ELSEVIER

Comput. Methods Appl. Mech. Engrg. 189 (2000) 1183–1204

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Solution adaptive grid strategies based on point redistribution

Bharat K. Soni *, Roy Koomullil, David S. Thompson, Hugh Thornburg

Aerospace Engineering, NSF Engineering Research Center for Computational Field Simulation, Mississippi State University, P.O. Box 9627, Mississippi State, MS 39762, USA

Received 19 May 1999

Abstract

Solution adaptive grid strategies based on the redistribution of a fixed number of points are described in this paper. The redistribution is performed using weight functions that vary based on significant flow features. The weight functions are evaluated using an equidistribution principle. In this paper, emphasis is placed on the development of weight functions applicable to compressible flows exhibiting large scale separated vortical flows, vortex–vortex and vortex–surface interactions, separated shear layers and multiple shocks of different intensities. Algebraic, elliptic and parabolic methods of grid generation have been utilized for structured grid redistribution. Additionally, a point movement scheme is presented for generalized (structured/unstructured/hybrid) grid adaptation. Computer Aided Geometry Design (CAGD) techniques are combined with redistribution schemes to maintain the fidelity of solid boundaries. In particular, solid boundaries are represented using Non-Uniform Rational B-Splines (NURBS). A grid generation software system – Parallel Multiblock Adaptive Grid generation (PMAG) – using an elliptic redistribution scheme is also described with emphasis placed on the parallel implementation for multiblock structured grids with unstructured blocking topologies and on interpolation issues. Computational examples demonstrating the influence of different weight functions and grid redistribution strategies are presented.

1. Introduction

Rapid access to highly accurate simulation data about complex configurations is needed for multidisciplinary optimization and design. In order to efficiently meet these requirements, a closer coupling between the analysis algorithms and the discretization process is needed. In some cases, such as free surface flows, temporally varying geometries, and fluid structure interactions, the need is unavoidable. In other cases, the need is to rapidly generate high quality grids and automatically adjust these grids in response to features in the solution. Techniques utilizing unstructured grid topologies and/or solution-adaptive grids can be used to speed the grid generation process and to automatically cluster mesh points in regions of interest. Important global flow features can be significantly affected by isolated regions of inadequately resolved flow. These regions may not exhibit high gradients and may be difficult to detect.

Several approaches have been employed for both structured and unstructured grid adaptation [1,7,11,12,14,15,22,32]. The most widely used approaches utilize grid point redistribution, local grid point enrichment/derefinement, or local modification of the actual flow solver. However, the success of any one of these methods ultimately depends on the feature detection algorithm used to identify candidate regions for refinement. Typically, weight functions are constructed to mimic the local truncation error and may require substantial user input. In the case of structured grids, most problems of engineering interest involve multiblock grids and widely disparate length scales. Hence, it is desirable that the feature detection algorithm recognize flow structures of different types as well as differing intensity, and adequately address the

* Corresponding author.

0045-7825/00/$ - see front matter
PII: S0045-7825(99)00373-4

issues of scaling and normalization across multiple blocks. These weight functions can then be used to construct blending functions for algebraic redistribution, interpolation functions for unstructured grid generation, forcing functions to attract/repel points in an elliptic system, or to trigger local refinement based upon application of an equidistribution principle.

In this paper, algorithms for grid adaptation based on redistribution/movement of a fixed number of grid points are presented. An overall approach to adaptation is presented. The weight functions developed utilize scaled derivatives and normalizing procedures to minimize or eliminate the need for user input. The ability to detect flow features of varying intensity and the lack of user defined inputs for definition of the weight functions are key characteristics of the developed weight functions. Redistribution algorithms using algebraic, elliptic, parabolic, and point movement techniques are presented. The application of Non-Uniform Rational B-Splines (NURBS) is explored for surface/volume grid redistribution, for maintaining fidelity of solid geometrical components, and for treatment of block interfaces. An adaptive grid system capable of automatically resolving complex flows with shock waves, expansion waves, shear layers, and complex vortex–vortex and vortex–surface interactions that is executable in a parallel computing environment is discussed. Computational examples to demonstrate the success of the methods utilized are presented.

2. Approach to adaptation

For structured grid topologies, redistribution of a fixed number of grid points, commonly referred to as *r*-refinement, is by far the simplest to implement and most widely utilized grid adaptation strategy [11]. For unstructured/hybrid or generalized grid topologies where the connectivity is specified explicitly, strategies based on *h*-refinement, derefinement, and point movement [27] are more attractive and are widely used. In this paper, redistribution schemes for structured grids with multiple, topologically distinct blocks are explored. Also considered is a point movement approach for generalized grid topologies.

Adaptive redistribution of points traces its roots to the principle of equidistribution of error [34] by which a point distribution is set so as to make the product of the spacing and a positive weight function, w, constant over the set of points

$$w\Delta x = \text{constant}. \tag{1}$$

With the point distribution defined by a function ξ_i, where ξ varies by a unit increment between points, the equidistribution principle can be expressed as

$$wx_{\xi} = \text{constant}. \tag{2}$$

This one-dimensional equation can be applied in each direction in an alternating fashion [7,11,12]. Direct extension to multiple dimensions using algebraic [7], variational [33,34], and elliptic [11,12,15] systems are well documented in the literature. The definition and evaluation of the weight function is the key to a successful adaptation.

