SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0011 $01.25/0

CONSTRUCTION OF A CURVILINEAR GRID*

BARBRO KREISS†

Abstract. The construction of overlapping grids is explained and applied to a system of hyperbolic differential equations.

Key words. overlapping grids, curvilinear grid, solution of PDE, interpolation.

1. Introduction. We want to solve a hyperbolic system of partial differential equations

$$\frac{\partial u}{\partial t} = A\frac{\partial u}{\partial x} + B\frac{u}{\partial y} + F \tag{1.1}$$

in a domain Ω_A bounded by a smooth curve $\partial\Omega_A$. At $t = 0$ we have the initial conditions

$$u(x, 0) = f(x),$$

and on $\partial\Omega_A$ boundary conditions

$$Lu = g$$

are given.

The crudest method is to cover Ω_A by a rectilinear grid (Fig. 1.1) and replace the differential equations and boundary conditions by difference equations.

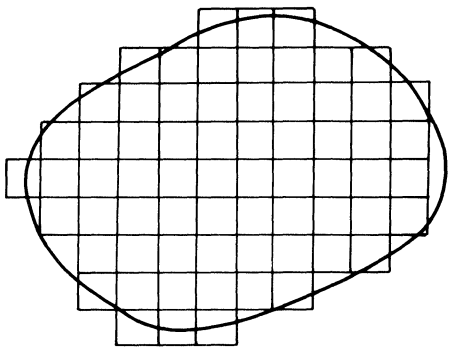

FIG. 1.1

Unfortunately this process is often very inaccurate. We shall instead use two different overlapping grids:

(1) One curvilinear grid, G_C, defined by the transformation

$$T: \begin{cases} x = (r, s), \\ y = y(r, s), \end{cases} \qquad 0 \leqq r, s \leqq 1, \tag{1.2}$$

* Received by the editors November 13, 1981, and in revised form June 25, 1982. This project was sponsored by the Office of Naval Research under contract N0014-80-C-0076, the U.S. Department of Energy under contract EY-76-S-03-070 and the Swedish Natural Science Council (NFR 2711-018) and the Swedish Board for Technical Development (STU 77-3690).

† Department of Computer Sciences, Uppsala University, Uppsala, Sweden. Present address, Department of Applied Mathematics, California Institute of Technology, Pasadena, California 91125.

which follows the boundary. It covers a domain $\Omega_C = \Omega_A - \Omega_B$, where Ω_B is bounded by another smooth curve $\partial\Omega_B$. This grid is constructed with help of splines.

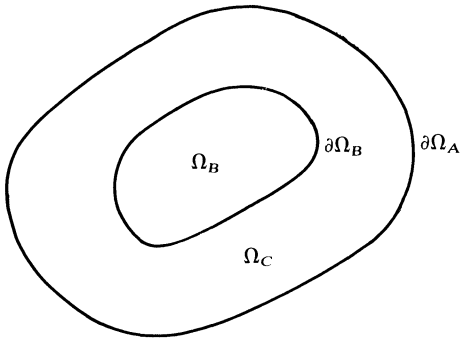

FIG. 1.2

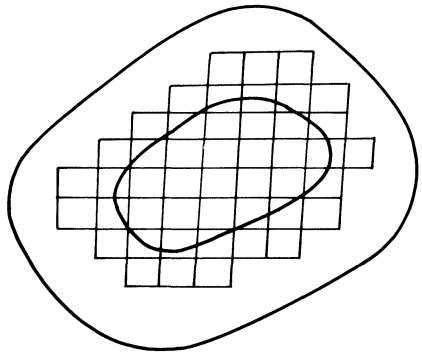

FIG. 1.3

(2) One rectilinear grid, G_B, covering Ω_B and partly overlapping Ω_C.

We replace (1.1) by difference equations. On Ω_B we use a standard leap-frog procedure. In order to obtain the difference equations on the curvilinear grid, we introduce r, s as new independent variables and obtain

$$\frac{\partial u}{\partial t} = \tilde{A}\frac{\partial u}{\partial s} + \tilde{B}\frac{\partial u}{\partial r} + F. \tag{1.3}$$

We replace (1.3) and the boundary conditions by difference equations and again use a standard leap-frog procedure.

