Collocation Software for Second-Order Elliptic Partial Differential Equations

E. N. HOUSTIS
Purdue University and University of Thessaloniki
and
W. F. MITCHELL AND J. R. RICE
Purdue University

We consider the collocation method for linear, second-order elliptic problems on rectangular and general two-dimensional domains. An overview of the method is given for general domains, followed by a discussion of the improved efficiencies and simplifications possible for rectangular domains. A very-high-level description is given of three specific collocation algorithms that use Hermite bicubic basic functions, (1) GENCOL (collocation on general two-dimensional domains), (2) HERMCOL (collocation on rectangular domains with general linear boundary conditions), and (3) INTCOL (collocation on rectangular domains with uncoupled boundary conditions). The linear system resulting from INTCOL has half the bandwidth of that from HERMCOL, which provides substantial benefit in solving the system. We provide some examples showing the range of applicability of the algorithms and some performance profiles illustrating their efficiency. Fortran implementations of these algorithms are given in the companion papers [10, 11].

Categories and Subject Descriptors: G.1.8 [**Numerical Analysis**]: Partial Differential Equations—*elliptic, finite element methods*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Hermite bicubics, rectangular domains, general domains, uncoupled boundary conditions, two-dimensional

1. INTRODUCTION

We consider a linear two-dimensional partial differential equation (PDE),

$$Lu \equiv au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + fu = g, \tag{1.1}$$

over a two-dimensional domain R in the x, y plane. It is assumed that the coefficients a, b, c satisfy the ellipticity condition $b^2 - 4ac < 0$ and that R is given by the boundary ∂R with clockwise orientation given parametrically as follows:

$$\left.\begin{array}{lll} x = x_i(p) & \text{for} & b_{1i} \le p \le b_{2i}, \\ y = y_i(p) & \text{for} & b_{1i} \le p \le b_{2i}, \end{array}\right\}, \quad i = 1, 2, \ldots, nbound. \tag{1.2}$$

The work of the first author was supported in part by the National Science Foundation under Grant MCS-78-04878 and in part by ARO Contract DAAG-29-33-K-0026. The work of the second and third authors was supported in part by Department of Energy Contract DE-AC01-ER81-01997.
Authors' address: Purdue University, Division of Mathematical Sciences, West Lafayette, IN 47907.

R is the interior of the domain defined by these *nbound* pieces; R need not be simply connected, but we ignore that case in this paper in order to simplify the discussion. In the case of a rectangular domain $R = [\mathrm{AX}, \mathrm{BX}] \times [\mathrm{AY}, \mathrm{BY}]$, the boundary ∂R is implicitly defined by the end points AX, BX, AY, BY. Further, we assume that the solution u of (1.1) is subject to the boundary conditions

$$\Lambda u \equiv \alpha u + \beta u_x + \gamma u_y = \delta \qquad \text{for} \quad (x, y) \in \partial R. \tag{1.3}$$

All the coefficients and right-hand sides in (1.1) and (1.3) may depend on x and y.

A large class of methods for approximating the solution u of (1.1), (1.3) involves first, the partition of R into a finite-element mesh Ω and, second, the determination of a piecewise polynomial approximation U defined over the partition Ω. This paper describes such a method, called collocation, and discusses specific implementations.

2. OVERVIEW OF THE COLLOCATION METHOD

Collocation is a finite-element method for approximating the solution $u(x, y)$ of (1.1), (1.3), consisting of the following three conceptual phases.

Phase 1. Overlay the domain of definition R by a rectangular grid G, identify the rectangular elements of G that are interior or exterior to R or that intersect ∂R, and associate boundary pieces with boundary elements. Define Ω, the finite-element mesh, as the union of the boundary elements $\partial\Omega$ and interior elements Ω'. (See Figure 1.)

Phase 2. Approximate $u(x, y)$ by a bicubic Hermite piecewise polynomial $U(x, y)$ over the finite element mesh Ω. Determine $U(x, y)$ so that

(i) for each element E of Ω,

$$[LU - g]|_{P_i} = 0, \qquad i = 1 \text{ to } 4,$$

where the P_i are four interior points of E. The P_i are called *interior collocation points*;

(ii) for each element E of $\partial\Omega$,

$$[\Lambda U - \delta]|_{Q_j} = 0, \qquad j = 1 \text{ to } n(E),$$

where the Q_j are $n(E)$ points of the piece of ∂R associated with E. The Q_j are called *boundary collocation points.*

Phase 3. Solve the resulting linear system from phase 2(i) and (ii) to obtain the coefficients of the approximation $U(x, y)$.

See [7, 8, 15] for further discussion.

We now present a more detailed, but still very-high-level, description of an implementation of the collocation method. The procedure is broken into seven steps given below and these are then described individually.

(1) Define the problem and I/O requirements
(2) Discretize the domain
(3) Generate the finite-element mesh

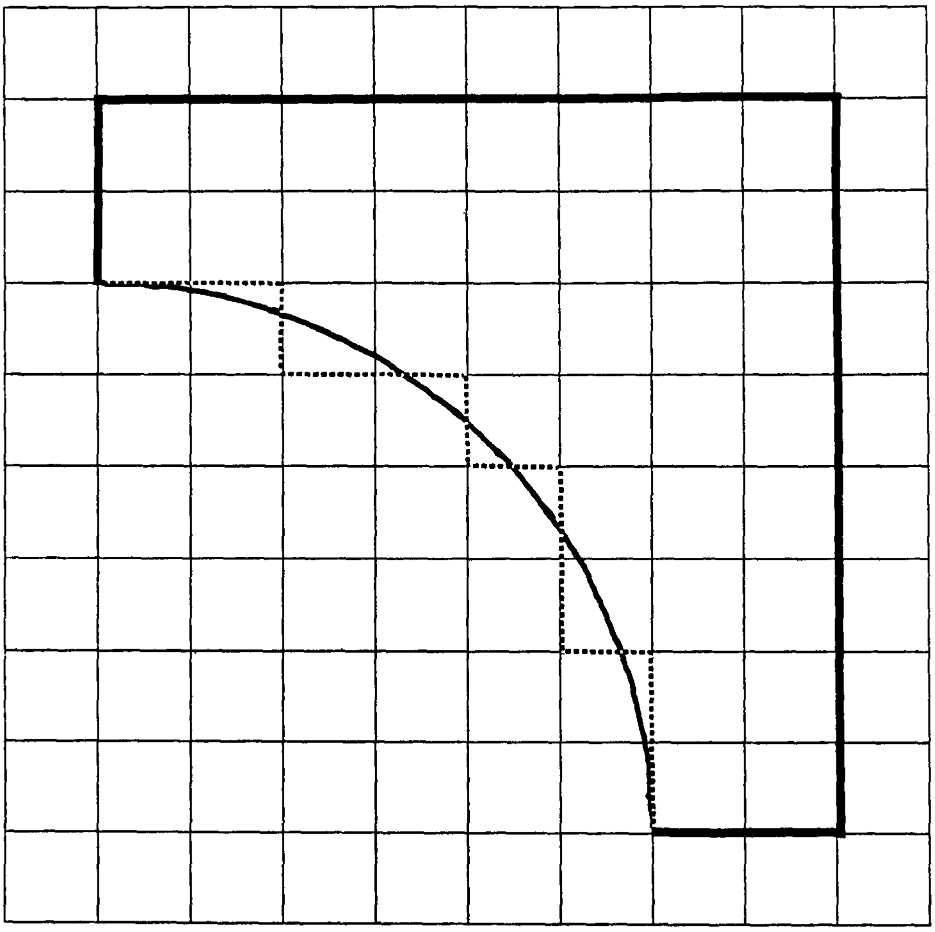


Fig. 1. A domain R (heavy lines) with finite-element mesh Ω embedded in the rectangular grid G (light lines). The boundary sides of Ω are shown dotted where they do not coincide with ∂R.

(4) Define the approximate solution
(5) Form the collocation equations
(6) Reorder the collocation equations
(7) Solve the collocation equations

2.1 Problem Definition and I/O Specification

Following the ELLPACK framework [19], an elliptic PDE problem is specified by a set of functions as follows:

(a) *Operator.* Function PDE(X, Y, CVALUS), where

CVALUS[1:7] = (a, b, c, d, e, f, g) at (x, y) in Ω and
PDE = $g(x, y)$.

(b) *Domain of Problem* (in parametric form)

$$(x(p, i), y(p, i)), \qquad i = 1 \text{ to } nbound, \quad b(1, i) \le p \le b(2, i),$$

where *nbound* is the number of boundary pieces and $b(2, i)$ is the endpoint of the parameter defining the ith boundary piece.

(c) *Boundary conditions.* Function BCOND(I, X, Y, BVALUS) where

BVALUS[1:4] = $(\alpha, \beta, \gamma, \delta)$ at (x, y) on the boundary piece i and BCOND = $\delta(x, y)$.

(d) *Uniqueness condition.* In case of Neumann boundary conditions, a unique solution is determined by knowing the solution at a boundary mesh point $(unqx, unqy)$. The information is supplied by the function $unqu(unqx, unqy)$.

(e) *Output specifications.* The output is specified by two arrays OUTFNC and OUTTYP as follows. The user specifies one of three functions for the ith output through OUTFNC(i) as follows:

$$\text{OUTFNC}(i) = \begin{cases} 1: & \text{approximation solution } U, \\ 2: & \text{error} = U\text{-TRUE}, \\ 3: & \text{residual} = LU - g, \end{cases}$$

where TRUE(x, y) is the exact solution of (1.1), (1.3), a function which must be supplied by the user. Other information for the ith output is specified by OUTTYP(i) as follows:

A grid for output of type 2 and 4 is specified by the parameters

tabx, *taby*:	vector of x and y coordinates of the grid,
ntabx, *ntaby*:	number of grid lines in *tabx* and *taby*.

Finally, the number of output specifications desired is specified by the parameter NOUT.

For example, if OUTFNC(1) = 2, OUTTYP(1) = 1, and NOUT = 1, then one requests the max, L^1, and L^2 norms of the error U-TRUE on the discretization grid.

2.2 Domain Discretization

Information must be generated that relates the problem domain R to the rectangular grid G. This geometric information must be fairly detailed, otherwise large amounts of code will appear in other parts of the implementation just to do a basic analysis of the geometry. We use the two-dimensional domain processor of [17, 18] and, for completeness, briefly describe its input and output here.

The rectangular grid G is defined by the variables

AX, BX:	left and right endpoints of x-interval,
AY, BY:	bottom and top endpoints of y-interval,
NGRIDX:	number of x-grid lines,
NGRIDY:	number of y-grid lines,
GRIDX:	vector of length NGRIDX containing values of x-grid lines,

GRIDY:	vector of length NGRIDY containing values of y-grid lines,
EPSGRD:	accuracy with which geometric data is to be determined.

With this information and the problem definition the domain processor generates the following information. A two-dimensional array which associates the grid points with the domain R and its boundary:

2.2.1 *Grid specification*

$$\text{GTYPE}(i, j),\ i = 1 \text{ to NGRIDX},\ j = 1 \text{ to NGRIDY}$$

The values in GTYPE indicate whether a grid point is interior or exterior and locate it relative to ∂R if it is a neighbor of ∂R.

The domain processor also generates seven one-dimensional arrays of length NBNDPT + 1, where NBNDPT is the number of boundary points. A boundary point is where ∂R intersects the grid G, and these are ordered along ∂R. If ∂R changes pieces off the grid G, then the point of change is also included as a boundary point. If B_i denotes the ith boundary point, the values of the arrays are defined as follows:

2.2.2 *Boundary specification*

XBOUND(i), YBOUND(i):	x and y coordinates of B_i,
BPARAM(i):	parameter value of B_i,
PIECE(i):	index of boundary piece to which B_i belongs (smallest index if there are two),
BPTYPE(i):	type of boundary point (horizontal, vertical, both or interior),
BNEIGH(i):	pointer to interior grid point neighbors of B_i,
BGRID(i):	$ix + 1000 * jy$ if B_i is in the grid element with lower left corner (ix, jy) in G.

The domain processor sets the (NBNDPT + 1)st value of the arrays to the initial values ($i = 1$) and it also requires that NBDIM be set to the actual dimension of above arrays.

Our implementation of the collocation method is based on this information. In the absence of the domain processor, this information must be provided directly as input. In the special case of rectangular domains, the domain discretization is implicitly defined by the vectors GRIDX and GRIDY, and no domain processing is required.

2.3 Finite-Element-Mesh Generation

In the case of rectangular regions, the finite-element mesh Ω coincides with the rectangular overlay G, and no further processing is needed. For nonrectangular regions each element is identified by the indices (ix, jy) of the lower left corner grid point, where

$$1 \le ix < \text{NGRIDX} \quad \text{and} \quad 1 \le jy < \text{NGRIDY}.$$

There may be boundary elements whose intersection with R is very small. In extreme cases, the use of these elements can make later computations numerically unstable. In any case, it is intuitively plausible that very small elements should be discarded just for the sake of efficiency. We thus define the *finite-element* *mesh* Ω to consist of the interior elements, plus those boundary elements E for which the ratio area of $E \cap R$ over area of E is greater than the value of the parameter DSCARE. The portions of ∂R from discarded elements are either allocated to neighboring elements or ignored, depending on the value of the logical variable GIVOPT. (GIVOPT = .TRUE. means to allocate discarded elements to neighboring elements.)

