Computers & Fluids Vol. 16, No. 1, pp. 47–58, 1988
Printed in Great Britain.

0045-7930/88 $3.00 + 0.00

A NUMERICAL GRID GENERATION TECHNIQUE

B. H. GILDING†
Faculty of Applied Mathematics, University of Twente, Enschede, The Netherlands

(*Received* 25 *February* 1987)

Abstract—The paper describes a technique for the generation of boundary-fitted curvilinear coordinate systems for the numerical solution of partial differential equations in two space dimensions. The technique is algebraic, has a transfinite character, and is based on the blending of shearing transformations. Applications to numerical grid generation for problems in the field of computational fluid dynamics are presented.

1. INTRODUCTION

The considerable progress in the numerical solution of partial differential equations on regions with arbitrarily-shaped boundaries in recent years has been greatly abetted by the development of techniques for numerical grid generation. This is especially true in the field of computational fluid dynamics, but also applies to other areas, such as heat transfer, mass transport, electromagnetism, solid mechanics, and structures [1–4].

Basically, techniques for numerical grid generation can be divided into two classes, viz. techniques which themselves involve solving partial differential equations, and algebraic interpolation techniques. In general, the preference for a technique involving the numerical solution of a partial differential equation above an algebraic technique is that it generates a smoother grid in which the propagation of boundary slope discontinuities is alleviated. On the other hand, algebraic techniques provide explicit grid control and, in comparison to the other class of techniques, require relatively few computations. Subsequently, in implementation, they have the advantage of speed and simplicity. This also makes them particularly attractive for use in conjunction with interactive computer graphics and for the generation of grids for simulations involving time-varying problem domains [2–6].

In this paper, a new algebraic technique for the generation of boundary-fitted curvilinear coordinate systems for numerically solving partial differential equations in two space dimensions is presented. The technique may be classified as transfinite in the sense that it involves interpolation between the trajectories of the boundary curves defining a problem domain. In other words, as opposed to a finite number of points, the coordinate system is generated by a non-denumerable number of boundary points. Notwithstanding, the technique does not conform to the standard concept of transfinite interpolation—it cannot be formulated in terms of the Boolean sum of a sequence of univariate projections [2–5, 7]. The technique has some affinity with the linear two-boundary technique with side constraints described in Ref. [5] and referred to in Ref. [3]. However, the technique may best be regarded as a kind of blended shearing transformation. Indeed, it arose out of the desire to extend a shearing transformation in which the coordinates of the computational points in a grid were obtained simply by normalizing a given boundary distribution between two other adjoining boundaries, to situations in which a blended normalization would take place from the first boundary to a fourth boundary completing the demarcation of the problem domain.

The technique described in this paper was actually developed in response to a desire to generate a computational grid such as that shown in Fig. 1. This mesh stems from an application of a computer program for the solution of the Navier–Stokes equations in two space dimensions using a finite difference technique based on curvilinear coordinates. Details of the computer program together with further applications can be found in Ref.

†Former affiliation: Delft Hydraulics Laboratory, Delft, The Netherlands.

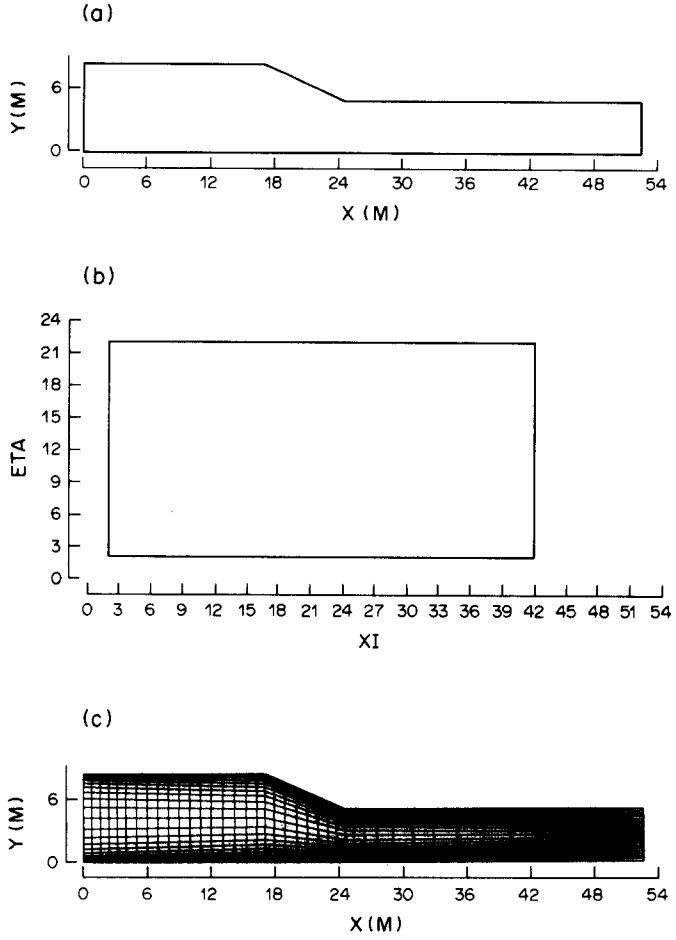

Fig. 1. Grid for computation of flow under a barge in a lock (a) physical boundary (b) computational domain (c) mesh.

[8]. This particular application was the computation of the flow under a barge moored in a lock, during the filling of the lock, with a view to calculating the forces exerted on the mooring ropes. The mesh represents a portion of a vertical profile lengthwise through the lock with inflow on the left and with the indentation in the upper boundary corresponding to the bow of the barge. The intersections of the lines in Fig. 1(c) indicate the location of computational points for the finite difference solution of the Navier–Stokes equations. This nodal distribution was motivated by a consideration of the expected flow pattern and the corresponding desired numerical accuracy, and of simplicity.

The salient feature of the numerical grid generation technique illustrated in Fig. 1 is the blending of the normalization between the upper and lower boundaries of the distributions of points on the left- and right-hand boundaries of the problem domain. A similar blending of the point distributions on the upper and lower boundaries also occurs. However, due to the choice of an identical horizontal spacing in this example this blending is not manifest.