The solutions on the two grids are connected by interpolation. The grid construction program generates a mapping function and its derivatives and the weight functions necessary for the interpolation.

Overlapping grids have earlier been used by [4]. However, we believe that our construction is simpler and more flexible. We are already working on extensions of the grid construction that include stretching in the r and s directions and allow other kinds of domains. Our applications include elliptic equations and a combination of hyperbolic and elliptic equations. The use of splines for grid construction has also been used by [5]. There is no overlapping of grids, though, which we think is a very important feature.

2. Grid construction. In practice the boundaries $\partial\Omega_A$ and $\partial\Omega_B$ are only known at J corresponding points

$$P_j, Q_j, j = 1, \cdots, J.$$

Therefore we define $\partial\Omega_A$ and $\partial\Omega_B$ by spline interpolation, as described in [1, p. 42] and [2, p. 50] and obtain a representation $P(s)$, $Q(s)$ of $\partial\Omega_A$, $\partial\Omega_B$ respectively, as functions of a parameter s, $0 \leqq s \leqq 1$. For every fixed s we connect $P(s)$, $Q(s)$ by a smooth curve $D(r, s)$, $0 \leqq r, s \leqq 1$. The simplest curve is a straight line, and this is the only type used so far (Fig. 2.1).

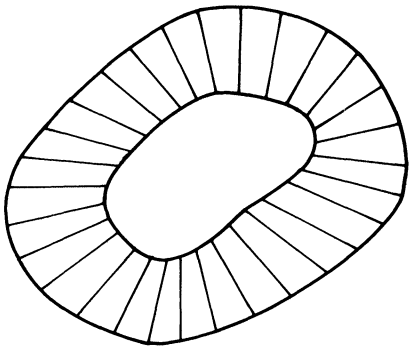

FIG. 2.1

This procedure defines the coordinate transformation (1.2).

We want to cover Ω_C by a curvilinear grid G_C. Let $\Delta s = 1/N$ and $\Delta r = 1/(M-1)$. The gridpoints of G_C are defined by

$$P_{i,\nu}(r, s) = P_{i,\nu}((i-1)\cdot\Delta r, \nu\cdot\Delta s), \qquad i = 1, \cdots, M, \quad \nu = 0, \cdots, N.$$

This curvilinear grid in the x, y-plane corresponds in the r, s-plane to a uniform grid over the rectangle $0 \leqq r, s \leqq 1$ (Fig. 2.2).

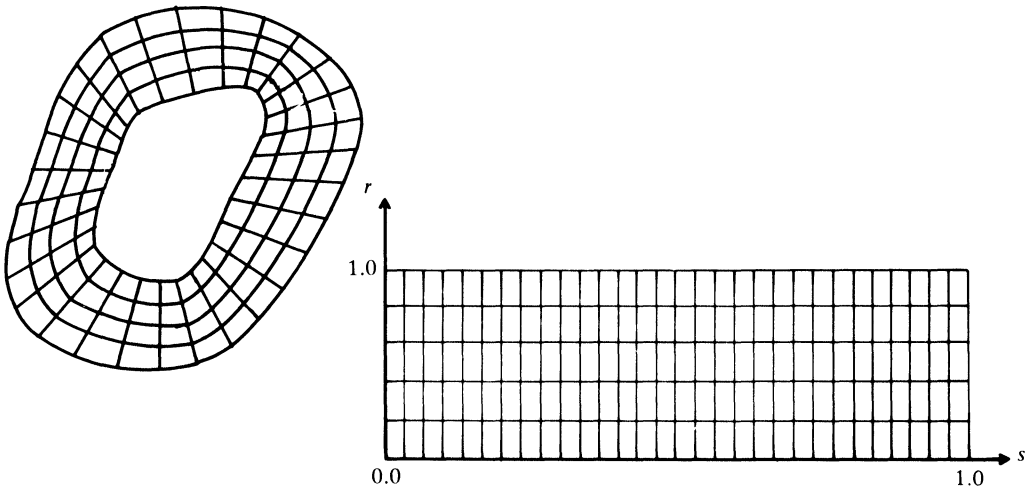

FIG. 2.2

We cover Ω_A by a uniform rectilinear grid. We use a method as given in [3, p. 358] and [4, p. 4.1] to restrict this grid in such a way that it covers Ω_B but is contained in Ω_A. Set $r = r_c$. We then obtain a coordinate curve

$$C(s)\colon x = x(r_c, s),\ y = y(r_c, s).$$

For every point on C we determine the closest point in the rectilinear grid. The points thus obtained are the boundary points R_L of the restricted grid. It is important that R_L forms a closed polygon. If two consecutive points on C give boundary points whose indices indicate a gap >1, then we define the intermediary boundary points as those which approximate the chord between the two points on C (Fig. 2.3).

