y-1.)
      T2 = EXP(-E*(Y-2.)*X*(X-4.))
      T3 = ((X-2.)**2+3.)*(Y*Y+3.)
      TRUE = T1*T2/T3
      RETURN
       END
      FUNCTION F(T,S,J)
      E = .0625
      GO TO(101,102),J
101   X = 4.*T
      Y = 4.*S
      F1=7.*Y
      F2=(X-2.)**2+Y*Y-1.
      F3=EXP(-E*(Y-2.)*X*(X-4.))
      F4=1./((X-2.)**2+3.)
      F5=1./(Y*Y+3.)
      DXF1=0.
      DXF2=2.*(X-2.)
      DXF3=(Y-2.)*(X-4.)*F3+(Y-2.)*X*F3
      DXF3 = -E* DXF3
      DXF4=-2.*(X-2.)/((X-2.)**2+3.)**2
      DXF5=0.
      DX2F1=0.
      DX2F2=2.
      DX2F3=(Y-2.)*F3+(Y-2.)*(X-4.)*DXF3+(Y-2.)*F3+(Y-2.)*X*DXF3
      DX2F3 = -E* DX2F3
      DX2F4=6.*((X-2.)**2-1.)/((X-2.)**2+3.)**3
      DX2F5=0.
      DYF1 = 7.
      DYF2=2.*Y
      DYF3=-E*X*(X-4.)*F3
      DYF4=0.
      DYF5=-2.*Y/(Y*Y+3.)**2
      DY2F1=0.
      DY2F2=2.
      DY2F3=E*E*X*X*(X-4.)**2*F3
      DY2F4=0.
      DY2F5=6.*(Y*Y-1.)/(Y*Y+3.)**3
       T1=F1*DX2F2*F3*F4*F5+F1*DXF2*DXF3*F4*F5+
     $    F1*DXF2*DXF3*F4*F5+F1*F2*DX2F3*F4*F5+
     $    F1*DXF2*F3*DXF4*F5+F1*F2*DXF3*DXF4*F5
     $    +F1*DXF2*F3*DXF4*F5+F1*F2*DXF3*DXF4*F5+
     $    F1*F2*F3*DX2F4*F5
      T2=DYF1*DYF2*F3*F4*F5+DYF1*F2*DYF3*F4*F5+
     $    DYF1*F2*F3*DYF4*F5+DYF1*F2*F3*F4*DYF5
      T3=DYF1*DYF2*F3*F4*F5+F1*DY2F2*F3*F4*F5+
     $    F1*DYF2*DYF3*F4*F5+F1*DYF2*F3*F4*DYF5
      T4=DYF1*F2*DYF3*F4*F5+F1*DYF2*DYF3*F4*F5+
     $    F1*F2*DY2F3*F4*F5+F1*F2*DYF3*F4*DYF5
      T5=DYF1*F2*F3*F4*DYF5+F1*DYF2*F3*F4*DYF5+
     $    F1*F2*DYF3*F4*DYF5+F1*F2*F3*F4*DY2F5
      F=(T1+T2+T3+T4+T5)*16.
      RETURN
102   F=TRUE(T,S)
      RETURN
      END
      FUNCTION COEF(X,Y,J)
      GO TO (1,2,3,4,5),J
1     COEF=1.
      RETURN
2     COEF=1.
      RETURN
3     COEF=0.
      RETURN
4     COEF=0.
      RETURN
5     COEF=0.
```

```
      RETURN
      END
      FUNCTION BCDEF(X,Y,J)
      GO TO (1,2,3),J
1     BCDEF=1.
      RETURN
2     BCDEF=0.
      RETURN
3     BCDEF=0.
      RETURN
      END
```