Structured grid generation methods using algebraic methods (where interpolation schemes or Computer Aided Geometry Design (CAGD) techniques are utilized) and methods using partial differential equations (where elliptic, parabolic, or hyperbolic equations are solved) can be utilized for adaptive redistribution of grid points. In general, the adaptive redistribution process consists of three main steps. The first step is to define appropriate weight functions representative of important solution features. The second, and probably the most crucial step, is to redistribute the grid points in the computational domain in a manner consistent with the aforestated weight function. It is crucial that the geometric fidelity of solid boundaries be maintained during the redistribution process. Also, grid quality, as measured by orthogonality, cell aspect ratio, and smoothness, must be maintained. The third and the final step is to modify the metric terms to reflect grid movement with a consistent grid speed or to re-evaluate the flow variables and metric terms at the new grid locations using an appropriate search/interpolation scheme. Time accuracy is achieved by transforming the time derivatives through the addition of convective-like

terms containing the grid speeds in a manner that does not alter the conservation properties of the governing PDEs.

The parametric, control points-based, NURBS [7,28] is utilized for algebraic grid adaptation and for redistribution of grid points on solid boundaries associated with the geometrical configuration under study. The convex hull, local support, shape preserving forms, and variation diminishing properties of NURBS are extremely attractive in engineering design applications and in geometry/grid generation. In fact, the NURBS representation has become the 'defacto' standard for the geometry description in most modern grid generation systems [25,28].

The NURBS representations [28] for a curve, surface, and volume are as follows:

$$\text{Curve}: \quad C(t) = \frac{\sum_{i=0}^{n} W(i)d(i)N_i^k(t)}{\sum_{i=0}^{n} W(i)N_i^k(t)}, \tag{3}$$

$$N_i^k(t) = \frac{(t - T(i))N_i^{k-1}(t)}{T(i+k-1) - T(i)} + \frac{(T(i+k) - t)N_{i+1}^{k-1}(t)}{T(i+k) - T(i+1)}, \tag{4}$$

where k is the order of the curve, T the knot sequence and

$$N_i^1(t) = \begin{cases} 1 & \text{if } T(i) \leqslant t < T(i+1), \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

W is the weight function, d the control polygon, N the basis function and n is the last index of control polygon

$$\text{Surface}: \quad S(s,t) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} W(i,j)d(i,j)N_i^{k1}(s)N_j^{k2}(t)}{\sum_{i=0}^{m}\sum_{j=0}^{n} W(i,j)N_i^{k1}(s)N_j^{k2}(t)}, \tag{6}$$

the basis functions are defined in the same manner as Eqs. (4) and (5),

$$\text{Volume}: \quad V(s,t,u) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{l=0}^{p} W(i,j,l)d(i,j,l)N_i^{k1}(s)N_j^{k2}(t)N_l^{k3}(u)}{\sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{l=0}^{p} W(i,j,l)N_i^{k1}(s)N_j^{k2}(t)N_l^{k3}(u)}, \tag{7}$$

the basis functions are defined in the same manner as Eqs. (4) and (5).

The relationship between the order k, number of knots n_{knot}, and the number of control points $(n+1)$ can be expressed by $n_{\text{knot}} = (n+1) + k$.

Curve/surface/volume grid point redistribution can be accomplished by evaluating the underlying NURBS representation at the specified parametric locations. The usual practice is to transform all pertinent geometrical entities into NURBS representations resulting in a common data structure. However, in many instances, only the grid information is available and the boundaries/surfaces of interest are represented by a discrete set of points. The inverse NURBS formulation can be used to uniquely transform discretized sets or networks of points into appropriate NURBS descriptions. For a curve in R^3, for example, the problem can be stated as: given a set of data points (X_i, Y_i, Z_i) $i = 1, \ldots, N$, find the control points d_i, $i = 0, \ldots, N+1$, which can be used to interpolate the given discrete points:

$$C(t_{j+2}) = \sum_{i=0}^{N+1} d_i N_i^4(t_{J+2}) = \mathbb{R}_j, \quad j = 1, \ldots, N, \tag{8}$$

where t is the knot sequence based on the arc length of the interval $[t_3, t_{N+2}]$ $\mathbb{R}_j = (X_j, Y_j, Z_j)$.