The foundations of the technique will be described in the next section. This will be cast in the form of a transformation of an arbitrarily-shaped simply-connected problem domain into a rectangular computational domain. In the subsequent section, the extension of the theory to problem domains containing "obstacles" and "islands" and in which the corresponding computational domain consists of a union of a number of rectangles will be presented. The extension enables a grid to be generated in an arbitrary block-structured multiply-connected physical domain without requiring the specification of a domain decomposition by the user.

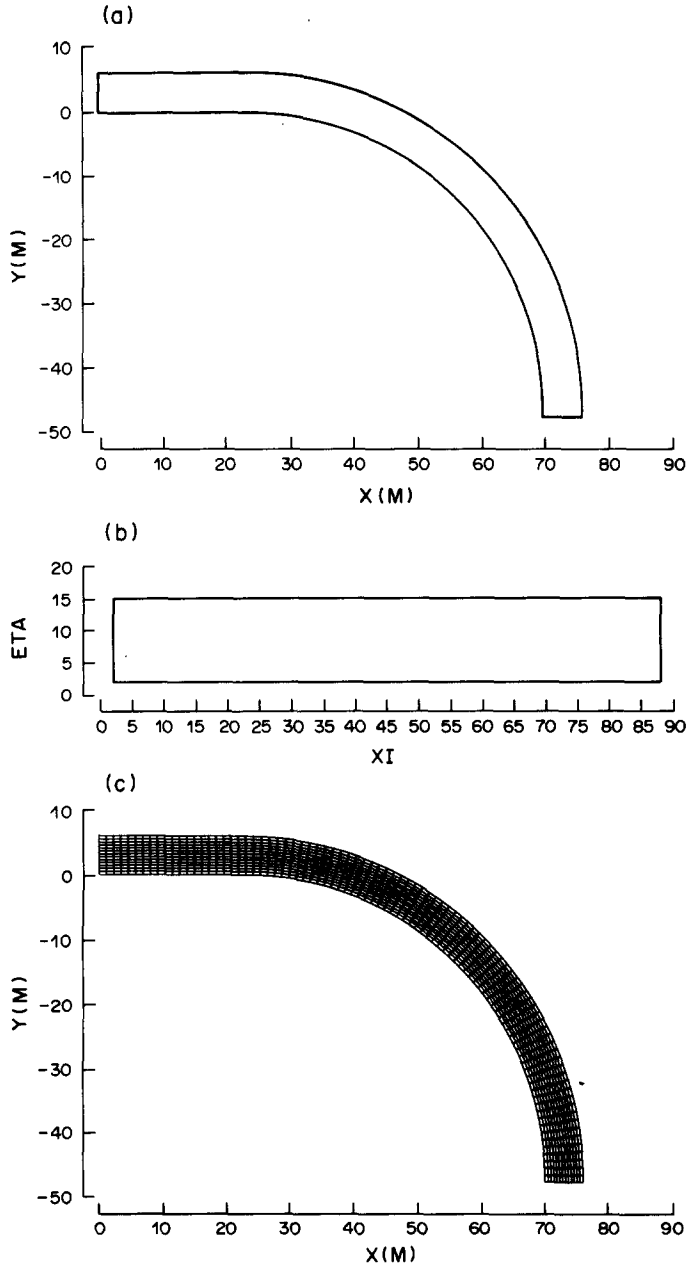

Fig. 2. Grid for computation of flow in river bend (a) physical boundary (b) computational domain (c) mesh.

In common with other algebraic methods, the numerical grid generation technique can be enhanced by the employment of algebraic stretching functions. However, in the interest of brevity, this aspect will not be dwelt upon.

Further to Fig. 1, the practical application of the numerical grid generation technique is illustrated in Figs 2–6. In all the illustrations, no supplementary domain decomposition, patching, stretching, or smoothing, which might be advantageously employed to improve the appearance of the meshes, has been effectuated.

Figure 2 depicts a mesh for the computation of the depth-averaged flow in a river bend. The objective of the computation was to obtain prototype data for the analysis of models simulating the total cross-sectional flow around the river bend as a one-dimensional phenomenon. Figures 3–5 represent applications of the numerical grid generation procedure to problems involving the numerical solution of the Navier–Stokes equations in two

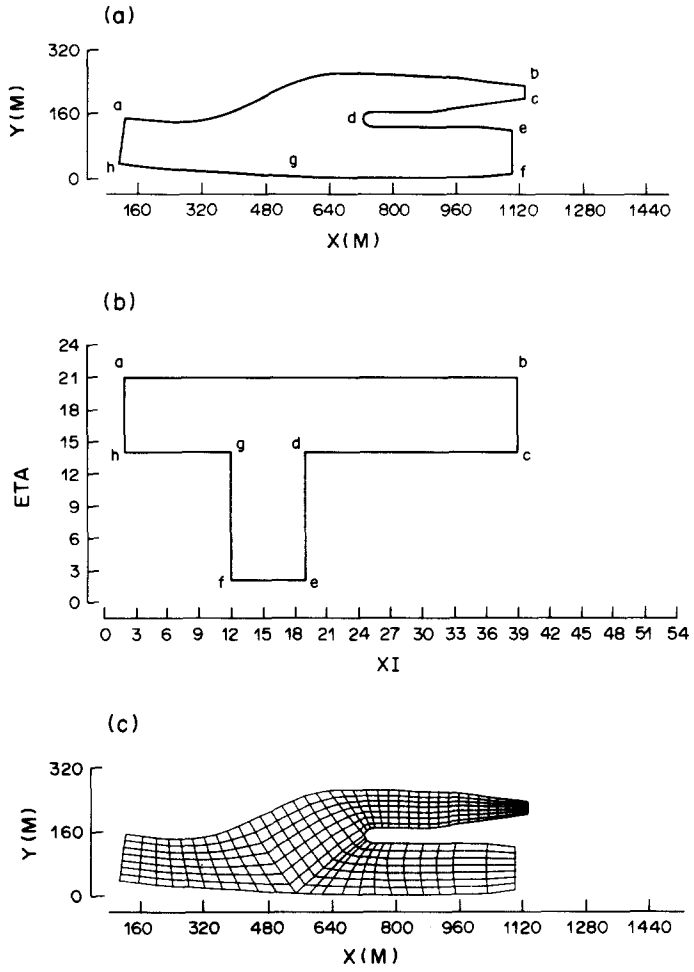

Fig. 3. Grid for computation of flow in junction of two waterways (a) physical boundary (b) computational domain (c) mesh.