```
*****  PROBLEM  15  DATA  *****
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF = 1.
      RETURN
102   COEF = 1.
      RETURN
103   COEF = 0.
      RETURN
104   COEF = 0.
      RETURN
105   COEF = 0.
      RETURN
      END
      FUNCTION F(X,Y,J)
      P = .1
      GO TO (101,102) , J
101   TEMP = -((X-.5)**2+(Y-.5)**2)/P**2
      F1 = EXP(TEMP)
      DXU = -2.*(X-.5)*TRUE(X,Y)/P**2 +
     $ F1*(2.*X-1.)*(Y-1.)*Y/P
      DX2U = -2.*(TRUE(X,Y)+(X-.5)*DXU)/P**2
     $-2.*(X-.5)*F1*(2.*X-1.)*(Y-1.)*Y/P**3 +
     $ 2.*F1*(Y-1.)*Y/P
      DYU = -2.*(Y-.5)*TRUE(X,Y)/P**2 +
     $ F1*(2.*Y-1.)*(X-1.)*X/P
      DY2U = -2.*(TRUE(X,Y)+(Y-.5)*DYU)/P**2
     $-2.*(Y-.5)*F1*(2.*Y-1.)*(X-1.)*X/P**3 +
     $ 2.*F1*(X-1.)*X/P
      F = (DX2U+DY2U)
      RETURN
102   F = 0.
      RETURN
      END
      FUNCTION TRUE(X,Y)
      P = .1
      TEMP = -((X-.5)**2+(Y-.5)**2)/P**2
      TRUE = EXP(TEMP)*(X-1.)*X*(Y-1.)*Y/P
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   BCOEF = 1.
      RETURN
102   BCOEF = 0.
      RETURN
103   BCOEF = 0.
      RETURN
      END
```

```
***** PROBLEM 16  DATA *****
      FUNCTION COEF(X,Y,J)
      GO TO (101,102,103,104,105),J
101   COEF = 1.
      RETURN
102   COEF = 1.
      RETURN
103   COEF = 0.
      RETURN
104   COEF = 0.
      RETURN
105   COEF = 0.
      RETURN
      END
      FUNCTION F(X,Y,J)
      GO TO(101,102),J
101   F = 2.*TRUE(X,Y)
      RETURN
102   F = TRUE(X,Y)
      RETURN
      END
      FUNCTION TRUE(X,Y)
      TRUE = EXP(X+Y)
      RETURN
      END
      FUNCTION BCOEF(X,Y,J)
      GO TO (101,102,103),J
101   BCOEF = 1.
      RETURN
102   BCOEF = 0.
      RETURN
103   BCOEF = 0.
      RETURN
      END
```

```
      FUNCTION F(X,Y,J)

      F COMPUTES EITHER THE TRUE SOLUTION OF PROBLEM 17 OR
      THE LAPLACIAN UXX + UYY .
      LOGICAL NODERV

      IF J .EQ. 1, THEN EVALUATE THE LAPLACIAN
                   ELSE EVALUATE THE TRUE SOLUTION

      NODERV = .TRUE.
      IF( J .EQ. 1 ) NODERV = .FALSE.

      CALL EVLATE(X,Y,NODERV,
     A            Q,QX,QY,QXX,QYY, IFLAG )

      F = 17770000000000000000B
      IF( NODERV ) F = Q
      IF( .NOT. NODERV  .AND.  IFLAG .EQ. 0 ) F = QXX + QYY

      RETURN
      END
      SUBROUTINE EVLATE(XX,YY,NODIN,
     A                  QV,QVX,QVY,QVXX,QVYY,IFLAG)

      INPUT  XX,YY,NODIN,ONLYNM,BUGALL,BUGGVAL
        LOCAL VARIABLES  X,Y,NODERV  FOR  XX,YY,NODIN
      OUTPUT QV,QVX,QVY,QVXX,QVYY,IFLAG

      IFLAG SET TO  0  IF SUCCESSFUL,  SET TO  1  IF NOT
        WHEN UNSUCCESSFUL, QV SET TO ZERO AND QVX,QVY,QVXX,QVYY
        ARE SET TO  INDEFINITE, THIS OCCURS AT REENTRANT BOUNDARY
        CORNERS