However, there are $N+2$ unknowns $(i = 0, 1, \ldots, N+1)$ to be evaluated. The additional two equations are obtained by specifying the end conditions (e.g. parametric derivatives at the end points). The entire system to be solved is as follows:

$$\begin{bmatrix} b_{00} & b_{01} & b_{02} & & \\ N_0^4(t_3) & N_1^4(t_3) & N_2^4(t_3) & & \\ & N_1^4(t_4) & N_2^4(t_4) & N_3^4(t_4) & \\ & & \ddots & & \\ & & & & \\ & N_{N-1}^4(t_{N+2}) & N_N^4(t_{N+2}) & N_{N+1}^4(t_{N+2}) & \\ & b_{N+1,N-1} & b_{N+1,N} & b_{N+1,N+1} & \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_N \\ d_{N+} \end{bmatrix} = \begin{bmatrix} \mathbb{R}_1' \\ \mathbb{R}_1 \\ \mathbb{R}_2 \\ \vdots \\ \vdots \\ \mathbb{R}_N \\ \mathbb{R}_N' \end{bmatrix}, \tag{9}$$

where $\mathbb{R}_1', \mathbb{R}_N'$ are the derivatives at $\mathbb{R}_1$ and $\mathbb{R}_N$.

$$b_{00} = \frac{-3N_1^3(t_3)}{t_4 - t_1}, \qquad b_{N+1,N} = \frac{3N_N^3(t_{N+2})}{t_{N+3} - t_N} - \frac{3N_{N+1}^3(t_{N+2})}{t_{N+4} - t_{N+1}},$$

$$b_{01} = \frac{3N_1^3(t_3)}{t_4 - t_1} - \frac{3N_2^3(t_3)}{t_5 - t_2}, \qquad b_{N+1,N+1} = \frac{3N_{N+1}^3(t_3)}{t_{N+4} - t_{N+1}},$$

$$b_{02} = \frac{3N_2^3(t_3)}{t_5 - t_2}, \qquad b_{N+1,N-1} = \frac{-3N_N^3(t_{N+2})}{t_{N+3} - t_N}.$$

It can be observed that the matrix in Eq. (9) can be readily transformed to a tridiagonal system with simple algebraic manipulation. Also, notice that Eq. (9) is valid even with duplicate and triplicate discrete points. Similar algorithms in higher dimensions can be obtained using a tensor product formulation [28].

3. Weight functions

Typically, weight functions for solution adaptive grid generation are constructed to mimic the local truncation error and may require substantial user input. These weight functions can then be used to construct blending functions for algebraic redistribution, interpolation functions for unstructured grid generation, forcing functions to attract/repel points in an elliptic system, or to trigger local refinement. Several papers, such as those by Thompson [11], Anderson [12], Soni and Yang [13] and Soni et al. [14], present applications where significant improvements in accuracy have been obtained through the use of adaptive grid procedures. Most of these techniques are based on weight functions comprised of combinations of flow variable derivatives, and require input for the selection of these variables, as well as their coefficients. Dwyer [15] and Soni and Yang [13] have developed scaling procedures to lessen the required user input for construction of these weight functions. It is believed that an adaptive grid system which requires no user input while ensuring an efficient grid point distribution would dramatically increase the routine use of CFD in the design and analysis environment. The development of such a system is the ultimate objective of this work. The weight functions developed here utilize extensive scaling and normalization, and hence, may not be the most efficient for a given problem, but are intended to remain useful for a wide range of problems with no user input. The weight function developed here utilizes scaled derivatives and normalizing procedures to minimize or eliminate the need for user input. In this work, solution adaptive redistribution of grid points is achieved via equidistribution of the weight function through the use of forcing functions in elliptic systems, parabolic systems, and algebraic systems.

3.1. Construction of weight functions

Application of the equidistribution law results in grid spacing inversely proportional to the weight function, and hence, the weight function determines the grid point distribution. Ideally, the weight function would be equal to the local truncation error ensuring a uniform distribution of error. Determination of this function is one of the most challenging areas of adaptive grid generation. The overall solution is only as accurate as the least accurate region. Excessive resolution in a given region does not necessarily increase the accuracy of the overall solution. Evaluation of higher-order derivatives from discrete data is progressively less accurate and is subject to noise. However, lower-order derivatives must be non-zero in regions where

the variation of higher-order derivatives is significant, and are proportional to the rate of variation. Therefore, it is possible to employ lower-order derivatives as proxies for the truncation error. Construction of weight functions often requires the user to specify which derivatives are to be used as well as their relative proportions. This can be a time consuming process. Also, due to the disparate strength of flow features, important features can be lost in the noise of dominant features.

Weight functions based on this paradigm have been developed by Soni and Yang [13] and Thornburg and Soni [8]. The weight function of Thornburg and Soni [8] has the attractive feature of requiring no user specified inputs. Relative derivatives are used to detect features of varying intensity, so that weaker but important structures, such as vortices, are accurately reflected in the weight function. In addition, each conservative flow variable is scaled independently. One-sided differences are used at boundaries. No-slip boundaries require special treatment since the velocity is zero and are handled in the same manner as zero velocity regions in the field. A small positive quantity, ϵ, is added to all normalizing quantities. In the present work, the weight function uses the Boolean sum construction method of Soni and Yang [13]. Additionally, the normalizing derivatives have been set to an initial or minimum value of ten percent of the freestream quantities. This alleviates problems encountered in flows without significant features to trigger adaption in one or more coordinate directions. Otherwise a few percent variation would be normalized to the same level as a shock or other strong feature.