space dimensions discussed in Ref. [8]. The configuration shown in Fig. 3, for instance, stems from the simulation of the change in flow pattern in a canal which would result from the construction of a hydropower station with an outlet on the canal. The tapered arm of the mesh corresponds to the projected outlet of the hydropower station, whilst the other arms represent the existing course of the canal. Figure 4 derives from a model of the coastal flow around the Amsterdam harbour entrance near Ijmuiden in North Holland. The interest here was in the erosion and sedimentation in the area. With the grid representing a total area of 13 × 7 km, the area protected by the harbour breakwaters was simulated as a single coastal obstacle. The last figure in this group, Fig. 5, results from a study into the scour around a pipeline on the sea bottom. In this case, the objective was to reproduce a laboratory experiment in which there was a current from left to right in the picture, and as a result of scouring of bottom material, the pipeline became suspended above the sea bed. The final illustration, Fig. 6, constitutes the most involved network of the examples presented. This network was generated for modelling the inshore flow pattern to be expected from the construction of breakwaters at Sabratah in Libya. The breakwaters had been designed to protect the harbour against heavy seas; and the goal of the modelling was to ascertain their effectiveness in also mitigating the fouling hazard to shipping caused by pestilent sea-weed growth in the area. A novel feature of this mesh is that it was so designed that it could easily function for the simulation of the present situation without the harbour breakwaters by completion of the blank portions. This facilitated easy comparison of the flow pattern in the present and projected situations, cf. Ref. [9].

A further application of the numerical grid generation technique involving patching, for the computation of tide- and wave-induced currents in a coastal zone, is to be found in Ref. [9].

Practical experience has borne out that the numerical grid generation technique described in this paper compares favourably with techniques involving the numerical solution of a partial differential equation. Compared with these more sophisticated techniques, two principal advantages have been identified. Firstly, the technique is computationally more economical. Secondly, the structure of the generated mesh is more easily controlled, particularly by the specification of computational points on the boundary or in the interior of the problem domain [9]. The technique provides an alternative to other algebraic grid generation techniques.

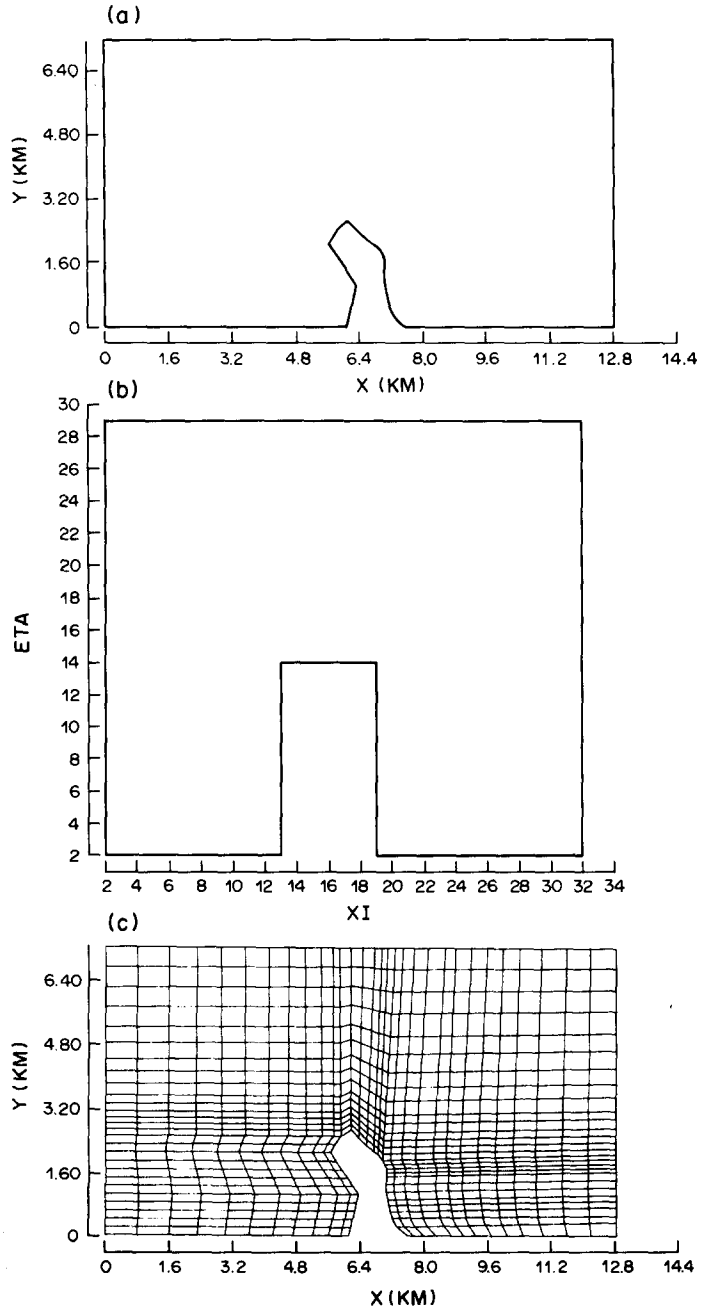

Fig. 4. Grid for computation of coastal flow around harbour area protected by breakwaters (a) physical boundary (b) computational domain (c) mesh.