The finite-element mesh Ω forms a rectangular approximation to R. Its exterior sides are called *boundary sides*, and they play a key role in the method. (See Figure 2 for an example.) The mesh lines of Ω define a partition of ∂R into *boundary segments* that are used to locate the boundary collocation points.

The construction of the finite-element mesh thus consists of the following three steps:

Generate finite-element mesh

2.3.1 Determine the boundary element types.
2.3.2 Determine the finite-element mesh.
2.3.3 Associate boundary segments with elements.
2.3.4 Number the nodes and elements of the finite-element mesh.

For steps 2.3.2 and 2.3.3 the elements of G are classified as *interior*, *boundary*, or *exterior*, depending on whether they are completely inside R, intersect ∂R, or are completely outside R. Some elements may be changed from boundary to exterior or from interior to boundary by the discard procedure. Our implementation of the above procedure depends very much on the following assumptions:

Assumptions for the finite-element mesh generation

a1. A boundary element does not contain an entire boundary piece, and there are at most two boundary pieces associated with it.
a2. If a boundary element has exactly two boundary sides, then they must be adjacent; a boundary element cannot have all its sides be boundary sides.
a3. If a boundary element is discarded, then no more than two of the four (4) neighboring elements can be without any boundary segment associated with them.
a4. The domain is parameterized clockwise.
a5. The boundary does not enter an element more than once, except when it leaves the element and reenters it without crossing a grid line and where the neighboring element it enters is discarded.

These assumptions are usually satisfied for a reasonably fine mesh.

We present a code outline for the finite-element mesh generation.

2.3.1 *Determine the boundary element types*

```
LOOP: FOR EACH BOUNDARY POINT $B_I$ DO:
  IF   THE BOUNDARY LEAVES AN ELEMENT AND ENTERS A NEW ELE-
       MENT ($IX$, $JY$) AT THIS POINT
  THEN SAVE THE BOUNDARY POINT INDICES FOR THAT NEW ELEMENT
       AS ELTYPE ($IX$, $JY$) := $IENTER$+1000 * $IEXIT$ WHERE $IENTER$ AND
       $IEXIT$ ARE THE INDICES OF THE BOUNDARY POINTS WHERE THE
       BOUNDARY ENTERS AND EXITS ELEMENT ($IX$, $JY$)
ENDLOOP;
```

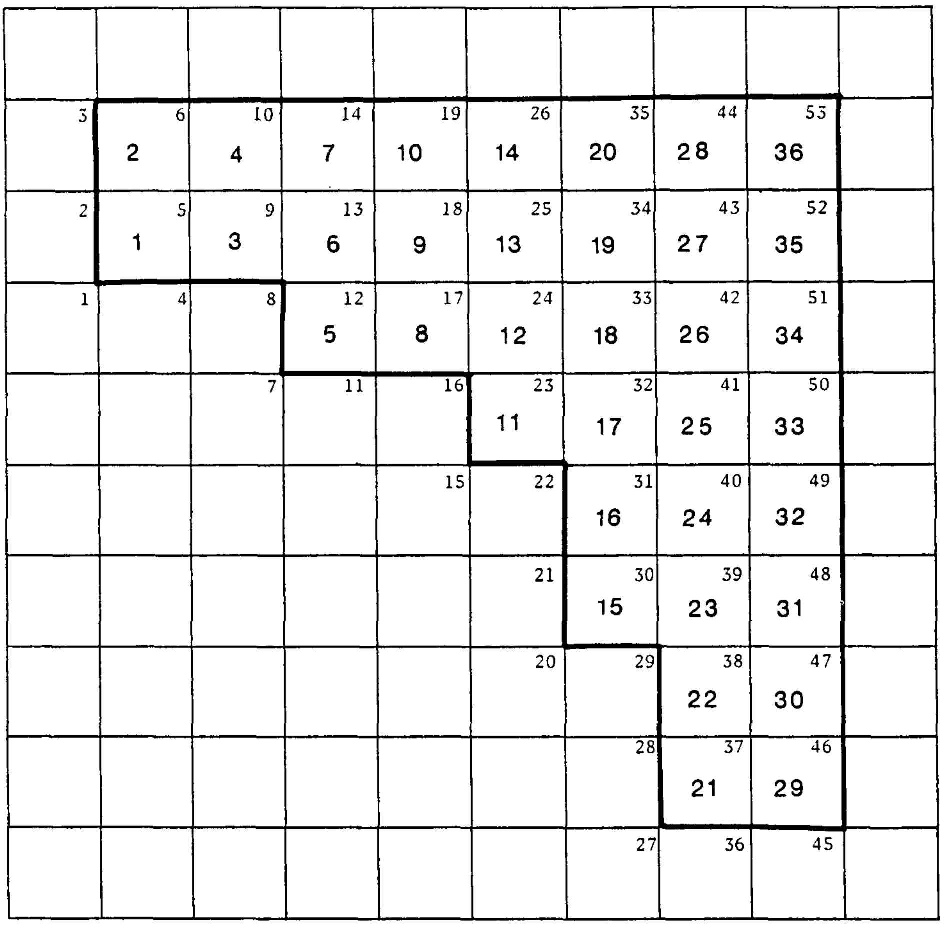


Fig. 2. Numbering of finite-element mesh and nodes for the domain of Fig. 1.

2.3.2 *Determine the finite-element mesh*

```
LOOP: FOR EACH ELEMENT ($IX$, $JY$) OF $G$ DO:
CASE TYPE OF ELEMENT ($IX$, $JY$)
  EXTERIOR:  ELTYPE($IX$, $JY$) := −1   /* DO NOT USE ELEMENT */
  INTERIOR:  ELTYPE($IX$, $JY$) := 0   /* USE ELEMENT */
  BOUNDARY:  IF $\frac{\text{AREA OF ELEMENT INTERSECTION}}{\text{AREA OF ELEMENT}} < \text{DSCARE}$
             THEN ELTYPE ($IX$, $JY$) := − ELTYPE ($IX$, $JY$)   /* DO NOT USE ELEMENT */
             ELSE ELTYPE = ($IENTER$+1000 * $IEXIT$)   /* AND THE ELEMENT IS USED */
ENDCASE
ENDLOOP
```

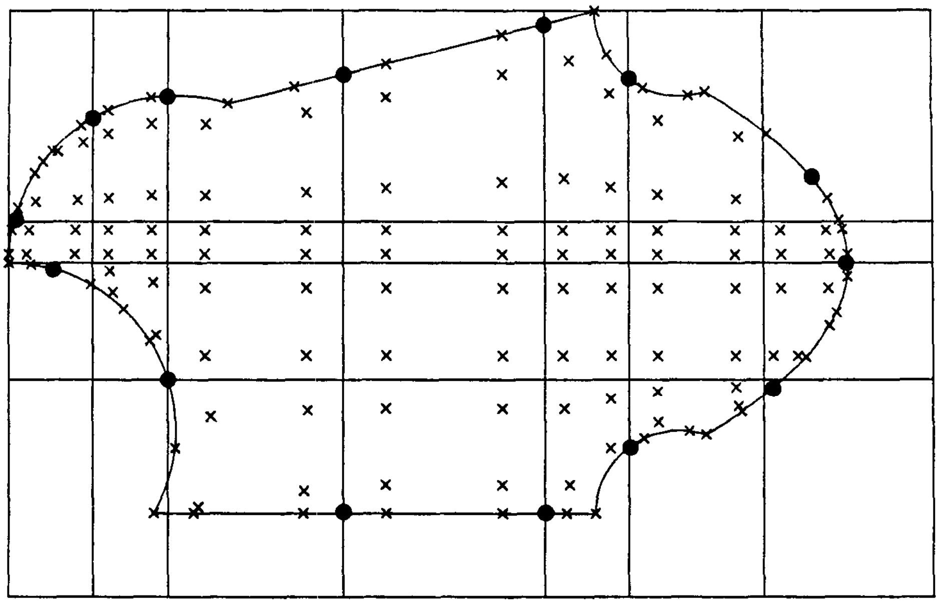


Fig. 3. A domain showing the discarded elements shaded and the ends (dots) of the boundary segments as associated with the retained elements for GIVOPT = TRUE. The x's are the collocation points for the operator and boundary conditions.

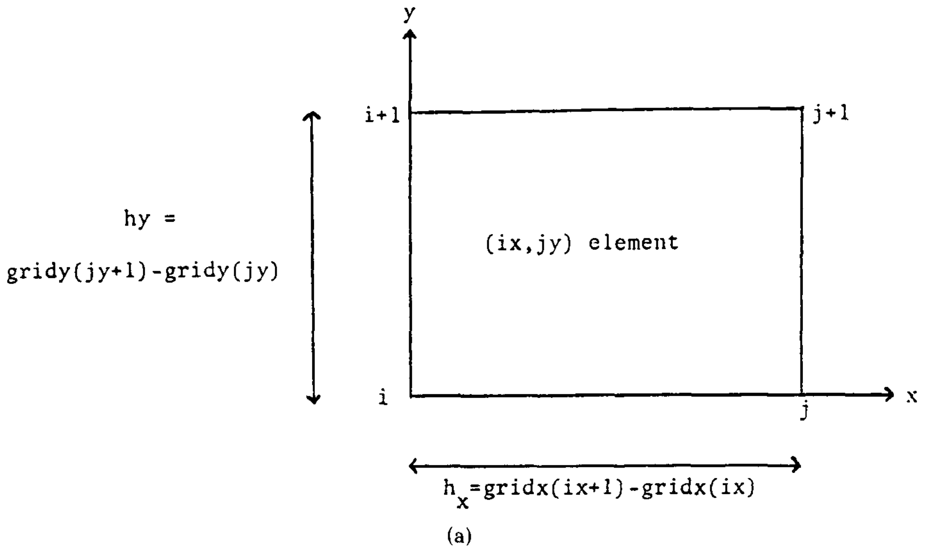
2.3.3 *Associate boundary segments with elements*

```
LOOP: FOR EACH BOUNDARY SEGMENT DO:
  /* IF SEGMENT IS IN ELEMENT (IX, JY) AND ELTYPE(IX, JY) < -1
     THEN THE BOUNDARY SEGMENT IN THE DISCARDED ELEMENT IS
       ASSIGNED TO A NEIGHBORING ELEMENT */
    IF   ANY NEIGHBORING ELEMENTS HAVE NO ASSOCIATED
         BOUNDARY SEGMENT
    THEN THE BOUNDARY SEGMENT IS SPLIT AMONG THEM (UP TO 2
         PIECES)
    ELSE IF GIVOPT = .TRUE.
         THEN THE BOUNDARY SEGMENT IS SPLIT BETWEEN THE
              TWO ELEMENTS WHOSE ASSOCIATED BOUNDARY SEG-
              MENTS ARE CONNECTED TO IT
ENDLOOP
/* NOTE: IF GIVOPT IS .FALSE. THEN THE PIECE OF THE BOUNDARY IN THE
   DISCARDED ELEMENT IS NOT USED */
```

2.3.4 *Number the nodes and elements of the finite-element mesh*

NODES AND ELEMENTS ARE NUMBERED VERTICALLY FROM BOTTOM TO TOP, THEN HORIZONTALLY FROM LEFT TO RIGHT. /* SEE FIGURE 2 */

Figure 3 shows a complex domain with the discarded elements shaded and the association of the boundary segments made. A circle indicates the end of each boundary segment and GIVOPT is .TRUE., that is, boundary segments from



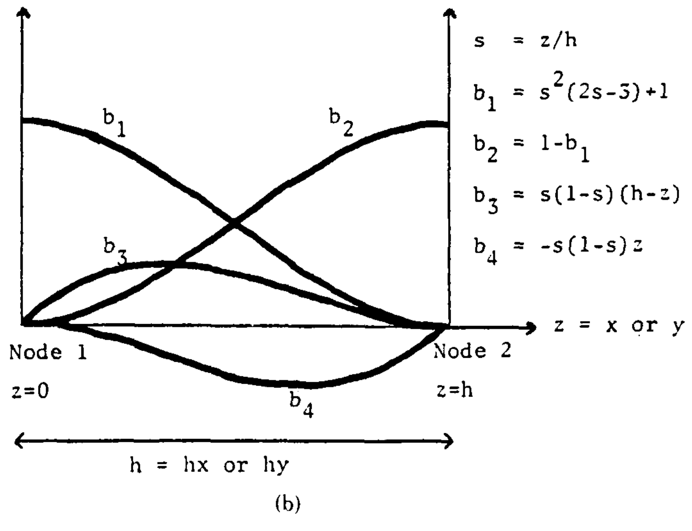


Fig. 4. (a) An interior element of the mesh Ω with the nodes numbered. (b) The four nonzero one-dimensional local basic functions for each variable in the element. These functions are half of the two standard Hermite cubic basis functions for each of the two nodes.

discarded segments are shared with neighboring elements. The default values of DSCARE and GIVOPT are 0.05 and .TRUE. respectively.