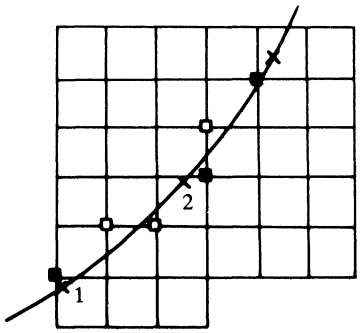

FIG. 2.3. $\times$ *Points on* C; ■ *boundary points from points* 1, 2, 3; □ *additional boundary points.*

This process needs a starting point that is known to be in the interior of Ω_B. Finally R_L is cleaned of corner points, which are not needed during the computation.

An integer image array is used to flag the points of the rectilinear grid. For the boundary points R_L the flag is 2. In these points we will interpolate. For interior points the flag is 1. There we will apply the leap-frog scheme. For outside points the flag is 0. The grid G_B consists of gridpoints, where the flag $\neq 0$.

3. Numerical solution of hyperbolic differential equations. Consider a symmetric hyperbolic system

$$\frac{\partial u}{\partial t} = A\frac{\partial u}{\partial x} + B\frac{\partial u}{\partial y} \tag{3.1}$$

for $(x, y) \in \Omega_A$, $t \geqq 0$. Here $u = (u^{(1)}, \cdots, u^{(n)})^T$ is a vector function and $A = A^*, B = B^*$ are $n \times n$ symmetric matrices which depend smoothly on x, y, t. At $t = 0$ initial conditions

$$u(x, 0) = f(x), \tag{3.2}$$

and on $\partial\Omega_A$ a number of linear relations between the components of u are given as boundary conditions

$$Lu = g, \qquad (x, y) \in \partial\Omega_A. \tag{3.3}$$

As we have described earlier, we cover Ω_A by two overlapping grids G_B and G_C. G_B is rectilinear in the x, y-plane and G_C in the r, s-plane, defined by the transformation (1.2). At all interior points of G_B we approximate the differential equation by the leap-frog scheme, i.e.,

$$u(x, y, t+\Delta t) = u(x, y, t-\Delta t) + 2\Delta t(A(x, y, t)D_{0x} + B(x, y, t)D_{0y})u(x, y, t).$$

Here

$$D_{ox}u(x, y, t) = (u(x+\Delta x, y, t) - u(x-\Delta x, y, t))/2\Delta x,$$

$$D_{0y}u(x, y, t) = (u(x, y+\Delta y, t) - u(x, y-\Delta y, t))/2\Delta y$$

denote the centered difference operators.

In order to obtain the difference approximation in G_C we use the transformation (1.2). We have

$$\frac{\partial u}{\partial x}=\left(\frac{\partial u}{\partial r}\cdot\frac{\partial y}{\partial s}-\frac{\partial u}{\partial s}\cdot\frac{\partial y}{\partial r}\right)\Big/\Delta, \qquad \Delta=\frac{\partial x}{\partial r}\cdot\frac{\partial y}{\partial s}-\frac{\partial x}{\partial s}\cdot\frac{\partial y}{\partial r},$$

$$\frac{\partial u}{\partial y}=\left(\frac{\partial u}{\partial s}\cdot\frac{\partial x}{\partial r}-\frac{\partial u}{\partial r}\cdot\frac{\partial x}{\partial s}\right)\Big/\Delta.$$

Therefore (3.1) becomes

$$\frac{\partial u}{\partial t}=\tilde{A}\frac{\partial u}{\partial r}+\tilde{B}\frac{\partial u}{\partial s}, \tag{3.4}$$

where

$$\tilde{A}=\frac{1}{\Delta}\left(A\frac{\partial y}{\partial s}-B\frac{\partial x}{\partial s}\right), \qquad \tilde{B}=\frac{1}{\Delta}\left(-A\frac{\partial y}{\partial r}+B\frac{\partial x}{\partial r}\right).$$

Here $u(r, s)$ is periodic in s. $r=0$ corresponds to $\partial\Omega_A$, and $r=1$ corresponds to $\partial\Omega_B$.