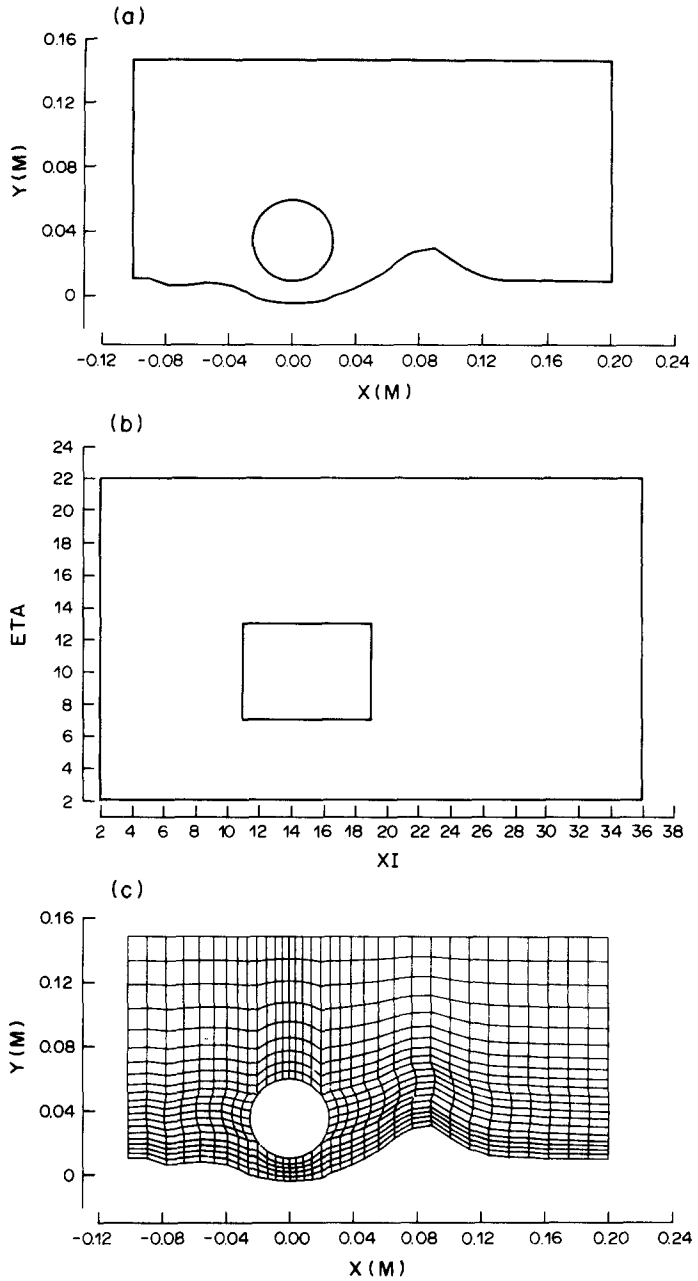

Fig. 5. Grid for computation of flow around pipe on sea bed (a) physical boundary (b) computational domain (c) mesh.

2. THEORETICAL FOUNDATION

Consider a problem domain D in a physical plane defined in terms of a Cartesian coordinate system (x, y). Suppose that we can specify a homeomorphism of the boundary of D onto the boundary of a canonical region R defined in terms of a coordinate system (ξ, η). Then the objective of numerical grid generation as considered here is to construct an extension of the boundary transformation to a homeomorphism between the closure of D and the closure of R. The coordinates (ξ, η) then constitute a boundary-fitted curvilinear coordinate system and define a computational plane. Simultaneously, the specification of a uniform grid on R in the computational plane generates a numerical grid in the physical problem domain.

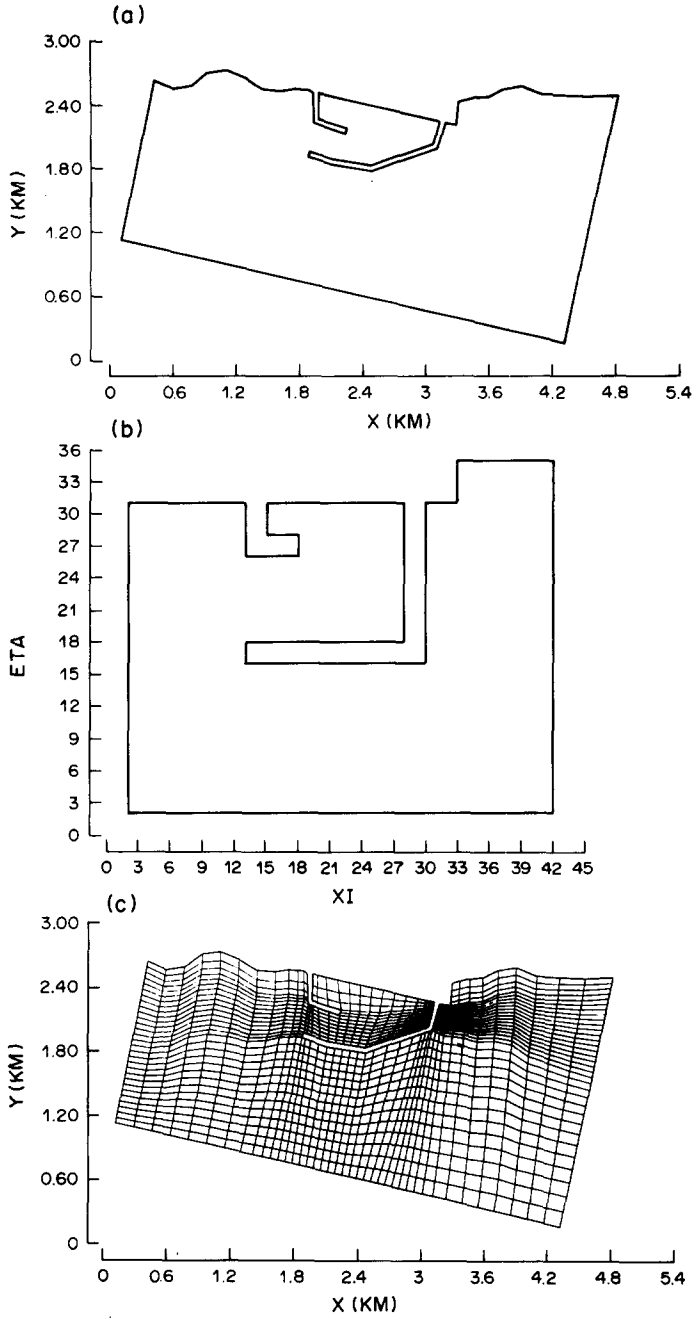

Fig. 6. Grid for computation of inshore flow around harbour breakwaters (a) physical boundary (b) computational domain (c) mesh.

Throughout this and the following section we shall suppose that D and R are non-empty open bounded two-dimensional spatial domains; and denote by ∂D the boundary of D and by $\bar{D}$ the closure of D, etc.

