MATHEMATICS OF COMPUTATION, VOLUME 26, NUMBER 118, APRIL 1972

An Improved Method for Numerical Conformal Mapping

By John K. Hayes, David K. Kahaner and Richard G. Kellner*

Abstract. A new technique for the numerical conformal mapping of a planar region onto the unit disk has been presented and tested by Symm. By elaborating on his methods, we have improved the accuracy of the numerical results by up to four orders of magnitude. For illustration, our methods have been applied to several of the same regions considered in the literature by Symm and Rabinowitz. A flexible FORTRAN code and User's Guide are reproduced on the microfiche card in this issue.

1. Introduction. A new technique for the numerical conformal mapping of a planar region onto the unit disk has been presented and tested by Symm [7], [8], [9]. By elaborating on his methods, we have improved the accuracy of the numerical results by up to four orders of magnitude. For illustration, our methods have been applied to several of the same regions considered in the literature by Symm [7] and Rabinowitz [6].

In this paper, we numerically approximate the univalent function $f(z)$ which maps the bounded, simply-connected region D of the complex plane onto the unit disk. Let L be the boundary of D and choose $z_0 \in D$ to be the point which is to be mapped into the center of the unit disk. It is known [7] that

$$w = f(z) = \exp[\log(z - z_0) + g(z) + ih(z)],$$

where g and h are real-valued harmonic conjugates, and g satisfies

$$\nabla^2 g(z) = 0 \quad \text{for } z \in D,$$

and

$$g(z) = -\log |z - z_0| \quad \text{for } z \in L.$$

The mapping function $f(z)$ above is determined only to within an arbitrary rotation. This depends upon the branch of the logarithm used in the computation and the additive constant chosen for the function h.

Symm [7] numerically solves the integral equation of the first kind

$$(1) \qquad \int_L \sigma(\zeta) \log |z - \zeta| \, |d\zeta| = -\log |z - z_0|, \qquad z \in L.$$

This may always be done, subject to a possible rescaling of the region D [3], [5]. Then, for any $z \in D + L$, g and h have the representation

Received June 21, 1971.

AMS 1970 *subject classifications.* Primary 30A28; Secondary 65E05.

Key words and phrases. Numerical conformal mapping, numerical solution of integral equations of the first kind.

* This research was supported by the U. S. Atomic Energy Commission under Contract W-7405-ENG-36.

(2) $$g(z) = \int_L \sigma(\zeta) \log |z - \zeta| \, |d\zeta|,$$

(3) $$h(z) = \int_L \sigma(\zeta) \arg(z - \zeta) \, |d\zeta|.$$

The function arg must be chosen in an appropriate manner [4].

2. Description of the Method. Our procedure for numerically mapping a region can be divided into two operational steps:

(i) Solve Eq. (1) for the function σ.

(ii) Evaluate Eqs. (2) and (3) for each point $z \in D$ where we want to find $f(z)$.

Let the curve L have the parametric representation $\{(v(t), w(t)) \mid t \in (0, d]\}$ with respect to arc length t. Here, d is the length of L. Define $\zeta(t) = v(t) + iw(t)$. With this notation, Eqs. (2) and (3) become

(4) $$g(z) = \int_0^d \sigma(t) \log |z - \zeta(t)| \, dt, \qquad z \in D + L,$$

(5) $$h(z) = \int_0^d \sigma(t) \arg(z - \zeta(t)) \, dt, \qquad z \in D + L,$$

where we have used $\sigma(t)$ for $\sigma(\zeta(t))$.

Now, we will sketch how we compute the function $\sigma(t)$ numerically. A detailed development is contained in [1]. Since $\sigma(t)$ is a function of arc length, we extend it continuously as a periodic function on $(-\infty, +\infty)$. For ease of explanation, assume $\sigma(t) \in C^3(-\infty, +\infty)$ and that L has no corners. Place on L a uniform mesh of n points (n even), each $h = d/n$ units apart. (In actual practice, one might wish to divide L into several sections. The mesh points on each section would then be uniform with respect to arc length on that section. See the user's guide in the microfiche portion of this issue and also Example 2 of this paper.) Define a set of piecewise polynomial functions $p_1(t), p_2(t), \cdots, p_n(t)$ by

$$\begin{aligned} p_1(t) &= (t - h)(t - 2h)/2h^2, && 0 \leqq t \leqq 2h, \\ &= (t + h)(t + 2h)/2h^2, && -2h \leqq t \leqq 0, \\ &= 0, && \text{otherwise}, \\ p_2(t) &= -t(t - 2h)/h^2, && 0 \leqq t \leqq 2h, \\ &= 0, && \text{otherwise}, \\ p_{2i+1}(t) &= p_1(t - 2ih), && i = 1, 2, \cdots, n/2 - 1, \end{aligned}$$

and

$$p_{2i}(t) = p_2(t - 2(i - 1)h), \qquad i = 2, 3, \cdots, n/2.$$

Define also $\tilde{\sigma}(t) = \sum_{i=1}^n \sigma(ih)p_i(t)$. It is true that

(i) $\tilde{\sigma}(t)$ is a polynomial of degree two on $[ih, (i + 2)h]$, for $i = 0, 2, 4, \cdots, n - 2$.

(ii) $\tilde{\sigma}(t) = \sigma(t)$ at $t = ih$, for $i = 0, 1, 2, \cdots, n$.

(6) (iii) $$\sigma(t) = \tilde{\sigma}(t) + O(h^3) = \sum_{i=1}^n \sigma_i p_i(t) + O(h^3),$$

where $\sigma_i = \sigma(ih)$ for $i = 1, 2, \cdots, n$.

Using the approximation Eq. (6) in Eq. (4), we get

$$\sum_{k=1}^{n} \sigma_k \int_0^d p_k(t) \log |z - \zeta(t)|\, dt = g(z) + O(h^3). \tag{7}$$

The function $g(z) = -\ln |z - z_0|$ for $z \in L$. Thus, we can evaluate Eq. (7) at the points $z = \zeta(ih)$ for $i = 1, 2, \cdots, n$, and we will get n linear equations with constant coefficients for the variables $\sigma_1, \sigma_2, \cdots, \sigma_n$. Set $A = (a_{ik})$ and $B = (b_i)$, where

$$a_{ik} = \int_0^d p_k(t) \log |\zeta(ih) - \zeta(t)|\, dt, \quad \text{for } i, k = 1, 2, \cdots, n,$$

$$b_i = -\log |\zeta(ih) - z_0|, \qquad \text{for } i = 1, 2, \cdots, n.$$

With this notation, Eq. (7) leads to the linear system $A\boldsymbol{\sigma} = B + \mathrm{O}(h^3)$, where $\mathrm{O}(h^3)$ is a vector, with each component bounded by $O(h^3)$, and $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_n)^T$.

The matrix equation we actually solve is

$$\tilde{A}\boldsymbol{\tau} = B, \tag{8}$$

where the elements of $\tilde{A}$ are approximations to those of A. Using our representation for the $p_k(t)$, it is evident that to compute A it is sufficient to evaluate integrals of the form

$$\int_{(i-1)h}^{ih} t^j \log |z - \zeta(t)|\, dt \tag{9}$$

for $i = 1, 2, \cdots, n$ and $j = 0, 1, 2$. The $\tilde{a}_{ij}$ are the result of approximating the integrals (9). For each fixed x, y and i, we approximate $|z - \zeta(t)|$ by a polynomial $q(t)$ of degree two on $((i - 1)h, ih)$. We choose $q(t)$ so that

$$q(t) = |z - \zeta(t)|^2 \quad \text{for } t = (i - 1)h,\ (i - \tfrac{1}{2})h,\ ih.$$

Then

$$\int_{(i-1)h}^{ih} t^j \log |z - \zeta(t)|\, dt \approx \frac{1}{2} \int_{(i-1)h}^{ih} t^j \log[q(t)]\, dt.$$

The integrals on the right-hand side above can be evaluated explicitly. For certain special cases, for instance when $|z - \zeta(t)| = 0$ on $[(i - 1)h, ih]$, the treatment is slightly different in that a higher order polynomial is used.

We then solve the matrix equation $\tilde{A}\boldsymbol{\tau} = B$ for the vector $\boldsymbol{\tau} = (\tau_1, \tau_2, \cdots, \tau_n)^T$ and use this as an approximation to $\boldsymbol{\sigma}$. Then

$$||\boldsymbol{\sigma} - \boldsymbol{\tau}|| \leqq ||(A^{-1} - \tilde{A}^{-1})B|| + ||A^{-1}||\, O(h^3).$$

We have found by experience on numerous problems that the error due to $A^{-1} - \tilde{A}^{-1}$ seldom if ever dominates the $||A^{-1}||O(h^3)$ term. Another analysis [2] strongly indicates that $||A^{-1}|| \leqq O(1/h)$.

Once $\sigma(t)$ has been computed, we may calculate $g(z)$ and $h(z)$.

$$g(z) = \int_0^d \sigma(t) \log |z - \zeta(t)|\, dt$$

$$\approx \sum_{k=1}^{n} \tau_k \int_0^d p_k(t) \log |z - \zeta(t)|\, dt.$$

These integrals are approximated as described above. The calculation of $h(z)$ is more difficult. Using integration by parts, and approximations similar to those above, we are led to integrals of the form

$$\int_{(i-1)h}^{ih} t^j \arg(z - \zeta(t))\, dt,$$

where $i = 1, 2, \cdots, n$ and $j = 0, 1, 2, 3$. The evaluation of these integrals is discussed in detail in [1].

The method set forth by Symm in [7] uses piecewise constant functions in Eq. (6) and evaluates the integrals in Eq. (7) by using Simpson's rule of integration.

3. Tests. A FORTRAN IV version of the algorithm described has been coded to run on our CDC 6600 and CDC 7600. This program is more or less machine independent, has flexible input, and is general enough to handle a large class of problems. It is a modification of a program described in [1] which has been in use for a few years. A limited number of copies of this deck and a user's guide are available from the authors. Using this code, we have computed some approximate conformal maps for several regions, including some used in [7]. Since our technique is an extension of the method used there, it is appropriate to compare our results with those. All of the regions selected for test have substantial symmetry. We have elected to ignore this symmetry in our code in order to give utmost flexibility. Taking advantage of symmetry ought to enhance the accuracy by reducing the volume of computation.

Symm has pointed out [8] that the maximum errors occur on the boundary of the region being mapped. Since points on the boundary have image points on the unit circle, it is easy to check the error in the modulus of an arbitrary boundary point. The data points themselves are constrained by the defining equations to be mapped onto $|w| = 1$, hence we check for modulus error at points midway between each of the data points. The columns labeled ERR-MOD contain the maxima of the quantities $||w| - 1|$ at these intermediate points. Computing the error in the argument is more difficult. Symm provides an estimate of this in [8], denoted E_A. Our experience has indicated that as the region becomes less circular and more elongated, errors, particularly those in the argument, increase in a monotonic way. Since the numbers E_A provided by Symm did not have this property, we considered them somewhat unreliable and decided to use an alternative technique. The columns labeled ERR-ARG represent the maximum difference in the argument at the data points between two computations, the second corresponding to the largest number of data points used for the domain in question. It is reasonable to examine the argument at the data points rather than the intermediate points, since the argument is not constrained in any way by Eq. (1). This procedure does yield the monotonicity we expect. In certain cases, analytic expressions for the conformal maps are available. It is then possible to compute the absolute errors in the argument exactly. These numbers compare extremely well with the approximate errors ERR-ARG described above, and constitute our main justification for this approach.