We approximate (3.4) for $0<r=j\cdot\Delta r<1$ and $s=0, \Delta s, 2\Delta s, \cdots, (N-1)\cdot\Delta s$ again by the leap-frog scheme

$$u(r, s, t+\Delta t)=u(r, s, t-\Delta t)+2\Delta t(\tilde{A}(r, s, t)D_{0s}+\tilde{B}(r, s, t)D_{0r})u(r, s, t). \tag{3.5}$$

For $r=0$ we use a one-sided formula. If

$$\tilde{A}(0, s)=\begin{pmatrix}-\Lambda_1 & 0\\ 0 & \Lambda_2\end{pmatrix}, \qquad \Lambda_j>0 \text{ diagonal}, \quad j=1, 2,$$

is diagonal, then the boundary formula becomes particularly simple. The positive eigenvalues correspond to the outgoing characteristics. Therefore we extrapolate these variables

$$\begin{aligned}u^{\mathrm{II}}(0, s, t+\Delta t)=u^{\mathrm{II}}(0, s, t)&+\frac{\sigma(\Delta s)^2}{2}D_{+s}D_{-s}u^{\mathrm{II}}(0, s, t)\\ &+\Delta t\,(\Lambda_2 D_{+r}u^{\mathrm{II}}(0, s, t)+(\tilde{B}D_{0s}u(0, s, t))^{\mathrm{II}}).\end{aligned} \tag{3.6}$$

Here σ is a parameter and

$$D_{+r}u^{\mathrm{II}}(0, s, t)=(u^{\mathrm{II}}(\Delta r, s, t)-u^{\mathrm{II}}(0, s, t))/\Delta r,$$

$$D_{+s}D_{-s}u^{\mathrm{II}}(0, s, t)=(u^{\mathrm{II}}(0, s+\Delta s, t)-2u^{\mathrm{II}}(0, s, t)+u^{\mathrm{II}}(0, s-\Delta s, t))/(\Delta s)^2.$$

For the other variables we use the boundary conditions.

If $\tilde{A}$ is not diagonal, then we proceed in the following way: At every boundary point $r=0$, $s=s_0$ we construct a unitary transformation U such that

$$U^*\tilde{A}U=\Lambda.$$

Then we introduce a new variable $\tilde{u}$ by $u=U\tilde{u}$ and transform (3.4). For the transformed equations we can use the approximation(3.6). Changing back to the original variables we obtain the boundary approximation.

Assume now that u is known at times t and $t-\Delta t$. Then we can use the above approximations to determine u at time $t+\Delta t$ in all gridpoints along $\partial\Omega_A$ and in all interior gridpoints of both G_C and G_B. The values of u at the gridpoints on the boundaries $\partial\Omega_B$ and R_L are obtained by interpolation, which will be explained in the next section.

4. Interpolation. Let $u(x, y)$ be a smooth function. Let P_i, $i = 1, \cdots, 9$ be nine points uniformly spaced in a rectilinear grid, and let $u(P_i)$ denote the function values in these points.

We want to determine an approximate value $u(Q)$ at a point Q inside the rectangle formed by P_1, P_3, P_7, P_9. See Fig. 4.1.

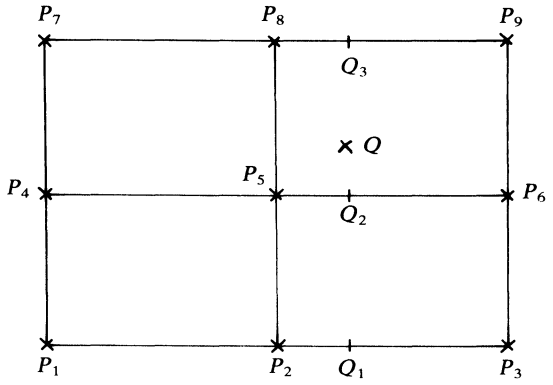

FIG. 4.1

Consider the 3-point formula

$$u(Q_I) = a_1(\alpha)u(P_{i-1}) + a_2(\alpha)u(P_i) + a_3(\alpha)u(P_{i+1}), \qquad 0 \leqq |\alpha| \leqq 1, \quad P_{i-1} \leqq Q_I \leqq P_{i+1} \tag{4.1}$$

with

$$a_1(\alpha) = \frac{\alpha(\alpha-1)}{2}, \quad a_2(\alpha) = \frac{(\alpha+1)(\alpha-1)}{-1}, \quad a_3(\alpha) = \frac{(\alpha+1)\alpha}{2}.$$