      EVALUATES  QV, QVX, QVY, QVXX, QVYY

      QV = SIZE*QVAL
        SIZE IS CONSTANT SET IN DATA
        QVAL = GVAL(0,0,0)/GVAL(A,B,C)
          A,B,C, ARE CONSTANT SET IN DATA
          GVAL = BVAL*FVAL(3)*FVAL(5)*FVAL(7)
            BVAL = (X - 1)*(Y - 1) + A*CIR2
              CIR2 = X**2 + Y**2 - RHOSQR
                RHOSQR IS CONSTANT SET IN DATA
              FVAL(I) = RDVAL(I)*THVAL(I)
                RDVAL(I) = ( ( X - XPT(I) )**2 + ( Y - YPT(I) )**2
                           + B*CIR2 )**(1/3)
                  XPT(I),XPT(I) IS I-TH REENTRANT CORNER POINT
                THVAL = SNVAL + C*CIR2
                  SNVAL(I) = SIN( 2*ANVAL(I)/3 )
                    ANVAL(I) = ARCTAN( (Y-YPT(I))/(X-XPT(I)) ) - PI/2
```



```
 COMMON /SUBCOM/
A PI,          PID2,       PID4,       TWOPI,      THPID2,
B TWTH,        TWTHSQ,     ONETHR,     FRTH,       NODERV,
C XD,          YD,         XDSQR,       YDSQR,       RDSQR,
D X          , Y          , XSQR       , YSQR       , ANGLE      ,
D SIZE       , A          , B          , C          ,
I RHO        , RHOSQR     , XPT(3)     , YPT(3)     , JUNK(20)   ,
E ANVAL      , ANVALX     , ANVALY     , ANVALXX    , ANVALYY    ,
G BVAL(2)    , BVALX(2)   , BVALY(2)   , BVALXX(2)  , BVALYY(2)  ,
H CIR        , CIRX       , CIRY       , CIRXX      , CIRYY      ,
I CIR2       , CIR2X      , CIR2Y      , CIR2XX     , CIR2YY     ,
J CT(2)      , CTX(2)     , CTY(2)     , CTXX(2)    , CTYY(2)    ,
K FVAL(2,3), FVALX(2,3), FVALY(2,3), FVALXX(2,3), FVALYY(2,3),
L GVAL(2)    , GVALX(2)   , GVALY(2)   , GVALXX(2)  , GVALYY(2)  ,
M RDVAL(2)   , RDVALX(2)  , RDVALY(2)  , RDVALXX(2) , RDVALYY(2) ,
N SNVAL      , DSNVAL     , DDSNVAL    ,
P THVAL(2)   , THVALX(2)  , THVALY(2)  , THVALXX(2) , THVALYY(2)

 DIMENSION CTALL(2,5), CR2ALL(5)
 EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

 DATA CTALL / 10*0.0 /

 LOGICAL NODERV, NODIN, ONLYNM, BUGALL, BUGGVL

 REAL INDEF
 DATA INDEF / 1777000000000000000B /

 DATA PI,PID2,PID4,  TWOPI,THPID2,TWTH,  TWTHSQ,ONETHR,FRTH /
A   3.14159265358979,    1.57079632679490,     .78539816339745,
B   6.28318530717959,    4.71238898038468,     .66666666666667,
C    .44444444444444,     .33333333333333,    1.33333333333333 /

