INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 13, 1–31 (1991)

ALGEBRAIC GRID GENERATION FOR COMPLEX GEOMETRIES

T. I-P. SHIH

Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890, U.S.A.

R. T. BAILEY

Department of Mechanical Engineering, University of Florida, Gainesville, FL 32611, U.S.A.

AND

H. L. NGUYEN AND R. J. ROELKE

National Aeronautics and Space Administration, Lewis Research Center, Cleveland, OH 44135, U.S.A.

SUMMARY

An efficient computer programme called GRID2D/3D has been developed to generate single and composite grid systems within geometrically complex two- and three-dimensional (2D and 3D) spatial domains that can deform with time. GRID2D/3D generates single grid systems by using algebraic grid generation methods based on transfinite interpolation. The distribution of grid points within the spatial domain is controlled by stretching functions and grid lines can intersect boundaries of the spatial domain orthogonally. GRID2D/3D generates composite grid systems by patching together two or more single grid systems. The patching can be discontinuous or continuous. For 2D spatial domains the boundary curves are constructed by using either cubic or tension spline interpolation. For 3D spatial domains the boundary surfaces are constructed by using a new technique, developed in this study, referred to as 3D bidirectional Hermite interpolation.

KEY WORDS Grid generation Surface generation

INTRODUCTION

Finite difference (FD) and finite volume (FV) methods are very powerful techniques for obtaining solutions to partial differential equations that govern fluid flow problems. However, in order to use these methods, it is necessary to replace the spatial domain of the problem being studied by a finite number of discrete points known as grid points. The process of replacing a spatial domain by a system of grid points is referred to as grid generation. Grid generation is a very important part of FD and FV methods because the system of grid points used strongly affects the accuracy, efficiency and ease with which these methods generate solutions. In some instances the ability or inability to generate an 'acceptable' grid system determines whether FD or FV methods can or cannot be used.

Even though tremendous advances have been made in grid generation techniques during the past 15 years,[1–10] the generation of acceptable grid systems for geometrically complex three-dimensional spatial domains remains a difficult problem. Recently, a very efficient and versatile computer programme called GRID2D/3D has been developed which can generate grid systems

0271–2091/91/110001–31$15.50

Received February 1990
Revised July 1990

inside complex-shaped two- and three-dimensional (2D and 3D) spatial domains. GRID2D/3D is so efficient that it is configured to run on PCs or PC-compatible computers, though it can also be used on workstations and mainframes. The high efficiency of GRID2D/3D makes it especially useful for spatial domains that deform with time. This is because for such spatial domains a different grid system must be generated at each time level and the number of time levels can be thousands or more. This paper describes GRID2D/3D.

The outline of the paper is as follows. First the various types of grid systems that can be generated by GRID2D/3D are described along with their advantages and disadvantages. Afterwards the methods used in GRID2D/3D to generate grid systems are presented. Then a discussion is given of methods for generating parametric representations of 2D and 3D boundaries. Finally several examples of grid systems generated by GRID2D/3D are presented.

TYPES OF GRID SYSTEMS

Eiseman and Erlebacher[9] classified all possible grid systems that can be used by FD and FV methods as follows. At the broadest level a grid system can be classified as structured, unstructured or mixed, depending upon how the grid points are connected to each other (Figure 1). A structured grid system in turn can be classified as a single grid or a composite grid. A single grid is one that is based on a single boundary-fitted co-ordinate system, whereas a composite grid is made up of two or more single grids patched together with each single grid having a different boundary-fitted co-ordinate system. Depending upon how the different single grids are patched together, a composite grid can further be classified as completely discontinuous, partially discontinuous, partially continuous or completely continuous (Figure 2). The continuity or discontinuity referred to here is concerned with that of the different boundary-fitted co-ordinate systems at locations where they are patched together in a composite grid.

Of the grid systems mentioned above, the unstructured grid system is the most versatile and the easiest to generate, especially for complicated-shaped spatial domains. However, the use of

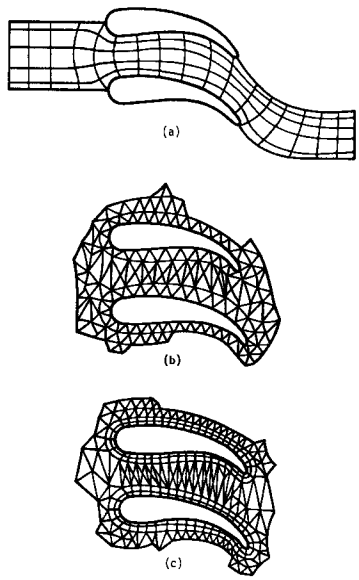

Figure 1. Types of grid systems: (a) structured; (b) unstructured; (c) mixed

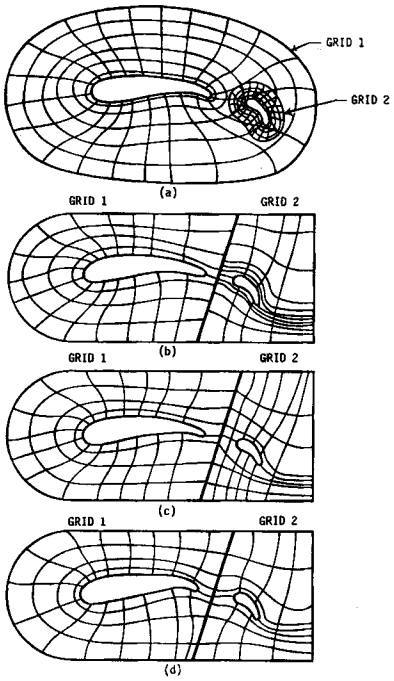

Figure 2. Types of composite grids: (a) completely discontinuous; (b) partially discontinuous; (c) partially continuous; (d) partially or completely continuous

unstructured grid systems with FD and FV methods is still at a state of development.[11-16] Presently, FD and FV methods almost exclusively use structured grid systems, and those are the ones that GRID2D/3D generates.

When a structured grid system is used with an FD or FV method to obtain solutions to fluid flow problems, the structured grid system generated by GRID2D/3D or any other computer programme should satisfy a number of conditions.

1. The total number of grid points in the grid system should be kept to the minimum needed for the FD or FV method to yield solutions of the desired accuracy. This condition is important for computational efficiency and can be achieved by clustering grid points in regions where they are needed (e.g. regions where gradients of the flow are large) and scattering them elsewhere.
2. One set of grid lines (co-ordinate lines of the boundary-fitted co-ordinate system) should always coincide with the boundary of the spatial domain regardless of the geometric complexity or motion of that boundary (i.e. the grid system should be boundary-conforming). This condition is important because it enables FD and FV methods to implement boundary conditions easily and accurately for geometrically complex and/or deforming spatial domains.
3. Grid lines that intersect a boundary should intersect that boundary perpendicularly so that derivative boundary conditions can be implemented more easily and accurately. At the interior of the spatial domain the angle of intersection between grid lines only needs to be nearly orthogonal (i.e. between 45° and 135°).

4. The spacings between grid points should change slowly from a region where grid points are concentrated to a region where grid points are sparsely distributed, especially in regions where gradients of the flow are large. This condition is important because Fourier components which make up the solution reflect and refract at interfaces where grid spacings change.
5. One set of grid lines should align with the flow direction. This condition is important for convection-dominated flows when the aspect ratio of the control volume about each grid point is very high and/or when the thin layer Navier–Stokes equations are used to study such flows.

For complicated 2D and 3D flows within geometrically complex spatial domains it is usually impossible to generate a single grid that would satisfy all of the above conditions at every part of the spatial domain. For such problems it is necessary to generate a number of different single grids, each of which satisfies the above five conditions at a different part of the spatial domain. These single grids are then patched together to form a composite grid, which as noted earlier can be completely discontinuous, partially discontinuous, partially continuous or completely continuous.

Since complicated geometries invariably imply composite grids which can be generated by using GRID2D/3D, below we describe the advantages and disadvantages of the various types of composite grids in order to know when a specific type should be used for a given problem.

Completely discontinuous composite (CDC) grids

The major advantage of CDC grids such as the chimera grid[17–21] (Figure 2(a)) is that they are the easiest to generate. For example, to generate a chimera grid over the spatial domain about an aircraft, all one has to do is generate a series of single grids, one about each component of the aircraft (e.g. one about the fuselage, another about the wing, and so on). The patching process simply involves laying each single grid over the appropriate component of the aircraft, deciding on the amount of overlap of different single grids and ensuring that the entire spatial domain is filled with grid points. Since the geometry for each single grid can be made as simple as desired and patching is trivial, the grid generation process is straightforward. Another important advantage of CDC grids is that the structure of each single grid can be different from each other; e.g. one single grid may have a C–C structure while another may have an O–O or an O–H structure. Thus it is possible to optimize each single grid for a different part of the spatial domain. Still another advantage is that this is the easiest grid on which to do local grid refinement. For a chimera grid one can refine the grid at any location by simply generating a very fine single grid and then overlaying it wherever desired. Also, this type of grid can easily be applied to problems in which one or more objects are moving relative to another object, such as the launching of missiles from an aircraft.[20] Finally, since composite grids are composed of a series of single grids, it is possible to do computations on one single grid at a time. This will reduce computer memory requirements considerably since only information on one single grid needs to reside in the computer at any one time. This advantage is shared by all composite grids, continuous or discontinuous.

The major disadvantage of CDC grids is that they are the most difficult to use in obtaining solutions when compared to other types of composite grids. This is because interpolation schemes are needed to transfer information from one single grid to another when and wherever two or more single grids overlap.[18–21] Also, schemes used should ensure that properties such as conservation and monotonicity are maintained in regions where two or more single grids overlap.

Finally, information is lost when transferred from a fine to a coarse grid; thus grid spacings of different single grids should be approximately equal in regions where they overlap.

Partially discontinuous composite (PDC) grids

PDC grids (Figure 2(b)) are generated in the following manner. First, the spatial domain of the problem being studied is partitioned into a number of non-overlapping, contiguous zones or blocks. Next, a single grid is generated within each zone. Finally, patching of the single grids simply involves putting each of the single grids into its respective zone. Thus PDC grids differ from CDC grids in that the single grids of PDC grids to not overlap each other.

PDC and CDC grids have two important similarities. First, each single grid in both cases can have a structure that is different from each other. Secondly, the number of grid points in each single grid can be different from each other. Because of these two similarities, the major advantages of PDC grids are the same as those of the CDC grids. However, since single grids in a PDC grid do not overlap each other, PDC grids are somewhat more difficult to generate but are easier to use than CDC grids. The main difficulty in using PDC grids is implementing boundary conditions at interfaces where different single grids meet.[22-25]

Partially and completely continuous composite grids

Completely continuous composite (CCC) grids are grid systems in which all grid lines (i.e. co-ordinate lines of the boundary-fitted co-ordinate system) and all of their derivatives of every order are continuous at all interfaces where different single grids meet. In general, it is not necessary to construct CCC grids. Typically, FD and FV methods only require continuity of the grid lines and their first- and, occasionally, second-order derivatives at the interfaces where different single grids meet. Composite grid systems with this limited degree of continuity are referred to as partially continuous composite (PCC) grids.

Figure 2(c) shows a PCC grid in which the grid lines are all continuous but their first-order derivatives have discontinuities. It can readily be seen in that figure that the slope of the grid lines and the spacing between the grid lines change suddenly at the interface where the two single grids meet. Figure 2(d) shows a PCC grid in which the grid lines and their first-order derivatives are continuous everywhere, including the interface where the two single grids meet. Such PCC grids have the same appearance as CCC grids.

The major advantage of PCC grids of the type shown in Figure 2(d) is that this is the easiest grid system for FD and FV methods to use. This is because boundary conditions can be implemented easily at interfaces where different single grids meet. In fact it is not even necessary to treat the interfaces where different single grids meet as boundaries since computations can be carried across them. For PCC grids the complete spatial domain of the problem can be mapped onto a single transformed domain, even though different boundary-fitted co-ordinate systems have been used in different parts of the spatial domain (Figure 3). Here it is important to note that not all grid systems which appear to be continuous are continuous. Figure 4 shows a composite grid that appears to be continuous but belongs to the PDC grids because it is impossible to map the entire spatial domain onto one transformed domain.

The major disadvantage of PCC grids is that they are the most difficult to generate when compared to CDC and PDC grids. Another disadvantage is that the structure and number of grid points in each single grid must satisfy certain compatibility conditions in order to ensure continuity. These compatibility conditions make it more difficult to optimize each single grid for a specific area. They also make it more difficult to do local grid refinement.

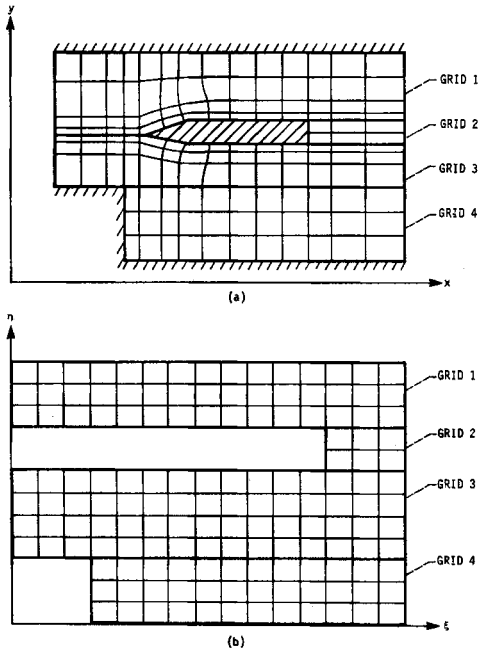

Figure 3. A continuous composite grid system made up of four single grids: (a) in x–y co-ordinate system; (b) in ξ–η co-ordinate system

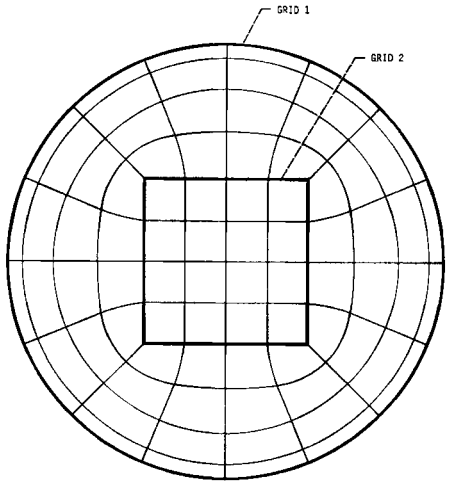

Figure 4. A partially discontinuous composite grid that appears like a partially or completely continuous composite grid

Thus there are many spatial domains for which it is extremely difficult, if not impossible, to generate a PCC grid that is acceptable. However, when it is possible then the generation of such grids is worthwhile because of the ease with which they can be used. References 26–31 show a number of examples of how to construct PCC grids for complex-shaped 2D and 3D spatial domains.