We use (4.1) with $\alpha = (x_Q - x_{P_5})/h_x$. First let $i = 2$ and we obtain

$$u(Q_1) = a_1(\alpha)u(P_1) + a_2(\alpha)u(P_2) + a_3(\alpha)u(P_3).$$

We obtain $u(Q_2)$ and $u(Q_3)$ when we let $i = 5, 8$, respectively. We replace α in (4.1) by $\beta = (y_Q - y_{P_5})/h_y$, P by Q and let $i = 2$ and obtain

$$u(Q) = a_1(\beta)u(Q_1) + a_2(\beta)u(Q_2) + a_3(\beta)u(Q_3)$$

or

$$u(Q) = a_1(\beta) \sum_{i=1}^{3} a_i(\alpha)u(P_i) + a_2(\beta) \sum_{i=1}^{3} a_i(\alpha)u(P_{i+3}) + a_3(\beta) \sum_{i=1}^{3} a_i(\alpha)u(P_{i+6}). \tag{4.2}$$

For every gridpoint Q on $\partial\Omega_B$ we find the gridpoint in G_B that is closest to Q. We denote this point P_5 and apply (4.2) using the function values of the nine gridpoints in G_B with P_5 in the center to obtain an approximate function value in Q.

The nine weight coefficients in (4.2) and the location of P_5 are computed once and stored to be available at computation time.

In our computation the grid G_C had to be considerably denser than G_B. Therefore we have found it to be satisfactory to use a four point formula when we interpolate among gridpoints in G_C to find an approximate value at the boundary points R_L of G_B.

Let $P \in R_L$. Denote by P_1 the gridpoint in G_C that has the shortest distance to P. We call three of $P_1 : s$ neighbor points in G_C P_2, P_3, P_4. See Fig. 4.2. Denote the vector from P_1 to P_2 by a, the vector from P_1 to P_4 by c and the vector from P_4 to P_3 by b.

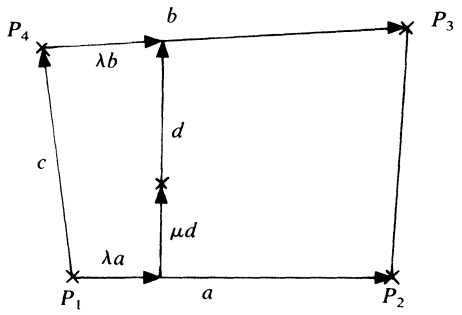

FIG. 4.2

The vector d through the point P is defined by

$$d = \lambda b + c - \lambda a, \qquad 0 \leqq \lambda \leqq 1. \tag{4.3}$$

We also have

$$P - P_1 = \lambda a + \mu d, \qquad 0 \leqq \mu \leqq 1. \tag{4.4}$$

Equations (4.3) and (4.4) give us

$$P - P_1 = \lambda a + \mu c + \lambda\mu (b - a). \tag{4.5}$$

We use an iterative process to solve (4.5) to obtain λ, μ. We have assumed that $\lambda, \mu \geqq 0$, i.e., P lies in "quadrant 1" of P_1. If the solution of (4.5) turns out negative values for λ or μ or both, then we try another "quadrant" of P_1, i.e., P_2, P_3, P_4 could be another set of points around P_1. Figure 4.3 shows the different "quadrants" and the arrows indicate the ascending order of the points P_2, P_3, P_4.