2.4 Definition of the Approximate Solution

Consider an interior element of the finite-element mesh Ω, it has the associated nodes numbered i, $i + 1$, j, $j + 1$ as shown in Figure 4. The approximate solution

$U(x, y)$ is a Hermite bicubic piecewise polynomial; it is defined on each element in terms of the one-dimensional local basis functions shown in Figure 4. If we use a local numbering of the 16 *degrees of freedom* (d.o.f.) of $U(x, y)$ on this element, then we have

$$\begin{aligned}U(x,y) = {} & q_1 v_1(x) w_1(y) + q_2 v_1(x) w_3(y) + q_3 v_3(x) w_1(y) + q_4 v_3(x) w_3(y) \\ & + q_5 v_2(x) w_1(y) + q_6 v_2(x) w_3(y) + q_7 v_4(x) w_1(y) + q_8 v_4(x) w_3(y) \\ & + q_9 v_2(x) w_2(y) + q_{10} v_2(x) w_4(y) + q_{11} v_4(x) w_2(y) + q_{12} v_4(x) w_4(y) \\ & + q_{13} v_1(x) w_2(y) + q_{14} v_1(x) w_4(y) + q_{15} v_3(x) w_2(y) + q_{16} v_3(x) w_4(y).\end{aligned}$$

It is worth noticing that there is a natural relation of the d.o.f. (the q coefficients) of U with the values at each node of

$$U, \quad \frac{\partial U}{\partial x}, \quad \frac{\partial U}{\partial y}, \quad \frac{\partial^2 U}{\partial x \partial y}. \tag{2.1}$$

We have, for example, $q_1 = U(\text{node}(i))$, $q_2 = U_x(\text{node}(i))$. The global numbering used for the four degrees of freedom (2.1) at the mth node is

$$\{4m - 3, 4m - 2, 4m - 1, 4m\}.$$

2.5 Formation of the Collocation Equations

The generation of the collocation equations is outlined by the following code skeleton.

Formation of the collocation equations

```
LOOP OVER ELEMENTS OF $\Omega$;
      INVOKE BOUNDARY COLLOCATION POINTS PROCEDURE
      IF ELEMENT = BOUNDARY
        THEN INVOKE BOUNDARY ELEMENT PROCEDURE
               INVOKE BOUNDARY CONDITION PROCEDURE
        ELSE
               INVOKE INTERIOR ELEMENT PROCEDURE
ENDLOOP
```

To generate the operator collocation equations for the interior elements in Ω', one first determines the interior collocation points P_i, $i = 1$ to 4, as the Gauss points of the rectangle and then forces U to satisfy the differential equation at these points. The collocation equations are represented by a data structure with two-dimensional arrays:

$$\begin{aligned}\text{COEF}(n, l) &= l\text{th coefficient value of equation } n, \\ \text{IDCOEF}(n, l) &= \text{index of the unknown associated with COEF}(n, l).\end{aligned}$$

These arrays have 17 columns, 16 for the coefficients and the 17th for the value of the right side g at the collocation point.

procedure INTERIOR ELEMENT

```
  DETERMINE THE $P_I$ AS THE FOUR GAUSS POINTS OF THE ELEMENT;
  LOOP WITH $P_I = (X_I, Y_I)$ FOR $I = 1$ TO 4 DO:
    /* NROW IS EQUATION INDEX */
    NROW := NROW+1
```

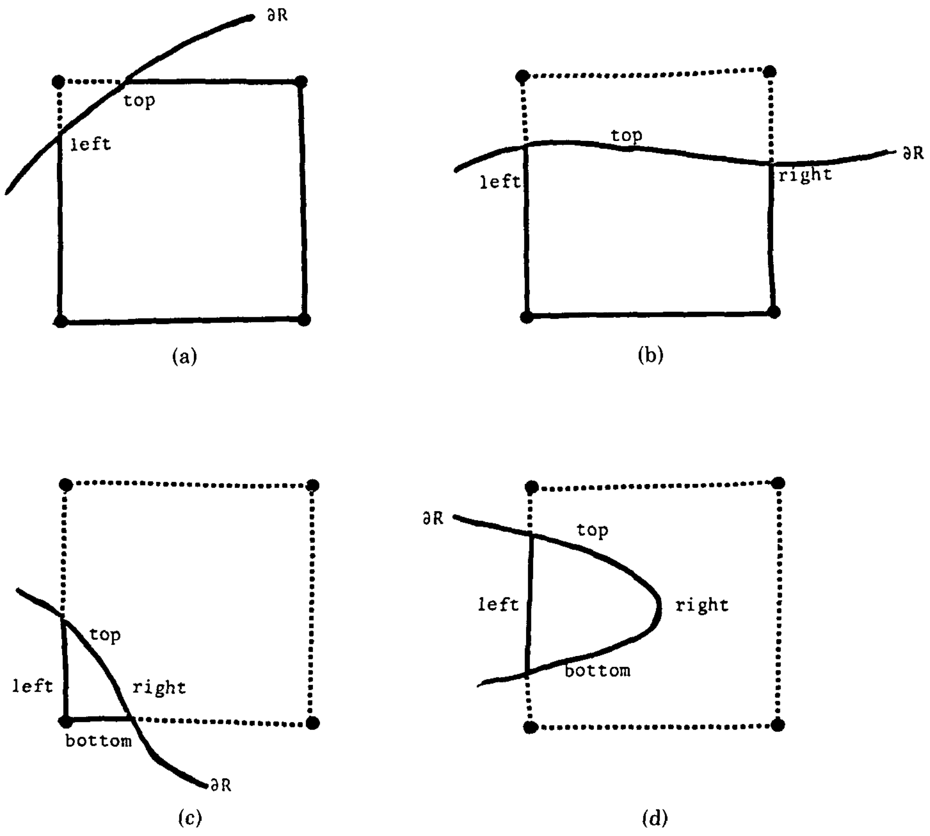


Fig. 5. The four basic mappings used for boundary elements. The dashed lines are the boundary sides external to R. The four element sides are mapped to the actual boundary of $E \cap R$ as indicated by the images of the element nodes. (a) One exterior node. (b) Two exterior nodes. (c) Three exterior nodes. (d) Four exterior nodes.

```
    LOOP FOR N, M = 1 TO 4 DO:
      K := N + 4 * (M−1)
      IDCOEF(NROW, K) := GLOBAL NUMBERING OF DOF K
      COEF(NROW, K) := $L(V_N(X_I) W_M(Y_I))$
    ENDLOOP
    COEF(NROW, 17) := PDE RIGHT SIDE AT $P_I = G(X_I, Y_I)$
  ENDLOOP
```

To generate the operator collocation equations for a boundary element E requires that E be mapped into $E \cap R$. The image of the four Gauss points under this mapping are the collocation points used. (See Figure 5.)

procedure BOUNDARY ELEMENT

```
/* INTERIOR COLLOCATION POINTS FOR 'BOUNDARY' TYPE ELEMENT */
  DEFINE A MAP FROM THE BOUNDARY $\partial E$ OF THE ELEMENT $E$ TO THE
  BOUNDARY OF THE INTERSECTION OF THE ELEMENT AND DOMAIN,
```

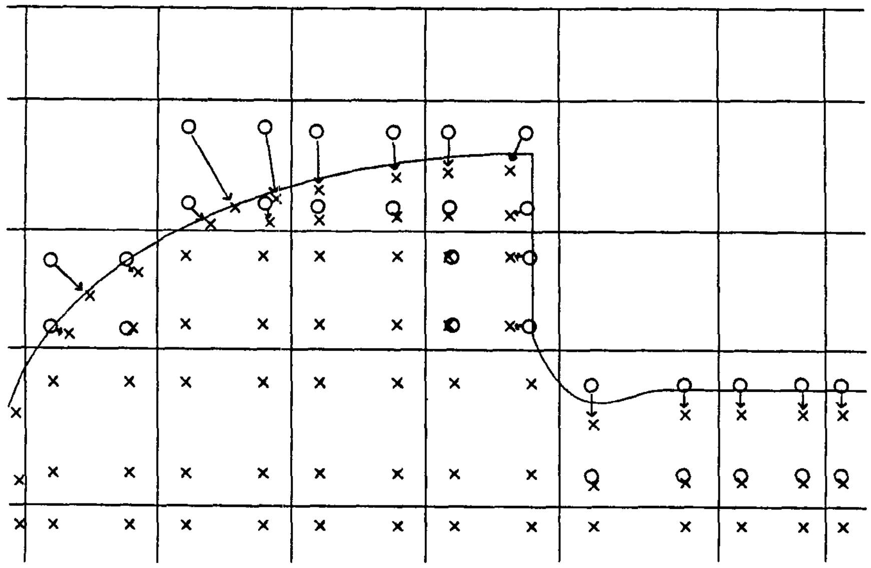


Fig. 6. The mappings of interior collocation points are along the boundary of the domain. The o's are the Gauss points of boundary elements, and the x's are their images under the mapping.

$\partial(E \cap R)$, BY PARTITIONING $\partial(E \cap R)$ INTO FOUR PARTS AND MAPPING EACH SIDE OF THE ELEMENT TO ONE OF THOSE PARTS (SEE FIGURE 5.)

DEFINE A MAP FROM E TO $E \cap R$ BY LINEARLY BLENDING THE FOUR MAPS OF THE BOUNDARY.

DETERMINE THE P_I'S AS THE IMAGES UNDER THIS MAP OF THE FOUR GAUSS POINTS OF THE ELEMENT.

FILL THE ARRAYS COEF AND IDCOEF AS IN THE PROCEDURE INTERIOR ELEMENT.

The map in the procedure boundary element from ∂E to $\partial(E \cap R)$ depends on several aspects of the geometry and is too complicated to give in complete detail here. However, most of the maps are variants of the four cases shown in Figure 5. See [6] for a discussion of linear blending in two dimensions.

It appears that if $E \cap R$ is convex, then the map from E to $E \cap R$ is one-to-one and onto. If $E \cap R$ is not convex, then the map might not be one-to-one and, if there is a strong concavity, might even map points from E to points outside $E \cap R$. However, a proper choice of grid will keep the images of the Gauss points inside $E \cap R$. An example for a portion of an actual domain is shown in Figure 6.

To generate the *boundary condition collocation equations*, one has to determine the location of the boundary collocation points Q_j, $j = 1, \ldots, n(E)$ associated with each boundary element E of the finite element mesh Ω. It can be shown that the method described gives $2s + 4$ boundary collocation points, where s is the number of boundary element sides of Ω. The process of the distribution of boundary points on the actual boundary is implemented in two passes.

The first pass is to place collocation points on the boundary sides of Ω (not on the boundary itself). Four collocation points are associated with each grid node,

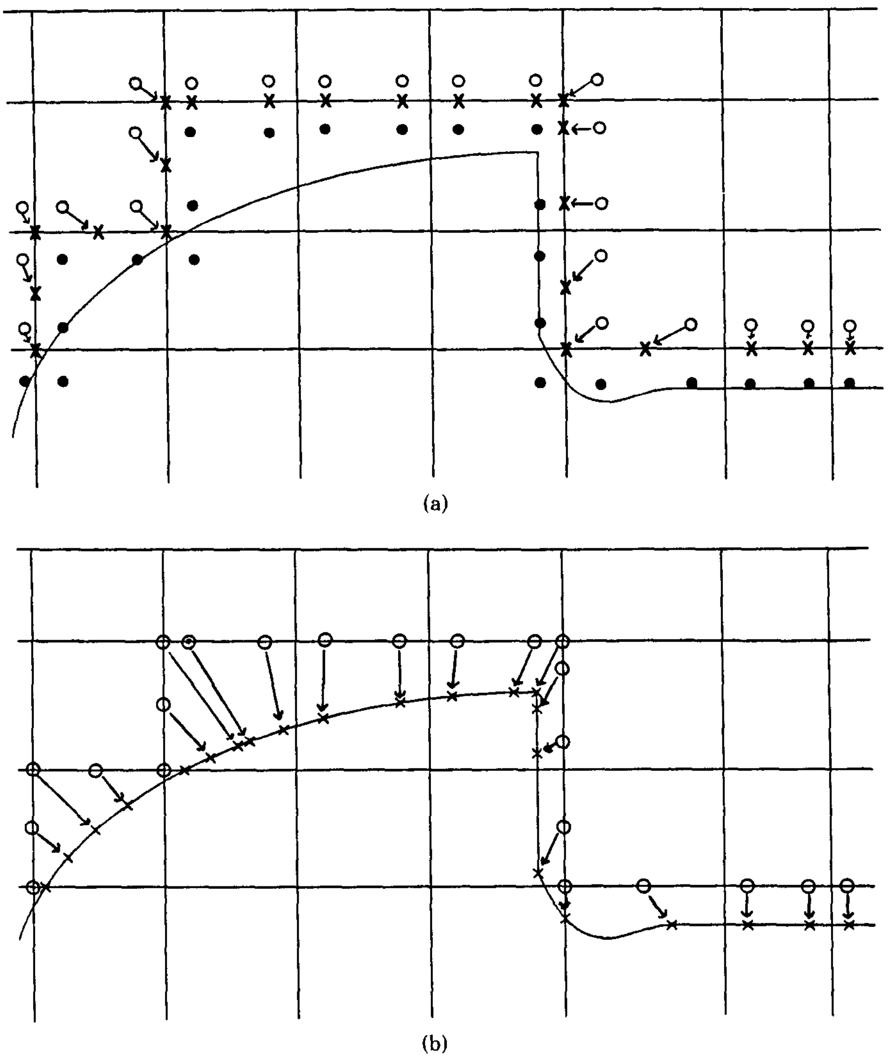


Fig. 7. Mapping of points representing degrees of freedom into boundary collocation points. (a) *First mapping of exterior points. Four points are associated with each node on the boundary of* Ω. Those exterior (o's) to Ω are mapped into the boundary (x's) and those interior (solid dots) are mapped to interior collocation points as in Figure 6. (b) Second mapping of exterior points. The points (o's) on the boundary are then mapped onto ∂R (x's) to the boundary collocation points.

one in each element adjacent to the node. Those points that are interior to Ω (possibly exterior to the domain R) are not considered further. Those points which are exterior to Ω are projected onto the boundary sides of the mesh and become the (intermediate) boundary side collocation points. See Figure 7 for an example.