Each of our test regions has its center point mapped into the origin. In what follows, h and n will have the same meaning as in Section 2.

Example 1. *Oval of Cassini.* This curve is defined by

$$[(x+1)^2 + y^2][(x-1)^2 + y^2] = \alpha^4, \qquad \alpha > 1.$$

For α near 1, the curve is elongated and nonconvex, becoming more circular as α increases. For $\alpha = 1.06$, the width to height ratio is about 5. Points are distributed uniformly on the entire boundary. The exact mapping is given by

$$f(z) = \alpha z/(\alpha^4 - 1 + z^2)^{1/2},$$

and we use this to compute errors in the argument.

TABLE I

Oval of Cassini

α	n	h	ERR-MOD	ERR-ARG
1.06	65	.11	2×10^{-3}	2×10^{-3}
1.06	129	.06	1×10^{-4}	2×10^{-4}
1.8	33	.34	1×10^{-4}	1×10^{-4}
1.8	65	.18	1×10^{-5}	1×10^{-5}
1.8	129	.09	6×10^{-7}	7×10^{-7}

The maximum error occurs at or near $x = 0$. The errors near $y = 0$ are smaller by a factor of 1/100. The comparison with Symm must be made carefully, since his data points are for the most part distributed uniformly with respect to x rather than t. As far as we can determine, errors in the modulus are from one to four orders of magnitude better than those in [7]. We should emphasize that our distribution of points is poor.

For $\alpha = 1.06$, $n = 65$, Table II indicates errors for points inside the curve.

TABLE II

Oval of Cassini

$\alpha = 1.06$	$n = 65$	ERRORS IN	
x	y	MODULUS	ARGUMENT
1.4	0.	6×10^{-7}	3×10^{-5}
1.26	0.	6×10^{-7}	3×10^{-5}
1.12	0.	1×10^{-6}	3×10^{-5}
0.98	0.	2×10^{-6}	3×10^{-5}
0.84	0.	2×10^{-6}	3×10^{-5}
0.7	0.	4×10^{-6}	3×10^{-5}
0.56	0.	7×10^{-6}	3×10^{-5}
0.42	0.	2×10^{-5}	4×10^{-5}
0.28	0.	5×10^{-5}	4×10^{-5}
0.14	0.	1×10^{-4}	4×10^{-5}

Example 2. *Rectangle.* $-1 \leqq x \leqq +1, -\alpha \leqq y \leqq \alpha$.

The case $\alpha = 1$ was computed exactly by the use of elliptic integrals. In the other cases, we use a comparison with the most accurate computed values. Points are uniformly spaced on each side, with $n/4$ points per side. See Table III.

TABLE III

Rectangle

α	n	ERR-MOD	ERR-ARG
0.1	516	4×10^{-5}	—
	260	6×10^{-4}	7×10^{-4}
	132	5×10^{-3}	6×10^{-3}
	68	1×10^{-2}	1×10^{-2}
	36	6×10^{-2}	5×10^{-2}
0.2	516	3×10^{-6}	—
	260	4×10^{-5}	4×10^{-5}
	132	5×10^{-4}	6×10^{-4}
	68	5×10^{-3}	7×10^{-3}
	36	1×10^{-2}	2×10^{-2}
0.4	260	3×10^{-6}	—
	132	4×10^{-5}	4×10^{-5}
	68	6×10^{-4}	7×10^{-4}
	36	5×10^{-3}	1×10^{-2}
0.5	260	1×10^{-6}	—
	132	2×10^{-5}	2×10^{-5}
	68	2×10^{-4}	2×10^{-4}
	36	2×10^{-3}	3×10^{-3}
0.8	260	2×10^{-7}	—
	132	3×10^{-6}	5×10^{-6}
	68	4×10^{-5}	8×10^{-5}
	36	6×10^{-4}	3×10^{-3}
1.0	260	9×10^{-8}	—
	132	1×10^{-6}	1×10^{-6}
	68	2×10^{-5}	3×10^{-5}
	36	2×10^{-4}	1×10^{-3}

Both ERR-MOD and ERR-ARG are monotonic with respect to n and α for $\alpha \leqq 1$. Errors in the modulus are from one to two orders of magnitude better than in [7]. It should be noted that for small α the distribution of boundary points is poor. This is true for most of the examples. The only reason for using the given distribution of points is to compare with [7]. Our experience shows that a good rule of thumb for the distribution of boundary points is to keep the distance between successive boundary points and the distance from the boundary points to the center in a nearly constant ratio. Thus, for α small, we want more points near the centers of the longer two sides and fewer points on the shorter two sides. This can be done by dividing the boundary into sections as mentioned in the paragraph following Eq. (5). We ran the problem with $\alpha = 0.1$ again, using a particularly simple redistribution of the boundary points. For a fixed number of points, the errors decreased by about 1/50. Using an optimal distribution of points, one would get more accuracy.

Example 3. *Ellipse.* $x^2/\alpha^2 + y^2 = 1$. The data points are uniformly distributed on the boundary of the ellipse. See Table IV.

TABLE IV

Ellipse

α	n	ERR-MOD	ERR-ARG
1.25	257	3×10^{-8}	—
	129	3×10^{-7}	2×10^{-7}
	65	4×10^{-6}	4×10^{-6}
	33	5×10^{-5}	4×10^{-5}
2.5	257	3×10^{-7}	—
	129	4×10^{-6}	5×10^{-6}
	65	5×10^{-5}	6×10^{-5}
	33	7×10^{-4}	9×10^{-4}
5.0	257	4×10^{-6}	—
	129	4×10^{-5}	5×10^{-5}
	65	7×10^{-4}	2×10^{-3}
	33	6×10^{-3}	5×10^{-3}
10	257	4×10^{-5}	—
	129	6×10^{-4}	6×10^{-4}
	65	5×10^{-3}	6×10^{-3}
	33	1×10^{-2}	6×10^{-2}
20	257	5×10^{-4}	—
	129	5×10^{-3}	6×10^{-3}
	65	1×10^{-2}	3×10^{-3}
	33	7×10^{-2}	5×10^{-2}

Again, we note monotonicity with respect to n and α for $\alpha \geqq 1$, with maximum error near the center of the side intersected by the minor axis. Improvements over [7] are from one to three orders of magnitude, with the least improvement for $\alpha = 20$.

Example 4. *Isosceles Triangle.* The corners of the triangle are at (0, 1), (2, −1), (−2, −1), and (0, 0) is mapped into the origin of the unit circle. There are equal numbers of points on each side.

TABLE V

Triangle

n	ERR-MOD	ERR-ARG
65	1×10^{-6}	—
33	2×10^{-5}	6×10^{-5}
17	2×10^{-4}	9×10^{-4}

4. Timing. There are three operations that are important as far as timing is concerned: (i) generating the matrix $\tilde{A}$ of Eq. (8), (ii) solving the matrix Eq. (8), and (iii) evaluating the function $f(z)$ at a given point. The time required for (i) is proportional to n^2 and is about 0.85 sec** for $n = 200$. The time required for (ii) is proportional to n^3 and is about 2 sec for $n = 200$. The time required for (iii) is proportional to n and is 0.016 sec for $n = 200$.

University of California
Los Alamos Scientific Laboratory
Los Alamos, New Mexico 87544

1. J. Hayes, *Four Computer Programs Using Green's Third Formula to Numerically Solve Laplace's Equation in Inhomogeneous Media,* Los Alamos Scientific Laboratory Report, LA-4423, April 1970.

2. J. Hayes & R. Kellner, *The Eigenvalue Problem for a Pair of Coupled Integral Equations Arising in the Numerical Solution of Laplace's Equation,* Los Alamos Scientific Laboratory Report, LA-DC-12009, October 1970.

→ M. A. Jaswon, "Integral equation methods in potential theory. I," *Proc. Roy. Soc. Ser. A,* v. 275, 1963, pp. 23–32. MR **27** #4034.

4. M. Maiti, "A note on the integral equation methods in potential theory," *Quart. Appl. Math.*, v. 25, 1968, pp. 480–484. MR **37** #447.

5. N. I. Muskhelišvili, *Singular Integral Equations. Boundary Problems of Function Theory and their Application to Mathematical Physics,* OGIZ, Moscow, 1946; English transl., Noordhoff, Groningen, 1953. MR **8,** 586; MR **15,** 434.

6. Philip Rabinowitz, "Numerical experiments in conformal mapping by the method of orthonormal polynomials," *J. Assoc. Comput. Mach.,* v. 13, 1966, pp. 296–303.

7. G. T. Symm, "An integral equation method in conformal mapping," *Numer. Math.,* v. 9, 1966, pp. 250–258. MR **34** #7056.

8. G. T. Symm, "Numerical mapping of exterior domains," *Numer. Math.,* v. 10, 1967, pp. 437–445. MR **36** #3525.

9. G. T. Symm, "Conformal mapping of doubly-connected domains," *Numer. Math.,* v. 13, 1969, pp. 448–457. MR **40** #3736.

** All times given here are for the CDC 7600.

to go to a more exact representation of this section of L?

Thus, there are three ways to approximate a boundary section.

ii. How many mesh points are to be required on this section? See Fig. 1.

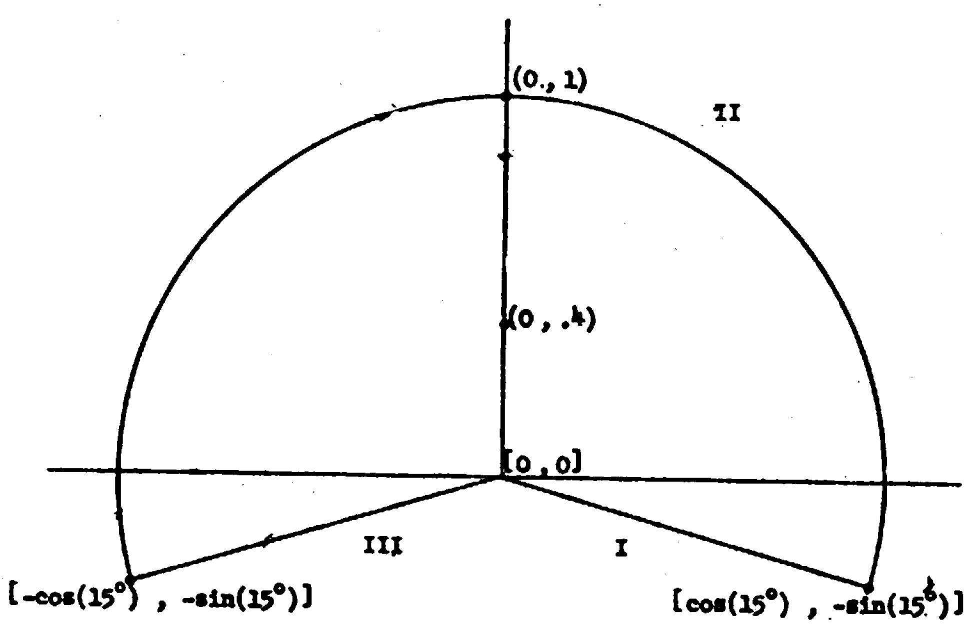

Fig. 1. Example

In the example there are three sections naturally imposed by the geometry. Sections I and III are exactly represented by line segments and Section II by a circular arc.