Analysis of the weight functions explored to date indicates that density or velocity derivatives alone are not adequate to represent the different types and strengths of flow features. Density, or pressure for that manner, varies insufficiently in the boundary layer to be used to construct weight functions for representation of these features. While velocity derivatives by themselves are adequate for resolving boundary layers in viscous flows, additional variables must be included to represent other flow features such as shocks or expansions.

The current weight function is as follows:

$$W = 1 + \frac{W^1}{\max(W^1, W^2, W^3)} \oplus \frac{W^2}{\max(W^1, W^2, W^3)} \oplus \frac{W^3}{\max(W^1, W^2, W^3)}, \tag{10}$$

where

$$W^k = \sum_{\oplus i=1}^{nq} \left[\frac{\frac{|q^i_{\xi k}|}{|q^i|+\varepsilon}}{\left[\frac{|q^i_{\xi k}|}{|q^i|+\varepsilon}\right]_{\max}} \right] \oplus \left[\frac{\frac{|q^i_{\xi k\,\xi k}|}{|q^i|+\varepsilon}}{\left[\frac{|q^i_{\xi k\,\xi k}|}{|q^i|+\varepsilon}\right]_{\max}} \right] \qquad (k = 1, 2, 3). \tag{11}$$

Usually $\varrho, \varrho u, \varrho v$, and ϱw are utilized for $i = 1, 2, 3$, and 4 respectively. However, the user may select variables for adaptation based on the particular application being considered.

The symbol $\oplus$ represents the Boolean sum, which, for two variables q_1 and q_2, is defined as

$$q_1 \oplus q_2 = q_1 + q_2 - q_1 q_2. \tag{12}$$

Note that the Boolean sum is defined only when $0 \leqslant q_i \leqslant 1$, which brings in both logical or/and effects in the evaluation of weights. Normalization and scaling applied in the computation of weight functions ensures the satisfaction of this criteria. Note that the directional weight functions are scaled using a common maximum in order to maintain the relative strength.

Intuition and experience of the user can be used to determine the location of relatively 'weak' features such as shear layers. Feature detection could be improved if this 'knowledge' could be incorporated into the weight function. It is critical for the adaptive procedure to recognize all flow features and not be dominated by a single feature such as a strong shock. Additionally, only structures that have been at least partially resolved by the flow solver can be detected by the weight function. Any feature completely missed in the simulation will not be detected. Hence the quality of the weight function is dependent upon the quality of the solution. The adaptation procedure and the flow solver should be coupled so that the adapted grid can reflect all the features that are detected as the solution progresses and improves due to adaptation. If the weight functions are to be used directly, such as in the algebraic redistribution technique, smoothing is

required. However, when performing approximate equidistribution via forcing functions, it seems to be more effective to smooth the forcing functions directly.

3.2. Simple modification to weight function

Grid adaptation can become tricky when the spacing of the initial grid is not adequate to resolve a significant flow field feature. This situation can occur where there is insufficient clustering of grid lines near a shock. Even though the shock may be captured within two cells, the resulting weight function is correspondingly smaller which results in inadequate adaptation near the shock. In order to remedy this situation, the weight function contribution in each direction is normalized individually. This gives equal importance to all the three directions. Also, the weight function is scaled by the normalized square of the spacing in each direction. The modified formulations of the weight functions is presented in Eqs. (13) and (14):

$$W = 1 + \frac{w^1}{\max(w^1)} \oplus \frac{w^2}{\max(w^2)} \oplus \frac{w^3}{\max(w^3)}, \tag{13}$$

where

$$w^k = \frac{W^k g_{kk}}{\max(g_{kk})} \qquad (k = 1, 2, 3), \tag{14}$$

and W^k is defined in Eq. (11).

4. Grid point redistribution: structured grids

4.1. Algebraic methods

An algebraic interpolation scheme based on transfinite interpolation is widely utilized for grid generation [25,26]. Transfinite interpolation is accomplished by the appropriate combination of 1D projectors F for the type of interpolation specified. For a three-dimensional interpolation from all six sides (surfaces) of a section, the combination of projectors is the Boolean sum of the three projectors

$$F_1 \oplus F_2 \oplus F_3 \equiv F_1 + F_2 + F_3 - F_1F_2 - F_2F_3 - F_3F_1 + F_1F_2F_3. \tag{15}$$

For adaptive grid redistribution, a NURBS representation is used as the projector F in each of the directions associated with the transfinite interpolation. Alternatively, a NURBS surface/volume description can be directly evaluated. The NURBS volume is defined by extending the surface definition in a tensor product form and is presented in Eq. (7).