GRID GENERATION METHODS

All grid generation techniques are classified as either differential equation or algebraic methods. Differential equation methods generate grid systems by solving a system of partial differential equations (PDEs) which describes how grid points are to be distributed within the spatial domain. In general, these methods require a significant amount of computational effort since the systems of PDEs that must be solved are quasi-linear and often as complicated as the PDEs that govern the fluid flow problem. Algebraic methods generate grid systems by interpolating between boundaries of the spatial domain. Since no PDE needs to be solved in the grid generation process, algebraic grid generation methods are computationally much more efficient than differential equation methods.

Whether one uses a differential equation or an algebraic method, the grid generation process is always iterative. This is because the 'acceptable' grid system is arrived at via trial and error after generating a series of unsatisfactory grids. The effort of the iterative process is, of course, compounded many times for spatial domains which can deform with time, since for such domains a different grid system is needed for each time level and the number of time levels can be thousands or more. Hence the efficiency of the grid generation process is extremely important for problems with 3D spatial domains and for problems in which the spatial domain can deform.

Since GRID2D/3D is intended for complex-shaped 2D and 3D spatial domains and for spatial domains that can deform with time, GRID2D/3D generates grid systems by using algebraic grid generation methods. Depending upon the complexity and dimensionality of the spatial domain, GRID2D/3D uses one of the following three methods, all of which are very similar and are based on transfinite interpolation:[32-34] the two-boundary method,[35,36] the four-boundary method[37,38] and the six-boundary method.[38,39] These methods were chosen because of their high efficiency and their ability to provide very precise controls over the distribution of grid points in the spatial domain when used in conjunction with stretching functions.[40,41] Also, these methods can generate grid lines that intersect boundaries orthogonally.

By using these algebraic grid generation methods, GRID2D/3D generates single grids with grid lines that are continuous and differentiable everywhere up to the second order. GRID2D/3D generates composite grids by patching together two or more single grids. The patching can be discontinuous or continuous. For continuous composite grids the grid lines are continuous and differentiable everywhere up to the second order except at interfaces where different single grids meet. At interfaces where different single grids meet, the grid lines are only differentiable up to the first order.

In order to use the two-, four- and six-boundary methods to generate grid systems, the boundaries of the spatial domains must be represented mathematically in parametric form. This is a difficult problem for complicated-shaped spatial domains because the boundaries of such domains are complicated as well. In GRID2D/3D, parametric equations for boundary curves of 2D spatial domains are generated by either spline interpolation[42] or tension spline interpolation.[43] Parametric equations for boundary surfaces of 3D spatial domains can be generated by a number of techniques, including linear Coons' interpolation,[32] bidirectional spline interpolation[44,45] and bihyperbolic spline interpolation.[46] In GRID2D/3D, parametric equations for these 3D surfaces are generated by a new technique, developed in this study, referred to as 3D bidirectional Hermite interpolation. The details of the algebraic grid generation methods used in GRID2D/3D are described below.

TWO-, FOUR- AND SIX-BOUNDARY METHODS

As mentioned, GRID2D/3D generates grid systems by using either the two-, four- or six-boundary method. All three methods can generate grid systems in 3D spatial domains; the two-

and four-boundary methods can also generate grid systems in 2D spatial domains. In this section the details of these methods are described. Our step-by-step descriptions of the methods follow closely those of Yang and Shih.[36] It is our intention to present the methods in a clear manner so that the reader might easily implement them.

The two-boundary method

The two-boundary method[35, 36] is intended for problems in which it is only necessary to map correctly two arbitrary-shaped boundaries of the spatial domain. If the remaining boundaries are straight lines in the 2D case or flat surfaces in the 3D case, then all boundaries can be mapped correctly. Implementation of this method involves the following eight major steps:[36] (1) define the nature of the co-ordinate transformation; (2) select a time-stretching function; (3) select two boundaries of the spatial domain that must be mapped correctly (these two boundaries cannot touch each other at any point); (4) describe the two boundaries selected in Step 3 in parametric form; (5) define curves that connect the two boundaries by using transfinite interpolation; (6) discretize the domain (i.e. replace the continuous domain of the problem by time levels and grid points); (7) control the distribution of the grid points with stretching functions; (8) calculate metric coefficients needed by the FD or FV method to obtain solutions.

The details of these eight steps are described below by generating a single grid in the 3D, deforming spatial domain shown in Figure 5(a) for the problem of flow through a converging–diverging channel. The spatial domain of interest is the region bounded by surfaces 1–6 in Figure 5(a), which deforms because surfaces 1 and 2 deform with time.

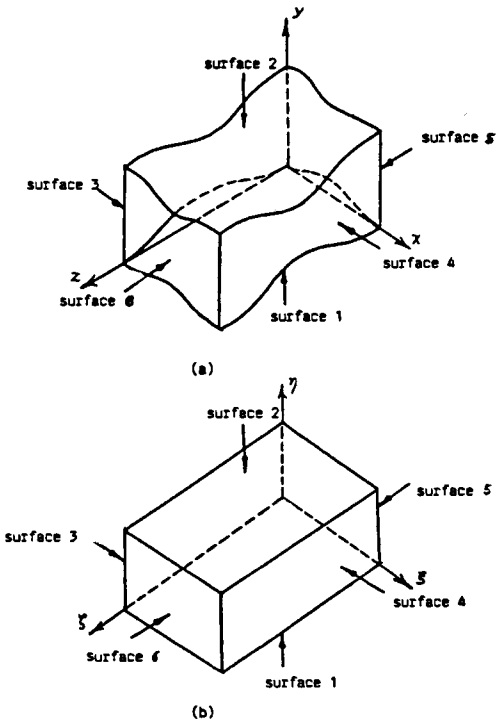

Figure 5. Spatial domain: (a) in x–y–z–t co-ordinate system; (b) in ξ–η–ζ–τ co-ordinate system

Step 1: Define the co-ordinate transformation. The first step is to define the co-ordinate transformation between the co-ordinate system of the spatial domain and the boundary-fitted co-ordinate system of the transformed domain. For 3D spatial domains with moving grid points, the following co-ordinate transformation is sought:

$$(x, y, z, t) \leftrightarrow (\xi, \eta, \zeta, \tau) \tag{1a}$$

or, more specifically,

$$t = t(\tau), \qquad x = x(\xi, \eta, \zeta, \tau), \qquad y = y(\xi, \eta, \zeta, \tau), \qquad z = z(\xi, \eta, \zeta, \tau), \tag{1b}$$

where x, y, z and t represent the co-ordinate system of the 'physical' domain and ξ, η, ζ and τ represent the boundary-fitted co-ordinate system of some transformed domain (Figures 5(a) and 5(b)).

Step 2: Select a time-stretching function. The next step is to define a relationship between t and τ. For our example we set t equal to τ, i.e.

$$t = \tau. \tag{2}$$

Thus no time-stretching function is used. Time stretching may be useful when variable time-step sizes are used with FD or FV schemes that involve information at more than two time levels.

Step 3: Select two boundaries of the spatial domain. The third step is to select the two boundaries of the spatial domain that are to be mapped correctly. These two boundaries must not intersect each other at any point. For the spatial domain of Figure 5(a) we select boundary surfaces 1 and 2. Since ξ, η, ζ and τ represent a boundary-fitted co-ordinate system, boundary surfaces of the spatial domain in the x–y–z–t co-ordinate system must correspond to co-ordinate planes in the ξ–η–ζ–τ co-ordinate system. We choose surfaces 1 and 2 to correspond to co-ordinate planes $\eta = 0$ and $\eta = 1$ respectively (Figure 5), i.e.

$$X_1 = x(\xi, \eta = 0, \zeta, \tau) = X_1(\xi, \zeta, \tau), \tag{3a}$$

$$Y_1 = y(\xi, \eta = 0, \zeta, \tau) = Y_1(\xi, \zeta, \tau), \tag{3b}$$

$$Z_1 = z(\xi, \eta = 0, \zeta, \tau) = Z_1(\xi, \zeta, \tau), \tag{3c}$$

$$X_2 = x(\xi, \eta = 1, \zeta, \tau) = X_2(\xi, \zeta, \tau), \tag{4a}$$

$$Y_2 = y(\xi, \eta = 1, \zeta, \tau) = Y_2(\xi, \zeta, \tau), \tag{4b}$$

$$Z_2 = z(\xi, \eta = 1, \zeta, \tau) = Z_2(\xi, \zeta, \tau), \tag{4c}$$

where X_i, Y_i and Z_i are the x-, y- and z-co-ordinates of surface i, $i = 1, 2$. The remaining four boundaries—surfaces 3, 4, 5 and 6—are mapped to co-ordinate planes $\xi = 0$, $\xi = 1$, $\zeta = 0$ and $\zeta = 1$ respectively (Figure 5).

Step 4: Describe the two boundaries selected in parametric form. Once the two boundaries have been selected, the next step is to represent these two boundaries in parametric form as suggested by the form of equations (3) and (4). Equations (3) and (4) also tell us that the three parameters which must be used to describe surfaces 1 and 2 are ξ, ζ and τ. In this example we assume that the parametric equations describing surfaces 1 and 2 are given and they are

$$X_1 = \xi/L_x, \qquad Y_1 = A \sin(w\tau)[1 - \cos(2\pi\xi)][1 - \cos(2\pi\zeta)], \qquad Z_1 = \zeta/L_y, \tag{5}$$

$$X_2 = \xi/L_x, \qquad Y_2 = L_y - A \sin(w\tau)[1 - \cos(2\pi\xi)][1 - \cos(2\pi\zeta)], \qquad Z_2 = \zeta/L_y, \tag{6}$$

where A, w, L_x, L_y and L_z are given constants. If parametric equations are not given, then GRID2D/3D can generate them by using either spline or tension spline interpolation (see next section).

Step 5: Define curves connecting boundaries using transfinite interpolation. A number of different transfinite interpolation techniques can be used to derive curves which connect the two boundaries chosen in Step 3. Here we consider two such methods: transfinite interpolations based on Lagrange and Hermite blending functions.

When Lagrange interpolation (also known as linearly blended transfinite interpolation) is used to generate connecting curves between surfaces 1 and 2, the resulting curves have the following functional form:

$$x(\xi, \eta, \zeta, \tau) = X_1(\xi, \zeta, \tau) l_1(\eta) + X_2(\xi, \zeta, \tau) l_2(\eta), \tag{7a}$$

$$y(\xi, \eta, \zeta, \tau) = Y_1(\xi, \zeta, \tau) l_1(\eta) + Y_2(\xi, \zeta, \tau) l_2(\eta), \tag{7b}$$

$$z(\xi, \eta, \zeta, \tau) = Z_1(\xi, \zeta, \tau) l_1(\eta) + Z_2(\xi, \zeta, \tau) l_2(\eta), \tag{7c}$$

where X_i, Y_i and Z_i ($i = 1, 2$) are given by equations (5) and (6). The functions l_1 and l_2 are blending functions which connect two points—one on each surface having the same ξ-, ζ- and τ-values. They are constrained by $l_1(\eta = 0) = 1$, $l_1(\eta = 1) = 0$, $l_2(\eta = 0) = 0$ and $l_2(\eta = 1) = 1$. With these constraints, l_1 and l_2 become

$$l_1(\eta) = 1 - \eta, \qquad l_2(\eta) = \eta. \tag{8}$$

Substitution of equation (8) in equation (7) yields the desired linear connecting curves:

$$x(\xi, \eta, \zeta, \tau) = X_1(\xi, \zeta, \tau)(1 - \eta) + X_2(\xi, \zeta, \tau)\eta, \tag{9a}$$

$$y(\xi, \eta, \zeta, \tau) = Y_1(\xi, \zeta, \tau)(1 - \eta) + Y_2(\xi, \zeta, \tau)\eta, \tag{9b}$$

$$z(\xi, \eta, \zeta, \tau) = Z_1(\xi, \zeta, \tau)(1 - \eta) + Z_2(\xi, \zeta, \tau)\eta. \tag{9c}$$