 DATA SIZE, A, B, C, RHO, RHOSQR / 100., -.5, .1, 7., .8, .64 /

 DATA XPT / .65 , .85 , .95 /
 DATA YPT / .7 , .5 , .3 /

 MAKE  XX,YY,NODIN  LOCAL IN  /SUBCOM/
    X      = XX
    Y      = YY
    NODERV = NODIN

    XSQR   = X*X
    YSQR   = Y*Y
    CALL CIRCLE
    CALL BVALS

    DO 20 I = 1 , 3
      XD    = X - XPT(I)
      YD    = Y - YPT(I)
      XDSQR = XD*XD
      YDSQR = YD*YD
      RDSQR = XDSQR + YDSQR
      IF( RDSQR .GT. 1.E-8 )                                    GO TO 10
        THEN TOO CLOSE TO  I-TH  BOUNDARY CORNER
          IFLAG = 1
          QV    = 0.
          QVX   = INDEF
          QVY   = INDEF
          QVXX  = INDEF
          QVYY  = INDEF
                                                                EXIT
                                                                GO TO 30
10      CONTINUE
        ELSE CAN EVALUATE
          CALL ANVALS
          CALL SNVALS
          CALL RDVALS
```

```
        CALL THVALS
   J = I
        CALL FVALS(FVAL(1,J),FVALX(1,J),FVALY(1,J),
  A                FVALXX(1,J),FVALYY(1,J))
        IF( BUGALL ) CALL DEBUG(1)
20     CONTINUE

   CALL GVALS
   CALL QVALS(QV,QVX,QVY,QVXX,QVYY)

   QV    = SIZE*QV

   IFLAG = 0

30 CONTINUE
                                                              RETURN
                                                              END
   SUBROUTINE CIRCLE

   FORM  CIR  = X**2 + Y**2 - RHOSQR
         CIR2 = CIR**2
         AND DERIVATIVES

   COMMON /SUBCOM/ *** REPEAT VARIABLES HERE ***

   DIMENSION CTALL(2,5), CR2ALL(5)
   EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

   LOGICAL NODERV

   CIR    = XSQR + YSQR - RHOSQR
   CIR2   = CIR**2
   IF( NODERV )                                               GO TO 10
     CIRX   = 2.*X
     CIRY   = 2.*Y
     CIRXX  = 2.
     CIRYY  = 2.

     CIR2X  = 2.*CIR*CIRX
     CIR2Y  = 2.*CIR*CIRY
     CIR2XX = 4.*(3.*XSQR +    YSQR - RHOSQR )
     CIR2YY = 4.*(   XSQR + 3.*YSQR - RHOSQR )
10   CONTINUE
                                                              RETURN
                                                              END
   SUBROUTINE BVALS

   FORM  BVAL  = (X-1 + A*CIR2)*(Y-1 + A*CIR2)
         AND DERIVATIVES

   COMMON /SUBCOM/ *** REPEAT VARIABLES HERE ***

   DIMENSION CTALL(2,5), CR2ALL(5)
   EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

   LOGICAL NODERV

   XM1     = X - 1.
   YM1     = Y - 1.
   ACIR2   = A*CIR2
   XFACT   = XM1 + ACIR2
   YFACT   = YM1 + ACIR2
   BVAL(1) = XM1*YM1
   BVAL(2) = XFACT*YFACT

   IF( NODERV )                                               GO TO 10

     BVALX(1)  = YM1
     BVALY(1)  = XM1
     BVALXX(1) = 0.
     BVALYY(1) = 0.
```

```
      XFACTX  = 1. + A*CIR2X
      YFACTY  = 1. + A*CIR2Y
      XFACTY  =      A*CIR2Y
      YFACTX  =      A*CIR2X
      XFACTXX =      A*CIR2XX
      YFACTYY =      A*CIR2YY
      XFACTYY = YFACTYY
      YFACTXX = XFACTXX

      BVALX(2)  = XFACTX*YFACT + XFACT*YFACTX
      BVALY(2)  = XFACTY*YFACT + XFACT*YFACTY
      BVALXX(2) = XFACTXX*YFACT + 2.*XFACTX*YFACTX + XFACT*YFACTXX
      BVALYY(2) = XFACTYY*YFACT + 2.*XFACTY*YFACTY + XFACT*YFACTYY
10    CONTINUE