The second pass is to map the boundary sides of a rectangular element onto the pieces of the boundary associated with the element; the images of the boundary-side collocation points are the boundary collocation points used in the discretization procedure.

There are two parameters, BCP1 and BCP2, to adjust the placement of the boundary collocation points in a boundary side. These allow one to vary placement from the two Gauss points to nodes and midpoints, etc. The default case (BCP1 = BCP2 = 0) selects the Gauss points on an element boundary side. A skeleton code for the placement of the boundary collocation points (BCPs) in the element E follows.

```
procedure BOUNDARY COLLOCATION POINTS
PASS 1: /* ASSOCIATE BOUNDARY COLLOCATION POINTS (BCPS) WITH
           BOUNDARY OF FINITE ELEMENT MESH */
  PLACE TWO BCPS ON EACH BOUNDARY SIDE OF $E$ IN THE SAME CONFIG-
      URATION AS PARAMETERS BCP1 AND BCP2 ARE PLACED IN THE IN-
      TERVAL (0, 1).
  PLACE ONE BCP AT EACH CORNER OF $\partial\Omega \cap E$.
  IF THE END OF THE LAST BOUNDARY SIDE IS A CONCAVE CORNER OF
      THE FINITE ELEMENT MESH
    THEN REPLACE THE TWO BCP OF THE LAST BOUNDARY SIDE WITH
        ONE BCP AT THE MIDPOINT OF THE SIDE
  IF THE BEGINNING OF THE FIRST BOUNDARY SIDE IS CONCAVE CORNER
      OF THE FINITE ELEMENT MESH
    THEN MOVE THE TWO BCPS OF THE FIRST SIDE SO THAT THE FIRST
        BCP IS AT THE BEGINNING OF THE FIRST SIDE AND THE SECOND
        BCP IS AT THE MIDPOINT OF THE FIRST SIDE
 /* THIS PLACEMENT IS REPRESENTED BY VALUES IN (0, 1) WITH 1/2 COR-
    RESPONDING TO THE CORNER IF THERE ARE TWO BOUNDARY SIDES
    AND 1/3 AND 2/3 CORRESPONDING TO THE CORNERS IF THERE ARE
    THREE BOUNDARY SIDES */
PASS 2: /* MAPPING THE BCP FROM $\partial\Omega$ TO $\partial R$ */
 /* THIS IS A MAPPING FROM (0, 1) TO THE SEGMENT OF $\partial R$ ASSOCIATED
    WITH THE ELEMENT $E$ */
  IF THE SEGMENT OF $\partial R$ IS CONTAINED IN ONE PIECE OF THE BOUNDARY
    THEN LINEARLY MAP (0,1) TO (PENTER=BPARAM(IENTER), PEXIT=
    BPARAM(IEXIT))
    DETERMINE THE BCPS FROM THE PASS 1 VALUES AND THE DEFINITION
    OF $\partial R$.
  ELSE IF THE SEGMENT OF $\partial R$ IS CONTAINED IN TWO PIECES OF THE
      BOUNDARY
  THEN LINEARLY MAP (0, 1/2) TO (PENTER, $B_{2,I}$) AND (1/2, 1) TO
      ($B_{1,I+1}$, PEXIT), WHERE $I$ IS THE NUMBER OF THE FIRST PIECE AND
      $B_{2,I}$, $B_{1,I+1}$ ARE FROM (1.2)
    DETERMINE THE BCPS FROM THE PASS 1 VALUES AND THE DEFINITION
        OF $\partial R$
  ELSE ERROR /* DO NOT ALLOW MORE THAN TWO PIECES OF BOUNDARY
      IN ONE ELEMENT */
```

Once the collocation points for the boundary conditions are determined by the above procedure, generating the rest of the boundary-condition collocation equations is simple.

```
procedure BOUNDARY CONDITION
/* BOUNDARY ELEMENT $E$ HAS $K$ BOUNDARY SIDES */
LOOP OVER BOUNDARY SIDES OF $E$ DO:
  LOOP OVER BOUNDARY COLLOCATION POINTS $Q_I = (X_I, Y_I)$ FOR BOUND-
    ARY SIDES DO:
    /* NROW IS EQUATION INDEX */
    NROW:=NROW+1
    LOOP FOR N, M = 1 TO 4 DO:
      K:=N+4*(M-1)
      COEF(NROW, K) = $\Lambda(V_N(X_I)W_M(Y_I))$
      IDCOEF(NROW, K) = GLOBAL NUMBERING OF KTH DOF
    ENDLOOP
    COEF(NROW, 17) = $\delta(X_I, Y_I)$
  ENDLOOP
ENDLOOP
```

2.6 Reordering of the Collocation Equations

The numbering of the equations and unknowns used in the previous section results in a system of linear equations that is banded in nature. If R is rectangular or close to rectangular, then the system has bandwidth about 4 * NGRIDY. As the domain R deviates more and more from being rectangular, the structure of the linear system becomes less and less regular and very little can be said for a completely general region.

The reordering generated in the actual algorithms discussed here is the natural extension of the *finite-element ordering* [21] to general domains. That is, if R were rectangular, this ordering would be obtained. There is a second ordering natural to collocation called the *collorder ordering.* This ordering is defined for rectangular domains in [4]; it can be extended to general domains in a straightforward way. The finite element ordering is attractive because it gives minimum band width in the rectangular case. The *collorder ordering* is attractive because it gives a non-zero diagonal and provides maximum numerical stability in the rectangular case. An example of these two ordering for a triangular domain is given in [15] and reproduced in Figure 8.

2.7 Solution of the Collocation Equations

It is customary for the linear equations arising from finite-element methods (such as this collocation method) to be solved by some form of Gauss elimination. If R is not far from rectangular, then the system can be made banded by a variety of orderings and considerable efficiency achieved compared to Gauss elimination for a general system of equations. The widely used frontal method [21] often provides the efficiency of bandedness even when R is far from rectangular, even though it is not guaranteed to do so.

A recent study by Rice [16] indicates that iterative methods are much more efficient than elimination methods for the Galerkin method equations (on a rectangle), and it is plausible that this is also true for the collocation equations. The usual finite-element ordering prevents iterative methods from being applied to collocation because there are mostly zeros on the diagonal. The collorder ordering remedies this, but the usual iterative methods diverge rapidly when

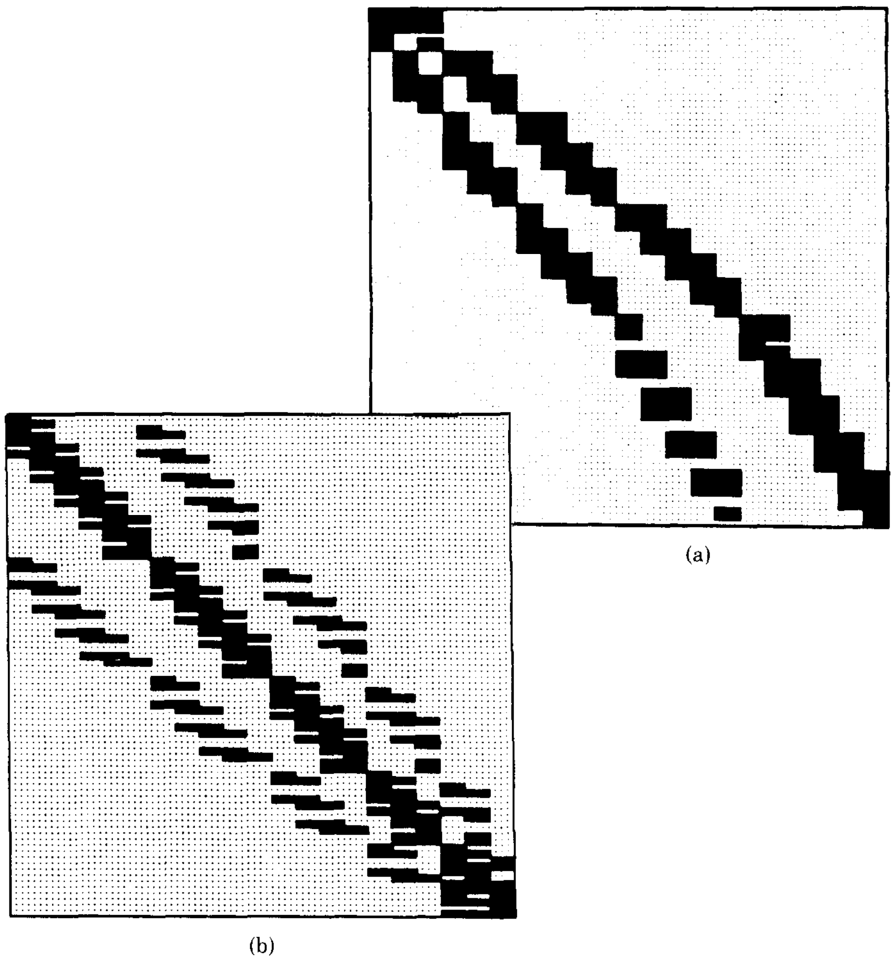


Fig. 8. The patterns of nonzeros in the collocation equations for a triangular domain with (a) the finite-element ordering of the equations and unknowns and (b) the collorder ordering.

directly applied. A convergent iteration method (for the model problem of Laplace's equation on a rectangle) has been presented by Balart et al [1]. It is still open as to how to define fast converging methods for the collocation equations in general, but this question should be viewed as one with good prospects for favorable results.

At this time the only reliable way to solve the collocation equations in general is by Gauss elimination with scaled partial pivoting.

3. THE SPECIAL CASES OF RECTANGULAR DOMAINS

The method described in Section 2 can be considerably simplified in case: (i) the domain R is rectangular and further simplified if (ii) the problem has *uncoupled* *boundary conditions*, that is,

$$u = \delta \qquad \text{in part of the boundary} \quad \partial R_1,$$
$$\frac{\partial u}{\partial n} = \delta \qquad \text{in the rest of the boundary} \quad \partial R_2 \equiv \partial R - \partial R_1.$$

The collocation method for rectangular domains with general mixed boundary conditions is called throughout *Hermite collocation*, while for rectangular regions with uncoupled boundary conditions it is called *interior collocation.* For rectangular domains, some of the collocation steps are implicitly defined by the input data. First, the domain discretization process is implicitly defined by the vectors GRIDX, GRIDY. Second, the finite-element mesh generator process is not needed, since the nodes of Ω coincide with the grid points of rectangular overlay G. The same local definition of the approximate solution is used, but the steps of generating the collocation equations are considerably simplified. We describe these steps for both interior and Hermite collocation.

3.1 Interior Collocation

In the case of uncoupled boundary conditions the boundary collocation equations can be solved explicitly during the discretization of the boundary conditions. Thus, one need only generate and solve the interior collocation equations. This step can be implemented by two parallel asynchronous processes and it is based on the *assumption* that

the boundary conditions only change type at the boundary nodes.