It is often desirable for connecting curves to intersect boundaries orthogonally so that derivative boundary conditions can be implemented accurately. Since one set of curves described by equation (9) are straight lines, they will not in general intersect boundaries orthogonally. One way to remedy this is to use transfinite interpolation based on Hermite interpolation to form the connecting curves. Hermite interpolation allows specification of derivatives at end points of curves, enabling one to force orthogonality at boundaries. When this method is used to generate connecting curves between surfaces 1 and 2, the resulting cubic curves have the following functional form:

$$\begin{aligned} x(\xi, \eta, \zeta, \tau) = {} & X_1(\xi, \zeta, \tau) h_1(\eta) + X_2(\xi, \zeta, \tau) h_2(\eta) \\ & + \frac{\partial x(\xi, \eta = 0, \zeta, \tau)}{\partial \eta} h_3(\eta) + \frac{\partial x(\xi, \eta = 1, \zeta, \tau)}{\partial \eta} h_4(\eta), \end{aligned} \tag{10a}$$

$$\begin{aligned} y(\xi, \eta, \zeta, \tau) = {} & Y_1(\xi, \zeta, \tau) h_1(\eta) + Y_2(\xi, \zeta, \tau) h_2(\eta) \\ & + \frac{\partial y(\xi, \eta = 0, \zeta, \tau)}{\partial \eta} h_3(\eta) + \frac{\partial y(\xi, \eta = 1, \zeta, \tau)}{\partial \eta} h_4(\eta), \end{aligned} \tag{10b}$$

$$\begin{aligned} z(\xi, \eta, \zeta, \tau) = {} & Z_1(\xi, \zeta, \tau) h_1(\eta) + Z_2(\xi, \zeta, \tau) h_2(\eta) \\ & + \frac{\partial z(\xi, \eta = 0, \zeta, \tau)}{\partial \eta} h_3(\eta) + \frac{\partial z(\xi, \eta = 1, \zeta, \tau)}{\partial \eta} h_4(\eta), \end{aligned} \tag{10c}$$

where X_i, Y_i and Z_i $(i = 1, 2)$ are given by equations (5) and (6). The functions h_1, h_2, h_3 and h_4 are blending functions which connect two points—one on each surface having the same ξ-, ζ- and τ-values. They are constrained by $h_1(\eta = 0) = 1$, $h_1(\eta = 1) = 0$, $\partial h_1(\eta = 0)/\partial\eta = 0$, $\partial h_1(\eta = 1)/\partial\eta = 0$, $h_2(\eta = 0) = 0$, $h_2(\eta = 1) = 1$, $\partial h_2(\eta = 0)/\partial\eta = 0$, $\partial h_2(\eta = 1)/\partial\eta = 0$, $h_3(\eta = 0) = 0$, $h_3(\eta = 1) = 0$, $\partial h_3(\eta = 0)/\partial\eta = 1$, $\partial h_3(\eta = 1)/\partial\eta = 0$, $h_4(\eta = 0) = 0$, $h_4(\eta = 1) = 0$, $\partial h_4(\eta = 0)/\partial\eta = 0$, and $\partial h_4(\eta = 1)/\partial\eta = 1$. With these constraints, h_1, h_2, h_3 and h_4 become[35]

$$h_1 = 2\eta^3 - 3\eta^2 + 1, \quad h_2 = -2\eta^3 + 3\eta^2, \quad h_3 = \eta^3 - 2\eta^2 + \eta, \quad h_4 = \eta^3 - \eta^2. \tag{11}$$

We choose values for $\partial x(\xi, \eta = 0, \zeta, \tau)/\partial\eta$, $\partial x(\xi, \eta = 1, \zeta, \tau)/\partial\eta$, $\partial y(\xi, \eta = 0, \zeta, \tau)/\partial\eta$, $\partial y(\xi, \eta = 1, \zeta, \tau)/\partial\eta$, $\partial z(\xi, \eta = 0, \zeta, \tau)/\partial\eta$ and $\partial z(\xi, \eta = 1, \xi, \tau)/\partial\eta$ so that connecting curves given by equation (10) will intersect surfaces 1 and 2 orthogonally. For surface 1 this will occur when the cross product of $\mathbf{n}$ (a vector normal to surface 1) and $\mathbf{e}_\eta$ (the vector tangent to the connecting curve) is zero. This will be the case when

$$\frac{\partial x(\xi, \eta = 0, \zeta, \tau)}{\partial\eta} = K_1(\xi, \zeta, \tau)\left(\frac{\partial Y_1}{\partial\zeta}\frac{\partial Z_1}{\partial\xi} - \frac{\partial Z_1}{\partial\zeta}\frac{\partial Y_1}{\partial\xi}\right), \tag{12a}$$

$$\frac{\partial y(\xi, \eta = 0, \zeta, \tau)}{\partial\eta} = -K_1(\xi, \zeta, \tau)\left(\frac{\partial X_1}{\partial\zeta}\frac{\partial Z_1}{\partial\xi} - \frac{\partial Z_1}{\partial\zeta}\frac{\partial X_1}{\partial\xi}\right), \tag{12b}$$

$$\frac{\partial z(\xi, \eta = 0, \zeta, \tau)}{\partial\eta} = K_1(\xi, \zeta, \tau)\left(\frac{\partial X_1}{\partial\zeta}\frac{\partial Y_1}{\partial\xi} - \frac{\partial Y_1}{\partial\zeta}\frac{\partial X_1}{\partial\xi}\right). \tag{12c}$$

Similarly, the connecting curves will intersect surface 2 orthogonally when

$$\frac{\partial x(\xi, \eta = 1, \zeta, \tau)}{\partial\eta} = K_2(\xi, \zeta, \tau)\left(\frac{\partial Y_2}{\partial\xi}\frac{\partial Z_2}{\partial\zeta} - \frac{\partial Z_2}{\partial\xi}\frac{\partial Y_2}{\partial\zeta}\right), \tag{13a}$$

$$\frac{\partial y(\xi, \eta = 1, \zeta, \tau)}{\partial\eta} = -K_2(\xi, \zeta, \tau)\left(\frac{\partial X_2}{\partial\xi}\frac{\partial Z_2}{\partial\zeta} - \frac{\partial Z_2}{\partial\xi}\frac{\partial X_2}{\partial\zeta}\right), \tag{13b}$$

$$\frac{\partial z(\xi, \eta = 1, \zeta, \tau)}{\partial\eta} = K_2(\xi, \zeta, \tau)\left(\frac{\partial X_2}{\partial\xi}\frac{\partial Y_2}{\partial\zeta} - \frac{\partial Y_2}{\partial\xi}\frac{\partial X_2}{\partial\zeta}\right). \tag{13c}$$

$K_1(\xi, \zeta, \tau)$ and $K_2(\xi, \zeta, \tau)$ in equations (12) and (13) are known as 'K-factors' and are chosen by trial and error so that no overlapping of the connecting curves takes place in the interior of the spatial domain. For our problem the 'K-factors' were set equal to a constant (i.e. $K_1(\xi, \zeta, \tau) = K_2(\xi, \zeta, \tau) = 0{\cdot}2$).

The values of $\partial X_1(\xi, \zeta, \tau)/\partial\xi$, $\partial Y_1(\xi, \zeta, \tau)/\partial\xi$, $\partial X_2(\xi, \zeta, \tau)/\partial\xi$, $\partial Y_2(\xi, \zeta, \tau)/\partial\xi$, etc. in equations (12) and (13) can easily be found by analytically differentiating equations (5) and (6) or by using finite difference formulae. Thus substitution of equations (11) to (13) in equation (10) yields the desired cubic connecting curves based on Hermite interpolation.

Step 6: Discretize the domain. Steps 1–5 map the domain in the x–y–z–t co-ordinate system onto the ξ–η–ζ–τ co-ordinate system. Now we need to discretize the domain, i.e. replace the continuous domain by time levels and grid points.

Here the time domain is replaced by equally incremented time levels, i.e.

$$\tau^n = n\Delta\tau, \quad n = 0, 1, 2, \ldots. \tag{14}$$

In the above equation, τ^n denotes the time at time level n and $\Delta\tau$ denotes the constant time step

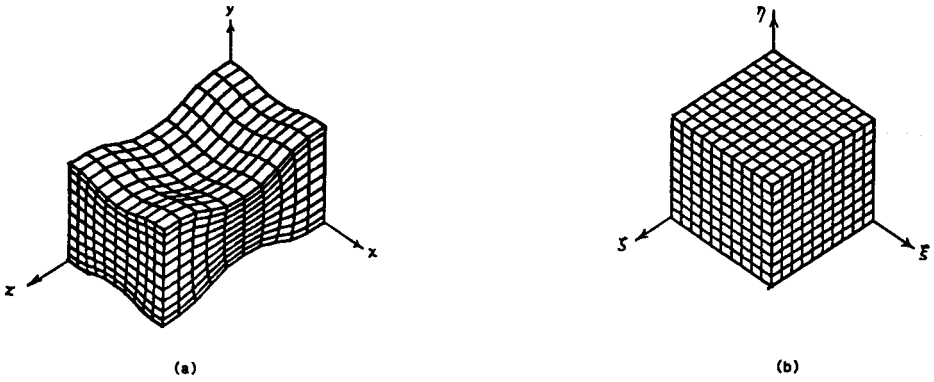

Figure 6. Grid system: (a) in x–y–z–t co-ordinate system; (b) in ξ–η–ζ–τ co-ordinate system

size. The spatial domain in the ξ–η–ζ–τ co-ordinate system is replaced by $IL \times JL \times KL$ equally spaced grid points (Figure 6(b)). The locations of these grid points are given by the ordered triples (ξ_i, η_j, ζ_k) defined by

$$\xi_i = (i-1)\Delta\xi, \qquad \eta_j = (j-1)\Delta\eta, \qquad \zeta_k = (k-1)\Delta\zeta, \tag{15a}$$

$$\Delta\xi = \frac{1}{IL-1}, \qquad \Delta_\eta = \frac{1}{JL-1}, \qquad \Delta\zeta = \frac{1}{KL-1}, \tag{15b}$$

where $i = 1, 2, \ldots, IL$, $j = 1, 2, \ldots, JL$ and $k = 1, 2, \ldots, KL$. By substituting equation (15) in equation (10), we obtain the locations of the grid points in the x–y–z–t co-ordinate system (Figure 6(a)).

Step 7: Control the distribution of grid points. At this point we need to examine the grid system shown in Figure 6(a) and ask: is the distribution of grid points satisfactory? In order to answer this question, we need to consider the physics of the problem for which the grid is generated. For accurate solutions, grid points should be clustered in regions of the spatial domain where sharp gradients in the dependent variables exist. Such clustering can be achieved by the use of stretching functions.[40,41] For flow within the 3D spatial domain shown in Figure 5(a) we expect steep gradients near walls (surfaces 1, 2, 3 and 4). Grid points can be clustered near surfaces 1 and 2 by replacing η in equation (10) by

$$\frac{(\beta_\eta + 1)[(\beta_\eta + 1)/(\beta_\eta - 1)]^{(2\eta-1)} - \beta_\eta + 1}{2\{1 + [(\beta_\eta + 1)/(\beta_\eta - 1)]^{(2\eta-1)}\}}, \tag{16}$$

where β_η is a constant greater than unity. More clustering takes place in the η-direction near $\eta = 0$ and $\eta = 1$ as β_η approaches unity. In a similar manner, grid points can be clustered near surfaces 3 and 4 by replacing ξ in equation (10) by

$$\frac{(\beta_\xi + 1)[(\beta_\xi + 1)/(\beta_\xi - 1)]^{(2\xi-1)} - \beta_\xi + 1}{2\{1 + [(\beta_\xi + 1)/(\beta_\xi - 1)]^{(2\xi-1)}\}}, \tag{17}$$

where β_ξ is a constant greater than unity that acts in the ξ-direction as β_η does in the η-direction. The new distribution of grid points in the x–y–z–t co-ordinate system after stretching is shown in Figure 7.

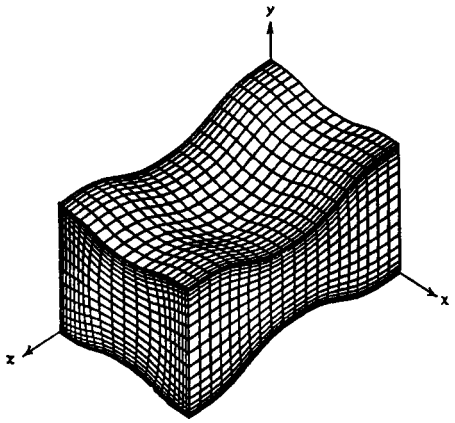

Figure 7. Grid system in the x–y–z–t co-ordinate system after stretching

Step 8: Calculate metric coefficients. Once we obtain a satisfactory distribution of grid points in the x–y–z–t co-ordinate system, we are ready to calculate metric coefficients. Metric coefficients appear in the governing equations when they are transformed from the co-ordinate system of the 'physical' domain (x–y–z–t in our example) to the boundary-fitted co-ordinate system (ξ–η–ζ–τ). With the co-ordinate transformation described by equation (1), the metric coefficients which appear are τ_t, ξ_t, η_t, ζ_t, ξ_x, η_x, ζ_x, ξ_y, η_y, ζ_y, ξ_z, η_z and ζ_z. These metric coefficients can be calculated by the following expressions:[2, 3, 8]

$$\tau_t = 1, \tag{18a}$$

$$\xi_x = (y_\eta z_\zeta - y_\zeta z_\eta)/J, \qquad \xi_y = -(x_\eta z_\zeta - x_\zeta z_\eta)/J, \qquad \xi_z = (x_\eta y_\zeta - x_\zeta y_\eta)/J, \tag{18b}$$

$$\xi_t = -[x_\tau(y_\eta z_\zeta - y_\zeta z_\eta) - y_\tau(x_\eta z_\zeta - x_\zeta z_\eta) + z_\tau(x_\eta y_\zeta - x_\zeta y_\eta)]/J, \tag{18c}$$

$$\eta_x = -(y_\xi z_\zeta - y_\zeta z_\xi)/J, \qquad \eta_y = -(x_\xi z_\zeta - x_\zeta z_\xi)/J, \qquad \eta_z = -(x_\xi y_\zeta - x_\zeta y_\xi)/J, \tag{18d}$$

$$\eta_t = [x_\tau(y_\xi z_\zeta - y_\zeta z_\xi) - y_\tau(x_\xi z_\zeta - x_\zeta z_\xi) + z_\tau(x_\xi y_\zeta - x_\zeta y_\xi)]/J, \tag{18e}$$

$$\zeta_x = (y_\xi z_\eta - y_\eta z_\xi)/J, \qquad \zeta_y = -(x_\xi z_\eta - x_\eta z_\xi)/J, \qquad \zeta_z = (x_\xi y_\eta - x_\eta y_\xi)/J, \tag{18f}$$

$$\zeta_t = -[x_\tau(y_\xi z_\eta - y_\eta z_\xi) - y_\tau(x_\xi z_\eta - x_\eta z_\xi) + z_\tau(x_\xi y_\eta - x_\eta y_\xi)]/J, \tag{18g}$$

where J is the Jacobian given by

$$J = x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi). \tag{18h}$$

The derivative terms x_ξ, x_η, x_ζ, y_ξ, y_η, y_ζ, z_ξ, z_η and z_ζ in equation (18) can be evaluated *either* analytically by differentiating equation (10) or numerically by using finite difference formulae. The correct way to evaluate these derivatives depends upon how the governing equations written in the boundary-fitted co-ordinate system are cast. This important topic is addressed in References 47–50. The derivatives x_τ, y_τ and z_τ should be evaluated numerically.