                                                          RETURN
                                                          END
   SUBROUTINE ANVALS

   FORM  ANVAL = ARCTAN( YD/XD ) - PI/2  AND DERIVATIVES

   D(ARCTAN(U/V))/DU = V/(V*V + U*U)

   ANGLE  MEASURED COUNTER-CLOCKWISE FROM  XD-AXIS
   ANVAL  MEASURED COUNTER-CLOCKWISE FROM  YD-AXIS
   BRANCH POINT AT  XD = YD = 0.
   BRANCH CUT ALONG  ANGLE = PI/4, ANVAL = -PI/4

    I                                  I ANVAL = 0
    I Y-AXIS                           I YD-AXIS
    I                                  I ANGLE = PI/2
    I                                  I
    I                                  I
    I                                  I
    I                                  I           ANGLE = 0
    I                                  I---------- XD-AXIS
    I                                              ANGLE = 2PI
    I                                              ANVAL = 3PI/2
    I
    I
    I             . (X,Y)
    I               (XD,YD)
    I
    I
    I
    I
    I
    I
    I------------------------------------------ X-AXIS


   COMMON /SUBCOM/ *** REPEAT VARIABLES HERE ***

   DIMENSION CTALL(2,5), CR2ALL(5)
   EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

   LOGICAL NODERV

   IF( NODERV )                                           GO TO 10
     COMPUTE DERIVATIVES
       ANVALX = -YD/RDSQR
       ANVALY =  XD/RDSQR
       ANVALXX= -2.*XD*ANVALX/RDSQR
       ANVALYY= -2.*YD*ANVALY/RDSQR
10   CONTINUE

   IF( ABS( YD ) .GT. ABS( XD ) )                         GO TO 20
     THEN  ANGLE  BETWEEN     0 AND PI/4  OR  3PI/4 AND 5PI/4
                          OR  7PI/4 AND 2PI
       ANGLE = ATAN( YD/XD )
       IF(    XD .LT. 0. ) ANGLE = PI    + ANGLE
```

```
        IF( ANGLE .LT. 0. ) ANGLE = TWOPI + ANGLE
                                                                GO TO 30
20    CONTINUE
      ELSE  ANGLE  BETWEEN  PI/4 AND 3PI/4  OR  5PI/4 AND 7PI/4
        ANGLE = PID2 - ATAN( XD/YD )
        IF(    YD .LT. 0. ) ANGLE = PI + ANGLE
30    CONTINUE

    SUBTRACT  PI/2  TO MAKE  ANVAL  BETWEEN  0 AND 3PI/2

    ANVAL = ANGLE - PID2

    ADJUST FOR BRANCH CUT

    IF( ANVAL .LT. -PID4 ) ANVAL = TWOPI + ANVAL
                                                                RETURN
                                                                END
    SUBROUTINE SNVALS

    FORM  SNVAL = SIN( 2*ANVAL/3 ), DSNVAL, DDSNVAL

    COMMON /SUBCOM/ *** REPEAT VARIABLES HERE ***

    DIMENSION CTALL(2,5), CR2ALL(5)
    EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

    LOGICAL NODERV

    ARG   = TWTH*ANVAL
    SNVAL = SIN( ARG )
    IF( NODERV )                                                GO TO 10
      COMPUTE DERIVATIVES
        DSNVAL  = TWTH*COS(ARG)
        DDSNVAL = -TWTHSQ*SNVAL
10    CONTINUE
                                                                RETURN
                                                                END
    SUBROUTINE THVALS

    FORM  THVAL = SNVAL + C*CIR2  AND DERIVATIVES

      C = 0.  FOR NUMERATOR

    COMMON /SUBCOM/ *** REPEAT VARIABLES HERE ***

    DIMENSION CTALL(2,5), CR2ALL(5)
    EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