On the data cards a line segment is defined by giving its two endpoints. A circular arc is defined by giving its two endpoints and any interior point on the arc. This defines the type of approximation used for that boundary section.

The data card is seven fields long. Each of the first six fields is ten characters in length and is read with an E10.0 format. If a card is used to describe a line segment, the first four fields are used for the (x,y) coordinates of the beginning and end of the segment. The fifth and sixth fields must be blank. If a card is used to describe a circular arc, the first six fields are used to give the (x,y) coordinates of the beginning, interior, and endpoint of the arc. The endpoint of section k (corresponding to data card k) must agree with the initial point of section k+1 (data card k+1), and the endpoint of the last boundary section must agree with the initial point of the first boundary section.

The input for the curve L must have a positive (counterclockwise) orientation. The same orientation suffices for interior or exterior problems. The direction of the curve L is determined by the order of the points on the data cards. The direction for a line segment or circular arc is from the initial point to the endpoint. It has been our experience that errors in the orientation of L are difficult to detect.

The seventh field on the data card determines the number of mesh points n_s per section. These points are uniformly distributed with respect to arc length along the approximating curve. This number is read with an I3 format in columns 61-63. It must be odd and equal to three or more.

One possible set of data cards for the curve given in Fig. 1 is

80 COLUMN ENTRY

5	10	15	20	25	30	35	40	45	50	55	60	65	70	72	80
	0.		0.		.96593		-.25882					31			
	.96593		-.25882		0.		1.		-.96593		-.25882	85			
	-.96593		-.25882		0.		0.					31			

Note the positive, counterclockwise orientation.

As a convenience, one can leave the first two fields blank on a data card used to describe a line segment or circular arc if the previous card was either a line segment or circular arc. An equivalent set of data cards for Fig. 1 is then

80 COLUMN ENTRY

5	10	15	20	25	30	35	40	45	50	55	60	65	70	72	80
	0.		0.		.96593		-.25882					31			
					0.		1.		-.96593		-.25882	85			
					0.		0.					31			

The next two sets of data cards represent input for the rectangle given in Fig. 2. Here the natural sections are line segments, but in the second data set below, the long sides of the rectangle have themselves been divided into three sections. This allows for a different mesh spacing, if desired, on each section.

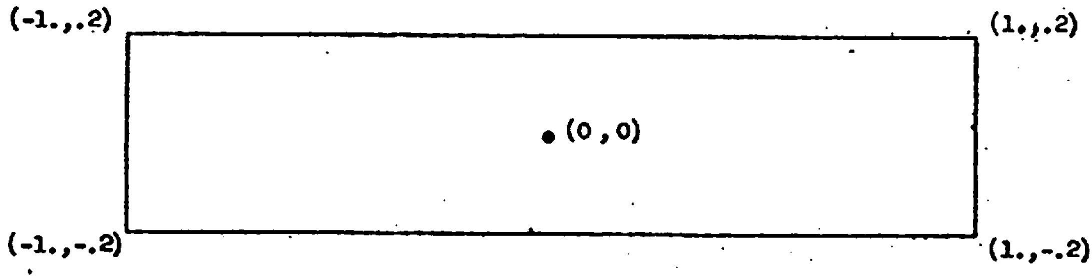

Fig. 2. Elongated rectangle.

80 COLUMN ENTRY

Cols 1–10	Cols 11–20	Cols 21–30	Cols 31–40	Cols 61–
-1.	-.2	1.	-.2	2 7
1.	-.2	1.	.2	3 7
1.	.2	-1.	.2	2 7
-1.	.2	-1.	-.2	3 7

80 COLUMN ENTRY

Cols 1–10	Cols 11–20	Cols 21–30	Cols 31–40	Cols 61–
-1.	-.2	-.2	-.2	2 2
		.2	-.2	2 1
		1.	-.2	2 1
		1.	.2	7
		.2	.2	2 1
		-.2	.2	2 1
		-1.	.2	2 1
		-1.	-.2	7

Occasionally, it is not satisfactory to represent a section of L by either a line segment or a circular arc. In that case "generalized" boundary input for that section is available in the form of a user provided subroutine BDRY. When line segments and circular arcs are the only type of input used, BDRY appears as a program that is referenced, but not loaded. This is normal and will cause no difficulty.

Suppose that the user decides to use "generalized" boundary data on the Ith and Jth section. He must then write a subroutine of the form

SUBROUTINE BDRY(K,T,X,Y,XP,YP,V).

Input to this subroutine will be K and T . Depending on when it is called K will take on the value I or J . T will be a real number ranging from zero to the length of the Kth subinterval. The pair of numbers K, T then uniquely defines a point T units in the positive direction along the Kth subinterval. If we denote that point by (X(T),Y(T)), the output of BDRY is

$$X : X(T)$$

$$Y : Y(T)$$

$$XP : \frac{dX(T)}{dT}$$

$$YP : \frac{dY(T)}{dT}$$

$$V = \frac{1}{2}\left[XP \cdot \frac{d^2Y(T)}{dT^2} - YP \cdot \frac{d^2X(T)}{dT^2}\right]$$

To keep the orientation correct (YP, - XP) must be the exterior normal at (X,Y).

To define the Kth section as being given by "generalized" boundary data, the Kth data card must have in columns 51-60 the arc length d of that section. Columns 1-50 must be blank and the number of mesh points appears as before in columns 61-63.

It is always possible to approximate sections of L by curves other than lines or circular arcs. In that case, the subroutine BDRY would output the parameters of the approximating curve, and d would be its length.

By way of illustration we give the input data and BDRY for the example of Fig. 1.

FORTRAN STATEMENT

```
      SUBROUTINE BDRY(I,T,X,Y,XP,YP,V)
      DATA  PI /3.14159265358979/
      GOTO(1,2,3),I
    1 X = T * COS(PI/12.)
      Y = -T*SIN(PI/12.)
      XP = X/T
      YP = Y/T
      V = 0.
      RETURN
    2 X = COS(T-PI/12.)
      Y = SIN(T - PI/12.)
      XP = -Y
      YP = X
      V = .5
      RETURN
    3 X = -COS(PI/12.) * (1.-T)
      Y = -SIN(PI/12.) * (1.-T)
      XP = -X/(1.-T)
      YP = -Y/(1.-T)
      V = 0.
      RETURN
      END
```

80 COLUMN ENTRY

1.	11
1.4142136	21
1.	31

It is, of course, necessary for the user to seperately compute the lengths of the "generalized" boundary sections.

Storage Requirements:

The present CMP requires about 37000_8 words of central memory. This includes all the subroutines, associated system programs, and internal dimensioned variables, but does not include the major storage needed for the matrix $\tilde{A}$ of Eq. (8).* If there are a total of n mesh points on the boundary then $n^2 + 1500_{10}$ words additional are needed. The CMP has been written to allow utmost flexibility in selecting the site for this storage. All the references to this data are through a subroutine ECRD with formal parameters

ECRD(A,M,L,K)

and additional entry point ECWR .

If the user operates on the LASL system with Extended Core Storage (ECS) available, this subroutine is generated by the system and he need not be concerned with it. His only requirement is to request $n^2 + 1500_{10}$ words of ECS and observe the restriction that $n \leq 630$ unless present dimension statements are changed.

If $n \leq 200$, the entire problem can be run in 200000_8 words of central memory. This is easily done by inserting into the deck the subroutine ECRD, a copy of which is included in the listings. In that subroutine the dimension of the local variable A must be at least $n^2 + 1500_{10}$. Problems with smaller n will run in even less central memory after the appropriate adjustments have been made to this dimension statement.

Users whose facilities prohibit the use of ECS or even 200000_8 words

*All referenced equations are in "An Improved Method for Numerical Conformal Mapping."

of central memory will find it possible to rewrite their own version of subroutine ECRD to store $\tilde{A}$ on tape or disk. To run a given problem the routine ECRD will need $n^2 + 1500_{10}$ words of some type of available storage. The entry point ECWR(A,M,L,K) should store L words from central memory starting at location A in available storage starting at location M . The entry point ECRD(A,M,L,K) should read L words from available storage starting at location M and store them starting at location A . The variable K is only used to make the subroutine compatible with the LASL system. K should merely be set equal to 0 on each entry.

To effect the conformal mapping it is necessary to call SUBROUTINE CONFORM(XO,YO,K). The parameters (XO,YO) define the point that we wish to map into the origin of the unit circle. The input parameter K can take on values 0 or 1, and determines whether an interior or exterior problem is to be solved. For K = 1 , an exterior problem, XO and YO are not used in the calculation. The call to this subroutine triggers the coding that will read in the data cards describing D (discussed above) and perform the calculations necessary to solve the integral equation of the first kind for the potential function $\sigma(t)$. The input data will be printed so that visual checks can be made. The only other printed output is a few possible error messages based on some consistency checks that are made on the data. All the other immediately available output, of which there is a great deal, resides in a COMMON block that the user must refer to (see below).

After the call to CONFORM has been completed, the image of any desired points can be obtained by calling

SUBROUTINE FM(X,Y,R,T)

with input X,Y . The output, R,T contains the image of point (X,Y) under the conformal map just executed. The image point is given in polar coordinates (R,Θ). This subroutine may be called as many times as necessary. Two restrictions on the use of FN are

(i) For interior problems, points (X,Y) outside D + L should not be given as input. For exterior problems points inside D + L should not be given.

(ii) Because of coding peculiarities, not all points on L can be given as input. The only completely acceptable input points on L are those appearing as mesh points on the boundary. Points halfway (in arc length) between the mesh points are also acceptable, but the value of T returned will be wrong. It is suggested that users wishing the images of boundary points other than mesh points perform quadratic interpolation through the mesh points. Errors in R and T are usually greatest for points near the boundary curve and least for points near the center of D. Exceptions to this are mesh points. Unless the nearest boundary section is a straight line, values of R and T obtained within h/4 of the boundary should not be trusted.