The four-boundary method

The four-boundary method for generating grid points is intended for problems where four boundaries of a spatial domain need to be mapped correctly. This method[37, 38] is an extension of the two-boundary method and as such consists of the same eight major steps. The eight steps of

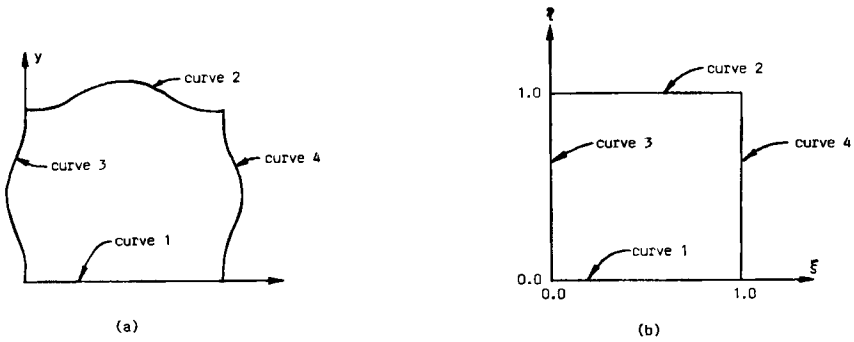

Figure 8. Spatial domain: (a) in x–y–t co-ordinate system; (b) in ξ–η–τ co-ordinate system

this method are described below by generating a single grid inside the 2D, non-deforming spatial domain shown in Figure 8(a).

Step 1: Define the co-ordinate transformation. For the 2D, non-deforming spatial domain shown in Figure 8(a) we seek a co-ordinate transformation of the form

$$(x, y, t) \leftrightarrow (\xi, \eta, \tau) \tag{19a}$$

or, more specifically,

$$t = t(\tau), \qquad x = x(\xi, \eta), \qquad y = y(\xi, \eta). \tag{19b}$$

Step 2: Select a time-stretching function. Here t is set equal to τ as shown by equation (2).

Step 3: Select four boundaries of the spatial domain. Since the spatial domain of Figure 8(a) has only four boundaries, all four boundaries are selected. We choose curves 1, 2, 3 and 4 to correspond to co-ordinate lines $\eta = 0$, $\eta = 1$, $\xi = 0$ and $\xi = 1$ respectively (Figure 8), i.e.

$$X_1 = x(\xi, \eta = 0) = X_1(\xi), \qquad Y_1 = y(\xi, \eta = 0) = Y_1(\xi), \tag{20a}$$

$$X_2 = x(\xi, \eta = 1) = X_2(\xi), \qquad Y_2 = y(\xi, \eta = 1) = Y_2(\xi), \tag{20b}$$

$$X_3 = x(\xi = 0, \eta) = X_3(\eta), \qquad Y_3 = y(\xi = 0, \eta) = Y_3(\eta), \tag{20c}$$

$$X_4 = x(\xi = 1, \eta) = X_4(\eta), \qquad Y_4 = y(\xi = 1, \eta) = Y_4(\eta). \tag{20d}$$

Here X_i and Y_i are the x- and y-co-ordinates of curve i, $i = 1, 2, 3, 4$.

Step 4: Describe the four boundaries selected in parametric form. Having selected four boundaries, we now need to represent these boundaries in parametric form as suggested by equation (20). Here tension spline[43] was used to obtain parametric equations for the four curves in terms of the parameter ξ for curves 1 and 2 and in terms of η for curves 3 and 4 (see next section).

Step 5: Map the spatial domain. The four-boundary method maps the spatial domain to the transformed domain in two steps. The first step is essentially the same as Step 5 of the two-boundary method in which we select two of the four boundaries which do not touch each other and which have been described parametrically using the same parameter (either ξ or η). As in Step 5 of the two-boundary method, curves that connect these two boundaries are specified by using Hermite transfinite interpolation. When this step is completed, the two boundaries that were selected will be mapped correctly, but the other two boundaries will in general be mapped

incorrectly. To remedy this, a second step is performed where the mapping constructed during the first step is modified so that the other two boundaries will also be mapped correctly.

Here we shall first define curves that connect curves 1 and 2 such that only curves 1 and 2 will be mapped correctly. Afterwards we will modify the connecting curves so that curves 3 and 4 will also be mapped correctly. Curves which connect curves 1 and 2 are described by the following Hermite interpolation expressions:

$$x'(\xi,\eta) = X_1(\xi)h_1(\eta) + X_2(\xi)h_2(\eta) + \frac{\partial x(\xi,\eta=0)}{\partial\eta}h_3(\eta) + \frac{\partial x(\xi,\eta=1)}{\partial\eta}h_4(\eta), \tag{21a}$$

$$y'(\xi,\eta) = Y_1(\xi)h_1(\eta) + Y_2(\xi)h_2(\eta) + \frac{\partial y(\xi,\eta=0)}{\partial\eta}h_3(\eta) + \frac{\partial y(\xi,\eta=1)}{\partial\eta}h_4(\eta), \tag{21b}$$

where h_1, h_2, h_3 and h_4 are given by equation (11). The values of the partial derivatives in equation (21) will be given shortly.

Equation (21) maps curves 1 and 2 correctly but not curves 3 and 4. In order to map curves 3 and 4 correctly, equation (21) must be adjusted as shown below:

$$x(\xi,\eta) = x'(\xi,\eta) + \Delta x(\xi,\eta), \qquad y(\xi,\eta) = y'(\xi,\eta) + \Delta y(\xi,\eta). \tag{22}$$

Here

$$\Delta x(\xi,\eta) = [x(\xi=0,\eta) - x'(\xi=0,\eta)]h_5(\xi) + [x(\xi=1,\eta) - x'(\xi=1,\eta)]h_6(\xi) + \left(\frac{\partial x(\xi=0,\eta)}{\partial\xi} - \frac{\partial x'(\xi=0,\eta)}{\partial\xi}\right)h_7(\xi) + \left(\frac{\partial x(\xi=1,\eta)}{\partial\xi} - \frac{\partial x'(\xi=1,\eta)}{\partial\xi}\right)h_8(\xi), \tag{23a}$$

$$\Delta y(\xi,\eta) = [y(\xi=0,\eta) - y'(\xi=0,\eta)]h_5(\xi) + [y(\xi=1,\eta) - y'(\xi=1,\eta)]h_6(\xi) + \left(\frac{\partial y(\xi=0,\eta)}{\partial\xi} - \frac{\partial y'(\xi=0,\eta)}{\partial\xi}\right)h_7(\xi) + \left(\frac{\partial y(\xi=1,\eta)}{\partial\xi} - \frac{\partial y'(\xi=1,\eta)}{\partial\xi}\right)h_8(\xi), \tag{23b}$$

where

$$\frac{\partial x'(\xi=0,\eta)}{\partial\xi} = h_1(\eta)\frac{\partial x(\xi=0,\eta=0)}{\partial\xi} + h_2(\eta)\frac{\partial x(\xi=0,\eta=1)}{\partial\xi} + h_3(\eta)\frac{\partial^2 x(\xi=0,\eta=0)}{\partial\xi\partial\eta} + h_4(\eta)\frac{\partial^2 x(\xi=0,\eta=1)}{\partial\xi\partial\eta}, \tag{23c}$$

$$\frac{\partial x'(\xi=1,\eta)}{\partial\xi} = h_1(\eta)\frac{\partial x(\xi=1,\eta=0)}{\partial\xi} + h_2(\eta)\frac{\partial x(\xi=1,\eta=1)}{\partial\xi} + h_3(\eta)\frac{\partial^2 x(\xi=1,\eta=0)}{\partial\xi\partial\eta} + h_4(\eta)\frac{\partial^2 x(\xi=1,\eta=1)}{\partial\xi\partial\eta}, \tag{23d}$$

$$\frac{\partial y'(\xi=0,\eta)}{\partial\xi} = h_1(\eta)\frac{\partial y(\xi=0,\eta=0)}{\partial\xi} + h_2(\eta)\frac{\partial y(\xi=0,\eta=1)}{\partial\xi} + h_3(\eta)\frac{\partial^2 y(\xi=0,\eta=0)}{\partial\xi\partial\eta} + h_4(\eta)\frac{\partial^2 y(\xi=0,\eta=1)}{\partial\xi\partial\eta}, \tag{23e}$$

$$\frac{\partial y'(\xi=1,\eta)}{\partial \xi} = h_1(\eta)\frac{\partial y(\xi=1,\eta=0)}{\partial \xi} + h_2(\eta)\frac{\partial y(\xi=1,\eta=1)}{\partial \xi} + h_3(\eta)\frac{\partial^2 y(\xi=1,\eta=0)}{\partial \xi \partial \eta} + h_4(\eta)\frac{\partial^2 y(\xi=1,\eta=1)}{\partial \xi \partial \eta}, \tag{23f}$$

$$h_5 = 2\xi^3 - 3\xi^2 + 1, \quad h_6 = -2\xi^3 + 3\xi^2, \quad h_7 = \xi^3 - 2\xi^2 + \xi, \quad h_8 = \xi^3 - \xi^2. \tag{24}$$

Substitution of equations (23) and (24) in equation (22) yields the desired expressions for $x(\xi, \eta)$ and $y(\xi, \eta)$ which describe the mapping between the 'physical' domain and the transformed domain.

We still need to specify the derivative terms in equations (21) and (23). Similar to the two-boundary method, the first-order derivative terms are chosen so that the connecting curves will intersect boundaries orthogonally. This time, however, the spatial domain is two-dimensional so that we must use the dot product instead of the cross product to specify orthogonality. Connecting curves will intersect curve 1 orthogonally when the dot product of $\mathbf{e}_\xi$ (a vector tangent to curve 1) and $\mathbf{e}_\eta$ (the vector tangent to the connecting curve) is zero. It can be shown that this will be the case when

$$\frac{\partial x(\xi,\eta=0)}{\partial \eta} = -K_1(\xi)\frac{\partial Y_1(\xi)}{\partial \xi}, \qquad \frac{\partial y(\xi,\eta=0)}{\partial \eta} = K_1(\xi)\frac{\partial X_1(\xi)}{\partial \xi}. \tag{25}$$

Following this line of reasoning, expressions for other first-order derivative terms in equations (21) and (23) are given below:

$$\frac{\partial x(\xi,\eta=1)}{\partial \eta} = -K_2(\xi)\frac{\partial Y_2(\xi)}{\partial \xi}, \qquad \frac{\partial y(\xi,\eta=1)}{\partial \eta} = K_2(\xi)\frac{\partial X_2(\xi)}{\partial \xi}, \tag{26a}$$

$$\frac{\partial x(\xi=0,\eta)}{\partial \xi} = K_3(\eta)\frac{\partial Y_3(\eta)}{\partial \eta}, \qquad \frac{\partial y(\xi=0,\eta)}{\partial \xi} = -K_3(\eta)\frac{\partial X_3(\eta)}{\partial \eta}, \tag{26b}$$

$$\frac{\partial x(\xi=1,\eta)}{\partial \xi} = K_4(\eta)\frac{\partial Y_4(\eta)}{\partial \eta}, \qquad \frac{\partial y(\xi=1,\eta)}{\partial \xi} = -K_4(\eta)\frac{\partial X_4(\eta)}{\partial \eta}. \tag{26c}$$

Here $K_1(\xi)$ and $K_2(\xi)$ were chosen to be equal to 0·3, while $K_3(\eta)$ and $K_4(\eta)$ were chosen to be equal to 0·1. Methods for determining the second-order derivative terms present in equation (23) are given in Reference 40. In our example these terms were set equal to zero.

Step 6: Discretize the domain. We discretize the domain in the ξ–η–τ co-ordinate system by replacing the temporal domain with equally incremented time levels and by replacing the spatial domain with $IL \times JL$ equally spaced grid points (Figure 9(b)). The time levels are described by equation (14). The grid points are located at (ξ_i, η_j); ξ_i and η_j are given by equation (15). By substituting equations (21), (23), (24), (25) and (26) in equation (22), we obtain the locations of the grid points in the x–y–t co-ordinate system (Figure 9(a)).

Step 7: Control the distribution of grid points. In this example we choose not to use stretching functions to redistribute the grid points within the spatial domain.

Step 8: Calculate metric coefficients. For our 2D, non-deforming spatial domain the metric coefficients which need to be evaluated are τ_t, ξ_x, η_x, ξ_y and η_y. These metric coefficients can be evaluated by using the following equations:

$$\tau_t = 1, \quad \xi_x = y_\eta/J, \quad \eta_x = -y_\xi/J, \quad \xi_y = -x_\eta/J, \quad \eta_y = x_\xi/J, \quad J = x_\xi y_\eta - x_\eta y_\xi. \tag{27}$$

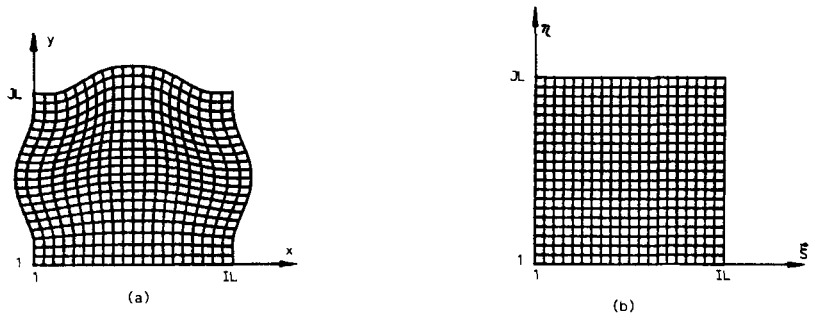

Figure 9. Grid system: (a) in x–y–t co-ordinate system; (b) in ξ–η–τ co-ordinate system

As before, the partial derivative terms in equation (27) can be evaluated either analytically or numerically by using finite difference formulae depending upon how the governing equations written in the boundary-fitted co-ordinate system are cast.