In this section we shall suppose that the domain D is simply-connected and consider a canonical rectangular region $R = \{(\xi, \eta): \xi_1 < \xi < \xi_2, \eta_1 < \eta < \eta_2\}$. We suppose furthermore that a homeomorphism between the boundary of D and the boundary of R is defined. Thus we can write $\partial D = \{(x(\xi, \eta_1), y(\xi, \eta_1)): \xi_1 \leqslant \xi < \xi_2\} \cup \{(x(\xi_2, \eta), y(\xi_2, \eta)): \eta_1 \leqslant \eta < \eta_2\} \cup \{(x(\xi, \eta_2), y(\xi, \eta_2)): \xi_2 \geqslant \xi > \xi_1\} \cup \{(x(\xi_1, \eta), y(\xi_1, \eta)): \eta_2 \geqslant \eta > \eta_1\}$. The objective is now to construct an extension of this mapping such that we can express $\bar{D} = \{(x(\xi, \eta), y(\xi, \eta)): (\xi, \eta) \in \bar{R}\}$.

For each point $(\xi, \eta) \in \bar{R}$ let $\alpha(\xi, \eta)$ denote the normalized projection of the point $(x(\xi, \eta), y(\xi, \eta))$ onto the line connecting the points $(x(\xi_1, \eta), y(\xi_1, \eta))$ and $(x(\xi_2, \eta), y(\xi_2, \eta))$ in the physical plane. The equation of the line joining the two last-mentioned points is:

$$\delta_x(\eta)\{y - y(\xi_1, \eta)\} = \delta_y(\eta)\{x - x(\xi_1, \eta)\},$$

where

$$\delta_x(\eta) = x(\xi_2, \eta) - x(\xi_1, \eta) \tag{1}$$

and

$$\delta_y(\eta) = y(\xi_2, \eta) - y(\xi_1, \eta). \tag{2}$$

Hence, the normal to this line, projecting the point $(x(\xi, \eta), y(\xi, \eta))$ onto it, is:

$$\delta_y(\eta)\{y - y(\xi, \eta)\} = -\delta_x(\eta)\{x - x(\xi, \eta)\}.$$

This gives the point (x_α, y_α) of the projection on the line as:

$$x_\alpha(\xi, \eta) = x(\xi_1, \eta) + \alpha(\xi, \eta)\,\delta_x(\eta) \tag{3}$$

$$y_\alpha(\xi, \eta) = y(\xi_1, \eta) + \alpha(\xi, \eta)\,\delta_y(\eta) \tag{4}$$

when

$$\alpha(\xi, \eta) = [\{x(\xi, \eta) - x(\xi_1, \eta)\}\delta_x(\eta) + \{y(\xi, \eta) - y(\xi_1, \eta)\}\delta_y(\eta)]/[\delta_x(\eta)^2 + \delta_y(\eta)^2]. \tag{5}$$

Similarly, one can define the normalized projection of the point $(x(\xi, \eta), y(\xi, \eta))$ onto the line with end-points $(x(\xi, \eta_1), y(\xi, \eta_1))$ and $(x(\xi, \eta_2), y(\xi, \eta_2))$ as:

$$x_\beta(\xi, \eta) = x(\xi, \eta_1) + \beta(\xi, \eta)\,\epsilon_x(\xi) \tag{6}$$

$$y_\beta(\xi, \eta) = y(\xi, \eta_1) + \beta(\xi, \eta)\,\epsilon_y(\xi), \tag{7}$$

where

$$\beta(\xi, \eta) = [\{x(\xi, \eta) - x(\xi, \eta_1)\}\epsilon_x(\xi) + \{y(\xi, \eta) - y(\xi, \eta_1)\}\epsilon_y(\xi)]/[\epsilon_x(\xi)^2 + \epsilon_y(\xi)^2], \tag{8}$$

$$\epsilon_x(\xi) = x(\xi, \eta_2) - x(\xi, \eta_1) \tag{9}$$

and

$$\epsilon_y(\xi) = y(\xi, \eta_2) - y(\xi, \eta_1). \tag{10}$$

Set

$$\alpha_i(\xi) = \alpha(\xi, \eta_i) \quad \text{for } i = 1, 2,$$

and

$$\beta_i(\eta) = \beta(\xi_i, \eta) \quad \text{for } i = 1, 2.$$

Observe that for $i = 1, 2$; $\alpha_i(\xi)$ is defined and continuous for $\xi_1 \leqslant \xi \leqslant \xi_2$ with $\alpha_i(\xi_1) = 0$ and $\alpha_i(\xi_2) = 1$. Similarly for $i = 1,2$; $\beta_i(\eta)$ is defined and continuous for $\eta_1 \leqslant \eta \leqslant \eta_2$ with $\beta_i(\eta_1) = 0$ and $\beta_i(\eta_2) = 1$.

The essence of the present technique is that we now blend the normalized projections by prescribing:

$$\alpha(\xi, \eta) = \{1 - \beta(\xi, \eta)\}\alpha_1(\xi) + \beta(\xi, \eta)\alpha_2(\xi),$$

and

$$\beta(\xi, \eta) = \{1 - \alpha(\xi, \eta)\}\beta_1(\eta) + \alpha(\xi, \eta)\beta_2(\eta),$$

for all $(\xi, \eta) \in R$. Elimination of unknowns in these two equations yields:

$$\alpha(\xi, \eta) = [\{1 - \beta_1(\eta)\}\alpha_1(\xi) + \beta_1(\eta)\alpha_2(\xi)]/d(\xi, \eta) \tag{11}$$

and

$$\beta(\xi,\eta)=[\{1-\alpha_1(\xi)\}\beta_1(\eta)+\alpha_1(\xi)\beta_2(\eta)]/d(\xi,\eta), \tag{12}$$

where

$$d(\xi,\eta)=1-\{\alpha_2(\xi)-\alpha_1(\xi)\}\{\beta_2(\eta)-\beta_1(\eta)\}, \tag{13}$$

by which means $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ are defined for all $(\xi,\eta)\in\bar{R}$.