There are four labelled common blocks used to store intermediate data by the CMP. They are called CONF, MODE, ZLAP, and LTT. The card images are in Fig. 3.

```
      COMMON /MODE/ IN
      COMMON /CONF/ [illegible],XO,YO,[illegible]
     1 [illegible]
     2 C,N
      COMMON /ZLAP/ V,[illegible],ZZ,[illegible],DP,AR,[illegible]
     1 ,BO,ZE,[illegible]
      COMMON /LTT/ [illegible]
```

Fig. 3. Card Images of Common Statements

MODE contains only the variable IN where the third parameter in the call statement to CONFORM is stored. Thus IN = 0 for interior problems, and IN = 1 for exterior problems. The labelled common ZLFP is used to transmit information between subroutines. A user would probably never need to interact with these variables. The labelled common COMP contains the variables of most interest to the user. The variable NDC is used to store the number of boundary sections for a given problem. For instance, for the example in Fig. 1 NDC = 3. The dimensioned variable KV is used to give a running total of the number of approximation points on the boundary sections. We have KV(1) = 0 and KV(J+1) - KV(J) is equal to the number of approximation points on the Jth boundary section. For the input given for the example in Fig. 1 we have KV(1) = 0, KV(2) = 31, KV(3) = 116, and KV(4) = 147. The variable N is the total number of approximation points for a given problem. Obviously N = KV(NDC+1). The dimensioned variable HD is used to store the distance between approximation points along the curve L for each of the boundary sections given as input. For the example in Fig. 1 we have HD(1) = HD(3) = 1/30 and HD(2) = $\pi(1 + 1/6)/84$. The variables XO and YO are the coordinates of the inverse image of the origin for an interior problem. For an exterior problem these two numbers are set equal to zero. The dimensioned variables X(I), Y(I), I = 1, 2, ..., N are coordinates of the approximation points. For any given boundary section there are two approximation points at the ends of the boundary section and the other approximation points for that boundary are equally spaced with respect to arc length along the section. Because we define our points this way, it is true that [X(KV(J+1)), Y(KV(J+1))]

= [X(KV(J + 1) + 1), Y(KV(J + 1) + 1)] for J = 2, 3, ..., NDC-1 and [X(1),Y(1)] = [X(N),Y(N)]. It may seem wasteful to the user to have two set of coordinates for the same point, but it makes the programming logic much simpler for certain cases. For example, we must allow for discontinuities in the unit normal for L at corners and we must allow for discontinuities in the values of σ at corners. By defining our approximation points in the above manner, it makes the indexing much easier in these cases.

For I = 1, 2, ..., N, we have that [XN(I), - YN(I)] is the unit normal vector for L at the point [X(I), Y(I)]. The approximate value of σ at the point (X(I), Y(I)) is stored in the dimensioned variable F(I) for I = 1, 2, ..., N. For exterior problems F(N + 1) contains the logarithm of the approximate transfinite diameter. F is also used for intermediate computations in CONFORM.

For I = 1, 2, ..., N the variable G(I) contains the value of $\frac{1}{2}\left[\frac{dx}{dt}\frac{d^2y}{dt^2} - \frac{dy}{dt}\frac{d^2x}{dt^2}\right]$ at the point [X(I), Y(I)]. Between each consecutive pair of approximation points on each boundary section we store the location of an intermediate point on the boundary L. The coordinates of the intermediate point are stored in the dimensioned variables [XI,YI], and the corresponding unit normal is stored in [XIN,YIN]. Thus the point midway on the curve L between the points [X(1), Y(1)] and [X(2), Y(2)] has its coordinates stored in [XI(2), YI(2)] and the unit normal at that point is [XIN(2), -YIN(2)]. Because of the manner in which we store our approximation points, the points [XI(KV(J) + 1), YI(KV(J) + 1)] for J = 1, 2, ..., NDC are not defined even though the corresponding points [X(KV(J) + 1), Y(KV(J) + 1)] are well defined. The two dimensioned variables CN and CD are used to transmit information between subroutines. The N different integrals that appear in Eq. (7) are stored in CN. The corresponding N integrals for h(s) are stored in CD.

The variables CN and CD are equivalenced to the variables A and AZ respectively and are also used for intermediate storage when solving Eq. (8).