When we have reached satisfactory values for λ and μ, we proceed using the following standard interpolation formula, in order to obtain an approximate function

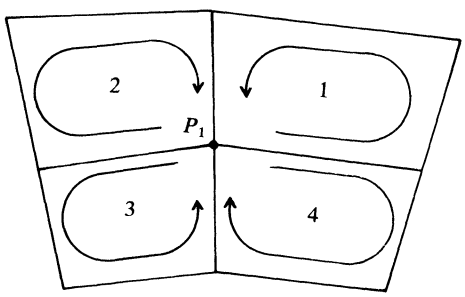

FIG. 4.3

value $u(P)$ from the function values $u(P_i)$, $i = 1, \cdots, 4$,

$$u(P) = (1-\lambda)(1-\mu)u(P_1) + \lambda(1-\mu)u(P2) + \lambda\mu \cdot u(P_3) + (1-\lambda)\mu \cdot u(P_4). \tag{4.6}$$

The four weight coefficients in (4.6) are computed once and stored to be available at computation time. Also available are the locations of P in G_B and of P_1 in G_C as well as the correct "quadrant" number. This is done for every point $P \in R_L$.

5. An example. As a test case we have solved (1.1) with

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad F = (\cos(x+2t), -3\sin(x+2t))^T,$$
$$u(x,0) = (\sin x, \cos x)^T, \qquad u(x, \Delta t) = (\sin(x+2\Delta t), \cos(x+2\Delta t))^T. \tag{5.1}$$

For the boundary $r = 0$ we have

$$a_1 u^{(1)} + a_2 u^{(2)} = g.$$

Remark. The coefficients a_1 and a_2 are chosen in such a way that we give the variable corresponding to the ingoing characteristic. The solution to this problem is

$$v = (\sin(x+2t), \cos(x+2t))^T.$$

Therefore we have been able to compute the error function

$$e = v - u.$$

Table 5.1 shows the maximum error, $e_{\max}$, and the global error for different grid densities. We used $\sigma = 0$ and $\Delta t = 0.001$. We constructed the grids from two sets of 7 points. They are listed in Table 5.2. In the iterative processes of the grid construction we used a tolerance $\varepsilon = 10^{-6}$. The grid construction program was run on a VAX 11/780 in time-sharing mode, the solution of (5.1) was done on an IBM 370/3032 at the California Institute of Technology, Pasadena.