Note that if $0<\alpha_1(\xi)$, $\alpha_2(\xi)<1$ and $0<\beta_1(\eta)$, $\beta_2(\eta)<1$ then the denominator (13) in expressions (11) and (12) is non-zero. Consequently, it can also be shown that $0<\alpha(\xi,\eta)<1$ and $0<\beta(\xi,\eta)<1$. Moreover, if $\alpha_1(\xi)=\alpha_2(\xi)$ for some $\xi\in[\xi_1,\xi_2]$, then expression (11) reduces to $\alpha(\xi,\eta)=\alpha_1(\xi)=\alpha_2(\xi)$ for all $\eta\in[\eta_1,\eta_2]$. Similarly, if $\beta_1(\eta)=\beta_2(\eta)$ for some $\eta\in[\eta_1,\eta_2]$, then (12) reduces to $\beta(\xi,\eta)=\beta_1(\eta)=\beta_2(\eta)$ for all $\xi\in[\xi_1,\xi_2]$.

With $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ defined by (11)–(13), inversion of (5) and (8) subsequently yields the proposed boundary-fitted mapping from R to D:

$$\begin{aligned}x(\xi,\eta)=&[\{x_\alpha(\xi,\eta)\delta_x(\eta)+y_\alpha(\xi,\eta)\delta_y(\eta)\}\epsilon_y(\xi)\\&-\{x_\beta(\xi,\eta)\epsilon_x(\xi)+y_\beta(\xi,\eta)\epsilon_y(\xi)\}\delta_y(\eta)]/\Delta(\xi,\eta)\end{aligned} \tag{14}$$

$$\begin{aligned}y(\xi,\eta)=&[\{x_\beta(\xi)\epsilon_x(\xi)+y_\beta(\xi,\eta)\epsilon_y(\xi)]\delta_x(\eta)\\&-\{x_\alpha(\xi,\eta)\delta_x(\eta)+y_\alpha(\xi,\eta)\delta_y(\eta)\}\epsilon_x(\xi)]/\Delta(\xi,\eta),\end{aligned} \tag{15}$$

in which $x_\alpha(\xi,\eta)$, $y_\alpha(\xi,\eta)$, $x_\beta(\xi,\eta)$ and $y_\beta(\xi,\eta)$ are defined by (3), (4), (6) and (7) respectively, and

$$\Delta(\xi,\eta)=\delta_x(\eta)\epsilon_y(\xi)-\epsilon_x(\xi)\delta_y(\eta). \tag{16}$$

Meshes generated by application of this technique as described above are illustrated in Figs 1 and 2. We remark that a mesh generated with this technique is invariant under rotation and under scaling of both of the coordinate axes in the physical plane, but not necessarily under scaling of only one of the coordinate axes.

The potential weakness of the numerical grid generation technique described in this section lies in the use of the projection of the curves in the physical plane defined by $\xi=\xi'$ and $\eta=\eta'$ onto the lines joining their end-points. If these projections are poorly defined, the technique may break down. For instance, if the computational domain contains a point (ξ',η') such that in the physical plane the lines joining the end-points of the curves defined by $\xi=\xi'$ and $\eta=\eta'$ are parallel, then the technique will not work since the denominator (16) in expressions (14) and (15) will become zero. Similarly, if the projection of a boundary curve onto the line joining its extremities is not one-to-one, the technique may fail to generate a satisfactory grid. Such difficulties can be avoided by exercising some additional control of the numerical grid. This can be achieved by, for example, specifying the physical coordinates of a number of points or curves in the interior of the problem domain and treating these as fictive boundaries which together with the original boundaries define a new domain, and then applying the extension of the technique described in the next section to the adapted domain.

3. EXTENSION TO MORE COMPLICATED SITUATIONS

Up to now, the discussion of the theoretical basis of the numerical grid generation technique has been restricted to the generation of a boundary-fitted grid on a simply-connected domain by considering a canonical rectangular region in the computational plane. In this section, we shall present the extension of the theoretical basis to more irregular physical domains in which corresponding computational domain consists of the union of a number of rectangles.

We consider a canonical region R defined in terms of a coordinate system (ξ, η) satisfying the following criterion. For each $(\xi, \eta) \in R$ let

$$\begin{aligned}
\xi_1(\xi, \eta) &= \inf\{\xi' : (\xi'', \eta) \in R \quad \text{for all } \xi'' \in (\xi', \xi]\} \\
\xi_2(\xi, \eta) &= \sup\{\xi' : (\xi'', \eta) \in R \quad \text{for all } \xi'' \in [\xi, \xi')\} \\
\eta_1(\xi, \eta) &= \inf\{\eta' : (\xi, \eta'') \in R \quad \text{for all } \eta'' \in (\eta', \eta]\} \\
\eta_2(\xi, \eta) &= \sup\{\eta' : (\xi, \eta'') \in R \quad \text{for all } \eta'' \in [\eta, \eta')\};
\end{aligned}$$

then the total set of coordinate values $\xi_i(\xi, \eta)$, $(\xi, \eta) \in R$, $i = 1, 2$, consists of a finite partition, $a_0 < a_1 < a_2 < \cdots < a_n$, and the total set of coordinate values $\eta_i(\xi, \eta)$, $(\xi, \eta) \in R$, $i = 1, 2$, consists of a finite partition, $b_0 < b_1 < b_2 < \cdots < b_m$. It follows that we can state that

$$R = \bigcup_{(i,j) \in I_1} (a_{i-1}, a_i) \times (b_{j-1}, b_j) \bigcup_{(i,j) \in I_2} (a_{i-1}, a_i) \times [b_j, b_j]$$

$$\bigcup_{(i,j) \in I_3} [a_i, a_i] \times (b_{j-1}, b_j) \bigcup_{(i,j) \in I_4} [a_i, a_i] \times [b_j, b_j]$$

for some or other (sub)collections of pairs of indices (i, j) with $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$; I_1, I_2, I_3 and I_4 say. Moreover, given R, this is the smallest union of rectangles with this property. Examples of such canonical regions R are given in Figs 1(b), 2(b), 3(b), 4(b), 5(b) and 6(b). The boundary of R may contain isolated points and isolated curves corresponding to slits in the physical domain. A problem domain D defined in the physical plane is characterized by the existence of a homeomorphic mapping from the boundary of R onto the boundary of D, viz. one has specified $\partial D = \{(x(\xi, \eta), y(\xi, \eta)) : (\xi, \eta) \in \partial R\}$.