The six-boundary method

The six-boundary method for generating grid points is intended for 3D spatial domains in which six boundaries of the spatial domain need to be mapped correctly. This method[38, 39] is an extension of the two- and four-boundary methods; thus it consists of the same eight major steps. We shall illustrate the six-boundary method by generating a grid system within the deforming, spatial domain shown in Figure 10(a). Of the eight major steps involved, only Steps 3 and 5 will be described since the other steps are identical to those of the two- and four-boundary methods.

Step 3: Select six boundaries of the spatial domain. Since the spatial domain of Figure 10(a) has only six boundaries, all six boundaries are selected. We choose surfaces 1, 2, 3, 4, 5 and 6 to correspond to co-ordinate planes $\eta = 0$, $\eta = 1$, $\xi = 0$, $\xi = 1$, $\zeta = 0$ and $\zeta = 1$ respectively, i.e.

$$X_1 = x(\xi, \eta = 0, \zeta, \tau) = X_1(\xi, \zeta, \tau), \qquad Y_1 = y(\xi, \eta = 0, \zeta, \tau) = Y_1(\xi, \zeta, \tau), \tag{28a}$$

$$X_2 = x(\xi, \eta = 1, \zeta, \tau) = X_2(\xi, \zeta, \tau), \qquad Y_2 = y(\xi, \eta = 1, \zeta, \tau) = Y_2(\xi, \zeta, \tau), \tag{28b}$$

$$X_3 = x(\xi = 0, \eta, \zeta, \tau) = X_3(\eta, \zeta, \tau), \qquad Y_3 = y(\xi = 0, \eta, \zeta, \tau) = Y_3(\eta, \zeta, \tau), \tag{28c}$$

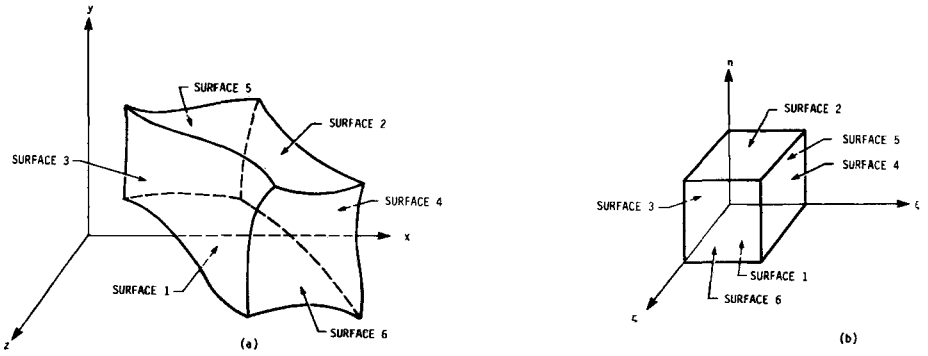

Figure 10. Spatial domain: (a) x–y–z–t co-ordinate system; (b) in ξ–η–ζ–τ co-ordinate system

$$X_4 = x(\xi = 1, \eta, \zeta, \tau) = X_4(\eta, \zeta, \tau), \qquad Y_4 = y(\xi = 1, \eta, \zeta, \tau) = Y_4(\eta, \zeta, \tau), \tag{28d}$$

$$X_5 = x(\xi, \eta, \zeta = 0, \tau) = X_5(\xi, \eta, \tau), \qquad Y_5 = y(\xi, \eta, \zeta = 0, \tau) = Y_5(\xi, \eta, \tau), \tag{28e}$$

$$X_6 = x(\xi, \eta, \zeta = 1, \tau) = X_6(\xi, \eta, \tau), \qquad Y_6 = y(\xi, \eta, \zeta = 1, \tau) = Y_6(\xi, \eta, \tau). \tag{28f}$$

Here X_i and Y_i are the x- and y-co-ordinates of surface i, $i = 1, 2, 3, 4, 5, 6$. The z-co-ordinates of surfaces 1, 2, 3, 4, 5 and 6 were omitted to shorten presentation.

Step 5: Map the spatial domain. We map the 'physical' domain to a transformed domain in three steps. In the first step, correctly map two of the six boundaries by using the two-boundary method. In the second step, correct the mapping completed in the first step by ensuring two more boundaries are mapped correctly (same as Step 5 of the four-boundary method). Finally, in the third step, correct the mapping completed in the second step by ensuring the remaining two boundaries are mapped correctly.

Surfaces 1 and 2 will be mapped correctly by the following Hermite interpolation expressions:

$$x'(\xi, \eta, \zeta, \tau) = X_1(\xi, \zeta, \tau)h_1(\eta) + X_2(\xi, \zeta, \tau)h_2(\eta) + \frac{\partial x(\xi, \eta = 0, \zeta, \tau)}{\partial \eta} h_3(\eta) + \frac{\partial x(\xi, \eta = 1, \zeta, \tau)}{\partial \eta} h_4(\eta), \tag{29a}$$

$$y'(\xi, \eta, \zeta, \tau) = Y_1(\xi, \zeta, \tau)h_1(\eta) + Y_2(\xi, \zeta, \tau)h_2(\eta) + \frac{\partial y(\xi, \eta = 0, \zeta, \tau)}{\partial \eta} h_3(\eta) + \frac{\partial y(\xi, \eta = 1, \zeta, \tau)}{\partial \eta} h_4(\eta). \tag{29b}$$

Here h_1, h_2, h_3 and h_4 are given by equation (11).

Surfaces 1, 2, 3 and 4 will be mapped correctly by the following equations:

$$x''(\xi, \eta, \zeta, \tau) = x'(\xi, \eta, \zeta, \tau) + \Delta x'(\xi, \eta, \zeta, \tau), \tag{30a}$$

$$y''(\xi, \eta, \zeta, \tau) = y'(\xi, \eta, \zeta, \tau) + \Delta y'(\xi, \eta, \zeta, \tau), \tag{30b}$$

where x' and y' are given by equation (29) and

$$\Delta x'(\xi, \eta, \zeta, \tau) = [X_3(\eta, \zeta, \tau) - x'(\xi = 0, \eta, \zeta, \tau)]h_5(\xi) + [X_4(\eta, \zeta, \tau) - x'(\xi = 1, \eta, \zeta, \tau)]h_6(\xi) + \left(\frac{\partial x(\xi = 0, \eta, \zeta, \tau)}{\partial \xi} - \frac{\partial x'(\xi = 0, \eta, \zeta, \tau)}{\partial \xi}\right)h_7(\xi) + \left(\frac{\partial x(\xi = 1, \eta, \zeta, \tau)}{\partial \xi} - \frac{\partial x'(\xi = 1, \eta, \zeta, \tau)}{\partial \xi}\right)h_8(\xi), \tag{31a}$$

$$\Delta y'(\xi, \eta, \zeta, \tau) = [Y_3(\eta, \zeta, \tau) - y'(\xi = 0, \eta, \zeta, \tau)]h_5(\xi) + [Y_4(\eta, \zeta, \tau) - y'(\xi = 1, \eta, \zeta, \tau)]h_6(\xi) + \left(\frac{\partial y(\xi = 0, \eta, \zeta, \tau)}{\partial \xi} - \frac{\partial y'(\xi = 0, \eta, \zeta, \tau)}{\partial \xi}\right)h_7(\xi) + \left(\frac{\partial y(\xi = 1, \eta, \zeta, \tau)}{\partial \xi} - \frac{\partial y'(\xi = 1, \eta, \zeta, \tau)}{\partial \xi}\right)h_8(\xi). \tag{31b}$$

In the above equations, h_5, h_6, h_7 and h_8 are given by equation (24).

All six surfaces of the spatial domain shown in Figure 10(a)—namely surfaces 1, 2, 3, 4, 5 and 6—will be mapped correctly by the following equations:

$$x(\xi, \eta, \zeta, \tau) = x''(\xi, \eta, \zeta, \tau) + \Delta x''(\xi, \eta, \zeta, \tau), \tag{32a}$$

$$y(\xi, \eta, \zeta, \tau) = y''(\xi, \eta, \zeta, \tau) + \Delta y''(\xi, \eta, \zeta, \tau), \tag{32b}$$

where x'' and y'' are given by equation (30) and

$$\begin{aligned}\Delta x''(\xi, \eta, \zeta, \tau) = {} & [X_5(\xi, \eta, \tau) - x''(\xi, \eta, \zeta = 0, \tau)]h_9(\zeta) \\ & + [X_6(\xi, \eta, \tau) - x''(\xi, \eta, \zeta = 1, \tau)]h_{10}(\zeta) \\ & + \left(\frac{\partial x(\xi, \eta, \zeta = 0, \tau)}{\partial \xi} - \frac{\partial x''(\xi, \eta, \zeta = 0, \tau)}{\partial \xi}\right)h_{11}(\zeta) \\ & + \left(\frac{\partial x(\xi, \eta, \zeta = 1, \tau)}{\partial \xi} - \frac{\partial x''(\xi, \eta, \zeta = 1, \tau)}{\partial \xi}\right)h_{12}(\zeta),\end{aligned} \tag{33a}$$

$$\begin{aligned}\Delta y''(\xi, \eta, \zeta, \tau) = {} & [Y_5(\xi, \eta, \tau) - y''(\xi, \eta, \zeta = 0, \tau)]h_9(\zeta) \\ & + [Y_6(\xi, \eta, \tau) - y''(\xi, \eta, \zeta = 1, \tau)]h_{10}(\zeta) \\ & + \left(\frac{\partial y(\xi, \eta, \zeta = 0, \tau)}{\partial \xi} - \frac{\partial y''(\xi, \eta, \zeta = 0, \tau)}{\partial \xi}\right)h_{11}(\zeta) \\ & + \left(\frac{\partial y(\xi, \eta, \zeta = 1, \tau)}{\partial \xi} - \frac{\partial y''(\xi, \eta, \zeta = 1, \tau)}{\partial \xi}\right)h_{12}(\zeta),\end{aligned} \tag{33b}$$

$$h_9 = 2\zeta^3 - 3\zeta^2 + 1, \qquad h_{10} = -2\zeta^3 + 3\zeta^2, \qquad h_{11} = \zeta^3 - 2\zeta^2 + \zeta, \qquad h_{12} = \zeta^3 - \zeta^2. \tag{34}$$

Substitution of equations (29)–(31) in equation (32) yields the desired expressions for $x(\xi, \eta, \zeta, \tau)$ and $y(\xi, \eta, \zeta, \tau)$ which describe the mapping between the 'physical' domain and the transformed domain. We still need to specify the derivative terms in equations (29), (31) and (33). Similar to the two- and four-boundary methods, the first-order derivative terms are chosen so that connecting curves will intersect boundaries orthogonally. Here all second-order derivatives were set equal to zero. Here we note that an expression for $z(\xi, \eta, \zeta, \tau)$ is also needed and can be derived in the same manner as $x(\xi, \eta, \zeta, \tau)$ and $y(\xi, \eta, \zeta, \tau)$.

Additional remarks

We conclude this section by mentioning three problems which must be dealt with when using the two-, four and six-boundary methods.

First, slope discontinuities present in the boundaries of spatial domains propagate into the interior of the grid systems generated by using these methods. These discontinuities in the slopes of the grid lines are undesirable since they can lead to errors in the solution. To correct for this, some technique should be used to smooth the grid. One way to smooth a grid system with slope discontinuity is to apply a Laplacian operator to the region near the discontinuity, as will be shown later.

Secondly, care must be taken when choosing the 'K-factors' and the stretching functions for the two-, four- and six-boundary methods. Large 'K-factors' tend to produce grid lines with more curvature. Such grid lines often overlap in the spatial domain or can at least form a grid system which is very skewed. In general, numerical values for the 'K-factors' and the stretching functions

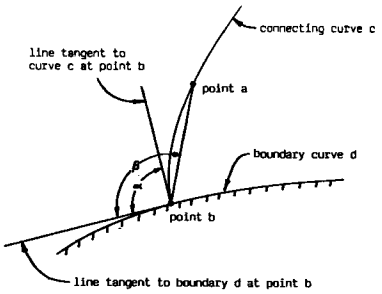

Figure 11. Diagram showing how orthogonality of a connecting curve at a boundary does not guarantee orthogonality after the connecting curve has been replaced by grid points

are arrived at in an iterative manner. A grid is first generated by using one set of inputs for the 'K-factors' and the stretching functions. Next, that grid is plotted (GRID2D/3D contains a graphics programme to plot grid systems that it generates) and inspected visually. On the basis of that inspection, the inputs are modified accordingly. This process repeats until a satisfactory grid has been obtained. Since GRID2D/3D is highly efficient, an acceptable grid system can be generated within a short time.

Thirdly, when connecting curves are discretized to form grid points, the orthogonality which was forced at the boundaries may be lost. Figure 11 illustrates this point. Between grid points a and b, curve c is approximated by line segment a–b. The original 90° angle α between boundary d and curve c has been replaced by angle β between boundary d and line segment a–b. In order for β to approximate α more closely, two things can be done. First, stretching functions can be used to move point a closer to point b. Secondly, larger 'K-factors' can be used to force the orthogonality further into the domain along curve c. In this way, orthogonality between the grid lines and the boundary curves can be maintained after the discretization of the spatial domain.

METHODS FOR GENERATING PARAMETRIC REPRESENTATION OF BOUNDARIES

In the previous section it was shown that in order to use the two-, four- and six-boundary methods, it is necessary to represent boundaries of the spatial domain in parametric form. These boundaries are curves for two-dimensional spatial domains and surfaces for three-dimensional ones.

Parametric representation of curves and surfaces

Curves and surfaces can be described mathematically in several different ways. For example, a curve in the x–y plane can be represented by $y = f(x)$ or $f(x, y) = 0$. Alternatively we can describe the same curve parametrically in terms of a parameter s by $x = g(s)$, $y = f[g(s)] = h(s)$. The choice of s is rather arbitrary, the only restriction being that s must increase monotonically along the curve. Here we note that all curves and surfaces can be represented by parametric equations and that there is no one unique way of representing a curve or surface in parametric form.

In grid generation, information about a curve or surface is given either by an analytical expression or by a set of co-ordinates which describe the locations of a finite number of discrete points on the curve or surface. When information about a curve or surface is provided in the form of an analytical expression such as $y = f(x)$, it is a straightforward matter to generate a set of parametric equations for the curve or surface. When information about a curve or surface is given

by a finite number of discrete points, then some type of interpolation schemes must first be used to approximate the curve or surface by an analytical expression before it can be represented in parametric form.