The labelled common LTT contains only the dimensioned variable DC . DC(J) for J = 1, 2, ..., NDC contains the arc length of the Jth boundary section.

```
      PROGRAM NEME (INPUT,OUTPUT)
C     TEST FOR CONFORMAL MAPPING PACKAGE
C     COMMON CONF IS NEEDED TO COMMUNICATE FOR ALL CALCULATIONS
      COMMON /CONF/ F(630),XO,YO,CD(630),CN(630),KV(210),HD(210),G(630),
     1X(630),XN(630),Y(630),YN(630),XI(630),XIN(630),YI(630),YIN(630),ND
     2C,N
C     DEFINE POINTS TO BE MAPPED INTO THE CENTER OF THE CIRCLE
      XO=YO=0.
C     SOLVE THE INTEGRAL EQUATION FOR SIGMA. FOR THIS PROBLEM, THE
C     BOUNDARY OF THE ELLIPSE IS DEFINED BY A USER SUPPLIED SUBROUTINE
      CALL CONFORM (XO,YO,1)
C     PRINT SIGMA
      PRINT 5
      PRINT 6, (F(M),M=1,N)
      D=EXP(F(N+1))
C     PRINT TRANSFINITE DIAMETER, STORED IN F(N+1).
C     THIS IS AN EXTERIOR PROBLEM
      PRINT 4, D
      PRINT 5
C     FIND THE IMAGES OF VARIOUS BOUNDARY POINTS
         DO 1 M=1,NDC
         M1=KV(M)+2
         M2=KV(M+1)
C     FIND THE IMAGES OF BOUNDARY MESH POINTS
         CALL FN (X(M1-1),Y(M1-1),R,TH)
         CALL OUTP (R,TH)
            DO 1 K=M1,M2
C     FIND THE IMAGES OF POINTS INTERMEDIATE TO MESH POINTS
            CALL FN (XI(K),YI(K),R,TH)
            TH=12345.
            CALL OUTP (R,TH)
            CALL FN (X(K),Y(K),R,TH)
1           CALL OUTP (R,TH)
C     THE FOLLOWING CALL CLEARS THE OUTPUT BUFFER
      CALL OUTC (R,TH)
C     THE FOLLOWING DO LOOP COMPUTES THE CONFORMAL MAP OF THREE
C     SEPARATE PROBLEMS.
C      1  A RECTANGLE WITH THE SAME NUMBER OF POINTS PER SIDE.
C      2. SAME RECTANGLE WITH A DIFFERENT DISTRIBUTION OF BOUNDARY
C     POINTS
C      3  A CIRCLE WITH A 160 DEGREE SECTOR CUT OUT.
C     THE FIRST TWO USE STRAIGHT LINE BOUNDARIES.  THE LAST USES
C     STRAIGHT LINE AND CIRCULAR ARC BOUNDARIES
         DO 3 I=1,3
         IF (I.EQ.3) YO=.4
         CALL CONFORM (XO,YO,0)
         PRINT 5
         PRINT 6, (F(M),M=1,N)
         PRINT 5
            DO 2 M=1,NDC
            M1=KV(M)+2
            M2=KV(M+1)
            CALL FN (X(M1-1),Y(M1-1),R,TH)
            CALL OUTP (R,TH)
               DO 2 K=M1,M2
               CALL FN (XI(K),YI(K),R,TH)
               TH=12345.
               CALL OUTP (R,TH)
               CALL FN (X(K),Y(K),R,TH)
2              CALL OUTP (R,TH)
         CALL OUTC (R,TH)
3        CONTINUE
C
4     FORMAT ('0','TRANSFINITE DIAMETER = ',E15.8)
5     FORMAT ('0')
6     FORMAT (1H 1P9E15.7)
      END
```

```
      SUBROUTINE OUTP (R,TH)
C     OUTPUT SUBROUTINE FOR TEST PROGRAM
      DIMENSION DAT(10), BCDDAT(10)
      DATA NUMB/0/
      DAT(NUMB+1)=R
      DAT(NUMB+2)=TH
      NUMB=NUMB+2
      IF (NUMB.NE.10) RETURN
 1    NUMB=0
         DO 2 I=1,10
         ENCODE (10,5,BCDDAT(I))DAT(I)
         IF (DAT(I).EQ.12345.) BCDDAT(I)=1H
 2       CONTINUE
      PRINT 4, (BCDDAT(I),I=1,10)
      RETURN
      ENTRY DUTC
 3    IF (NUMB.EQ.10) GO TO 1
      DAT(NUMB+1)=12345.
      DAT(NUMB+2)=12345.
      NUMB=NUMB+2
      GO TD 3
C
 4    FORMAT (10(2XA10))
 5    FORMAT (F10.7)
      END

      SUBROUTINE BDRY (M,S,X,Y,XP,YP,V)
C     GENERALIZED BOUNDARY ROUTINE FOR CONFORMAL MAP OF ELLIPSE
C     DIFFERENT SUBSECTIONS ARE TO UTILIZE SYMMETRY ON CURVE TO
C     AVOID REPEATING CALCULATIONS
C     ELLI IS AN ELLIPTIC INTEGRAL SUBROUTINE NEEDED IN THE CALCULATIONS
C     FOR THE BOUNDARY OF THE ELLIPSE
      DATA IFLAG/0/
      IF (IFLAG.EQ.1) GO TO 1
      AA=5.
      EK=SQRT(24.)/5.
      CALL ELLI (1.570796326,EK,Z1,Z2)
      BLO4=5.*Z2
      IFLAG=1
 1    IF (S.GT.BLO4*1.00000001) GO TO 2
      S1=S2=1.
      T=S
      GO TO 5
 2    IF (S.GT.BLO4*2.00000002) GO TO 3
      S1=1.
      S2=-1.
      T=2.00000002*BLO4-S
      GO TO 5
 3    IF (S.GT.BLO4*3.00000003) GO TD 4
      S1=-1.
      S2=-1.
      T=S-2.00000003*BLO4
      GO TO 5
 4    S1=-1.
      S2=1.
      T=4.00000004*BLO4-S
 5    TMI=0.
      TMA=1.570796326
         DO 6 J=1,40
         ZI=.5*(TMA+TMI)
         CALL ELLI (ZI,EK,Z,SI)
         IF (SI*AA.GE.T) TMA=ZI
         IF (SI*AA.LE.T) TMI=ZI
 6       CONTINUE
      X=-AA*SIN(ZI)*S1
      Y=SQRT(1.-(X/AA)**2)*S2
      A2=Y*AA**2
      XP=-A2/SQRT(A2**2+X**2)
      YP=X/SQRT(A2**2+X**2)
      V=.5*AA**4/(X**2+A2**2)**1.5
      F=0.
      RETURN
      END
```

```
      SUBROUTINE ELLI (CHI,CAY,F,E)
C     THE FOLLOWING ROUTINE IS AN ELLIPTIC INTEGRAL ROUTINE USED ONLY IN
C     THE CALCULATION OF THE BOUNDARY OF THE ELLIPSE
      DIMENSION MESG1(10), MESG2(10)
      DATA MESG1/50HELLI - K.GT. ONE           F=E=PHI                 ,5
     1 0HCELLI - K.GT. ONE           F=E=PHI             /
      DATA MESG2/50HELLI- K=1,PHI.GE.PI/2,SO F SET TO SIGN(PHI)*1.E+294,5
     1 0HCELLI - K=1, SO F SET TO 1.E+294                /
      DATA P12,PI,TOP/17206220773250420551B,3.14159265358979,17175057400
     1 333447104B/
      DATA EPS,EPS2/1.E-13,2.E+13/
      PHI=CHI
      IND=1
1     F=PHI
      E=F
      PSI=ABS(PHI)
      IF ((PSI.LT.EPS).OR.(CAY.EQ.0.)) RETURN
      S1=CAY*CAY
      RAD=1.-S1
      IF (RAD) 10,9,2
2     ALPHA=1.
      BETA=SQRT(RAD)
      S2=0.
      PR2=1.
      PWR2=1.
      FINT=PSI*TOP
      NQUAD=INT(FINT)
      FINT=FINT-FLOAT(NQUAD)
      IF (AMIN1(FINT,1.-FINT).LT..5E-13) GO TO 6
      TANPSI=TAN(PSI)
      NQUAD=NQUAD+1
3     IF (ABS(ALPHA-BETA).LE.EPS) GO TO 7
      PWR2=2.*PWR2
      PR2=PWR2
      DENOM=ALPHA-BETA*TANPSI**2
      TOP=(ALPHA+BETA)*TANPSI
      CN=.5*(ALPHA-BETA)
      S1=S1+PWR2*CN*CN
      TEMP=SQRT(ALPHA*BETA)
      ALPHA=(ALPHA+BETA)*.5
      BETA=TEMP
      IF (DENOM.EQ.0.) GO TO 4
      TANPSI=TOP/DENOM
      IF (ABS(TANPSI).GE.EPS2) GO TO 4
      NQUAD=2*NQUAD
      IF (TANPSI.GT.0.) NQUAD=NQUAD-1
      IF (ABS(TANPSI).LE.EPS) GO TO 5
      MQUAD=MOD(NQUAD,4)
      SINP=TANPSI/SQRT(1.+TANPSI*TANPSI)
      IF ((MQUAD.EQ.3).OR.(MQUAD.EQ.2)) SINP=-SINP
      S2=S2+CN*SINP
      GO TO 3
4     FINT=2*NQUAD-1
      PSI=FINT*P12
      SINP=1.
      IF (AMOD(FINT,4.).GT.2.) SINP=-1.
      S2=S2+CN*SINP
      GO TO 6
5     FINT=NQUAD/2
      PSI=FINT*PI
6     IF (ABS(ALPHA-BETA).LT.EPS) GO TO 8
      PR2=2.*PR2
      CN=.5*(ALPHA-BETA)
      S1=S1+PR2*CN*CN
      TEMP=SQRT(ALPHA*BETA)
      ALPHA=(ALPHA+BETA)*.5
      BETA=TEMP
      GO TO 6
7     FINT=NQUAD/2
      PSI=ATAN(TANPSI)+FINT*PI
```

```
8         F=PSI/(ALPHA*PWR2)
          S1=1.-0.5*S1
          E=S2+S1*F
          F=SIGN(F,PHI)
          E=SIGN(E,PHI)
          RETURN
9         FINT=AINT(TOP*PSI)
          E=FINT+ABS(SIN(PSI)-SIN(FINT*P12))
          E=SIGN(E,PHI)
          IF (PSI.GE.P12) GO TO 11
          DENOM=1.-SIN(PHI)
          IF ((DENOM.EQ.0.).OR.(DENOM.EQ.2.)) GO TO 11
          F=.5*ALOG((2.-DENOM)/DENOM)
          RETURN
          ENTRY CELLI
          PHI=P12
          IND=6
          GO TO 1
10        STOP
          RETURN
11        F=SIGN(1.E+294,PHI)
          STOP
          RETURN
          END

          SUBROUTINE CONFORM (XP,YP,IP)
C         ************* CONFORMAL MAPPING PACKAGE************************
C         ************* SUBROUTINES REQUIRED ****************************
C                         CONFORM
C                         ATAN3
C                         FN
C                         BDRZ
C                         QV
C                         ECSW
C                         ECRD, IFF ALL CALCULATIONS ARE ARE DONE IN CORE
C                         BDRY, IFF GENERALIZED BOUNDARY DATA ARE USED
C                         PIVOTL,  IFF THE DO LOOP IN CONFORM ENDING AT
C                              STATEMENT 28 IS LEFT OUT
C         ***************************************************************
C
C         .......   ...... VARIABLES USED ..........................
C
C         X,Y   VECTORS OF BOUNDARY POINTS
C         (XN(I),YN(I))    UNIT NORMAL VECTOR AT (X(I),Y(I))
C         G(I)  CURVATURE AT (X(I),Y(I))
C         XI,YI   VECTORS OF POINTS ON BOUNDARY INTERMEDIATE TO X,Y
C         (XIN(I),YIN(I))  UNIT NORMAL AT I TH INTERMEDIATE BOUNDARY POINT
C         KV(I)  IS THE NUMBER OF BOUNDARY POINTS ON THE FIRST THROUGH I-IST
C              BOUNDARY SECTION
C         NDC   TOTAL NUMBER OF BOUNDARY SECTIONS