4.2. Elliptic methods

The elliptic generation system

$$\sum_{i=1}^{3}\sum_{j=1}^{3} g^{ij} r_{\xi^i\xi^j} + \sum_{k=1}^{3} g^{kk} P_k r_{\xi^k} = 0, \tag{16}$$

where r is the position vector, g^{ij} the contravariant metric tensor, ξ^i the curvilinear coordinate, and P_k the control function, is widely used for grid generation [1]. Control of the distribution and characteristics of a grid system can be achieved by varying the values of the control functions P_k in Eq. (16) [1]. The application of the one-dimensional form of Eq. (16) combined with equidistribution of the weight function results in the definition of a set of control functions for three dimensions given by

$$P_i = \frac{(W_i)_{\xi^i}}{W_i} \quad (i = 1, 2, 3). \tag{17}$$

These control functions were generalized by Eiseman [2] as

$$P_i = \sum_{j=1}^{3} \frac{g^{ij}}{g^{ii}} \frac{(W_i)_{\xi^i}}{W_i} \quad (i = 1, 2, 3). \tag{18}$$

In order to conserve some of the geometrical characteristics of the original grid, the definition of the control functions is extended as

$$P_i = (P_{\text{initial geometry}}) + c_i(P_{\text{wt}}) \quad (i = 1, 2, 3), \tag{19}$$

where $P_{\text{initial geometry}}$ is the control function based on initial geometry, P_{wt} the control function based on current solution, and c_i is the constant weight factor.

These control functions are evaluated based on the current grid at the adaptation step. This can be formulated as

$$P_i^{(n)} = P_i^{(n-1)} + c_i(P_{\text{wt}})^{(n-1)} \quad (i = 1, 2, 3), \tag{20}$$

where

$$P_i^{(1)} = P_i^{(0)} + c_i(P_{\text{wt}})^{(0)} \quad (i = 1, 2, 3). \tag{21}$$

A flow solution is first obtained with an initial grid. Then the control functions P_i are evaluated in accordance with Eqs. (17) and (20), which are based on a combination of the geometry of the current grid and the weight functions associated with the current flow solution [11].

Evaluation of the forcing functions corresponding to the current grid geometry has proven to be troublesome. Direct solution of Eq. (16) for the forcing functions using the input grid coordinates via Cramer's rule or IMSL libraries was not successful. For some grids with very high aspect ratio cells and/or very rapid changes in cell size, the forcing functions became very large. The use of any differencing scheme other than the one used to evaluate the metrics, such as the hybrid upwind scheme [8], would result in very large mesh point movements. An alternative technique for evaluating the forcing functions based on derivatives of the metrics was implemented [3].

$$P_i = \frac{1}{2} \frac{(g_{11})_{\xi^i \xi^i}}{g_{11}} + \frac{1}{2} \frac{(g_{22})_{\xi^i \xi^i}}{g_{22}} + \frac{1}{2} \frac{(g_{33})_{\xi^i \xi^i}}{g_{33}} \quad (i = 1, 2, 3). \tag{22}$$

This technique has proven to be somewhat more robust, but research efforts are continuing in this area.

4.3. Parabolic methods

The effectiveness of utilizing systems of partial differential equations to generate solution adaptive node distributions for discrete solutions of field equations is well documented. In addition to the elliptic techniques described in the previous section, hyperbolic systems [16,17] have also been used to generate adaptive grids for a variety of configurations.

The two main advantages provided by elliptic systems are: (1) the smoothness of the resulting mesh and (2) the ability to control grid point locations. The primary disadvantage is the execution time required to solve the resulting elliptic system. The main advantage of the hyperbolic system is the execution time needed to generate the grid. Hyperbolic systems can be solved in much less time than elliptic systems – typically one to two orders of magnitude less. However, purely hyperbolic systems do not guarantee the same degree of smoothness because of a lack of dissipation. This shortcoming is remedied through the addition of dissipative terms [19].

Parabolic systems provide a compromise between these two approaches since grids generated using parabolic methods can be obtained in times competitive with those generated using hyperbolic schemes. In addition, the presence of dissipation helps to ensure a smooth grid point distribution. The approach used here is based on the initial work of Nakamura [18] for static grids, later extended by Noack and Anderson [20] and Noack and Parpia [21] to solution adaptive grids.

The method employed here is discussed in some detail in the sections that follow. Although only two-dimensional domains are considered here, extension to three dimensions is straightforward and has previously been demonstrated [21] for a similar approach.

4.3.1. Description of method

As with any initial value problem, computation for a parabolic scheme is begun from an initial data line – typically the body surface. The next layer of points is computed using the specified algorithm. Once computation of this layer is complete, it is used as the initial data line for the next layer, etc.

Direct application of the elliptic grid generation equations as a marching scheme results in an ill-posed problem. In parabolic methods, the concept of a local reference grid [18,20] is used to make the problem well posed. Two layers of a local reference grid are generated from the initial data line. The elliptic equations are applied to the intermediate layer using the points in the outer layer as known values. The application of the elliptic grid generation equations to the intermediate layer can be viewed as a smoothing of the reference grid. The outer layer location can be altered during the smoothing process by recomputing the outer layer grid point positions using the reference grid generation technique. It should be noted that the resulting smoothed grid still exhibits many characteristics of the reference grid. This point has important ramifications for solution adaptive grids. Specifically, the reference grid must be constructed to be consistent with the grid spacing implied by the chosen weight function used in the smoothing equation.