    LOGICAL NODERV

    THVAL(1) = SNVAL
    THVAL(2) = THVAL(1) + C*CIR2
    IF( NODERV )                                                GO TO 10
      THVALX (1) =  DSNVAL*ANVALX
      THVALY (1) =  DSNVAL*ANVALY
      THVALXX(1) = DDSNVAL*(ANVALX**2) + DSNVAL*ANVALXX
      THVALYY(1) = DDSNVAL*(ANVALY**2) + DSNVAL*ANVALYY

      THVALX (2) = THVALX (1) + C*CIR2X
      THVALY (2) = THVALY (1) + C*CIR2Y
      THVALXX(2) = THVALXX(1) + C*CIR2XX
      THVALYY(2) = THVALYY(1) + C*CIR2YY
10    CONTINUE
                                                                RETURN
                                                                END
    SUBROUTINE RDVALS

    FORM  RDVAL = (RDSQR + B*CIR2)**(1/3)  AND DERIVATIVES
            B = 0. FOR NUMERATOR

    COMMON /SUBCOM/ *** REPEAT VARIABLES HERE ***
```

```
      DIMENSION CTALL(2,5), CR2ALL(5)
      EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

      LOGICAL NODERV

      SET CIRCLE-TEMP FOR DENOMINATOR (NUMERATOR SET TO ZERO IN DATA)

      DO 10 IDERV = 1, 5
        CTALL(2,IDERV) = B*CR2ALL(IDERV)
   10   CONTINUE

      EVALUATE FOR NUMERATOR AND DENOMINATOR (NUM = 1, DEN = 2)

      DO 30 NMDN = 1, 2
        RVALUE       = RDSQR + CT(NMDN)
        RDVAL(NMDN) = RVALUE**ONETHR
        IF( NODERV )                                          GO TO 20
          COMPUTE DERIVATIVES
            T1              =  ONETHR*RDVAL(NMDN)/RVALUE
            T2              = -TWTH*T1/RVALUE
            XDERV           =  2.*XD + CTX(NMDN)
            YDERV           =  2.*YD + CTY(NMDN)
            RDVALX (NMDN) =  T1*XDERV
            RDVALY (NMDN) =  T1*YDERV
            RDVALXX(NMDN) =  T2*XDERV*XDERV + T1*(2. + CTXX(NMDN))
            RDVALYY(NMDN) =  T2*YDERV*YDERV + T1*(2. + CTYY(NMDN))
   20     CONTINUE
   30   CONTINUE
                                                              RETURN
                                                              END

      SUBROUTINE FVALS(FV,FVX,FVY,FVXX,FVYY)

      FORMS  FV = ( RSQ**(1/3) )*SIN( 2*ANVAL/3 )

      COMMON /SUBCOM/ *** REPEAT VARIABLES HERE ***

      DIMENSION CTALL(2,5), CR2ALL(5)
      EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

      LOGICAL NODERV

      DIMENSION FV(2),FVX(2),FVY(2),FVXX(2),FVYY(2)
      FOR NUMERATOR (1) AND DENOMINATOR (2)

      DO 20 N = 1, 2
        FV(N) = RDVAL(N)*THVAL(N)
        IF( NODERV )                                          GO TO 10
          COMPUTE DERIVATIVES
            FVX(N)  = RDVALX(N)*THVAL(N) + RDVAL(N)*THVALX(N)
            FVY(N)  = RDVALY(N)*THVAL(N) + RDVAL(N)*THVALY(N)
            FVXX(N) = RDVALXX(N)*THVAL(N) + 2.*RDVALX(N)*THVALX(N)
     A               +RDVAL(N)*THVALXX(N)
            FVYY(N) = RDVALYY(N)*THVAL(N) + 2.*RDVALY(N)*THVALY(N)
     &               +RDVAL(N)*THVALYY(N)
   10     CONTINUE
   20   CONTINUE
                                                              RETURN
                                                              END