Note that by definition, given any point $(\xi, \eta) \in R$, if j is such that $b_{j-1} \leqslant \eta \leqslant b_j$; although $\xi_1(\xi, \eta)$ and $\xi_2(\xi, \eta)$ may span several segments of the partition $a_0, a_1, \ldots, a_n$; the points $(\xi_1(\xi, \eta), b_{j-1})$, $(\xi_2(\xi, \eta), b_{j-1})$, $(\xi_1(\xi, \eta), b_j)$, and $(\xi_2(\xi, \eta), b_j) \in \partial R$; whilst the rectangle $(\xi_1(\xi, \eta), \xi_2(\xi, \eta)) \times (b_{j-1}, b_j)$ is wholly contained in R. Hence, for $j = 1, 2, \ldots, m$, we can inductively define:

$$\begin{aligned}
\alpha_1(\xi, \eta) = [&\{x(\xi, b_{j-1}) - x(\xi_1(\xi, \eta), b_{j-1})\}\delta_{x,1} \\
&+ \{y(\xi, b_{j-1}) - y(\xi_1(\xi, \eta), b_{j-1})\}\delta_{y,1}]/[\delta_{x,1}^2 + \delta_{y,1}^2]
\end{aligned}$$

where

$$\begin{aligned}
\delta_{x,1} &= x(\xi_2(\xi, \eta), b_{j-1}) - x(\xi_1(\xi, \eta), b_{j-1}) \\
\delta_{y,1} &= y(\xi_2(\xi, \eta), b_{j-1}) - y(\xi_1(\xi, \eta), b_{j-1})
\end{aligned}$$

and

$$\begin{aligned}
x(\xi, b_{j-1}) &= \lim_{\eta \uparrow b_{j-1}} [x(\xi_1(\xi, \eta), \eta) + \alpha_1(\xi, \eta)\{x(\xi_2(\xi, \eta), \eta) - x(\xi_1(\xi, \eta), \eta)\}] \\
y(\xi, b_{j-1}) &= \lim_{\eta \uparrow b_{j-1}} [y(\xi_1(\xi, \eta), \eta) + \alpha_1(\xi, \eta)\{y(\xi_2(\xi, \eta), \eta) - y(\xi_1(\xi, \eta), \eta)\}] \\
&\quad \text{if } (\xi, b_{j-1}) \notin \partial R,
\end{aligned}$$

for all $(\xi, \eta) \in R$ with $b_{j-1} \leqslant \eta < b_j$. Similarly, we can inductively define

$$\alpha_2(\xi, \eta) = [\{x(\xi, b_j) - x(\xi_1(\xi, \eta), b_j)\}\delta_{x,2} + \{y(\xi, b_j) - y(\xi_1(\xi, \eta), b_j)\}\delta_{y,2}]/[\delta_{x,2}^2 + \delta_{y,2}^2]$$

where

$$\begin{aligned}
\delta_{x,2} &= x(\xi_2(\xi, \eta), b_j) - x(\xi_1(\xi, \eta), b_j) \\
\delta_{y,2} &= y(\xi_2(\xi, \eta), b_j) - y(\xi_1(\xi, \eta), b_j)
\end{aligned}$$

and

$$x(\xi, b_j) = \lim_{\eta \downarrow b_j} [x(\xi_1(\xi, \eta), \eta) + \alpha_2(\xi, \eta)\{x(\xi_2(\xi, \eta), \eta) - x(\xi_1(\xi, \eta), \eta)\}]$$

$$y(\xi, b_j) = \lim_{\eta \downarrow b_j} [y(\xi_1(\xi, \eta), \eta) + \alpha_2(\xi, \eta)\{y(\xi_2(\xi, \eta), \eta) - y(\xi_1(\xi, \eta), \eta)\}]$$

$$\text{if } (\xi, b_j) \notin \partial R,$$

for all $(\xi, \eta) \in R$ with $b_j \geqslant \eta > b_{j-1}$, for $j = m, m-1, m-2, \ldots, 1$. Likewise, given any $(\xi, \eta) \in R$, if i is such that $a_{i-1} \leqslant \xi \leqslant a_i$; although $\eta_1(\xi, \eta)$ and $\eta_2(\xi, \eta)$ may span several segments of the partition $b_0, b_1, \ldots, b_m$; the points $(a_{i-1}, \eta_1(\xi, \eta))$, $(a_{i-1}, \eta_2(\xi, \eta))$, $(a_i, \eta_1(\xi, \eta))$, and $(a_i, \eta_2(\xi, \eta)) \in \partial R$, whilst the rectangle $(a_{i-1}, a_i) \times (\eta_1(\xi, \eta), \eta_2(\xi, \eta))$ is wholly contained in R. So that, for $i = 1, 2, \ldots, n$, we can inductively define

$$\beta_1(\xi, \eta) = [\{x(a_{i-1}, \eta) - x(a_{i-1}, \eta_1(\xi, \eta))\}\epsilon_{x,1} + \{y(a_{i-1}, \eta) - y(a_{i-1}, \eta_1(\xi, \eta))\}\epsilon_{y,1}]/[\epsilon_{x,1}^2 + \epsilon_{y,1}^2]$$

where

$$\epsilon_{x,1} = x(a_{i-1}, \eta_2(\xi, \eta)) - x(a_{i-1}, \eta_1(\xi, \eta))$$

$$\epsilon_{y,1} = y(a_{i-1}, \eta_2(\xi, \eta)) - y(a_{i-1}, \eta_1(\xi, \eta))$$

and

$$x(a_{i-1}, \eta) = \lim_{\xi \uparrow a_{i-1}} [x(\xi, \eta_1(\xi, \eta)) + \beta_1(\xi, \eta)\{x(\xi, \eta_2(\xi, \eta)) - x(\xi, \eta_1(\xi, \eta))\}]$$

$$y(a_{i-1}, \eta) = \lim_{\xi \uparrow a_{i-1}} [y(\xi, \eta_1(\xi, \eta)) + \beta_1(\xi, \eta)\{y(\xi, \eta_2(\xi, \eta)) - y(\xi, \eta_1(\xi, \eta))\}]$$