As shown in [20,21], it is possible to impose an outer boundary location by modification of the reference grid generation procedure.

4.3.2. Reference grid generation

An orthogonal reference grid is generated using an algebraic scheme. Starting from an initial data line j, the grid points at the next marching layer $j+1$ are computed using

$$\boldsymbol{r}_{i,j+1} = \boldsymbol{r}_{i,j} + \delta_{i,j}\boldsymbol{n}_{i,j}, \tag{23}$$

where $\boldsymbol{r}$ is the position vector, δ a specified spacing function, and $\boldsymbol{n}$ is the local unit normal to the $\eta =$ constant line j. Eq. (23) is applied again after incrementing j to generate the second or outer layer of the reference grid. It should be noted that this method is equivalent to solving a hyperbolic system. In some cases, it is helpful to smooth the unit normals in strongly non-convex regions. If no adaptation is required or if there is no outer boundary, the $j+1$ layer of the reference grid is smoothed as described below.

If there is a specified outer boundary, the reference grid is modified so it smoothly merges with the outer boundary following the method of Noack and Anderson [20]. This merging is accomplished using a blending function σ as follows:

$$\boldsymbol{r}_{i,j+1} = \sigma\boldsymbol{r}_g + (1-\sigma)\boldsymbol{r}_{i,j}, \tag{24}$$

where $\boldsymbol{r}_g$ is computed using a linear interpolation based on distance from the outer boundary

$$\boldsymbol{r}_g = \boldsymbol{r}_{i,j} + \left(\frac{\delta_{i,j}}{l_{i,j}}\right)\left(\boldsymbol{r}_{i,j\max} - \boldsymbol{r}_{i,j}\right), \tag{25}$$

where $l_{i,j}$ is the distance from the point (i,j) to the point $(i,j\max)$ computed using a summation of δ. The blending function σ is a user specified, continuous function that is zero at the wall and unity at the outer boundary.

Finally, as noted above, the smoothed grid exhibits characteristics of the reference grid. Therefore, δ (the spacing function for reference grid) should be modified based on the weight function used in the smoothing step. The approach taken here is to apply equidistribution along each $\xi =$ constant coordinate line. For the weight function w, equidistribution can be shown to be equivalent to

$$s_{\xi\xi} + \phi s_{\xi} = 0, \tag{26}$$

where s is the arc length along the $\xi =$ constant line and $\phi = w_{\xi}/w$. δ is computed using the distance between adjacent grid points on the $\xi =$ constant coordinate line.

4.3.3. Smoothing of reference grid points

Once the reference grid has been generated, the grid point distribution is smoothed by applying the elliptic Poisson equation

$$g_{22}(\boldsymbol{r}_{\xi\xi} + \phi r_{\xi}) - 2g_{12}\boldsymbol{r}_{\xi\eta} + g_{11}(\boldsymbol{r}_{\eta\eta} + \psi \boldsymbol{r}_{\eta}) = 0 \tag{27}$$

at $i, j+1$. All derivatives in the above expression are approximated using central differences. The resulting matrix system is solved using a scalar tridiagonal inversion algorithm. In practice, it has been found that reducing the grid point movement induced by the smoothing by an order of magnitude results in a grid that is sufficiently smooth. Typically, this requires the equivalent of two to three iterations of an elliptic solver. Domains containing strongly non-convex regions may require additional iterations.

5. Grid point movement

One of the many techniques for grid adaptation by node movement uses a weighted Laplacian approach [22]. This approach is simple to implement and can be used for any grid topology. The weights are calculated using the same approach as used for structured grids. The gradient at the cell center, used in evaluating the weight function, is calculated by applying Gauss' theorem

$$\nabla f = \frac{1}{V}\oint_{\partial\Omega} f\underline{n}\,\mathrm{d}A = \frac{1}{V}\sum_{\text{faces}} f\underline{n}\,\mathrm{d}A, \tag{28}$$

where V is the cell volume, $\underline{n}$ the positive outward unit normal to the control surface, and $\mathrm{d}A$ is the control surface area. The gradients and the corresponding weight functions are calculated in physical coordinates. A typical form of the weighted Laplacian [22] is written as

$$\underline{r}^{n+1} = \underline{r}^{n} + \omega\frac{\sum_{\text{edges}} W_{i0}(\underline{r}_i^n - \underline{r}_0^n)}{\sum_{\text{edges}} W_{i0}}, \tag{29}$$

where $\underline{r}$ is the position vector, superscript $n+1$ and n the indicate relaxation levels, W_{i0} the weight function for the edge connecting nodes i and 0, and ω is the relaxation parameter.