C         DC(M)    LENGTH OF M TH BOUNDARY SECTION ,SET IN SUBROUTINE BDRZ
C         HD(M)    ARCLENGTH SPACING BETWEEN BOUNDARY POINTS ON M TH SECTION
C         N=KV(NDC+1)  TOTAL NUMBER OF BOUNDARY POINTS
C         F     1.  AT EXIT FROM LAPLACE  F STORES  SIGMA AT BOUNDARY POINTS
C                  (X(I), Y(I)) ,I=1,N
C               2.  DURING  COMPUTATIONS  F STORES RIGHT HAND SIDE OF MATRIX
C                  EQUATION
C         IP EQUALS 0 FOR INTERIOR PROBLEM, 1 FOR EXTERIOR PROBLEM
C         ...............................................................
          DIMENSION A(630), AA(630)
          COMMON /MODE/ IN
          COMMON /CONF/ F(630),XO,YO,CD(630),CN(630),KV(210),HD(210),G(630),
     1    X(630),XN(630),Y(630),YN(630),XI(630),XIN(630),YI(630),YIN(630),ND
     2    C,N
          COMMON /ZLAP/ V,Q(12),H,HS,HSO2,HCO3,HI,HIS,ZZ,BZ,DP,AR,ME1,HSM,FB
     1    ,BQ,ZE,RI,W
          COMMON /LTT/ DC(210)
          LOGICAL BZ,IT,T1,T2
          EQUIVALENCE (A,CN), (AA,CD)
C         THE LOGICAL VARIABLE BZ=F MEANS THE PROBLEM HAS BEEN
C         RUN TO COMPLETION
```

```
      BZ=.T.
      IF (IP.NE.1.AND.IP.NE.0) STOP
      XO=XP
      YO=YP
      IN=IP
      KV(1)=0
      M=1
      IF (IP.NE.1) GO TO 1
      XO=0.
      YO=0.
1     CALL BDRX (M,S,Z1,Z2,Z3,Z4,Z5)
C     READ BOUNDARY DATA CARD CORRESPONDING TO SECTION M.
C     Z1=0. IMPLIES A BLANK CARD
      IF (Z1.EQ.0.) GO TO 3
      NDC=M
      KV(M+1)=KV(M)+IFIX(Z2)
      HD(M)=DC(M)/FLOAT(KV(M+1)-KV(M)-1)
      M1=KV(M)+1
      N=KV(M+1)
C     THE FOLLOWING DO LOOP GENERATES BOUNDARY AND INTERMEDIATE POINTS
C     FOR SECTION M
        DO 2 L=M1,N
        Z=AMIN1(DC(M),FLOAT(L-M1)*HD(M))
        IF (L.NE.M1) CALL BDRZ (M,-0.5*HD(M),XI(L),YI(L),YIN(L),XIN(L),
     1  A)
        CALL BDRZ (M,Z,X(L),Y(L),YN(L),XN(L),G(L))
2       CONTINUE
      M=M+1
      GO TO 1
C     LAST DATA CARD FOR THIS COMPUTATION HAS BEEN READ
3     CONTINUE
C     ALL DATA READ. START GENERATING EQUATIONS
        DO 4 KO=1,N
C     COMPUTE AND STORE EACH ROW OF THE MATRIX A AND VECTOR E.
        CALL FN (X(KO),Y(KD),B,C)
C     EACH CALL TO FN GENERATES THE LEFT HAND SIDE OF THE KD TH EQUATION
C     AND STORES IT IN CN
        F(KD)=.5*ALOG((X(KD)-XP)**2+(Y(KD)-YP)**2)
C     ECSW TRANSFERS ROW OF MATRIX TO SUPPLEMENTAL STORAGE.
C     DISK, CORE, TAPE, ECS, AT USERS OPTION
        CN(N+1)=1.
4       CALL ECSW (CN,KD,1,N+IN)
C     INSERT THE CORNER CONDITIONS
        DO 10 M=1,NDC
        M1=KV(M+1)
          DO 5 L=1,NDC
          M2=KV(L)+1
          IF (ABS(X(M2)-X(M1))+ABS(Y(M2)-Y(M1)).LT.HD(M)*1.E-4)
     1    GO TO 6
5         CONTINUE
        PRINT 19, M
        RETURN
C     AT THIS POINT IT MUST BE TRUE THAT THE END OF M-TH BOUNDARY
C     SECTION JOINS THE START OF THE L-TH BOUNDARY SECTION
C     AN+PI IS THE ANGLE AT WHICH THE CURVE M INTERSECTS THE CURVE L.
6       AN=ATAN2(YN(M1)*XN(M2)-XN(M1)*YN(M2),XN(M1)*XN(M2)+YN(M1)*YN(M2
     1  ))
        IF (ABS(AN).LT.1.E-4.OR.ABS(1.5707963-ABS(AN)).LT.1.E-4.OR.IN.E
     1  Q.1
     1  ) GO TO 7
        CALL FN (XI(M2+1),YI(M2+1),B,C)
        A(N+1)=1.
        CALL ECSW (A,M2,1,N+IN)
        F(M2)=.5*ALOG((XI(M2+1)-XP)**2+(YI(M2+1)-YP)**2)
        CALL FN (XI(M1),YI(M1),B,C)
        A(N+1)=1.
        CALL ECSW (A,M1,1,N+IN)
        F(M1)=.5*ALOG((XI(M1)-XP)**2+(YI(M1)-YP)**2)
        GO TO 10
7         DO 8 L1=1,N
8         A(L1)=0.
        A(N+1)=0.
        CALL ECSW (A,M1,1,N+IN)
        F(M1)=0.
        IF (ABS(AN).LT.1.E-4.OR.IN.EQ.1) GO TO 9
```

```
      CALL ECSW (A,M2,1,N+IN)
      F(M2)=0.
      CALL ECSW (1.,M1,M1,1)
      CALL ECSW (1.,M2,M2,1)
      GO TO 10
9     CONTINUE
C    ANGLE BETWEEN TWO BOUNDARY SECTIONS IS ZERO, HENCE SIGMA IS
C    THE SAME AT BOTH ENDPOINTS
C    -1. IN A(M1,M2)
C    1. IN A(M1,M1)
      CALL ECSW (-1.,M1,M2,1)
      CALL ECSW (1.,M1,M1,1)
10    CONTINUE
      DO 12 M=1,NDC
      M1=KV(M)+2
      M2=KV(M+1)
         DO 11 J=M1,M2,2
         CN(J)=4.*HD(M)
11       CN(J+1)=2.*HD(M)
      CN(M1-1)=HD(M)
12    CN(M2)=HD(M)
     CN(N+1)=0.
     IF (IN.EQ.1) CALL ECSW (CN,N+1,1,N+IN)
     F(N+1)=0.
     L=N+IN
C    SOLVE THE MATRIX EQUATION AF=D USING GAUSSIAN ELIMINATION.
     NM=L-1
      DO 17 J=1,NM
      M2=J+1
C    THE FOLLOWING 14 STATEMENTS ARE FOR ROW PIVOTING ON THE LARGEST
C    ELEMENT OF THE J-TH COLUMN.
      CALL ECSR (T,J,J,1)
      T=ABS(T)
      M1=J
         DO 13 I=M2,L
         CALL ECSR (Z,I,J,1)
         Z=ABS(Z)
         IF (Z.LE.T) GO TO 13
         T=Z
         M1=I
13       CONTINUE
      CALL ECSR (AA(J),M1,J,L+1-J)
      IF (M1.EQ.J) GO TO 14
      CALL ECSR (A(J),J,J,L+1-J)
      CALL ECSW (A(J),M1,J,L+1-J)
      T=F(J)
      F(J)=F(M1)
      F(M1)=T
14    F(J)=F(J)/AA(J)
         DO 15 I=J,L
         K=L+J-I
15       AA(K)=AA(K)/AA(J)
      CALL ECSW (AA(J),J,J,L+1-J)
         DO 16 I=M2,L
         CALL ECSR (A(J),I,J,L+1-J)
         CALL PIVOTL (A,AA,M2,L)
C
C    ********************MACHINE LANGUAGE REPLACEMENT****************
C          THE PREVIOUS CALL IS TO A MACHINE LANGUAGE CODE TO REPLACE
C          THE L**3 DO LOOP BELOW.  IF THIS CALL IS REMOVED, AND THE
C          COMMENTS REMOVED FROM THE DO, THE CODE WILL BE ALL FORTRAN
C    DO28K=M2,L
C    28 A(K)=A(K)-AA(K)*A(J)
C    **************MACHINE LANGUAGE REPLACEMENT**********************
         F(I)=F(I)-F(J)*A(J)
16       CALL ECSW (A(M2),I,M2,L+1-M2)
17    CONTINUE
C    THE FOLLOWING TEN STATEMENTS ARE FOR BACK SUBSTITUTION.
     CALL ECSR (ZS,L,L,1)
```

```
      F(L)=F(L)/ZS
      CALL ECSW (1.,L,L,1)
         DO 18 I=2,L
         N1=L+1-I
         CALL ECSR (A(N1),N1,N1,L+1-N1)
         I1=I-1
           DO 18 J=1,I1
           L1=L+1-J
18         F(N1)=F(N1)-F(L1)*A(L1)
      BZ=.F.
      RETURN
C
19    FORMAT (50H THE BOUNDARY IS NOT CLOSED AT THE END OF SECTION ,I2)
      END

      REAL FUNCTIONATAN3(Y,X)
      ATAN3=ATAN2(Y,X)
C     THIS ROUTINE COMPUTES THE ARCTAN OF Y/X .  THE COMPLICATED LOGIC
C     INSURES THAT THIS FUNCTION IS ALWAYS INCREASING AS THE CURVE
C     IS TRAVERSED IN POSITIVE SENSE IF THE DOMAIN IS CONVEX
C     IF THE SITUATION 0/0. OCCURS WE FORCE THE ANGLE TO INCREASE
1     IF (ABS(ATAN3-ANGPRE).LT.3.1416) GO TO 3
      ATAN3=ATAN3+2.*3.14159265358988
      GO TO 1
      ENTRY ATAN4
      ATAN3=0.
      ANGPRE=ATAN2(Y,X)
      IF (ANGPRE.LE.1.E-12) ANGPRE=ANGPRE+2.*3.14159265358988
      RETURN
      ENTRY ATAN5
      ATAN3=ATAN2(Y,X)
2     IF (ATAN3.GT.ANGPRE+.05) GO TO 3
      ATAN3=ATAN3+2.*3.14159265358988
      GO TO 2
3     ANGPRE=ATAN3
      RETURN
      END

      SUBROUTINE FN (S,T,R,TH)
C     FN SERVES DUAL FUNCTION
C     1. GENERATE ONE ROW OF MATRIX, AND RETURN
C     2. COMPUTES (R,TH) CORRESPONDING TO INPUT (S,T)
      LOGICAL BZ,RI,BQ
      COMMON /ZLAP/ V,Q(12),H,HS,HSO2,HCO3,HI,HIS,ZZ,BZ,DP,AR,ME1,HSM,FB
     1,BQ,ZE,RI,W
      COMMON /CONF/ F(630),XO,YO,CD(630),CN(630),KV(210),HD(210),G(630),
     1X(630),XN(630),Y(630),YN(630),XI(630),XIN(630),YI(630),YIN(630),ND
     2C,N
      COMPLEX GP
      IF (ABS(X(1)-S)+ABS(Y(1)-T).GE.1.E-3*HD(1)) GM=ATAN4(Y(1)-T,X(1)-S
     1)
      IF (ABS(X(1)-S)+ABS(Y(1)-T).LT.1.E-3*HD(1)) GM=ATAN4(-XN(1),-YN(1)
     1)
      GM=HM=0.
C     PUT THE VARIABLES IN COMMON.
      V=S
      W=T
         DO 1 ME0=1,NDC
C     INITIALIZE VARIABLES FOR THE ROUTINE QV.
         M0=KV(ME0)+1
         ME1=KV(ME0+1)-2
         H=HD(ME0)
         ZZ=ALOG(H)
         HSO2=H**2/2.
         HCO3=H**3/3.
         TPIH2=.25/(3.14159265358988*HSO2)
         TPIH=.5*TPIH2
         HS=HSO2*2.
         HS2=2.*HS
         H2=2.*H
         H3=3.*H
```

```
      HI=1./H
      HSM=.08*HS
      HIS=1./HSO2
      ZE=H/(45.*3.14159265358979)
      CN(M0)=CD(M0)=0.
C     THE ENTRY POINT QVF INITIALIZES VARIABLES USED BY QV.
      CALL QVF (M0+1)
C     STORES MATRIX ROW IN CN
C     BQ FALSE --COMPUTE INTEGRAL THE LONG WAY --POLYNOMIAL  REPLACEMENT
C     BQ TRUE --COMPUTE INTEGRAL THE SHORT WAY  --SIMPSONS RULE
      BQ=.F.
         DO 1 M1=M0,ME1,2
         CALL QVS (M1+1)
         IF (BQ) GO TO 1
C     Q(K+1) . K=0,1,2 CONTAIN INTEGRALS OF T**K*THETA FOR 2. ABOVE
C     Q(K+4) . K=0,1,2 CONTAIN INTEGRALS OF T**K*LOG(R) FOR 2. ABOVE.