$$\text{if } (a_{i-1}, \eta) \notin \partial R,$$

for all $(\xi, \eta) \in R$ with $a_{i-1} \leqslant \xi < a_i$; and, for $i = n, n-1, n-2, \ldots, 1$,

$$\beta_2(\xi, \eta) = [\{x(a_i, \eta) - x(a_i, \eta_1(\xi, \eta))\}\epsilon_{x,2} + \{y(a_i, \eta) - y(a_i, \eta_1(\xi, \eta))\}\epsilon_{y,2}]/[\epsilon_{x,2}^2 + \epsilon_{y,2}^2]$$

where

$$\epsilon_{x,2} = x(a_i, \eta_2(\xi, \eta)) - x(a_i, \eta_1(\xi, \eta))$$

$$\epsilon_{y,2} = y(a_i, \eta_2(\xi, \eta)) - y(a_i, \eta_1(\xi, \eta))$$

and

$$x(a_i, \eta) = \lim_{\xi \downarrow a_i} [x(\xi, \eta_1(\xi, \eta)) + \beta_2(\xi, \eta)\{x(\xi, \eta_2(\xi, \eta)) - x(\xi, \eta_1(\xi, \eta))\}]$$

$$y(a_i, \eta) = \lim_{\xi \downarrow a_i} [y(\xi, \eta_1(\xi, \eta)) + \beta_2(\xi, \eta)\{y(\xi, \eta_2(\xi, \eta)) - y(\xi, \eta_1(\xi, \eta))\}]$$

$$\text{if } (a_i, \eta) \notin \partial R,$$

for all $(\xi, \eta) \in R$ with $a_i \geqslant \xi > a_{i-1}$. The formulae for the boundary-fitted mapping of R to D are now given by (1)–(4), (6), (7), and (9)–(16), with $\alpha_i(\xi)$ replaced by $\alpha_i(\xi, \eta)$, $\beta_i(\eta)$ replaced by $\beta_i(\xi, \eta)$, ξ_i replaced by $\xi_i(\xi, \eta)$, and η_i replaced by $\eta_i(\xi, \eta)$, for $i = 1, 2$.

In principal, the above formulae can be used to generate a numerical grid on an arbitrarily-shaped multiply-connected domain without further ado. Nonetheless, tests have revealed that this does not always lead to the optimal form, since, in general, as it were, the coordinates generated in a point (ξ', η') will be predominantly determined by the behaviour of the physical boundary corresponding to the lines $\xi = \xi_1(\xi', \eta')$,

$\xi = \xi_2(\xi', \eta')$, $\eta = \eta_1(\xi', \eta')$, and $\eta = \eta_2(\xi', \eta')$ at the expense of ignoring "obstacles" and "islands" in the physical plane in the neighbourhood of the image point $(x(\xi', \eta'), y(\xi', \eta'))$. The form improves through the simple measure of taking the grid generation in stages.

First, physical coordinates for those points in the computational plane (ξ, η) with the property that $[\xi_1(\xi, \eta), \xi_2(\xi, \eta)] \times [\eta_1(\xi, \eta), \eta_2(\xi, \eta)]$ is contained in $\bar{R}$ are generated. Once the physical coordinates for these points have been generated, a new computational domain consisting of the remaining points in R is automatically defined, and the procedure is repeated. Since the originally-specified computational domain is composed of the union of a finite number of rectangles, this strategy need only be repeated a limited number of times to satisfactorily generate an entire grid in the original domain.

Compared with decomposing D into subdomains corresponding to rectangular subregions of R and subsequently patching meshes generated independently on each subdomain, the essential feature of the extension of the grid generation technique described above is that it is automatic. That is to say, it does not require the specification of any internal dividing curves on the part of the user (although naturally one is free to define imaginary internal boundaries if one so wishes).

Meshes generated by the extended technique as described in this section are shown in Figs 3–6. These meshes illustrate the scope of the technique. In particular, Figs 4 and 6 depict meshes in which the physical domain contains one or more "obstacles", whilst Fig. 5 depicts a mesh in which the physical domain contains an "island". All of these meshes are derived from practical applications in the field of computational fluid dynamics which are reviewed in the Introduction.

Acknowledgements—The technique described in this paper was developed whilst the author was employed by the Delft Hydraulics Laboratory. The subsequent consent of the laboratory to the publication of this work is thus gratefully acknowledged.

REFERENCES

1. J. F. Thompson (Editor), *Numerical Grid Generation*. Elsevier, New York (1982).
2. J. F. Thompson, Z. U. A. Warsi and C. W. Mastin, Boundary-fitted coordinate systems for numerical solution of partial differential equations—a review. *J. Comput. Phys.* **47**, 1–108 (1982).
3. J. F. Thompson, Grid generation techniques in computational fluid dynamics. *AIAA J.* **22**, 1505–1523 (1984).
4. L.-E. Eriksson, Practical three-dimensional mesh generation using transfinite interpolation. *SIAM J. Sci. Stat. Comput.* **6**, 712–741 (1985).
5. R. E. Smith, Algebraic grid generation. In: *Numerical Grid Generation* (Edited by J. F. Thompson), pp. 137–170. Elsevier, New York (1982).
6. S.-L. Yang and T. I.-P. Shih, An algebraic grid generation technique for time-varying two-dimensional spatial domains. *Int. J. numer. Meth. Fluids* **6**, 291–304 (1986).
7. W. J. Gordon and L. C. Thiel, Transfinite mappings and their application to grid generation. In: *Numerical Grid Generation* (Edited by J. F. Thompson) , pp. 171–192. Elsevier, New York (1982).
8. M. J. Officier, C. B. Vreugdenhil and H. G. Wind, Applications in hydraulics of numerical solutions of the Navier–Stokes equations. In: *Computational Techniques for Fluid Flow* (Edited by C. Taylor, J. A. Johnson and W. R. Smith), pp. 115–147. Pineridge Press, Swansea (1986).
9. M. J. Officier and A. K. Wiersma, Experience with numerical grid generation techniques and their application in flow problems at the Delft Hydraulics Laboratory. In: *Numerical Grid Generation in Computational Fluid Dynamics* (Edited by J. Häuser and C. Taylor), pp. 641–652. Pineridge Press, Swansea (1986).