In GRID2D/3D, two interpolation schemes can be used to generate curves in 2D and 3D, namely cubic spline interpolation[42] and tension spline interpolation.[43] Tension spline interpolation should be used whenever the curve to be approximated possesses extreme curvature. GRID2D/3D approximates 3D surfaces by using a new technique, developed in this study, referred to as 3D bidirectional Hermite interpolation. Here we only present the 3D bidirectional Hermite interpolation for generating 3D surfaces.

Three-dimensional bidirectional Hermite interpolation

It turns out that linear Coons' interpolation (also known as transfinite bilinear interpolation) sometimes does not produce satisfactory surface approximations. Also, if a boundary surface is broken up into two or more subsurfaces with each subsurface generated by transfinite bilinear interpolation, then that boundary surface will have discontinuous first-order derivatives at all interfaces where different subsurfaces meet. Here a new technique for generating surfaces, referred to as 3D bidirectional Hermite interpolation, has been developed which is as efficient as linear Coons' surface but does not have its shortcomings.

To illustrate the 3D bidirectional Hermite interpolation, consider the boundary surface bounded by four twisted curves shown in Figure 12. Information about the four twisted curves may be given in analytical form, or they may be put into analytical form from co-ordinates of a set of discrete points located on the curves by using either cubic spline or tension spline interpolation. In either case we end up with parametric equations describing each of the curves in the following form:

$$X_i = X_i(s_i), \qquad Y_i = Y_i(s_i), \qquad Z_i = Z_i(s_i), \tag{35}$$

where X_i, Y_i and Z_i describe curve i, $i = 1, 2, 3, 4$. The parameter s_i represents the approximate arc length along curve i. Recall that for the two-, four- and six-boundary methods, boundary surfaces in the spatial domain are mapped onto co-ordinate planes in the transformed domain. Here we map the boundary surface shown in Figure 12 onto the ξ–η co-ordinate plane located at $\zeta = 0$ in which both ξ and η varied between zero and unity. Thus s_i in equation (35) must be related to either ξ or η, depending upon i.

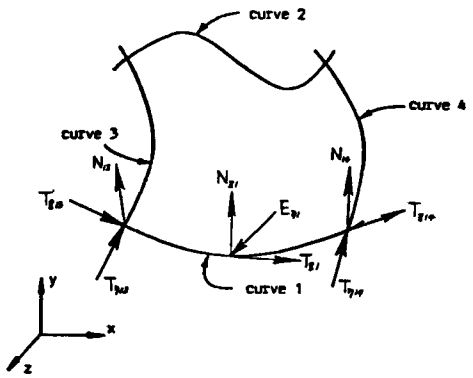

Figure 12. A boundary surface bounded by four twisted curves in the x–y–z–t co-ordinate system along with definitions of vectors tangent and orthogonal to curve 1

By using the four-boundary method with Hermite interpolants, we obtain

$$x(\xi, \eta, \zeta = 0) = x'_{12}(\xi, \eta, \zeta = 0) + \Delta x(\xi, \eta, \zeta = 0), \tag{36a}$$

$$y(\xi, \eta, \zeta = 0) = y'_{12}(\xi, \eta, \zeta = 0) + \Delta y(\xi, \eta, \zeta = 0), \tag{36b}$$

$$z(\xi, \eta, \zeta = 0) = z'_{12}(\xi, \eta, \zeta = 0) + \Delta z(\xi, \eta, \zeta = 0), \tag{36c}$$

where

$$\begin{aligned} x'_{12}(\xi, \eta, \zeta = 0) = {} & X_1(\xi)h_1(\eta) + X_2(\xi)h_2(\eta) \\ & + \frac{\partial x(\xi, \eta = 0, \zeta = 0)}{\partial \eta} h_3(\eta) + \frac{\partial x(\xi, \eta = 1, \zeta = 0)}{\partial \eta} h_4(\eta), \end{aligned} \tag{36d}$$

$$\begin{aligned} y'_{12}(\xi, \eta, \zeta = 0) = {} & Y_1(\xi)h_1(\eta) + Y_2(\xi)h_2(\eta) \\ & + \frac{\partial y(\xi, \eta = 0, \zeta = 0)}{\partial \eta} h_3(\eta) + \frac{\partial y(\xi, \eta = 1, \zeta = 0)}{\partial \eta} h_4(\eta), \end{aligned} \tag{36e}$$

$$\begin{aligned} z'_{12}(\xi, \eta, \zeta = 0) = {} & Z_1(\xi)h_1(\eta) + Z_2(\xi)h_2(\eta) \\ & + \frac{\partial z(\xi, \eta = 0, \zeta = 0)}{\partial \eta} h_3(\eta) + \frac{\partial z(\xi, \eta = 1, \zeta = 0)}{\partial \eta} h_4(\eta), \end{aligned} \tag{36f}$$

$$\begin{aligned} \Delta x(\xi, \eta, \zeta = 0) = {} & [x(\xi = 0, \eta, \zeta = 0) - x'_{12}(\xi = 0, \eta, \zeta = 0)]h_5(\xi) \\ & + [x(\xi = 1, \eta, \zeta = 0) - x'_{12}(\xi = 1, \eta, \zeta = 0)]h_6(\xi) \\ & + \left(\frac{\partial x(\xi = 0, \eta, \zeta = 0)}{\partial \xi} - \frac{\partial x'_{12}(\xi = 0, \eta, \zeta = 0)}{\partial \xi}\right) h_7(\xi) \\ & + \left(\frac{\partial x(\xi = 1, \eta, \zeta = 0)}{\partial \xi} - \frac{\partial x'_{12}(\xi = 1, \eta, \zeta = 0)}{\partial \xi}\right) h_8(\xi), \end{aligned} \tag{36g}$$

$$\begin{aligned} \Delta y(\xi, \eta, \zeta = 0) = {} & [y(\xi = 0, \eta, \zeta = 0) - y'_{12}(\xi = 0, \eta, \zeta = 0)]h_5(\xi) \\ & + [y(\xi = 1, \eta, \zeta = 0) - y'_{12}(\xi = 1, \eta, \zeta = 0)]h_6(\xi) \\ & + \left(\frac{\partial y(\xi = 0, \eta, \zeta = 0)}{\partial \xi} - \frac{\partial y'_{12}(\xi = 0, \eta, \zeta = 0)}{\partial \xi}\right) h_7(\xi) \\ & + \left(\frac{\partial y(\xi = 1, \eta, \zeta = 0)}{\partial \xi} - \frac{\partial y'_{12}(\xi = 1, \eta, \zeta = 0)}{\partial \xi}\right) h_8(\xi), \end{aligned} \tag{36h}$$

$$\begin{aligned} \Delta z(\xi, \eta, \zeta = 0) = {} & [z(\xi = 0, \eta, \zeta = 0) - z'_{12}(\xi = 0, \eta, \zeta = 0)]h_5(\xi) \\ & + [z(\xi = 1, \eta, \zeta = 0) - z'_{12}(\xi = 1, \eta, \zeta = 0)]h_6(\xi) \\ & + \left(\frac{\partial z(\xi = 0, \eta, \zeta = 0)}{\partial \xi} - \frac{\partial z'_{12}(\xi = 0, \eta, \zeta = 0)}{\partial \xi}\right) h_7(\xi) \\ & + \left(\frac{\partial z(\xi = 1, \eta, \zeta = 0)}{\partial \xi} - \frac{\partial z'_{12}(\xi = 1, \eta, \zeta = 0)}{\partial \xi}\right) h_8(\xi), \end{aligned} \tag{36i}$$

$$\frac{\partial x'_{12}(\xi=0,\eta,\zeta=0)}{\partial\xi} = h_1(\eta)\frac{\partial x(\xi=0,\eta=0,\zeta=0)}{\partial\xi} + h_2(\eta)\frac{\partial x(\xi=0,\eta=1,\zeta=0)}{\partial\xi} + h_3(\eta)\frac{\partial^2 x(\xi=0,\eta=0,\zeta=0)}{\partial\xi\,\partial\eta} + h_4(\eta)\frac{\partial^2 x(\xi=0,\eta=1,\zeta=0)}{\partial\xi\,\partial\eta}, \tag{36j}$$

$$\frac{\partial x'_{12}(\xi=1,\eta,\zeta=0)}{\partial\xi} = h_1(\eta)\frac{\partial x(\xi=1,\eta=0,\zeta=0)}{\partial\xi} + h_2(\eta)\frac{\partial x(\xi=1,\eta=1,\zeta=0)}{\partial\xi} + h_3(\eta)\frac{\partial^2 x(\xi=1,\eta=0,\zeta=0)}{\partial\xi\,\partial\eta} + h_4(\eta)\frac{\partial^2 x(\xi=1,\eta=1,\zeta=0)}{\partial\xi\,\partial\eta}, \tag{36k}$$

$$\frac{\partial y'_{12}(\xi=0,\eta,\zeta=0)}{\partial\xi} = h_1(\eta)\frac{\partial y(\xi=0,\eta=0,\zeta=0)}{\partial\xi} + h_2(\eta)\frac{\partial y(\xi=0,\eta=1,\zeta=0)}{\partial\xi} + h_3(\eta)\frac{\partial^2 y(\xi=0,\eta=0,\zeta=0)}{\partial\xi\,\partial\eta} + h_4(\eta)\frac{\partial^2 y(\xi=0,\eta=1,\zeta=0)}{\partial\xi\,\partial\eta}, \tag{36l}$$

$$\frac{\partial y'_{12}(\xi=1,\eta,\zeta=0)}{\partial\xi} = h_1(\eta)\frac{\partial y(\xi=1,\eta=0,\zeta=0)}{\partial\xi} + h_2(\eta)\frac{\partial y(\xi=1,\eta=1,\zeta=0)}{\partial\xi} + h_3(\eta)\frac{\partial^2 y(\xi=1,\eta=0,\zeta=0)}{\partial\xi\,\partial\eta} + h_4(\eta)\frac{\partial^2 y(\xi=1,\eta=1,\zeta=0)}{\partial\xi\,\partial\eta}, \tag{36m}$$

$$\frac{\partial z'_{12}(\xi=0,\eta,\zeta=0)}{\partial\xi} = h_1(\eta)\frac{\partial z(\xi=0,\eta=0,\zeta=0)}{\partial\xi} + h_2(\eta)\frac{\partial z(\xi=0,\eta=1,\zeta=0)}{\partial\xi} + h_3(\eta)\frac{\partial^2 z(\xi=0,\eta=0,\zeta=0)}{\partial\xi\,\partial\eta} + h_4(\eta)\frac{\partial^2 z(\xi=0,\eta=1,\zeta=0)}{\partial\xi\,\partial\eta}, \tag{36n}$$

$$\frac{\partial z'_{12}(\xi=1,\eta,\zeta=0)}{\partial\xi} = h_1(\eta)\frac{\partial z(\xi=1,\eta=0,\zeta=0)}{\partial\xi} + h_2(\eta)\frac{\partial z(\xi=1,\eta=1,\zeta=0)}{\partial\xi} + h_3(\eta)\frac{\partial^2 z(\xi=1,\eta=0,\zeta=0)}{\partial\xi\,\partial\eta} + h_4(\eta)\frac{\partial^2 z(\xi=1,\eta=1,\zeta=0)}{\partial\xi\,\partial\eta}. \tag{36o}$$

In the above equations, h_1, h_2, h_3 and h_4 are given by equation (11) and h_5, h_6, h_7 and h_8 are given by equation (24). The method for evaluating the partial derivative terms appearing on the right-hand sides of equation (36) is described below.

The first-order partial derivative terms in equation (36) determine the shape of the surface shown in Figure 12 bounded by the four twisted curves given by equation (35). These derivative terms also determine whether grid lines on that surface will intersect the four twisted curves orthogonally or not. Since the only information we have about the surface shown in Figure 12 is

the four twisted curves, these curves are used to derive expressions for the first-order derivative terms in equation (36).

We shall illustrate how first-order derivative terms in equation (36) are calculated by deriving expressions for $\partial x(\xi, \eta = 0, \zeta = 0)/\partial\eta$, $\partial y(\xi, \eta = 0, \zeta = 0)/\partial\eta$ and $\partial z(\xi, \eta = 0, \zeta = 0)/\partial\eta$. These three derivative terms are evaluated along curve 1 in which $\eta = 0, \zeta = 0$ and ξ varies between zero and unity. The method for deriving first-order derivative terms along the other three twisted curves (i.e. curves 2, 3 and 4 in Figure 12) will be similar to the procedure described below.

Expressions for the first-order partial derivative terms along curve 1 are derived in five steps: (1) determine a vector which is perpendicular to the plane formed by the vectors tangent to curves 1 and 3 at one end of curve 1; (2) determine a vector which is perpendicular to the plane formed by the vectors tangent to curves 1 and 4 at the other end of curve 1; (3) linearly interpolate between the two vectors determined in Steps 1 and 2 and denote this vector function as $\mathbf{N}_{\xi 1}$; (4) determine a vector function that is parallel to the cross product of $\mathbf{N}_{\xi 1}$ and a vector function tangent to curve 1 (the vector function thus determined is assumed to be tangent to the surface that we wish to approximate along curve 1); (5) determine first-order partial derivative terms along curve 1 by forcing the vector function formed by the first-order derivatives to be parallel to the vector function determined in Step 4. The details of these five steps are described below.

Step 1: Determine normal vector at one end of curve 1. The vectors tangent to curves 1 and 3 at $\xi = 0$ and $\eta = 0$ are given by

$$\mathbf{T}_{\xi 13} = \frac{\partial X_1(\xi = 0)}{\partial \xi}\mathbf{I} + \frac{\partial Y_1(\xi = 0)}{\partial \xi}\mathbf{J} + \frac{\partial Z_1(\xi = 0)}{\partial \xi}\mathbf{K}, \tag{37a}$$

$$\mathbf{T}_{\eta 31} = \frac{\partial X_3(\eta = 0)}{\partial \eta}\mathbf{I} + \frac{\partial Y_3(\eta = 0)}{\partial \eta}\mathbf{J} + \frac{\partial Z_3(\eta = 0)}{\partial \eta}\mathbf{K}, \tag{37b}$$

where **I**, **J** and **K** are unit vectors pointing in the x-, y- and z-directions respectively.