A code skeleton for the two processes follows.

```
  /* OPERATOR DISCRETIZATION */
LOOP OVER ALL ELEMENTS $E$ DO:
  INVOKE INTERIOR ELEMENT PROCEDURE
  /* THIS PROCEDURE IS GIVEN IN SECTION 2.5 */
ENDLOOP
  /* BOUNDARY DISCRETIZATION */
LOOP OVER EACH BOUNDARY PIECE:
  LOOP OVER EACH NODE $T_I$ OF BOUNDARY PIECE;
    DETERMINE THE LEFT OR RIGHT HALF INTERVAL ($[T_{I-1/2}, T_I]$ OR
        $[T_I, T_{I+1/2}]$) WHERE THE BOUNDARY CONDITION $BC$ IS OF THE SAME
        TYPE AS AT $T_I$.
    /* DENOTE THE INTERVAL BY $\Delta$ AND LET $\tau_1$, $\tau_2$ BE ITS TWO GAUSS
    POINTS */
    $S = \{\tau_1, \tau_2$ AND END POINTS OF $\Delta\}$;
    CASE $BC$ TYPE IS:
      DIRICHLET ($U = \delta$): DETERMINE $U_X$ (OR $U_Y$) AT $T_I$ BY INTERPOLATING
                          $\delta$ BY A CUBIC AT THE POINTS S; IDENTIFY ACTIVE
                          DOFS;
      NEUMANN ($\partial U/\partial N = \delta$): DETERMINE $U_{XY}(= U_{YX})$ AT $T_I$ BY INTERPOLAT-
                               ING $\delta$ BY A CUBIC AT THE POINTS S; IDENTIFY
                               ACTIVE DOF;
    ENDCASE;
  ENDLOOP;
ENDLOOP;
```

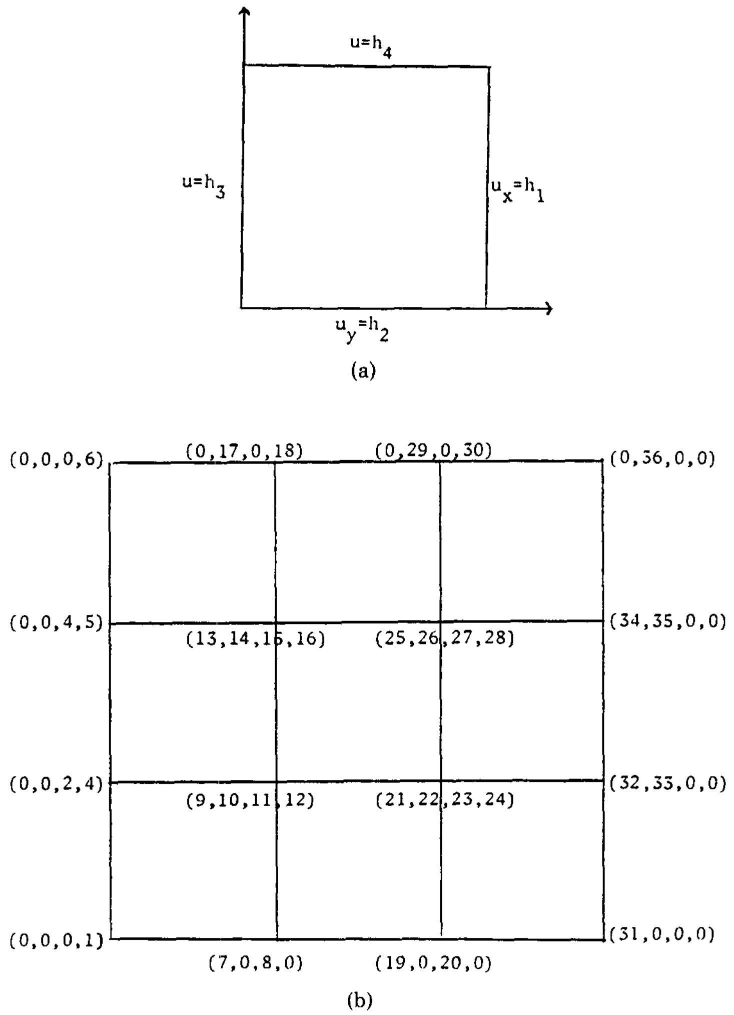


Fig. 9. Example of the interior collocation process of identifying degrees of freedom. (a) Indicates boundary conditions of problem. (b) Numbering of active degree of freedom associated with nodal values of (u, u_y, u_x, u_{xy}) for the boundary conditions of (a). The inactive or predetermined degrees of freedom are denoted by 0.

Figure 9 shows the numbering of active d.o.f. at the end of the boundary discretization process. Finally, the nonactive d.o.f. predetermined in the boundary discretization process are eliminated from equations generated in the operator discretization process.

3.2 Hermite Collocation

In the case of mixed boundary conditions the boundary condition collocation equations are explicitly generated along with the operator collocation equations.

The user can set the location of the two boundary collocation points at each boundary element side by using the parameters $0 \leq BCP1, BCP2 \leq 1$. The default case ($BCP1 = BCP2 = 0$) selects the Gauss points in each boundary element side.

The Hermite collocation method is a direct simplification of the general method described previously. A code skeleton of the method follows.

```
LOOP OVER ELEMENTS $E$ OF $\Omega$ DO:
  INVOKE INTERIOR ELEMENT PROCEDURE
  IF ELEMENT = BOUNDARY
  THEN LOOP OVER BOUNDARY COLLOCATION POINTS $(X_I, Y_I)$ DO:
           NROW = NROW+1
           LOOP FOR N,M=1 TO 4 DO:
             K:=N+4*(M−1)
             COEF(NROW,K) = $\Lambda(V_N(X_I)W_M(Y_I))$
             IDCOEF(NROW,K) = GLOBAL NUMBERING OF KTH DOF
           ENDLOOP
           COEF(NROW, 17) = $\delta(X_I, Y_I)$
         ENDLOOP
ENDLOOP
```

4. THE ALGORITHMS GENCOL, INTCOL, AND HERMCOL

The collocation methods described above have been implemented as three Fortran programs [10, 11]. The initial comments of those programs provide a concise summary of each algorithm; we do not repeat that here. We identify the three algorithms and their input. Each applies to the operator (1.1) and boundary conditions (1.3).

GENCOL: General Collocation

INPUT: GENERAL DOMAIN SPECIFIED IN PARAMETRIC FORM CLOCKWISE
RECTANGULAR OVERLAY: [AX,BX] × [AY,BY]
7 PDE COEFFICIENT FUNCTIONS: CUXX(X,Y), . . . , CU(X,Y),G(X,Y)
4 BOUNDARY CONDITION FUNCTIONS: α(X,Y), β(X,Y),γ(X,Y),δ(X,Y)
2 OUTPUT SPECIFICATION ARRAYS: OUTFNC(I),OUTTYP(I), I=1 TO NOUT

INTCOL: Interior Collocation

INPUT: RECTANGULAR DOMAIN
RECTANGULAR GRID: POINTS X(I), I=1 TO NX+1; Y(J), J=1 TO NY+1
7 PDE COEFFICIENT FUNCTIONS: CUXX(X,Y), . . . , CU(X,Y), G(X,Y)
4 UNCOUPLED BOUNDARY CONDITION FUNCTIONS: α(X,Y), β(X,Y), γ(X,Y), δ(X,Y)
2 OUTPUT SPECIFICATION ARRAYS: OUTFNC(I), OUTTYP(I), I=1 TO NOUT

HERMCOL: Hermite Collocation

INPUT: RECTANGULAR DOMAIN
RECTANGULAR GRID: POINTS X(I), I=1 TO NX+1; Y(J), J=1 TO NY+1
7 PDE COEFFICIENT FUNCTIONS: CUXX(X,Y), . . . , CU(X,Y), G(X,Y)
4 BOUNDARY CONDITION FUNCTIONS: α(X,Y), β(X,Y), γ(X,Y), δ(X,Y)
2 BOUNDARY SPECIFICATION ARRAYS: OUTFNC(I), OUTTYP(I), I=1 TO NOUT

5. EXAMPLES

A wide class of elliptic PDEs have been solved by GENCOL, INTCOL, and HERMCOL. Many results can be found in the references [7, 8, 9, 12]. We include a set of problems to illustrate the use of this software and to give a general indication of its applicability.

5.1 Example 1: Incompressible Flow in a Circular Tube

This example involves an elliptic PDE that models an incompressible Newtonian fluid flow in an internally finned circular tube [13]. The problem is defined by

$$\begin{array}{rlllll}
\text{PDE:} & u_{xx} + \frac{1}{x^2} u_{yy} + \frac{1}{x} u_x = -1; & & & & \\
\text{DOMAIN:} & (0, 1) \times (0, \alpha); & & & & \\
\text{BC:} & u = 0 & \text{at} & x = 1 & \text{and} & 0 \le y \le \alpha, \\
 & u = 0 & \text{at} & y = \alpha & \text{and} & l \le x \le 1, \\
 & u_y = 0 & \text{at} & y = \alpha & \text{and} & 0 < x < l, \\
 & u_x = 0 & \text{at} & x = 0 & \text{and} & 0 \le y \le \alpha, \\
 & u_y = 0 & \text{at} & y = 0 & \text{and} & 0 < x < 1.
\end{array}$$

We choose $\alpha = \pi/4$, $l = .5$ and a uniform spaced 32×17 mesh. Note that for this example all three algorithms can be applied. The one used is INTCOL (interior collocation), as it is the most efficient whenever it is applicable. Figure 10 presents a contour plot of the approximation to the unknown solution of this problem. This problem has been solved with a variety of meshes, and the agreement between the solutions for finer meshes is quite good, which suggests that the solutions are accurate.

5.2 Example 2: Distribution of Diffused Particles

This example involves a non-self-adjoint problem used to model the distribution of diffused particles [20]. The problem is defined by

$$\begin{array}{rl}
\text{PDE:} & u_{xx} + \frac{1}{x^2} u_{yy} + \frac{2}{x} u_x + (1/(x \tan y)) u_y = g, \\
\text{DOMAIN:} & (0, 1) \times (0, 1), \\
\text{BC:} & u = h.
\end{array}$$

In order to test convergence, the functions g and h are chosen so that $u = e^{x+y}$. In this case, all three algorithms can be applied. The problem is solved for various meshes, and for each mesh various performance indicators are computed. These data are summarized in Table I. These data indicate that the rate of convergence of the collocation method is of order 3.8. This is similar to the fourth-order convergence in the approximation with bicubic Hermite polynomials; order 4 is

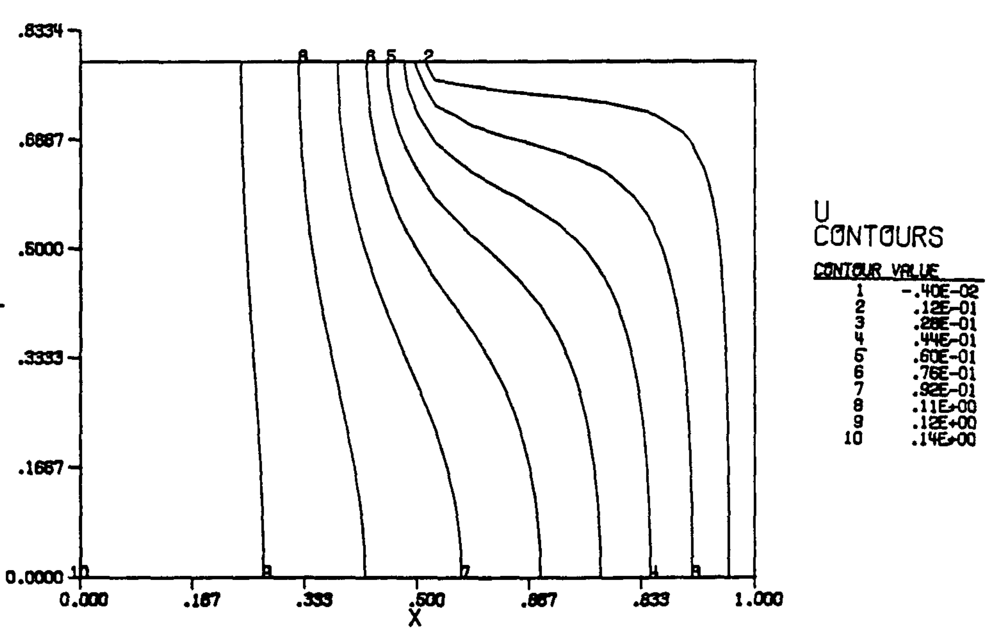


Fig. 10. Contour plot of the solution of Example 1 by INTCOL.

Table I. Performance Data for Example 2[a]

HERMCOL		INTCOL		HERMCOL		GENCOL			
Grid	Equations	Time-D	Time-T	Time-D	Time-T	Time-D	Time-T	Error	Order
3 × 3	36	0.08	0.20	0.17	0.47	0.15	0.53	2.84e − 4	
5 × 5	100	0.13	0.75	0.28	1.83	0.27	1.81	2.39e − 5	3.57
9 × 9	324	0.60	5.72	0.58	11.94	0.72	11.92	1.70e − 6	3.81
13 × 13	676	1.17	22.12	1.10	41.67	1.48	45.38	3.50e − 7	3.90
17 × 17	1156	2.40	57.83	1.62	109.88	2.55	109.47	1.15e − 7	3.87

[a] Time-*D* is the time in seconds on a VAX 11/780 for discretization, Time-T is the total time for problem solution (excluding I/O). Error and order are estimates of the maximum error and the order of convergence as a function of Δx. The same band Gauss elimination solver was used for each discretization, and the same errors were obtained.

the highest possible order of convergence. The order is estimated at the *i*th grid by

$$\text{order} = \frac{\log(\text{error}(i)/\text{error}(i-1))}{\log(\Delta x_i/\Delta x_{i-1})}.$$

Recall that INTCOL has fewer equations and produces an insignificantly different solution.