6. Surface point redistribution

Accurate representation of the flow field in the vicinity of boundaries is critical for an acceptable overall solution. Physical processes occurring near the boundaries often drive the flow physics in other regions of the domain. This is especially true for no-slip surfaces. Hence, the quality and distribution of the grid in no-slip regions is of critical importance. Mesh orthogonality may also be required for the implementation of a turbulence model. When using an adaptive procedure based on a redistribution strategy, the interior points move as the grid is adapted. This leads to distorted cells if the boundary points are not redistributed in a consistent manner as the grid is adapted. Both grid quality and geometric fidelity must be maintained during the redistribution process. In this approach, all surfaces of individual blocks are treated in the same manner – whether they are block interfaces or physical boundaries. The NURBS description of the underlying geometry associated with the interface has already been presented. This description is used to generate the redistributed surface based on a user specified distribution mesh. The entire surface or a subregion can be redistributed. Subregions can be used to fix points, such as sharp corners or transition points between boundary condition types. For solid surfaces, the distribution mesh is based on the nearest interior surfaces. The spacing between surfaces is small and the surfaces are of a similar geometric shape resulting in a nearly orthogonal coordinate system. Block interfaces are treated by redistributing the current block surface based on its corresponding surface in the neighboring block.

7. Parallel multiblock adaptive grid system

A parallel grid adaptation system for general three-dimensional structured multiblock domains (Parallel Multiblock Adaptive Grid System, PMAG) [31], developed utilizing described techniques, is presented in this section. The PMAG was developed as a stand-alone module that reads in PLOT3D [4] grids and solution files as input. The grid adaptation scheme is based on the redistribution of grid points using an elliptic solver with weight functions as described previously. The grid blocks are treated as individual domains that may be distributed over multiple processors. MPI [29,30] is used for message passing. On shared memory machines, each block can be split over multiple threads. The weight function used has been previously described in Section 3. A Neumann surface boundary condition has been implemented using NURBS representations for the geometrical entities described previously to maintain geometric fidelity. A parallel multiblock solution interpolation algorithm has been incorporated to guarantee accurate adaptation. The system can also be used as a multiblock elliptic grid generator [9]. The PMAG system was developed with the following goals in mind:

1. There should be absolutely no restriction on block connectivity. A block can be connected to any block including itself, thus supporting a wide range of complex three-dimensional topologies.
2. The algorithm should adapt a multiblock grid concurrently with each block solved in an individual process. These processes could be run on a shared memory parallel machine or distributed over a network of workstations. The algorithm should be scalable. Communication latency should be minimized.
3. Grid adaption should require minimum user interaction to resolve all important flow features. The Neumann boundary conditions should be incorporated to maintain grid quality near body surfaces. Solution interpolation must be included.

7.1. Parallel implementation

The distribution of grid points in each block is performed as an independent process. PMAG spawns processes equal to the number of topological blocks. Each block is stored in a separate disk file. This is done to enable concurrent reading and writing of grid files by each process. The user needs to supply the connectivity information for each block. This includes information about shared faces and fixed patches. Grid lines must be continuous across adjoining blocks. The inter-block faces, edges, and vertices that do not describe a fixed body must be free to float in the space. This requires that the point distribution on the block faces be computed with an exchange of information across the faces. Each block has a layer of ghost cells that contain data from the neighboring blocks. These data are updated at intermediate intervals using asynchronous communication. Individual processors are responsible for computing the control functions and executing the elliptic solver. A global norm is computed after each iteration for comparison with a specified convergence criterion. Solution interpolation and boundary point movement using a NURBS surface definition is performed after every n iterations as specified by the user.

This algorithm achieves scalability through the use of threads. If excess processors are available, the processes subdivide their domains by unrolling the outermost loop of the solver and the search algorithm and spawning a thread for each subdomain. Controlling the number of threads spawned by each process aids in load balancing. The larger blocks are allowed to spawn more threads than the smaller blocks. Splitting the domain into subdomains also leads to better cache performance. Threads are used with MPI although the MPICH implementation is *not* thread safe. Therefore, only one thread is active at the time of inter-process communication. The rest of the threads stop during this time. A further enhancement to the current version would have other threads continuing the computations with locks on the data while one thread is dedicated to communications.

7.2. Handling shared faces

To guarantee complete continuity of grid lines across block faces, each block sends a face to its neighboring block as shown in Fig. 1. The elliptic generator can then run using the face from the neighboring block as a Dirichlet boundary. After every iteration, the updated faces are transmitted to the neighboring block. This way, the face point positions are computed using the elliptic equations at every

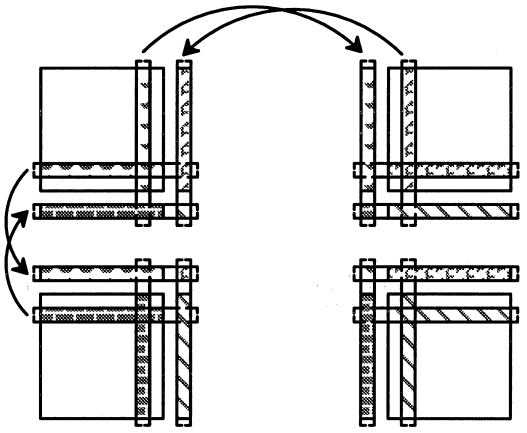

Fig. 1. Exchange of information.

step. However, the face point locations are computed independently for each block. Hence it is possible that the face points may not coincide after an iteration. To eliminate this discontinuity, some sequential multiblock codes keep track of the faces, edges, and vertices belonging to each block. Connectivity tables are maintained to identify shared vertices, edges, and faces. At the end of each iteration, these connectivity tables are used to select the copies of the common face, edge, or vertex and then an averaged value (or a value computed using the elliptically solver) is broadcast to all the owners.