The vector perpendicular to the plane formed by the above two vectors is given by

$$\begin{aligned}\mathbf{N}_{13} &= \mathbf{T}_{\xi 13} \times \mathbf{T}_{\eta 31} \\ &= \left(\frac{\partial Y_1(\xi = 0)}{\partial \xi}\frac{\partial Z_3(\eta = 0)}{\partial \eta} - \frac{\partial Z_1(\xi = 0)}{\partial \xi}\frac{\partial Y_3(\xi = 0)}{\partial \eta}\right)\mathbf{I} \\ &\quad - \left(\frac{\partial X_1(\xi = 0)}{\partial \xi}\frac{\partial Z_3(\eta = 0)}{\partial \eta} - \frac{\partial Z_1(\xi = 0)}{\partial \xi}\frac{\partial X_3(\eta = 0)}{\partial \eta}\right)\mathbf{J} \\ &\quad + \left(\frac{\partial X_1(\xi = 0)}{\partial \xi}\frac{\partial Y_3(\eta = 0)}{\partial \eta} - \frac{\partial Y_1(\xi = 0)}{\partial \xi}\frac{\partial X_3(\xi = 0)}{\partial \eta}\right)\mathbf{K}.\end{aligned} \tag{38}$$

Step 2: Determine normal vector at other end of curve 1. The vectors tangent to curves 1 and 4 at $\xi = 1$ and $\eta = 0$ are given by

$$\mathbf{T}_{\xi 14} = \frac{\partial X_1(\xi = 1)}{\partial \xi}\mathbf{I} + \frac{\partial Y_1(\xi = 1)}{\partial \xi}\mathbf{J} + \frac{\partial Z_1(\xi = 1)}{\partial \xi}\mathbf{K}, \tag{39a}$$

$$\mathbf{T}_{\eta 41} = \frac{\partial X_4(\eta = 0)}{\partial \eta}\mathbf{I} + \frac{\partial Y_4(\eta = 0)}{\partial \eta}\mathbf{J} + \frac{\partial Z_4(\eta = 0)}{\partial \eta}\mathbf{K}. \tag{39b}$$

The vector perpendicular to the plane formed by the above two vectors is given by

$$\begin{aligned}\mathbf{N}_{14} &= \mathbf{T}_{\xi 14} \times \mathbf{T}_{\eta 41} \\ &= \left(\frac{\partial Y_1(\xi=1)}{\partial \xi}\frac{\partial Z_4(\eta=0)}{\partial \eta} - \frac{\partial Z_1(\xi=1)}{\partial \xi}\frac{\partial Y_4(\xi=0)}{\partial \eta}\right)\mathbf{I} \\ &\quad - \left(\frac{\partial X_1(\xi=1)}{\partial \xi}\frac{\partial Z_4(\eta=0)}{\partial \eta} - \frac{\partial Z_1(\xi=1)}{\partial \xi}\frac{\partial X_4(\eta=0)}{\partial \eta}\right)\mathbf{J} \\ &\quad + \left(\frac{\partial X_1(\xi=1)}{\partial \xi}\frac{\partial Y_4(\eta=0)}{\partial \eta} - \frac{\partial Y_1(\xi=1)}{\partial \xi}\frac{\partial X_4(\xi=0)}{\partial \eta}\right)\mathbf{K}. \end{aligned} \tag{40}$$

Step 3: Linearly interpolate between two normal vectors. In this step we linearly interpolate between the two normal vectors obtained in Steps 1 and 2 to produce the following vector function:

$$\mathbf{N}_{\xi 1} = (1-\xi)\mathbf{N}_{13} + \xi \mathbf{N}_{14}, \tag{41}$$

where $\mathbf{N}_{13}$ and $\mathbf{N}_{14}$ are given by equations (38) and (40).

Step 4: Determine a vector function tangent to the surface. Since there are an infinite number of surfaces that can pass through any given curve, a strategy must be developed to determine which surface is to approximate the surface shown in Figure 12. Here the surface selected is the one that will pass through curve 1 as well as curves 3 and 4. Thus the vector function tangent to the surface at curve 1 can be approximated by

$$\mathbf{E}_{\eta 1} = \mathbf{T}_{\xi 1} \times \mathbf{N}_{\xi 1}, \tag{42}$$

where $\mathbf{N}_{\xi 1}$ is given by equation (41) and $\mathbf{T}_{\xi 1}$ is a vector function tangent to curve 1 given by

$$\mathbf{T}_{\xi 1} = \frac{\partial X_1(\xi)}{\partial \xi}\mathbf{I} + \frac{\partial Y_1(\xi)}{\partial \xi}\mathbf{J} + \frac{\partial Z_1(\xi)}{\partial \xi}\mathbf{K}. \tag{43}$$

Step 5: Determine expressions for first-order derivatives. Now that we know along which surface partial derivatives with respect to η at curve 1 can be made, we can derive expressions for the first-order partial derivative terms $\partial x(\xi, \eta = 0, \zeta = 0)/\partial \eta$, $\partial y(\xi, \eta = 0, \zeta = 0)/\partial \eta$ and $\partial z(\xi, \eta = 0, \zeta = 0)/\partial \eta$ that appear in equation (36). Since we desire grid lines on the surface to be perpendicular to the boundary curve (curve 1 in this case), the vector function tangent to the η-direction, $\mathbf{T}_{\eta 1}$, must be parallel to the vector function $\mathbf{E}_{\eta 1}$ derived in Step 4. This can be expressed mathematically as

$$\mathbf{T}_{\eta 1} \times \mathbf{E}_{\eta 1} = \mathbf{0}, \tag{44a}$$

where

$$\mathbf{T}_{\eta 1} = \frac{\partial x(\xi, \eta = 0, \zeta = 0)}{\partial \eta}\mathbf{I} + \frac{\partial y(\xi, \eta = 0, \zeta = 0)}{\partial \eta}\mathbf{J} + \frac{\partial z(\xi, \eta = 0, \zeta = 0)}{\partial \eta}\mathbf{K}. \tag{44b}$$

The desired expressions for the first-order partial derivative terms are obtained from equation (44) in a manner very similar to the derivation of equation (12). Again, the 'K-factors' which appear in equation (12) can also be used here to ensure that grid lines do not overlap each other and to create a smoother surface.

GENERATION OF SINGLE AND COMPOSITE GRIDS

In the previous sections the details of the algebraic grid generation methods used in GRID2D/3D have been described. In this section we demonstrate the usefulness of GRID2D/3D by using it to generate a number of single and composite grids within complex-shaped 2D and 3D spatial domains. Recall that a single grid is a grid system based on one boundary-fitted co-ordinate system and a composite grid is a grid system made up of two or more single grids patched together.

Single grids

While illustrating the two-, four- and six-boundary methods, several single grids were generated (Figures 6, 7 and 9). In this subsection we present a brief discussion on how to smooth discontinuities in grid systems that arise from boundary discontinuities.

Figure 13(a) shows a 2D single grid around a sharp bend generated by using the two-boundary method. This figure illustrates how boundary discontinuities can propagate into the interior of the grid. The slope discontinuity of the η-grid lines along $i = I_1$ can be eliminated by applying the following equations:

$$y_{i,j}^{n+1} = y_{i,j}^{n} + \alpha(y_{i+1,j}^{n} - 2y_{i,j}^{n} + y_{i-1,j}^{n}), \tag{45a}$$

$$x_{i,j}^{n+1} = x_{i,j}^{n} + \alpha(x_{i+1,j}^{n} - 2x_{i,j}^{n} + x_{i-1,j}^{n}), \tag{45b}$$

where α is a constant ($\leq 0{\cdot}5$), $i = I_1 - m, \ldots, I_1, \ldots, I_1 + m$ (m can be zero or some positive integer constant) and $j = 2, 3, \ldots, JL - 1$. In order to smooth the grid shown in Figure 13(a), equation (45) needs to be applied a number of times, i.e. $n = 0, 1, 2, 3, \ldots$ with $n = 0$ being the grid shown in Figure 13(a), $n = 1$ being the first correction, $n = 2$ being the second correction and so on. By using equation (45) repeatedly, the grid shown in Figure 13(a) was smoothed as shown in Figure 13(b).

Composite grids

GRID2D/3D can generate composite grids which are completely discontinuous, partially discontinuous and partially continuous (Figure 2). For completely or partially discontinuous composite grids such as those shown in Figures 2(a) and 2(b), each single grid within the composite grid can be different from any other in structure and in the number of grid points. Thus each single grid within such composite grids can be generated independently of the other single grids in the manner described in the section 'Two-, four- and six-boundary methods'. How the

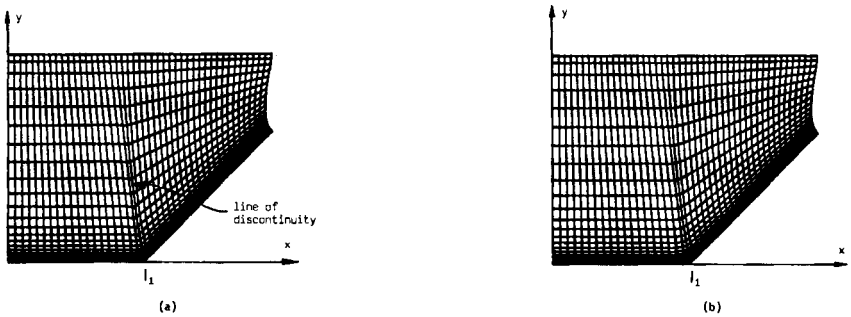

Figure 13. Spatial domain with a boundary discontinuity: (a) grid system generated by the two-boundary method; (b) grid system after smoothing

different single grids of such composite grids are patched together once they are generated is described in the section 'Types of grid systems'. Since discontinuous composite grids can be generated rather easily and patching is trivial, no further discussions concerning them will be given.

For partially continuous composite (PCC) grids, each single grid within them must satisfy a number of compatibility conditions so that when the different single grids are patched together the resultant composite grid will have some degree of continuity (see section 'Types of grid systems'). PCC grids such as the one shown in Figure 2(d) are often difficult to generate. However, when it is possible to generate them, they are the easiest to use with FD and FV methods to obtain solutions to partial differential equations. Below we discuss how PCC grids can be generated.

Partially continuous composite grids

PCC grids are generated by GRID2D/3D in two major steps. The first step involves partitioning the spatial domain into zones. The second step involves constructing grid systems that will be continuous from one zone to another. These two steps are described in detail below.

Partitioning. The first step in constructing a PCC grid is to partition the spatial domain of interest into a finite number of contiguous zones. The partitioning process involves answering three interrelated questions: (1) How should the spatial domain be partitioned into zones? (2) Based on that partition, what grid structure is to be used within each zone (e.g. C–C, C–H, O–H, . . .)? (3) Based on that partition and grid structure selections, how should the different zones be mapped to the transformed domain so that the resultant composite grid will have some degree of continuity? The answers to these questions depend on the geometry of the spatial domain and the physics of the problem for which the grid is being generated. For any given problem, several different choices are usually possible. However, for simplicity, the choice containing the least number of zones is often the most desirable.

An example illustrating one strategy for partitioning is shown in Figure 3(a). The boundaries of that spatial domain contain a backward-facing step, a flat wall and a wedge-shaped obstacle. The fluid is flowing from left to right. In Figure 3 the spatial domain is partitioned by using free streamline theory. More specifically, a new zone is set up wherever separation is expected (e.g. at the backward step and at the rear of the wedge-shaped obstacle) or where a flow will separate into two streams (e.g. at the nose of the wedge-shaped obstacle). The structure of the grid within each zone in Figure 3(a) was chosen to be H-type. This structure aligns the main flow direction with the grid lines, which is important because it reduces dissipation error and permits the use of the thin layer Navier–Stokes equations. Also, this structure allows grid lines to be clustered near solid wall boundaries readily as well as allowing continuity of grid lines and their derivatives from one zone to another. Figure 3(b) shows how the different zones in Figure 3(a), each of which will have a different boundary-fitted co-ordinate system, are mapped onto the same transformed domain.

As a second example, consider the 2D inlet of an aircraft engine shown in Figure 14(a). This geometry would be awkward to handle with a single grid owing to the separation of the internal flow from the external flow by the cowl. Figures 14(a) and (14b) show a partitioning of this spatial domain into two zones and the transformed domain to which it is mapped. Other configurations involving more than two zones are certainly possible, but more complicated configurations are unnecessary unless the physics of the flow dictates their use.

Grid generation with patching. Once we have partitioned the spatial domain of interest into zones, we are ready to generate a grid for each zone. For partially continuous composite (PCC)

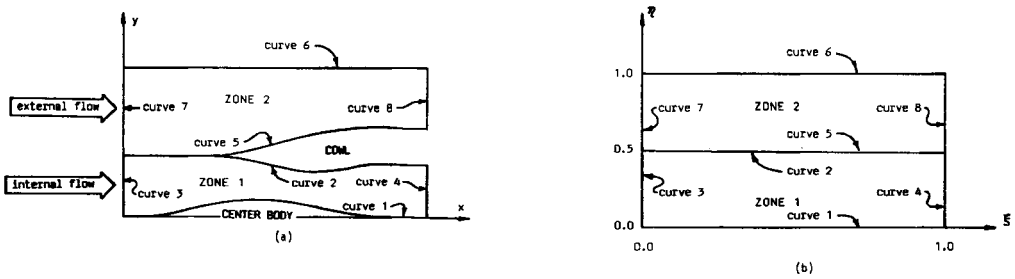

Figure 14. Inlet of a turbojet engine; (a) partitioning in the x–y–t co-ordinate system; (b) partitioning in the ξ–η–τ co-ordinate system

grids, the grid generated for each zone must be such that when they are patched together the resultant composite grid has some degree of continuity. It turns out that this requirement necessitates a few minor modifications in the two-, four- and six-boundary methods presented earlier. In this section these modifications are described in the framework of the two-boundary method by applying it to generate a PCC grid in the 2D inlet of the aircraft engine shown in Figure 14(a).