C     OR R SMALL RELATIVE TO H
C     CN(K)  INTEGRAL OF PIECEWISE POLYNOMIAL OF DEGREE 2. TIMES LOG
C     CD(K)  INTEGRAL OF PIECEWISE POLYNOMIAL OF DEGREE 2. TIMES ARCTAN
         TT=Q(3)-H3*Q(2)+HS2*Q(1)
         TS=Q(6)-H3*Q(5)+HS2*Q(4)
         CD(M1+1)=H2*Q(2)-Q(3)
         CN(M1+1)=H2*Q(5)-Q(6)
         CD(M1+2)=Q(3)-H*Q(2)
         CN(M1+2)=Q(6)-H*Q(5)
         CALL QV (M1+2)
         CD(M1)=CD(M1)+(TT+Q(3)-H*Q(2))*TPIH2
         CN(M1)=CN(M1)+(TS+Q(6)-H*Q(5))*TPIH
         CD(M1+1)=(CD(M1+1)+Q(1)*HS-Q(3))*TPIH2*2.
         CN(M1+1)=(CN(M1+1)+Q(4)*HS-Q(6))*TPIH*2.
         CD(M1+2)=(CD(M1+2)+Q(3)+H*Q(2))*TPIH2
         CN(M1+2)=(CN(M1+2)+Q(6)+H*Q(5))*TPIH
1        CONTINUE
      IF (BZ) RETURN
C     COMPUTE APPROXIMATION OF G AND H BY COMPUTING DOT PRODUCT OF
C     SIGMA WITH CN AND CD
        DO 2 M=1,N
        GM=GM-CN(M)*F(M)
        HM=HM+CD(M)*F(M)
2       CONTINUE
      GP=CMPLX(S-XO,T-YO)*CEXP(CMPLX(GM-F(N+1),HM))
      R=CABS(GP)
      TH=ATAN2(AIMAG(GP),REAL(GP))
      RETURN
      END

      SUBROUTINE BDRZ (M,S,XX,YY,XP,YP,G)
C     GIVEN INPUT-M-BOUNDARY SECTION, S=ARCLENGTH ALONG THAT SECTION
C     OUTPUT  XX,YY,XP,YP,G. =COORDINATES OF POINT . DXDS, DYDS, AND
C     CURVATURE
C     BDRZ ENTRY SETS UP DATA POINTS ON BOUNDARY CURVE AFTER BDRX PICKS
C     TYPE OF SECTION
      COMMON /LTT/ DC(210)
C     THE FUNCTION ABE IS USED TO DETERMINE IF A CARD FIELD IS BLANK
C     ABE=-1  IF AND ONLY IF X=-0.
      ABE(X)=ABS(X)+SIGN(1.,X)
      DIMENSION A(7)
      COMPLEX Z2,Z3
      EQUIVALENCE (A(1),GM), (A(3),AL), (A(4),BE), (A(5),R), (A(6),D), (
     1 A(7),GA1)
      CALL ECSR7W (A,M,0,7)
      IF (GM.NE.0.) GO TO 1
      IF (SIGN(1.,GM).NE.-1.) GO TO 2
C     CALL BDRY IF THE BOUNDARY DATA IS GENERALIZED
      CALL BDRY (M,S,XX,YY,XP,YP,G)
      RETURN
C     GENERATE BOUNDARY INFORMATION FOR CIRCULAR ARC
1     XX=R*COS(D*S/R+GA1)+AL
      YY=R*SIN(D*S/R+GA1)+BE
      XP=-(YY-BE)*D/R
```

```
            YP=(XX-AL)*O/R
            GO TO 3
C           GENERATE BOUNDARY INFORMATION FOR A LINE SEGMENT
2           XX=AL*S+D
            YY=BE*S+R
            XP=AL
            YP=BE
3           G=GM
            RETURN
            ENTRY BDRX
C           ENTRY BDRX READS DATA CARDS AND DECIDES IF BOUNDARY SECTION IS
C           LINE, CIRCULAR ARC, OR GIVEN EXPLICITLY BY USER SUBROUTINE
            XX=0.
C           READ BOUNDARY DATA CARD
4           READ 11, X1,Y1,X2,Y2,X3,Y3,KV
            IF (ABE(X1)+ABE(Y1)+ABE(X2)+ABE(Y2)+ABE(X3)+ABE(Y3).NE.-6.) GO TO
     1      5
            IF (M.NE.1) RETURN
C           IF ANY CARD IS BLANK EXCEPT THE FIRST ONE, RETURN WITH XX=0.
            GO TO 4
5           XX=1.
            KV=MAX0(KV,3)+MOD(MAX0(KV,3),2)-1
C           REQUIRES NUMBER OF POINTS ON A SECTION BE 3 OR MORE AND ODD
            YY=FLOAT(KV)+.1
            IF (MOD(M,55).EQ.1) PRINT 12
C           55 LINES PER PAGE, NEW HEADING ON EACH PAGE
            IF (ABE(X1)+ABE(Y1)+ABE(X2)+ABE(Y2)+ABE(X3).GT.-5.) GO TO 6
C           THIS SECTION HAS GENERALIZED BOUNDARY DATA
            PRINT 13, M,Y3,KV
            CALL ECSR7W (-0.,M,1,1)
C           ECSR7W IS ENTRY POINT TO ECSW, WRITES 7 WORDS INTO ECS,
C           CORRESPONDING TO ONE BOUNDARY SECTION. FOR GENERALIZED BOUNDARY
C           THIS IS WRITTEN AS -0.
            DC(M)=Y3
            IF (DC(M).GT.0.) RETURN
            PRINT 14
            STOP
6           IF (ABE(X1)+ABE(Y1).NE.-2.OR.M.EQ.1) GO TO 7
C           CHECK TO SEE IF BOUNDARY SECTION IS LINE OR CIRCULAR ARC
C           IF FIRST TWO FIELDS BLANK,BOUNDARY SECTION STARTS AT THE
C           END OF LAST SECTION
            X1=XO
            Y1=YO
7           IF (ABE(X3)+ABE(Y3).EQ.-2.) GO TO 8
            PRINT 15, M,X1,Y1,X2,Y2,X3,Y3,KV
            GO TO 10
C           PROCESS BOUNDARY DATA
C
8           PRINT 16, M,X1,Y1,X2,Y2,KV
C           THIS SECTION IS FOR PROCESSING BOUNDARY CARDS HAVING LINE SEGMENTS
            GM=0.
            F=SQRT((X1-X2)**2+(Y1-Y2)**2)
            D=X1
            R=Y1
            AL=(X2-X1)/F
            BE=(Y2-Y1)/F
            XO=X2
            YO=Y2
9           CALL ECSR7W (A,M,1,7)
            DC(M)=F
            RETURN
C           THIS SECTION IS FOR PROCESSING BOUNDARY CARDS HAVING CIRCULAR ARCS
10          R2=(X1**2+Y1**2)-(X2**2+Y2**2)
            R3=(X1**2+Y1**2)-(X3**2+Y3**2)
            DET=4.*((X1-X2)*(Y1-Y3)-(Y1-Y2)*(X1-X3))
            AL=2.*((R2)*(Y1-Y3)-(R3)*(Y1-Y2))/DET
            BE=2.*((X1-X2)*R3-(X1-X3)*(R2))/DET
            R=SQRT((X1-AL)**2+(Y1-BE)**2)
            GA1=ATAN2(Y1-BE,X1-AL)
```

```
      Z2=CMPLX(X2-AL,Y2-BE)/CMPLX(X1-AL,Y1-BE)
      Z3=CMPLX(X3-AL,Y3-BE)/CMPLX(X2-AL,Y2-BE)
      D=SIGN(1.,AIMAG(Z2))
      T1=ATAN2(AIMAG(Z2),REAL(Z2))
      T2=ATAN2(AIMAG(Z3),REAL(Z3))
      IF (ABS(T1).GT.ABS(T2)) D=SIGN(1.,AIMAG(Z3))
      F=(ATAN2(Y3-BE,X3-AL)-GA1)*R*O
      IF (F.LT.0.) F=F+6.28318530717958*R
      XD=X3
      YD=Y3
      GM=D*.5/R
      GO TO 9
C
11    FORMAT (6E10.0,I3)
12    FORMAT ('1SECTION',72X,'Y3, OR SPECIAL        POINTS IN
     1 '/' NUMBER',6X,'X1',12X,'Y1',12X,'X2',12X,'Y2',12X,'X3',9X,'BOUND
     2ARY LENGTH       SECTION              ')
13    FORMAT (1H I3,74X,1PE14.6,12X,I3)
14    FORMAT ('0 THE NEXT      BOUNDARY SECTION HAS NON-POSITIVE LENGTH.'
     1)
15    FORMAT (1H I3,4X,1P6E14.6,2X,10X,I3)
16    FORMAT (1H I3,4X,1P4E14.6,40X,I3)
      END

      SUBROUTINE QV (I)
      LOGICAL BQ
      COMMON /ZLAP/ V,Q(12),H,HS,HSO2,HCO3,HI,HIS,ZZ,BZ,DP,AR,ME1,HSM,FB
     1,BQ,ZE,RI,W
      COMMON /CONF/ F(630),XO,YO,CD(630),CN(630),KV(210),HD(210),G(630),
     1X(630),XN(630),Y(630),YN(630),XI(630),XIN(630),YI(630),YIN(630),ND
     2C,N
      DIMENSION HP(13)
C     THE VARIABLES R3 AND AL3 ARE STORED FROM A PREVIOUS CALL TO QV.
1     AL1=AL3
      R1=R3
      BQ=.F.
C     THE VARIABLES R1,R2 AND R3 ARE USED TO CONSTRUCT THE POLYNOMIAL
C     Q(T). FOLLOWING EQ. (9) OF THE WRITE-UP.
      R2=(XI(I)-V)**2+(YI(I)-W)**2
      R3=(X(I)-V)**2+(Y(I)-W)**2
      AL2=(XI(I)-V)*XIN(I)-(YI(I)-W)*YIN(I)
      AL3=(X(I)-V)*XN(I)-(Y(I)-W)*YN(I)
C     THE FOLLOWING TWO STATEMENTS ARE USED TO DETERMINE IF AND WHERE
C     Q(T)=0.
      IF ((R1.GE.HSM).AND.(R2.GE.HSM).AND.(R3.GE.HSM)) GO TO 2
C     THE FOLLOWING TWO STATEMENTS ARE USED TO DETERMINE WHERE Q(T)=0.
      IF ((R3.LT.HS*1.E-6).OR.(R1.LT.HS*1.E-6)) GO TO 10
      IF (R2.LT.HS*1.E-6) GO TO 15
C     THIS SECTION OF THE PROGRAM EVALUATES THE INTEGRALS IN EQ. (9)
2     A=((R1+R3)-2.*R2)*HIS
      B=(4.*R2-(3.*R1+R3))*HI
      AL=(AL1+AL3-2.*AL2)*HIS
      BE=(4.*AL2-(3.*AL1+AL3))*HI
      ELC=EL
      EL=ALOG(R3)
      IF (ABS(A).LT.R1*HIS*1.E-4) GO TO 7
C     IF A*H**2 IS VERY SMALL COMPARED WITH R1 AND ONE USES THE EXPLICIT
C     EVALUATION OF THE INTEGRALS,THEN ROUND-OFF ERROR WILL BE A PROBLEM
      P=4.*A*R1-B*B
      E=2.*R1+B*H
      PS=SQRT(ABS(P))
C     THE FOLLOWING TWO STATEMENTS DETERMINE WHETHER THE DISCRIMINANT OF
C     Q(T) IS POSITIVE,NEGATIVE OR APPROXIMATELY ZERO.
      IF (P.GE.HS*1.E-8) GO TO 4
      IF (P.GE.-HS*1.E-8) GO TO 3
      Q0=ALOG((E+H*PS)/(E-H*PS))/PS
      GO TO 5
3     Q0=2.*H/E
      GO TO 5
4     Q0=2.*ATAN2(H*PS,E)/PS
5     E=1./A
      ZZZ=ATAN3(Y(I)-W,X(I)-V)
      Q1=.5*E*(EL-ELC-B*Q0)
      Q2=E*(H-B*Q1-R1*Q0)
      Q3=E*(HSO2-B*Q2-R1*Q1)
      Q4=E*(HCO3-B*Q3-R1*Q2)
      Q5=E*(H**4/4.-B*Q4-R1*Q3)
6     Q(1)=AL*Q3+BE*Q2+AL1*Q1-H*ZZZ
      Q(2)=(AL*Q4+BE*Q3+AL1*Q2)/2.-HSO2*ZZZ
      Q(3)=(AL*Q5+BE*Q4+AL1*Q3)/3.-HCO3*ZZZ
```

```
      Q(4)=EL*H-2.*A*Q2-B*Q1
      Q(5)=EL*HSO2-A*Q-0.5*B*Q2
      Q(6)=EL*HCO3-(2.*A*Q4+B*Q3)*.3333333333333
      RETURN
C     IF B*H IS ALSO SMALL IN COMPARISON WITH R1 THEN ANOTHER METHOD IS
C     NEEDED TO EVALUATE Q0,Q1.....Q4.
7     IF (ABS(B).LT.R1*HI*1.E-3) GO TO 8
      ZZZ=ATAN3(Y(I)-W,X(I)-V)
      BI=1./B
      Q0=BI*ALOG(1.+B*H/R1)
      Q1=BI*(H-R1*Q0)
      Q2=BI*(HSO2-R1*Q1)
      Q3=BI*(HCO3-R1*Q2)
      Q4=BI*(HSO2**2-R1*Q3)
      Q5=BI*(H**5/5.-R1*Q4)
      GO TO 6
8        DO 9 J=4,12
9        HP(J+1)=H**J/FLOAT(J)
      ZZZ=ATAN3(Y(I)-W,X(I)-V)
      HP(1)=1.
      P=A/R1
      E=B/R1