This example is modified to make the domain nonrectangular. The Dirichlet boundary condition is kept the same so the problem is defined by $u = h(x, y)$ on

lines: (1.0, 0.0) to (0.0, 0.0) to (0.1, 0.5) to (0.5, 0.5),
arc: $x = 0.5 + .5 * \sin t, y = .5 * \cos t$ for $t = 0.$ to $\pi/2$.

Table II. Performance Data for Example 2 with a Nonrectangular Domain[a]

		GENCOL			
Grid	Equations	Time-D	Time-T	Error	Order
3 × 3	36	0.15	0.46	1.04e − 4	
5 × 3	60	0.27	1.16	1.02e − 5	3.78
9 × 5	176	0.67	4.64	7.40e − 7	3.78
17 × 9	560	1.73	21.90	7.96e − 8	3.22
15 × 13	1172	3.38	68.92	1.21e − 8	4.65

[a] The notation is the same as for Table I.

The performance results of GENCOL for this modified problem are given in Table II. These data suggest that the rate of convergence of the collocation method is about 3.7. There is no theoretical basis upon which to base a conjecture about the rate one should expect here, but this example suggests that the convergence may be about the same as collocation on rectangular domains.

5.3 Example 3: Flux Distribution in Magnetic Materials (Nonlinear) Problem

The calculation of flux distribution in magnetic material with saturation leads to the nonlinear elliptic PDE

$$\frac{\partial}{\partial x}\left(\frac{1}{\mu}\frac{\partial \psi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{1}{\mu}\frac{\partial \psi}{\partial y}\right)=0,$$

where ψ is the flux function and μ is the permeability which can be expressed as the ratio of the magnitudes of the flux vector B and field vector H. It is shown by Poritsky [14] that $B = (\psi_x^2 + \psi_y^2)^{1/2}$ and $H = (\phi_x^2 + \phi_y^2)^{1/2}$, where ϕ is the potential function. In [14] a number of methods are applied to determine the distribution of magnetic flux in a transformer core with periodic circular bolt holes (Figure 11a), for an (H, B)-relation shown in Table III and average flux density B_0 = 15,000 lines per centimeter across a 6-inch lamination width. We denote by H_0, μ_0 the values of H, μ corresponding to $B_0(H_0 = 3.1, \mu_0 = 5{,}000)$. Because of symmetry, it is sufficient to solve the PDE in the domain shown in Figure 11b with the indicated boundary conditions. A dimensionless form of the problem is obtained by replacing μ by μ/μ_0, where μ_0 is an average value. Similarly, B and H are replaced by B/B_0 and H/H_0 in the dimensionless form.

The PDE can be written in the form

$$\psi_{xx}+\psi_{yy}-\frac{\mu_x}{\mu}\psi_x-\frac{\mu_y}{\mu}\psi_y=0.$$

In this form, μ_0 and H_0 do not enter into the calculations. We select $B_0 = 3333$ so that $\| B \|_\infty$ (the maximum magnitude of B) is about 18000 and Table III can be used to calculate μ as a function of B.

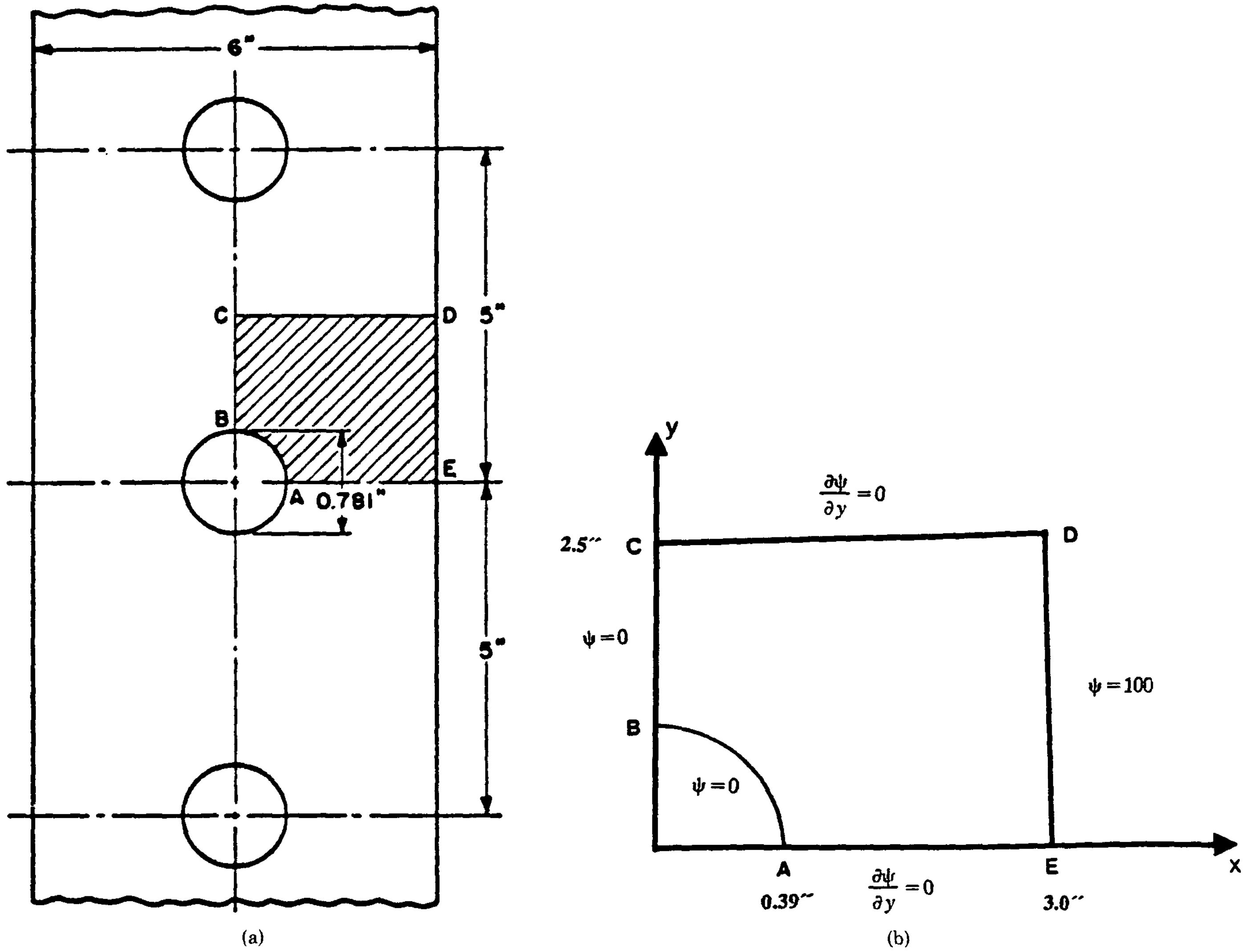

6"
C
D
5"
B
E
A
0.781"
5"
(a)
y
$\frac{\partial\psi}{\partial y}=0$
2.5"
C
D
$\psi=0$
$\psi=100$
B
$\psi=0$
A
0.39"
$\frac{\partial\psi}{\partial y}=0$
E
3.0"
x
(b)

Fig. 11. Domains for Example 3. (a) The physical arrangement. (b) The domain after symmetry is exploited.

Table III. Flux Distributions with Saturation for Example 3

B (lines/cm^2)	H (Gilberts/cm)
500	0.0500
1,000	0.0645
2,000	0.0850
4,000	0.117
6,000	0.149
8,000	0.190
10,000	0.259
12,000	0.399
13,000	0.544
14,000	0.815
15,000	1.67
16,000	6.50
17,000	17.7
18,000	41.0

Table IV. Maximum Differences between Iterates for Example 3

K	DIFF(K)
1	6.500e + 01
2	3.645e − 03
3	3.740e − 04
4	1.764e − 05
5	2.861e − 06
6	1.907e − 06

The nonlinear problem is solved by the following simple iteration.

```
GUESS $\psi^{(0)} = 10*X*Y$
LOOP FOR K = 1 TO L DO
    SOLVE $\psi_{XX}^{(K)} + \psi_{YY}^{(K)} + \frac{\mu_X^{(K-1)}}{\mu^{(K-1)}}\psi_X^{(K)} + \frac{\mu_Y^{(K-1)}}{\mu^{(K-1)}}\psi_Y^{(K)} = 0$    FOR $\psi^{(K)}$
ENDLOOP
```

The term $\mu^{(k-1)}$ is computed by

$$\mu^{(k-1)}(x, y) = \frac{B^{(k-1)}(x, y)}{H^{(k-1)}(x, y)},$$

where

$$B^{(k-1)}(x, y) = ((\psi_x^{(k-1)}(x, y))^2 + (\psi_y^{(k-1)}(x, y))^2)^{1/2}$$

and $H^{(k-1)}(x, y)$ is computed by linear interpolation of Table III. The terms $\mu_x^{(k-1)}$ and $\mu_y^{(k-1)}$ are computed similarly.

The four methods used in [14] were (1) ordinary finite differences, (2) a graphical method, (3) an analog device, and (4) a hodograph method. None were very satisfactory, but that is no surprise since the paper is more than 30 years old. The simple iteration converges very well. Table IV indicates the convergence as measured by

$$\text{DIFF}(K) = \max_{(x,y)\in\text{grid points}} |\psi^{(k)}(x, y) - \psi^{(k-1)}(x, y)|.$$

The contour plot obtained after six iterations with a 13 × 11 grid is shown in Figure 12. It agrees qualitatively with the crude results obtained by Poritsky.

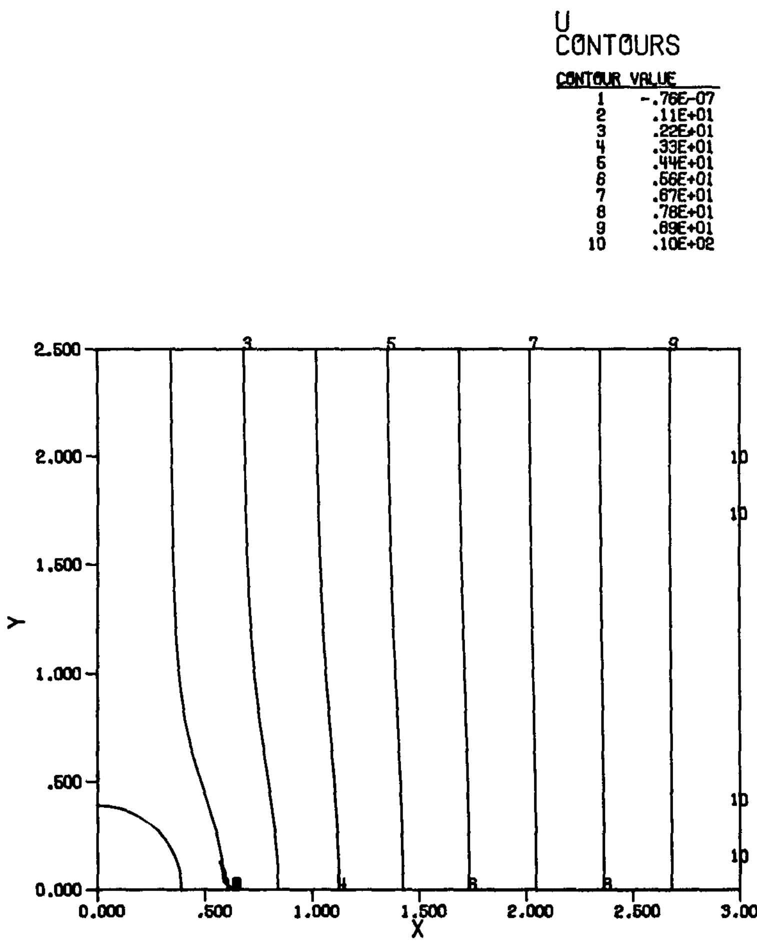


Fig. 12. Contour plot of the solution of the nonlinear problem of Example 3.

5.4 Example 4: Gas Lubrication (Nonlinear Problem)

The Navier–Stokes equations for compressible, nonviscous fluid flow in thin films reduce to Reynold's equation. It models the pressure distribution in the gas films that lubricate high speed devices such as gyroscope bearings or magnetic read heads. The PDE is

$$(uh^3u_x) + (uh^3u_y)_y + c(uh)_x = 0,$$

with boundary conditions $u = 1.0$ everywhere. The function $h(x, y)$ is the height of the thin lubrication film. The parameter c is a physical constant, when c is small (low speed), the problem is easy, and when c is large (high speed), the problem becomes quite difficult.