Step 1: Define the co-ordinate transformation. We seek a co-ordinate transformation of the form given by equation (19).

Step 2: Select a time-stretching function. τ is set equal to t according to equation (2).

Step 3: Select two boundaries of each zone. We choose curves 1 and 2 for zone 1 and curves 5 and 6 for zone 2 (Figure 14). For each zone we map these two curves to co-ordinate lines $\eta = \eta_{\text{low}}$ and $\eta = \eta_{\text{high}}$ respectively. Here η_{low}, η_{high}, ξ_{low} and ξ_{high} are defined as the co-ordinate lines in the transformed domain which correspond to the boundaries of a zone. Thus for zone 1 we have

$$X_1 = x(\xi, \eta = \eta_{\text{low}}) = X_1(\xi), \qquad Y_1 = y(\xi, \eta = \eta_{\text{low}}) = Y_1(\xi), \tag{46a}$$

$$X_2 = x(\xi, \eta = \eta_{\text{high}}) = X_2(\xi), \qquad Y_2 = y(\xi, \eta = \eta_{\text{high}}) = Y_2(\xi), \tag{46b}$$

where $\eta_{\text{low}} = 0$, $\eta_{\text{high}} = 0{\cdot}5$ and X_i and Y_i are the x- and y-co-ordinates of curve i. For zone 2 we have

$$X_5 = x(\xi, \eta = \eta_{\text{low}}) = X_5(\xi), \qquad Y_5 = y(\xi, \eta = \eta_{\text{low}}) = Y_5(\xi), \tag{47a}$$

$$X_6 = x(\xi, \eta = \eta_{\text{high}}) = X_6(\xi), \qquad Y_6 = y(\xi, \eta = \eta_{\text{high}}) = Y_6(\xi), \tag{47b}$$

where $\eta_{\text{low}} = 0{\cdot}5$, $\eta_{\text{high}} = 1$ and X_i and Y_i are the x- and y-co-ordinates of curve i. The other two curves in each zone (curves 3 and 4 in zone 1 and curves 7 and 8 in zone 2) are mapped to co-ordinate lines $\xi = \xi_{\text{low}}$ and $\xi = \xi_{\text{high}}$ in the transformed domain. For our example, $\xi_{\text{low}} = 0$ and $\xi_{\text{high}} = 1$ for both zones 1 and 2.

Step 4: Describe the two boundaries of each zone in parametric form. We now need to represent the four curves selected in Step 3 (two for each zone) in parametric form. Here tension spline interpolation[43] is used.

Step 5: Define curves that connect the two boundaries in each zone. $x(\xi, \eta)$ and $y(\xi, \eta)$ in each zone can be defined by performing two mappings. The first mapping is from (x, y) to (ξ', η') with the boundaries of both ξ' and η' between zero and unity. The second mapping is from (ξ', η') to (ξ, η) with the boundaries of ξ between $\xi = \xi_{\text{low}}$ and $\xi = \xi_{\text{high}}$ and the boundaries of η between

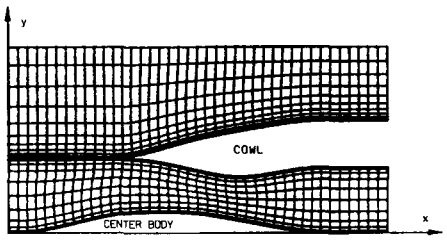

Figure 15. Grid system in the x–y–t co-ordinate system after stretching

$\eta = \eta_{low}$ and $\eta = \eta_{high}$. The first mapping can be accomplished by using the equations derived in the section 'Two-, four- and six-boundary methods' with ξ and η replaced by ξ' and η'. The second mapping is given by $\xi = \xi_{low} + \xi'(\xi_{high} - \xi_{low})$ and $\eta = \eta_{low} + \eta'(\eta_{high} - \eta_{low})$.

Step 6: Discretize the domain. We discretize the domain in the ξ–η–τ co-ordinate system by replacing the temporal domain with equally incremented time levels and by replacing each zone of the spatial domain with equally spaced grid points (see equations (14) and (15)).

Step 7: Control the distribution of grid points in each zone. For high-speed flows through the inlet we expect large velocity gradients next to all solid surfaces. Thus grid points need to be clustered near solid surfaces. With this in mind we use stretching functions to cluster grid points near curves 1 and 2 for zone 1 and near curve 5 for zone 2. The new distribution of grid points is shown in Figure 15.

Here we make a few comments about the continuity of grid lines and grid spacings across zonal interfaces. In the above example, grid lines remained continuous along the portion of the zonal interface which does not represent a 'physical' boundary. This is a desirable situation since it simplifies the numerical algorithm which will be used to investigate the flow in the spatial domain. Proper alignment of grid lines at the zonal interface in this case is made possible by using an appropriate stretching function in each zone. Thus one should exercise care when constructing a grid to ensure that grid lines remain continuous across the sections of the zonal interfaces which touch each other in the spatial domain. For our example we also note that the grid spacing in the η-direction must not change too abruptly across the zonal interface. The grid spacing in the η-direction is defined as the distance between adjacent grid points along a grid line of constant ξ. As with grid alignment, proper grid spacing at the zonal interface depends upon using an appropriate stretching function in each zone.

Step 8: Calculate metric coefficients. For the spatial domain shown in Figure 14, the five metric coefficients which need to be evaluated are τ_t, ξ_x, η_x, ξ_y and η_y. These metric coefficients can be determined by using equation (27). We note that one-sided differencing should be used when calculating partial derivative terms at grid points which lie along portions of the zonal interface which do not touch in the spatial domain. Central differencing can be used at the other grid point locations along the zonal interface.

ACKNOWLEDGEMENT

This research was supported by NASA grants NAG3-929 and NAG3-997. The authors are grateful to NASA Lewis Research Center for this support.

REFERENCES

1. R. E. Smith (ed.), *Numerical Grid Generation Techniques, NASA CP-2166*, 1980.
2. J. F. Thompson (ed.), *Numerical Grid Generation*, Elsevier, New York, 1982.
3. 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).
4. I. Babuška, J. Chandra and J. E. Flaherty (eds), *Adaptive Computational Methods for Partial Differential Equations*, SIAM, Philadelphia, 1983.
5. K. N. Ghia and U. Ghia (eds), *Advances in Grid Generation, FED Vol. 5*, ASME, New York, 1983.
6. J. F. Thompson, 'Grid generation techniques in computational fluid dynamics', *AIAA J.*, **22**, 1505–1523 (1984).
7. P. Eiseman, 'Grid generation for fluid mechanics computations', *Ann. Review of Fluid Mech.*, **17**, 487–522 (1985).
8. J. F. Thompson, Z. U. A. Warsi and C. W. Mastin, *Numerical Grid Generation*, Elsevier, New York, 1985.
9. P. R. Eiseman and G. Erlebacher, 'Grid generation for the solution of partial differential equations', *ICASE Report No. 87-57*, NASA Langley Research Center, Hampton, VA, 1987; also *NASA CR-178365*, 1987.
10. S. Sengupta, J. Hauser, P. R. Eiseman and J. F. Thompson (eds), *Numerical Grid Generation in Computational Fluid Mechanics '88*, Pineridge Press, Swansea, 1988.
11. A. Jameson and D. Mavriplis, 'Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh', *AIAA Paper 85-0435*, 1985.
12. D. Mavriplis and A. Jameson, 'Multigrid solution of the two-dimensional Euler equations on unstructured triangular meshes', *AIAA Paper 87-0353*, 1987.
13. J. A. Desideri and A. Dervieux, 'Compressible flow solvers using unstructured grids', *Von Karman Institute Lecture Series 1988-05*, 7–11 March 1988, pp. 1–115.
14. S. R. Allmaras and M. B. Giles, 'A second order flux split scheme for the unsteady 2-D Euler equations on arbitrary meshes', *AIAA Paper 87-1119*, 1987.
15. D. J. Mavriplis, 'Accurate multigrid solution of the Euler equations on unstructured and adaptive meshes', *AIAA/ASME/SIAM/APS First Natl Fluid Dynamics Congr.*, 1988.
16. T. J. Barth and D. C. Jespersen, 'The design and application of upwind schemes on unstructured meshes', *AIAA Paper 89-0366*, 1989.
17. E. H. Atta, 'Component-adaptive grid embedding', in R. E. Smith (ed.), *Numerical Grid Generation Techniques, NASA CP-2166*, 1980, pp. 157–174.
18. E. H. Atta and J. Vadyak, 'A grid overlapping scheme for flowfield computations about multicomponent configurations', *AIAA J.*, **21**, 1271–1277 (1983).
19. J. L. Steger, F. C. Dougherty and J. A. Benek, 'A chimera grid scheme', in K. N. Ghia and U. Ghia (eds), *Advances in Grid Generation, FED Vol. 5*, ASME, New York, 1983, pp. 59–69.
20. F. C. Dougherty, J. A. Benek and J. L. Steger, 'On applications of chimera grid schemes to store separation', *NASA TM-88193*, 1985.
21. J. A. Benek, T. L. Donegan and N. E. Suh, 'Extended chimera grid embedding scheme with application to viscous flows', *AIAA Paper 87-1126*, 1987.
22. M. M. Rai, 'A relaxation approach to patched-grid calculations with the Euler equations', *J. Comput. Phys.*, **66**, 99–131 (1986).
23. M. M. Rai, 'A conservative treatment of zonal boundaries for Euler equation calculations', *J. Comput. Phys.*, **62**, 472–503 (1986).
24. K. Hessenius and M. M. Rai, 'Three-dimensional, conservative, Euler computations using patched grid systems and explicit methods', *AIAA Paper 86-1081*, 1986.
25. M. M. Rai, 'Navier–Stokes simulations of rotor/stator interaction using patched and overlaid grids', *J. Propulsion Power*, **3**, 387–396 (1987).
26. P. D. Thomas, 'Composite three-dimensional grids generated by elliptic systems', *AIAA J.*, **20**, 1195–1202 (1982).
27. K. D. Lee, N. J. Y. Huang and P. E. Rubbert, 'Grid generation for general three-dimensional configurations', in R. E. Smith (ed.), *Numerical Grid Generation Techniques, NASA CP 2166*, 1980, pp. 355–366.
28. P. E. Rubbert and K. D. Lee, 'Patched coordinate systems', in J. F. Thompson (ed.), *Numerical Grid Generation*, Elsevier, New York, 1982, pp. 235–252.
29. R. L. Sorenson, 'Grid generation by elliptic partial differential equations for a tri-element augmentor-wing airfoil', in J. F. Thompson (ed.), *Numerical Grid Generation*, Elsevier, New York, 1982, pp. 653–665.
30. R. M. Coleman, 'Generation of boundary-fitted coordinate systems using segmented computational regions', in J. F. Thompson (ed.), *Numerical Grid Generation*, Elsevier, New York, 1982, pp. 633–651.
31. J. F. Thompson, 'Composite grid generation code for general 3-D regions—the EAGLE code', *AIAA J.*, **26**, 271–272 (1988).
32. S. A. Coons, 'Surfaces for computer-aided design of space forms', *MIT MACTR-41*, Cambridge, MA, 1967.
33. W. J. Gordon and C. A. Hall, 'Construction of curvilinear coordinate systems and applications to mesh generation', *Int. j. numer. methods eng.*, **7**, 461–477 (1973).
34. W. J. Cook, 'Body-oriented (natural) co-ordinates for generating three-dimensional meshes', *Int. j. numer. methods eng.*, **8**, 27–43 (1974).
35. R. E. Smith, 'Two-boundary grid generation for the solution of the three-dimensional compressible Navier–Stokes equations', *NASA TM-83123*, 1981.

36. S.-L. Yang and T. I-P. Shih, 'An algebraic grid generation technique for time-varying two-dimensional spatial domains', *Int. j. numer. methods fluids*, **6**, 291–304 (1986).
37. M. Vinokur and C. K. Lombard, 'Algebraic grid generation with corner singularity', in K. N. Ghia and U. Ghia (eds), *Advances in Grid Generation, FED Vol. 5*, ASME, New York, 1983, pp. 99–106.
38. A. Rizzi and L. E. Eriksson, 'Transfinite mesh generation and damped Euler equation algorithm for transonic flow around wing–body configurations', *AIAA Paper 81-0999*, 1981.
39. L. E. Eriksson, 'Generation of boundary-conforming grids around wing–body configurations using transfinite interpolation', *AIAA J.*, **20**, 1313–1320 (1982).
40. G. O. Roberts, 'Computational meshes for boundary layer problems', *Lecture Notes in Physics, Vol. 8*, Springer, Berlin, 1971, pp. 171–177.
41. M. Vinokur, 'On one-dimensional stretching functions for finite-difference calculations', *J. Comput. Phys.*, **50**, 215–234 (1983).
42. G. Birkhoff and H. Garabedian, 'Smooth surface interpolation', *J. Math. Phys.*, **39**, 258–268 (1960).
43. D. G. Schweikert, 'An interpolation curve using a spline in tension', *J. Math. Phys.*, **45**, 312–317 (1966).
44. C. de Boor, 'Bicubic spline interpolation', *J. Math. Phys.*, **41**, 212–218 (1962).
45. J. Ferguson, 'Multivariable curve interpolation', *J. ACM*, **11**, 221–228 (1964).
46. H. Spath, 'Algorithm 16: two-dimensional exponential splines', *Computing*, **4**, 225–233 (1969).
47. P. D. Thomas and C. K. Lombard, 'Geometric conservation law and its application to flow computations on moving grids', *AIAA J.*, **17**, 1030–1037 (1979).
48. R. G. Hindman, 'Generalized Coordinate forms of governing fluid equations and associated geometrically induced errors', *AIAA J.*, **20**, 1359–1367 (1982).
49. J. L. Steger, 'On application of body conforming curvilinear grids for finite difference solution of external flow', in J. F. Thompson (ed.), *Numerical Grid Generation*, Elsevier, New York, 1982, pp. 295–315.
50. M. Vinokur, 'An analysis of finite-difference and finite-volume formulations of conservation laws', *J. Comput. Phys.*, **81**, 1–52 (1989).