      Q5=(HP(7)-E*HP(8)+(E*E-P)*HP(9)+E*(2.*P-E*E)*HP(10)+P*(P-3.*E*E)*H
     1P(11)-3.*E*P*P*HP(12)-(P**3)*HP(13))/R1
      Q4=(HP(6)-E*HP(7)+(E*E-P)*HP(8)+E*(2.*P-E*E)*HP(9)+P*(P-3.*E*E)*HP
     1(10)-3.*E*P*P*HP(11)-(P**3)*HP(12))/R1
      Q3=H**4/(4.*R1)-E*Q4-P*Q5
      Q2=HCO3/R1-E*Q3-P*Q4
      Q1=HSO2/R1-E*Q2-P*Q3
      Q0=H/R1-E*Q1-P*Q2
      GO TO 5
10    IF (R1.LT.HS*1.E-6) GO TO 11
C     IN THIS CASE R3 IS ZERO.
      AL=(AL1/R1+G(I)-2.*AL2/R2)*HIS
      BE=(4.*AL2/R2-3.*AL1/R1-G(I))*HI
      AL1=AL1/R1
      A=R1/HS
      B=(8.*R2-2.*R1)/(H**3)
      ZS=1.
      ZZZ=ATAN3(-XN(I),-YN(I))
      GO TO 12
C     IN THIS CASE R1 IS ZERO.
11    AL=(G(I-1)+AL3/R3-2.*AL2/R2)*HIS
      BE=(4.*AL2/R2-3.*G(I-1)-AL3/R3)*HI
      AL1=G(I-1)
      A=(6.*R2-R3)/HS
      B=2.*(R3-4.*R2)/(H**3)
      EL=ALOG(R3)
      ZS=0.
      ZZZ=ATAN5(Y(I)-W,X(I)-V)
12    Q1=B*H/A
      P=ALOG(A)+2.*ZZ
      R1=1.
      Q2=Q1**2
      Q3=Q2*Q1
      IF (ABS(Q1).LT.5.6E-4) GO TO 14
      E=ALOG(1.+Q1)
      Q(4)=H*(P-2.+((1.+Q1)*E-Q1)/Q1)
      Q(5)=HSO2*(P-(1.+2.*ZS)+((Q2-1.)*E+Q-0.5*Q2)/Q2)
      Q(6)=HCO3*(P-(2./3.+3.*ZS)+((Q3+1.)*E-Q1+.5*Q2-Q3/3.)/Q3)
13    Q(1)=(AL*H**4/4.+BE*HCO3+AL1*HSO2)/R1-ZZZ*H
      Q(2)=(AL*H**5/5.+BE*HSO2**2+AL1*HCO3)/(2*R1)-ZZZ*HSO2
      Q(3)=(AL*H**6/6.+BE*H**5/5.+AL1*H**4/4.)/(3.*R1)-ZZZ*HCO3
      RETURN
14    Q(4)=H*(P-2.+Q1*(.5-Q1/6.+Q2/12-0.05*Q3))
      Q(5)=HSO2*(P-(1.+2.*ZS)+Q1*(.666666666666-0.25*Q1+Q2/7.5-Q3/12.))
      Q(6)=HCO3*(P-(2./3.+3.*ZS)+Q1*(.7-0.3*Q1+Q2/6.+Q3/9.333333333333))
      GO TO 13
C     IN THIS CASE R2 IS ZERO. THE VARIABLE G HAS BEEN COMPUTED JUST
C     PREVIOUS TO THE CALL TO FN. IT IS NOT THE VALUE OF G(I)
```

```
C         CORRESPONDING TO THE POINT (X(I),Y(I)).
15        AL=(AL1/R1+AL3/R3-2.*G)*HIS
          BE=(4.*G-3.*AL1/R1-AL3/R3)*HI
          AL1=AL1/R1
          R1=1.
          ELC=EL
          EL=ALOG(R3)
          Q(4)=H*((ELC+EL)*6-2.)
          Q(5)=HSO2*((ELC-EL-8.)*.25+EL)
          Q(6)=HCO3*((ELC-EL-13.3333333333)*.125+EL)
          ZZZ=ATAN3(Y(I)-W,X(I)-V)
          GO TO 13
          ENTRY QVF
C         THIS ENTRY POINT IS FOR INITIALIZING THE VARIABLES AL3 AND R3.
          AL3=(X(I-1)-V)*XN(I-1)-(Y(I-1)-W)*YN(I-1)
          R3=(X(I-1)-V)**2+(Y(I-1)-W)**2
          IF (R3.GT.0.) EL=ALOG(R3)
          RETURN
          ENTRY QVS
          IF (R3.LT.50.*HS) GO TO 1
C         IF R30.GE.50.*HS,THEN ON THE TWO INTERVALS TO BE CONSIDERED ONE
C         CANNOT GUARANTEE THAT THE DISTANCE REMAINS GREATER THAN 5*H.
C         THIS IS NOT THE BEST POSSIBLE TEST, BUT THE LOGIC NEEDED FOR
C         THE BEST POSSIBLE TEST IS RELATIVELY DIFFICULT.
C         THESE VARIABLES ARE USED IN THE CALCULATION BELOW.
          IF (.NOT.BZ) GO TO 1
          R1=R3
          R12=(X(I)-V)**2+(Y(I)-W)**2
          R2=(X(I)-V)**2+(Y(I)-W)**2
          R13=(X(I+1)-V)**2+(Y(I+1)-W)**2
          R3=(X(I+1)-V)**2+(Y(I+1)-W)**2
          IF (BQ) GO TO 16
C         IN THIS CASE THE CONTRIB. FROM THE PART OF THE BOUNDARY BETWEEN
C         (X(I-1),Y(I-1)) AND (X(I+1),Y(I+1)) IS ADDED TO CD(I-1)
C         AND CN(I-1).
          CN(I-1)=CN(I-1)+ZE*ALOG(R1*((R1*R12**2)**3/R13**2)**2)/2.
          GO TO 17
C         THE TWO INTEGRALS STORED IN CD(I-1) AND CN(I-1) EVALUATED IN ONE
C         STEP. THIS INVOLVES AN INTEGRATION OVER THE PART OF THE BOUNDARY
C         BETWEEN (X(I-3),Y(I-3)) AND (X(I+1),Y(I+1)).
16        CN(I-1)=ZE*ALOG(R1*((R1*R12*RIO2)**3/(R13*RIO3))**2)
C         THE TWO INTEGRALS STORED IN CD(I) AND CN(I) ARE EVAL. IN ONE STEP.
C         THIS ONLY INVOLVES AN INTEGRATION OVER THE PART OF THE BOUNDARY
C         BETWEEN (X(I-1),Y(I-1)) AND (X(I+1),Y(I+1)).
17        CN(I)=ZE*6.*ALOG((R12*R13)**2*R2)
C         THESE FOUR VARIABLES ARE STORED FOR POSSIBLE USE IN THE NEXT CALL TO
C         QV.
          RIO3=R12
          RIO2=R13
          BQ=.T.
C         THE FOLLOWING STATEMNT ASKS IF THE POINT (X(I+1),Y(I+1)) IS AT END
C         OF THE BOUNDARY SECTION OR IF THE POINT IS TOO CLOSE TO (V,W) FOR
C         SIMPSONS RULE INTEGRATION TO BE USED
          IF (R3.GT.50.*HS.AND.I-1.NE.ME1) RETURN
C         IN THIS CASE THE CONTRIBUTNS TO THE INTEGRALS CD(I+1) AND CN(I+1)
C         FROM THE PART OF THE BOUNDARY BETWEEN (X(I-1),Y(I-1)) AND (X(I+1),
C         Y(I+1)) MUST BE ADDED IN NOW.
          EL=ALOG(R3)
          CN(I+1)=ZE*ALOG(R3*((R3*R13**2)**3/R12**2)**2)/2.
          RETURN
          END
```

```
      SUBROUTINE ECSW (A,I,J,L)
      COMMON /MODE/ IN
      COMMON /CONF/ F(630),XO,YO,CD(630),CN(630),KV(210),HD(210),G(630),
     1X(630),XN(630),Y(630),YN(630),XI(630),XIN(630),YI(630),YIN(630),NO
     2C,N
      COMMON /ZLAR/ V,Q(12),H,HS,HSO2,HCO3,HI,HIS,ZZ,BZ,DP,AR,ME1,HSM,FB
     1,BO,ZE,RI,W
C     THIS ENTRY POINT WRITES L WORDS INTO THE  I-TH ROW OF THE MATRIX A
C     STARTING IN COLUMN J.
      KN=5
      M=(I-1)*(N+IN)+J-1+1470
1     CALL ECWR (A,M,L,JJ)
      IF (JJ.EQ.0) RETURN
      KN=KN-1
      IF (KN.GT.0) GO TO 1
      PRINT 3, L,M
      STOP
      ENTRY ECSR
C     THIS ENTRY POINT READS L WORDS FROM THE I-TH ROW OF THE MATRIX A
C     STARTING IN COLUMN J.
      M=(I-1)*(N+IN)+J-1+1470
      KN=5
2     CALL ECRD (A,M,L,JJ)
      IF (JJ.EQ.0) RETURN
      KN=KN-1
      IF (KN.GT.0) GO TO 2
      PRINT 4, L,M
      STOP
      ENTRY ECSR7W
C     THIS ENTRY POINT IS USED TO READ AND WRITE
C     SELDOM USED BOUNDARY DATA.
      M=7*(I-1)
      KN=5
      IF (J.EQ.1) GO TO 1
      GO TO 2
C
3     FORMAT (43H ECS WRITE ERROR FLAG SET. TRYING TO WRITE ,I3,28H WORD
     1S STARTING AT LOCATION ,I7,1H.)
4     FORMAT (41H ECS READ ERROR FLAG SET. TRYING TO READ ,I3,28H WORDS
     1 STARTING AT LOCATION ,I7,1H.)
      END

      SUBROUTINE ECRD (AA,I,J,JJ)
C     THIS ROUTINE CAN BE REMOVED IF E.C.S. IS AVAILABLE
C     LEAVING THIS SUBROUTINE IN CAUSES THE ENTIRE PROBLEM TO
C     BE RUN IN CORE
C     IF THIS SUBROUTINE IS RETAINED THE VECTOR A BELOW MUST BE
C     DIMENSIONED N*N+1500
C     IN TEST DECK 23404 = 148*148 + 1500
C     THIS CORRESPONDS TO THE LARGEST N FOR OUR TEST PROBLEMS
      DIMENSION A(23404)
      DIMENSION AA(2)
      JJ=0
      I1=I+1
      I2=I+J
      M=1
        DO 1 K=I1,I2
        AA(M)=A(K)
1       M=M+1
      RETURN
      ENTRY ECWR
      JJ=0
      M=1
      I1=I+1
      I2=I+J
        DO 2 K=I1,I2
        A(K)=AA(M)
2       M=M+1
      RETURN
      END
```

```
           IDENT  MACHLNG
           ENTRY  PIVOTL
********** ********* THIS MACHINE LANGUAGE ROUTINE  REPLACES THE  L**3
********** ********* LOOP IN SUBROUTINE CONFORM
           VFD    42/6LPIVOTL,18/4
PIVOTL     DATA   0
           SA1    B3
           SA2    B4+08
+          SB6    2
           IX3    X2-X1
+          SB7    X1-1
           SB5    X3-1
+          SA2    B1+B7
           SA3    B2+B7
           NG     B5,PIV4
+          SA5    A2-1
           SA4    A3+1
+          SA1    A2-2
           SA2    A5
           BX0    X5
+          LT     B5,B6,PIV2
PIV1       SA1    A1+B6
           SA2    A2+2
           FX6    X3*X0
+          FX7    X4*X5
           SA3    A3+B6
           SA4    A4+2
+          FX6    X1-X6
           FX7    X2-X7
           NX6    B0,X6
           SB5    B5-B6
+          NX7    B7,X7
           SA6    A1
           SA7    A2
           GE     B5,B6,PIV1
PIV2       SA1    A1+B6
           SA2    A2+B6
+          FX6    X3*X0
           FX7    X4*X5
           FX6    X1-X6
           FX7    X2-X7
+          NX6    B0,X6
           SA6    A1
           NX7    B0,X7
           SA7    A2
           GE     B0,B5,PIVOTL
           SA3    A3+B6
           SA1    A1+B6
           FX6    X3*X0
           FX6    X1-X6
           NX6    B0,X6
           SA6    A1
           EQ     PIVOTL
PIV4       SA5    A2-1
           FX6    X5*X3
           FX6    X2-X6
           NX6    B0,X6
           SA6    A2
           EQ     PIVOTL
           END
```

The editorial committee would welcome readers' comments about this microfiche feature. Please send comments to Professor Eugene Isaacson, MATHEMATICS OF COMPUTATION, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012.

Mathematics of Computation

TABLE OF CONTENTS

APRIL 1972