Table V. Maximum Differences Between Iterates for Example 4

K	DIFF(K)
1	1.418e + 00
2	6.064e − 02
3	3.116e − 03
4	1.102e − 03
5	3.788e − 06
6	1.295e − 07

This problem is solved by Newton's method, see [15]and [19] for a derivation of this iteration. The Newton iteration is

```
GUESS U^0(X,Y)
FOR K=0 TO L DO:
  (1) SOLVE THE LINEARIZED PROBLEM
      $U^{(K)}U_{XX} + U^{(K)}U_{YY} + D(X,Y)U_Y + E(X,Y)U_X + F(X,Y)U = G(X,Y)$
  (2) SET $U^{(K+1)} = U(X,Y)$
ENDLOOP
```

where

$$\begin{aligned} d(x, y) &= 2u_x^{(k)} + \frac{3h_x u^{(k)}}{h} + \frac{c}{h^2}, \\ e(x, y) &= 2u_y^{(k)} + \frac{3h_y u^{(k)}}{h}, \\ f(x, y) &= u_{xx}^{(k)} + u_{yy}^{(k)} + \frac{3(u_x^{(k)} + u_y^{(k)})}{h} + \frac{ch_x}{h^3}, \\ g(x, y) &= u^{(k)}(u_{xx}^{(k)} + u_{yy}^{(k)}) + (u_x^{(k)})^2 + (u_y^{(k)})^2 + \frac{3h(u_x^{(k)} + u_x^{(k)})}{u^{(k)}}. \end{aligned}$$

We choose as an example a simple slider bearing, that is, $h(x, y)$ is linear in x and constant in y. The bearing geometry is a square pad with a half disk on the leading edge. Thus we have

$$h(x, y) = 1 + 2x,$$

and the domain is defined by

lines: (1.0, 0.0) to (0.0, 0.0) to (0.0, 1.0) to (1.0, 1.0),
arc: $x = 1.0 + 0.5 \sin p$, $y = 0.5(1 + \cos p)$ for $p = 0.0$ to π.

We also choose $c = 12.0$, a moderate value for this physical parameter.

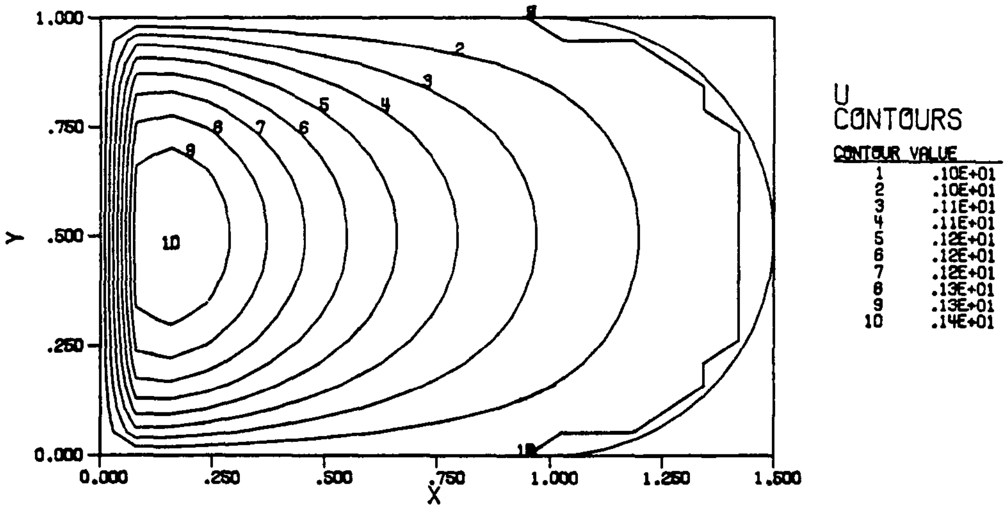


Fig. 13. Contour plot of the solution to Example 4, Reynold's equation, computed by Newton's method and GENCOL.

With the initial guess of $u^{(0)} = 1.0$ (no motion at all) the values of DIFF as defined in Example 3 are given in Table V for a 12 by 9 grid. We see that the Newton iteration converges rapidly. A contour plot of the final $u(x, y)$ is shown in Figure 13. The problem has been solved again with a 16×12 grid; the contour plot is essentially the same and the error using the 12×9 grid is about 9.6×10^{-4}. This is estimated by comparing with a solution on a 10×10 grid.

5.5 Example 5: Minimal Surface (Nonlinear Problem)

The minimal surface equation (or *plateau problem*) is

$$(1 + u_y)^2 u_{xx} - 2u_x u_y u_{xy} + (1 + u_x)^2 u_{yy} = 0.$$

Its solution is the shape a soap film takes on a wire loop defined by the boundary conditions (see [2]).

We select the example of an elliptical domain and boundary condition that have an upward peak at the top of the ellipse and a downward peak at the bottom of the ellipse. Specifically, the domain and boundary conditions are given by

$$u = e^{-4x^2} \quad \text{on} \quad x = 2 \sin t, \quad y = \cos t \qquad \text{for} \quad t = -\frac{\pi}{2} \text{ to } \frac{\pi}{2},$$
$$u = e^{-4x^2} \quad \text{on} \quad x = 2 \sin t, \quad y = \cos t \qquad \text{for} \quad t = \frac{\pi}{2} \text{ to } \frac{3\pi}{2}.$$

The problem is solved by Newton's method; see, for example, [19, ch. 5] for a

Table VI. Maximum differences between iterates for Example 5

K	DIFF(K)
1	1.000e + 00
2	1.020e − 01
3	9.719e − 02
4	2.156e − 02
5	1.408e − 03
6	8.613e − 06
7	8.138e − 06

derivation of the following iteration:

```
GUESS $U^{(0)} = 0$
FOR K = 0 TO L DO
  SOLVE THE LINEARIZED PROBLEM
  $A(X,Y)U_{XX} + B(X,Y)U_{XY} + C(X,Y)U_{YY} + D(X,Y)U_X + E(X,Y)U_Y = G(X,Y)$
  $U^{(K+1)} = U(X,Y)$
ENDLOOP
```

where

$$\begin{aligned}
a(x, y) &= (1 + u_y^{(k)})^2,\\
b(x, y) &= -2u_x^{(k)}u_y^{(k)},\\
c(x, y) &= (1 + u_x^{(k)})^2,\\
d(x, y) &= 2(u_x^{(k)}u_{yy}^{(k)} - u_y^{(k)}u_{xy}^{(k)}),\\
e(x, y) &= 2(u_y^{(k)}u_{xx}^{(k)} - u_x^{(k)}u_{xy}^{(k)}),\\
g(x, y) &= 2((u_x^{(k)}u_{yy}^{(k)} - u_y^{(k)}u_{xy}^{(k)})u_x^{(k)} + (u_y^{(k)}u_{xx}^{(k)} - u_x^{(k)}u_{xy}^{(k)})u_y^{(k)}).
\end{aligned}$$

Again, the iteration converges well. Table VI shows the differences between iterates for a 17×9 grid. Figure 14 shows the contour plots of the solution for the first three iterations. The plot of the final solution is nearly identical to the third plot.

6. PERFORMANCE EVALUATION

There are four principal performance questions for software such as considered here: (1) How much time does it take to run? (2) How much memory does it use? (3) How much accuracy does it achieve? (4) How reliable is it? We do not attempt a scientific analysis of reliability at all. We merely note that the three algorithms have been used on a large number of varied problems. The only difficulty that has been observed is in GENCOL's handling of nonrectangular geometry. Sometimes a grid used does not satisfy the assumptions stated in Section 2.3 and must be modified. Less frequently, but still possible, there are domains with sharp corners where considerable care must be taken in selecting the grid overlay so that reasonably accurate results are obtained. We have also noted for very large problems that linear equation solvers which do not do scaling (such as the LINPACK software) may produce unacceptable magnification of round-off errors [5].

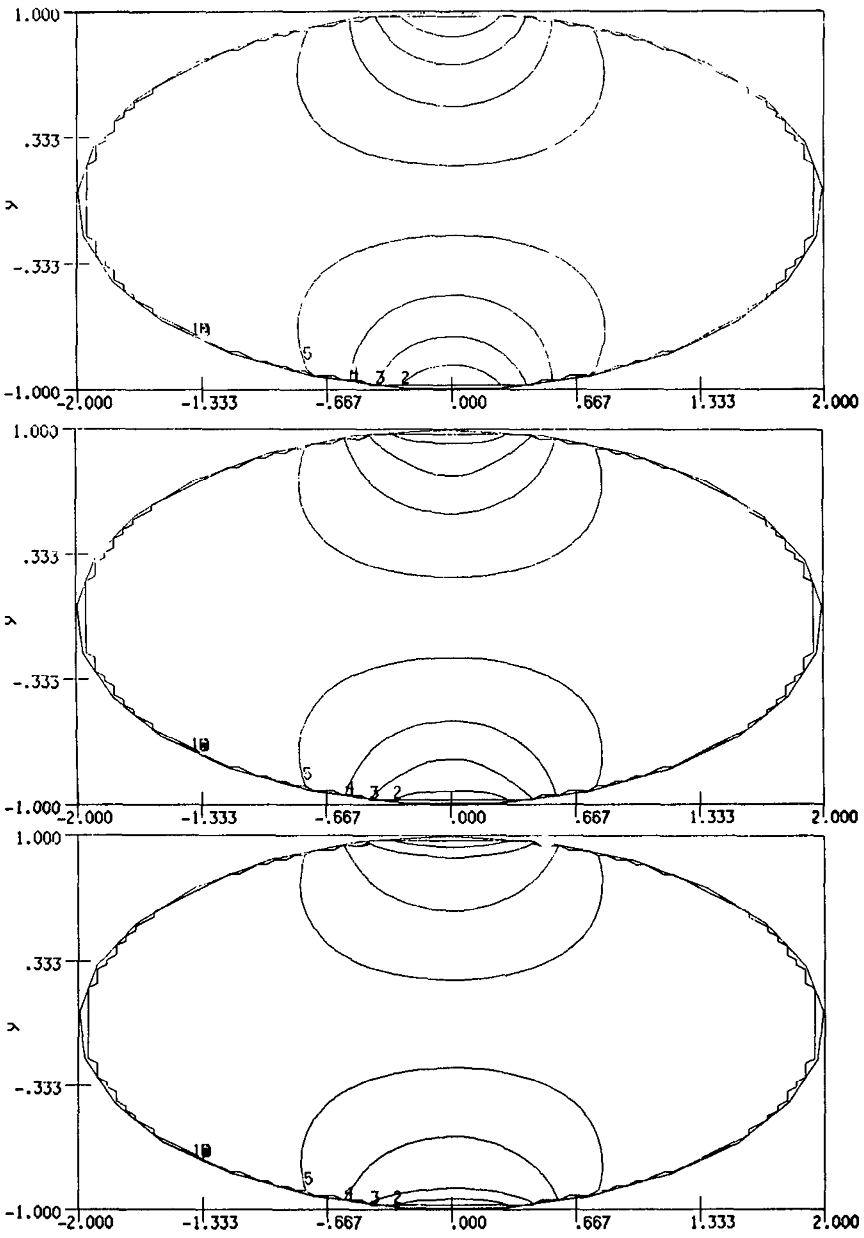


Fig. 14. Contour plots for Example 5. The contours for the first three iterations are shown in top-to-bottom order. The width of the two peaks decreases by a factor of about 3 from the initial Laplace solution estimate (the resolution of the plotter and does not allow this to be seen clearly).

Time and memory use are the standard and most easily handled measures of performance. The algorithms INTCOL and HERMCOL are parameterized by the grid sizes NGRIDX and NGRIDY, and time and memory can be expressed in terms of these two parameters. We distinguish two phases of the solution: the

Table VII. Time and Memory Performance Estimates for INTCOL and HERMCOL

	INTCOL	HERMCOL
Time-D	$O(NX * NY)$	$O(NX * NY)$
Memory-D	$34 * NX * NY$	$34 * (NX + 1) * (NY + 1)$
Time-T	$O(NX * NY^2)$	$O(NX * NY^2)$
Memory-T	$O(NX * NY^2)$	$O(NX * NY^2)$

Table VIII. Time and Memory Performance Data for GENCOL[a]

	Example			
Ratio	2	3	4	5
Time-D/$(NX * NY)$				
$NX = 4$	0.017		0.067	0.021
$NX = 8$	0.011	0.057	0.056	0.016
$NX = 12$	0.010	0.059	0.051	0.012
$NX = 16$	0.010	0.056	0.059	0.010
Memory-D/$(NX * NY)$				
$NX = 4$	212.813		214.375	212.813
$NX = 8$	172.203	173.469	169.344	163.953
$NX = 12$	159.646	159.903	155.319	141.313
$NX = 16$	153.551	153.133	148.492	132.926
Time-T/$(NX * NY^2)$				
$NX = 4$	0.033		0.041	0.032
$NX = 8$	0.026	0.032	0.028	0.024
$NX = 12$	0.028	0.032	0.024	0.016
$NX = 16$	0.028	0.029	0.024	0.015
Memory-T/$(NX * NY^2)$				
$NX = 4$	200.609		196.953	195.797
$NX = 8$	109.990	109.287	108.727	107.912
$NX = 12$	86.435	86.083	85.666	84.438
$NX = 16$	75.782	75.547	75.241	74.208

[a] In each case we have $NX = NY$. Time is measured in seconds on a VAX 11/780 in single precision with code compiled by F77.

discretization (which is fixed for each algorithm) and the solution of the linear system (which may be changed by the user). Table VII presents basic estimates of performance for these two algorithms. We use the following notation in this table:

NX,NY =	NGRIDX-1, NGRIDY-1,
Time-D =	Time to discretize problem (in units of one arithmetic operation),
Time-T =	Time to solve problem using Gauss elimination software for band matrices (in units of one arithmetic option),
Memory-D =	Memory used to discretize problem,
Memory-T =	Memory used to solve problem.