      SUBROUTINE GVALS

      FORM  GVAL = BVAL*F3*F5*F7  AND DERIVATIVES

      COMMON /SUBCOM/ *** REPEAT VARIABLES HERE ***

      DIMENSION CTALL(2,5), CR2ALL(5)
      EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

      LOGICAL NODERV
      DIMENSION F357(2),F357X(2),F357XX(2),F357Y(2),F357YY(2),
     A          F57(2), F57X(2), F57XX(2), F57Y(2), F57YY(2)
```

```
      COMPUTE NUMERATORS (1) AND DENOMINATORS (2)

      DO 20 N = 1, 2
        F57(N)  = F5(N)*F7(N)
        F357(N) = F3(N)*F57(N)
        GVAL(N) = BVAL(N)*F357(N)
        IF( NODERV )                                          GO TO 10
          COMPUTE DERIVATIVES
            F57X(N)   = F5X(N)*F7(N) + F5(N)*F7X(N)
            F57Y(N)   = F5Y(N)*F7(N) + F5(N)*F7Y(N)

            F357X(N)  = F3X(N)*F57(N) + F3(N)*F57X(N)
            F357Y(N)  = F3Y(N)*F57(N) + F3(N)*F57Y(N)

            GVALX(N)  = BVALX(N)*F357(N) + BVAL(N)*F357X(N)
            GVALY(N)  = BVALY(N)*F357(N) + BVAL(N)*F357Y(N)

            F57XX(N)  = F5XX(N)*F7(N) + 2.*F5X(N)*F7X(N)
     A                  +F5(N)*F7XX(N)
            F57YY(N)  = F5YY(N)*F7(N) + 2.*F5Y(N)*F7Y(N)
     A                  +F5(N)*F7YY(N)

            F357XX(N) = F3XX(N)*F57(N) + 2.*F3X(N)*F57X(N)
     A                  +F3(N)*F57XX(N)
            F357YY(N) = F3YY(N)*F57(N) + 2.*F3Y(N)*F57Y(N)
     A                  +F3(N)*F57YY(N)

            GVALXX(N) =  BVALXX(N)*F357(N) + 2.*BVALX(N)*F357X(N)
     A                  +BVAL(N)*F357XX(N)
            GVALYY(N) =  BVALYY(N)*F357(N) + 2.*BVALY(N)*F357Y(N)
     A                  +BVAL(N)*F357YY(N)

 10       CONTINUE
 20   CONTINUE
                                                              RETURN
                                                              END

      SUBROUTINE QVALS(QV,QVX,QVY,QVXX,QVYY)

      FORM  QV = SIZE*GVAL(1)/GVAL(2)
        AND DERIVATIVES

      COMMON /SUBCOM/ *** REPEAT VARIABLES HERE ***

      DIMENSION CTALL(2,5), CR2ALL(5)
      EQUIVALENCE (CTALL(1,1), CT(1)), (CR2ALL(1), CIR2)

      LOGICAL NODERV

      QV = GVAL(1)/GVAL(2)
      IF( NODERV )                                            GO TO 10

        COMPUTE DERIVATIVES.

          FACT   = 1./GVAL(2)
          FACTSQ = FACT*FACT
          FACTX  = -GVALX(2)*FACTSQ
          FACTY  = -GVALY(2)*FACTSQ
          FACTXX = (2.*(GVALX(2)**2)*FACT - GVALXX(2))*FACTSQ
          FACTYY = (2.*(GVALY(2)**2)*FACT - GVALYY(2))*FACTSQ
          QVX    = GVALX(1)*FACT + GVAL(1)*FACTX
          QVY    = GVALY(1)*FACT + GVAL(1)*FACTY
          QVXX   = GVALXX(1)*FACT + 2.*GVALX(1)*FACTX + GVAL(1)*FACTXX
          QVYY   = GVALYY(1)*FACT + 2.*GVALY(1)*FACTY + GVAL(1)*FACTYY
 10   CONTINUE
      RETURN
      END
```