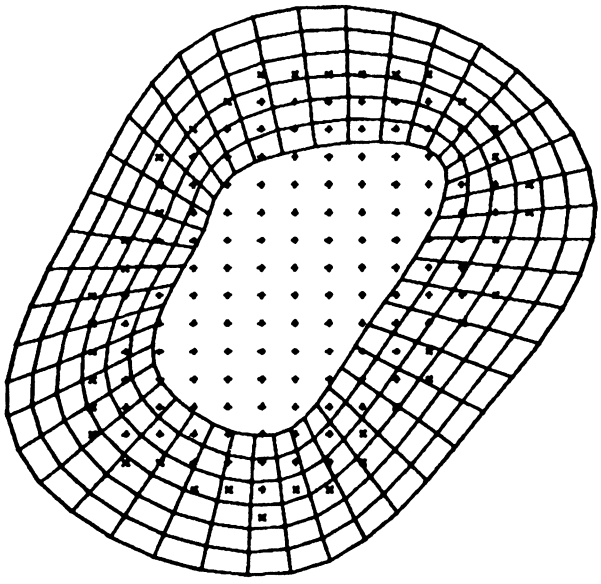

FIG. 5.1

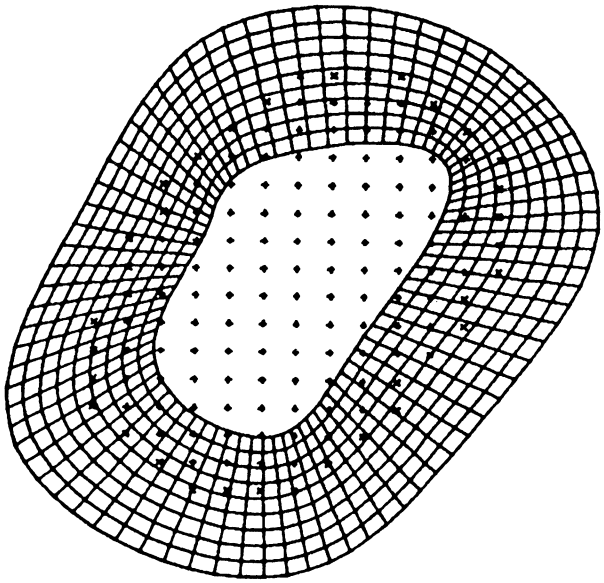

FIG. 5.2

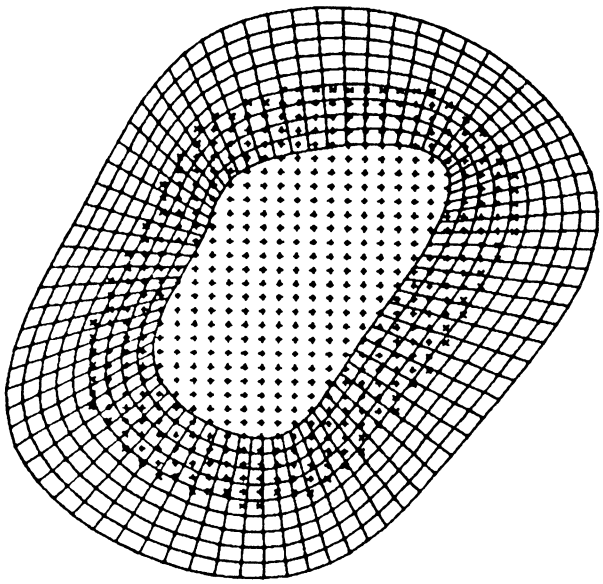

FIG. 5.3

TABLE 5.1

		G_C		G_B		G_C		G_B		
	time	N	M	h_x	h_y	$e_{\max}u^{(1)}$	$e_{\max}u^{(2)}$	$e_{\max}u^{(1)}$	$e_{\max}u^{(2)}$	global e
I	0.2	43	7	0.05	0.05	0.00289	0.00319	0.00228	0.00231	0.00118
	0.4					0.00280	0.00263	0.00269	0.00304	0.00147
	0.6					0.00224	0.00276	0.00497	0.00437	0.00130
	0.8					0.00195	0.00316	0.00689	0.00461	0.00114
	1.0					0.00252	0.00316	0.00712	0.00608	0.00159
II	0.2	78	10	0.05	0.05	0.00118	0.00158	0.00080	0.00127	0.00124
	0.4					0.00130	0.00180	0.00109	0.00183	0.00156
	0.6					0.00151	0.00144	0.00164	0.00184	0.00116
	0.8					0.00136	0.00114	0.00169	0.00183	0.00101
	1.0					0.00150	0.00181	0.00198	0.00336	0.00138
III	0.2	78	10	0.025	0.025	0.00100	0.00151	0.00082	0.00130	0.00079
	0.4					0.00130	0.00172	0.00125	0.00185	0.00098
	0.6					0.00154	0.00154	0.00160	0.00214	0.00067
	0.8					0.00139	0.00116	0.00183	0.00146	0.00064
	1.0					0.00153	0.00164	0.00208	0.00181	0.00076

The grids I, II, III (see Table 5.1) are plotted in Fig. 5.1, 5.2, 5.3 respectively.

TABLE 5.2.

$\partial\Omega_A$		$\partial\Omega_B$	
x	y	x	y
0.49195	0.00291	0.46714	0.24867
0.77552	0.28246	0.58529	0.40381
0.94802	0.66032	0.71172	0.63728
0.68927	0.99824	0.65264	0.77706
0.36907	0.96906	0.43406	0.73712
0.19184	0.67414	0.36198	0.61270
0.09495	0.22102	0.29229	0.37616

REFERENCES

[1] GÖRAN STARIUS: *Composite mesh difference methods for elliptic boundary value problems*, Numer. Math., 28 (1977), pp. 243–258.

[2] J. H. AHLBERG, E. N. NILSSON AND J. L. WALSCH: *The Theory of Splines and Their Applications*, Academic Press, London–New York, 1967.

[3] G. E. FORSYTHE AND W. R. WASOW: *Finite Difference Methods for Partial Differential Equations*, John Wiley, New York–London, 1960.

[4] GÖRAN STARIUS: *Numerical treatment of boundary layers for perturbed hyperbolic equations.* Depart. of Computer Science Report No. 69, Uppsala Univ., Sweden, 1978.

[5] PETER R. EISEMAN: *Geometric methods in computational fluid dynamics*, Rep. 80-11 Institute for Computational Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, April 18, 1980.