The orders given in Table VII do not distinguish much between INTCOL and HERMCOL; INTCOL is more efficient in both time and memory. The paper [3] gives specific data for a large number of problems.

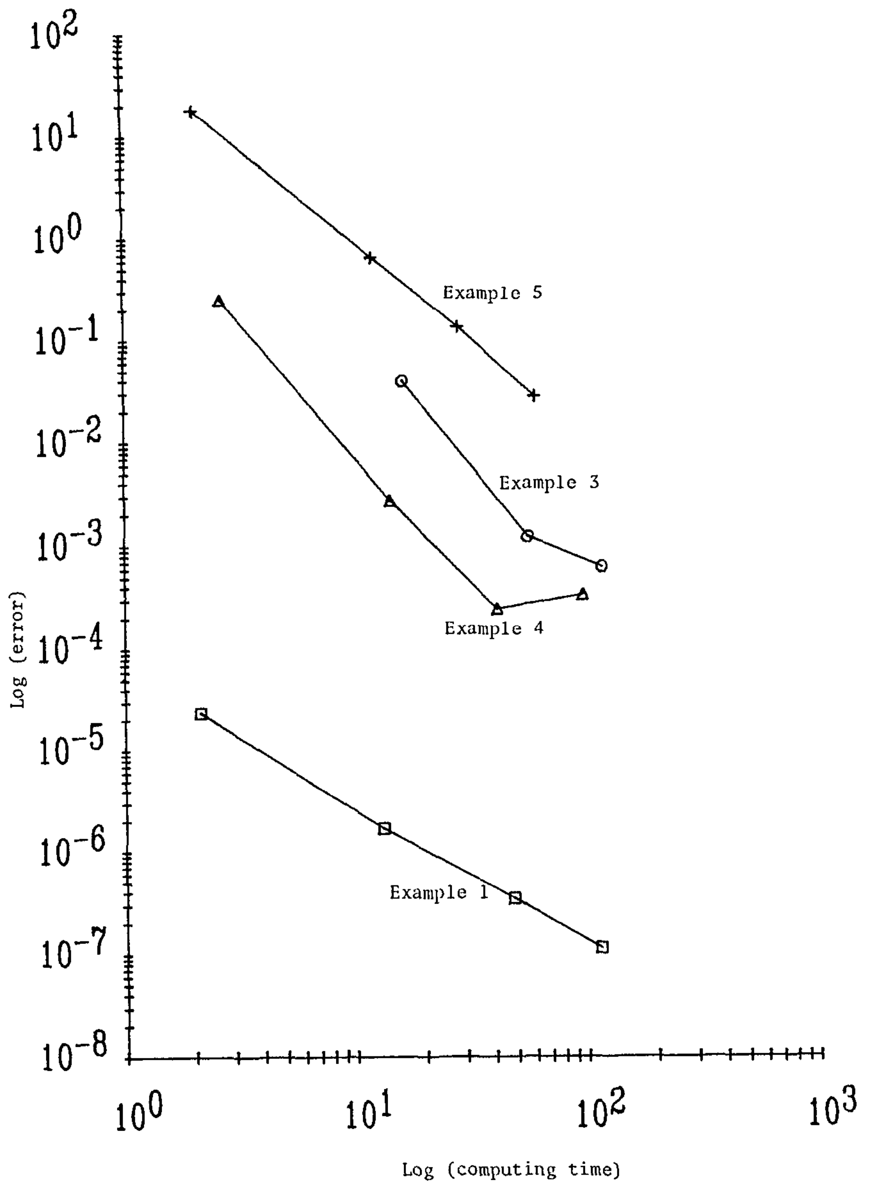


Fig. 15. Performance of GENCOL applied to Examples 2–5. The plot is log(error) versus log(computing time).

Time and memory use for GENCOL are less easily parameterized because the shape of the domain R enters. The orders given in Table VII for HERMCOL are applicable provided (a) the domain is reasonably "compact" and (b) the grid G just covers R. In Table VIII we give specific performance data for four of the examples from Section 5.

Accuracy achieved is perhaps the most important measure of performance, and yet it is sometimes not considered at all. High efficiency is only meaningful when

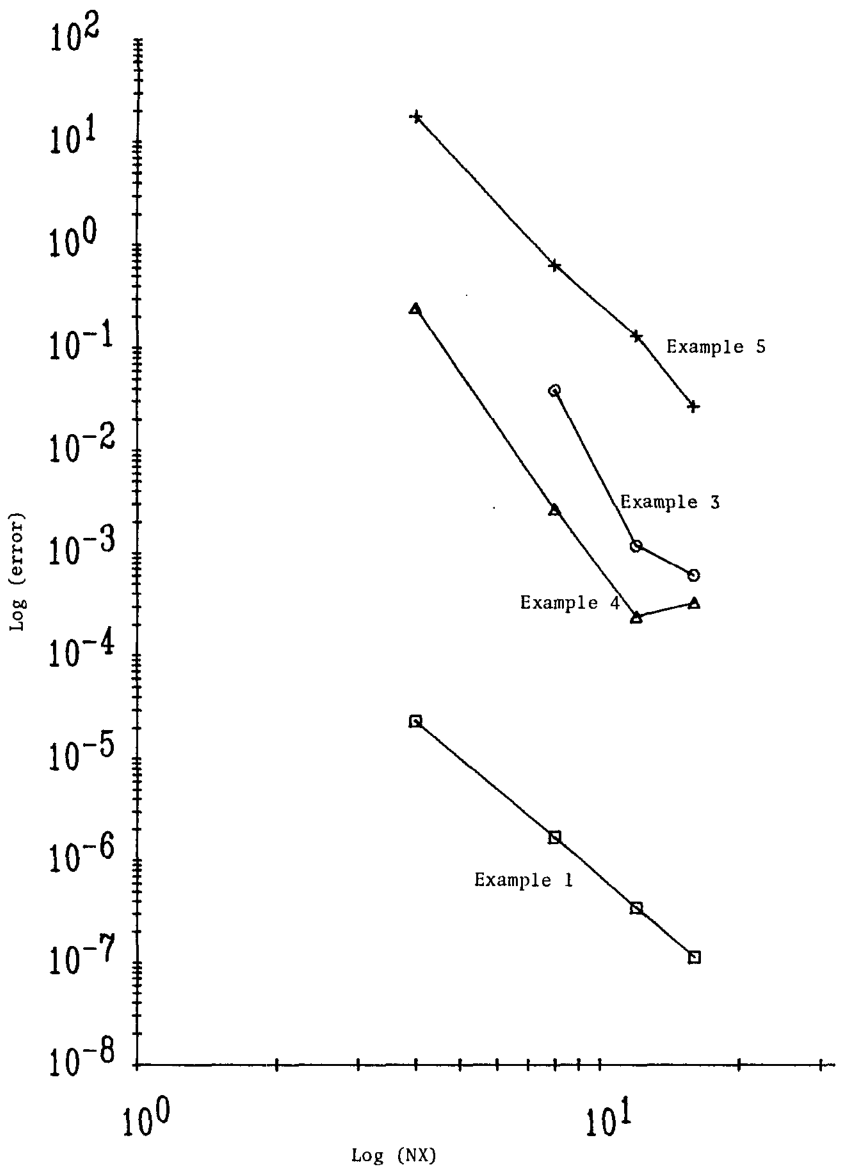


Fig. 16. Performance of GENCOL applied to Examples 2–5. The plot is log(error) versus log(NX); the slopes of the lines are estimates of the order of convergence of the collocation method.

related to accuracy achieved. Accuracy performance is highly problem-dependent, whereas time and memory performance are almost problem-independent. Therein lies the difficulty of measuring accuracy performance in a broadly meaningful way; see [12] for more details on this topic. There is theoretical reason to expect that collocation on rectangular domains is a fourth-order method. That is, the error $|u(x, y) - U(x, y)|$ should be of the order of $1/N^4$, where

N = min(NGRIDX, NGRIDY), provided the problem is well-behaved. This expectation is correct; the data in [3, 7, 8, 9] provides ample evidence of this and there is much more such evidence elsewhere; see, for example, [19].

The accuracy to be expected by GENCOL is unknown at present. It surely should be at least $O(1/N^2)$; it should sometimes be as good as $O(1/N^4)$ and one can hope that it is usually $O(1/N^3)$, or even $O(1/N^4)$. Figures 15 and 16 show accuracy versus NX and versus total computing time for GENCOL applied to Examples 2–5 of Section 5. The error data is plotted on a log-log scale and the slopes of the data estimate the order of convergence. These figures strongly suggest fourth-order convergence (error versus grid size or computer time). The reliability of this suggestion is suspect because this is a very small sample and the errors for three of these examples are only estimates because the true solutions are unknown.

REFERENCES

1. Balart, R., Houstis, E. N., and Papatheodorou, T. S. On the iterative solution of collocation method equations. In *10th IMACS World Congress*, (Montreal, Canada) vol. 1, R. Vishnevetsky, ed. 1982, pp. 98–100.
2. Courant, R. *Dirichlet's Principle, Conformal Mappings, and Minimal Surfaces.* Vol. 3. Interscience, New York, 1950.
3. Dyksen, W. R., Houstis, E. N., Lynch, R. E., and Rice, J. R. The performance of the collocation and Galerkin methods with Hermite bicubics. *SIAM J. Numer. Anal. 21* (1984) 695–715.
4. Dyksen, W. R., and Rice, J. R. A New ordering scheme for the Hermite bicubic collocation equations. In *Elliptic Problem Solvers II*, G. Birkhoff and A. Schoenstat, Eds. Academic Press, New York, 1984, pp. 467–480.
5. Dyksen, W. R., and Rice, J. R. The importance of scaling for the Hermite bicubic collocation equations. *SIAM J. Stat. Sci. Comput.* To be published.
6. Gordon, W. J., and Hall, C. A. Construction of curvilinear coordinate systems and applications to mesh generation. *Int. J. Numer. Meth. Eng. 7* (1973), 461–477.
7. Houstis, E. N., Lynch, R. E., Papatheodorou, T. S., and Rice, J. R. Evaluation of numerical methods for elliptic partial differential equations. *J. Comput. Phy.* 27 (1978), 323–350.
8. Houstis, E. N., Mitchell, W. F., and Papatheodorou, T. S. A C1-collocation method for mildly nonlinear elliptic equations on general domains. In *Advances in Computer Methods for Partial Differential Equations II* (R. Vishnevetsky, ed.) *IMACS*, Rutgers University, 1979, 13–17.
9. Houstis, E. N., Mitchell, W. F., and Papatheodorou, T. S. Performance evaluation of algorithms for mildly nonlinear elliptic problems. *Int. J. Numer. Meth. Eng. 19* (1983), 665–709.
10. Houstis, E. N., Mitchell, W. F., and Rice, J. R. Algorithm GENCOL: Collocation on general domains with bicubic Hermite polynomials. Rep. *ACM Trans. Softw. 12* (1985), 413–415.
11. Houstis, E. N., Mitchell, W. F., and Rice, J. R. Algorithms INTCOL and HERMCOL: Collocation on rectangular domains with bicubic Hermite polynomials. *ACM Trans. Softw. 12* (1985), 416–418.
12. Houstis, E. N., and Rice, J. R. An experimental design for the computational evaluation of partial differential equation solvers. In *Production and Assessment of Numerical Software*, M. Delves and M. Hennell, Eds. Academic Press, London, pp. 57–66, 1980.
13. Masliyah, J. H., and Kumar, D. Application of orthogonal collocation on finite elements to a flow problem. *Math. Comput. Simulation 22*, (1980), 49–54.
14. Poritsky, H. Calculation of flux distributions with saturation. *AIEE Trans. 70* (1951), 309–319.
15. Rice, J. R. *Numerical Methods, Software, and Analysis*, McGraw-Hill, New York, 1983, ch. 10.
16. Rice, J. R. Performance analysis of 13 methods to solve the Galerkin method equations. *Linear Alg. Appl. 53*, (1983), 533–546.

17. RICE, J. R. Numerical computation with general two dimensional domains. *ACM Trans. Math. Softw. 10*, (1984), 443–452.
18. RICE, J. R. Algorithm 625: A two dimensional domain processor. *ACM Trans. Math. Softw. 10*, (1984), 453–462.
19. RICE, J. R., AND BOISVERT, R. F. *Solving Elliptic Problems with ELLPACK.* Springer-Verlag, New York, 1985.
20. RICE, J. R., HOUSTIS, E. N., AND DYKSEN, W. R. A population of linear, second order elliptic partial differential equations on rectangular domains. *Math. Comput. 36* (1981), 475–484.
21. ZIENKIEWICZ, O. *The Finite Element Method in Engineering Science.* McGraw-Hill, London, 1971.

Received May 1984; revised May 1985; accepted September 1985