Another method especially amenable for C language codes uses pointers to faces, so that only a single copy of a shared face is maintained, with both blocks pointing to the same memory location. Clearly, the later strategy cannot be used when the blocks have been assigned to different processors. The first strategy of collecting all copies and then broadcasting an average can work in a parallel implementation but only with a communication penalty. In the case of a simple structured topology with eight vertices, twelve edges, and six faces per block, each face can be shared by only one other block. One edge is shared by four blocks and a vertex is shared by eight blocks. In such a case, to retrieve unique face, edge, and vertex locations, each block needs to communicate with 26 neighboring blocks. Since an elliptic system has a three-point stencil, and since both blocks have identical copies of all three faces required for computation of the block boundary face, the face points calculated for the each block using the elliptic system should be the same. Hence, the face point locations are computed individually using point Jacobi iteration (to guarantee a three-point stencil).

The point distribution in the interior of the block is computed using the standard tridiagonal system. The control functions for the points on the shared faces are also evaluated using only a three-point stencil guaranteeing that neighboring blocks compute identical locations for the shared points. The simplicity of this scheme is its major advantage. No complicated global connectivity tables need to be maintained. This makes the code extremely flexible, enabling it to handle a wide variety of block topologies, including 0 grids embedded in H grids, periodic boundary conditions, etc. For the above strategy to be successful, the basic premise is that all blocks sharing a particular point must have identical copies of its complete stencil. The user input specifies only the blocks that share a face with the current block. This means that a block has no information about its diagonally opposite neighbor. A communication strategy that allows each block to acquire information from its diagonal neighbors is described in the next section.

7.3. Communication between shared faces

Each block is aware only of the neighboring blocks with shared faces. This means the block cannot directly access the corner points from its diagonally opposite block. To overcome this problem, the blocks communicate faces inclusive of the extra points received from the neighboring blocks (shown by the dashed-blocks). However, this means that the blocks that send the points before receiving the faces from the neighbors would send out old points to the neighbors. One way to overcome this problem is to perform

communication in a cyclic loop. However, the resulting communication process would be sequential and result in a large communication latency. The problem can be overcome by simply performing the communication process twice. This strategy allows results in the use of asynchronous communication, so that more than just two blocks communicate at any instant. In case of the simple topology discussed in the earlier section, each block would now communicate only 12 times. Note that doing the entire communication process twice does involve transmission of some redundant information. This could be avoided by sending only edge and vertex information in the second communication process. This feature has yet to be implemented in PMAG.

7.4. Parallel multiblock interpolation

Once the points are redistributed, the original solution needs to be interpolated onto the new grid. A parallel grid adaptation algorithm supporting general multiblock topologies makes solution interpolation significantly more complex. The grid points can move outside the original domain of a block. In such a case, the block does not have enough information to interpolate the solution and adaptive functions to all its points. Each block now needs to query all other blocks for the points that are no longer within its domain. The search algorithm starts with a search and interpolation of all points found within each block. Each process creates a list of points that are not found within its domain. These lists are concatenated into a global list which is broadcast to all processes for search. The processes then locate and compute the interpolated solutions for the points found within their domains. A final all-reduce over all processes makes the solutions known to all the processes. This operation takes 4 log P communications. A shift operation (the list of external points are shifted in a circle through all processes) would take P communications to complete. Hence a shift would be faster for domains where the number of blocks <16. Ideally a polyalgorithm should be used to switch methods according to the number of processes. The shift has not yet been implemented in PMAG.

8. Applications

In this section, application of the strategies discussed in the previous sections is illustrated using several examples. These sample cases range from aerodynamic simulations in the low speed, supersonic, and hypersonic flight regimes to simulation of shock propagation after a nuclear collision.

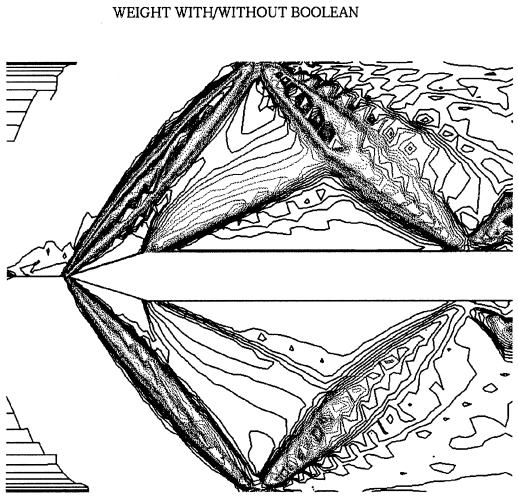

Fig. 2. Comparison of weight functions.

8.1. Influence of weight functions

Two-dimensional flow past a wedge at Mach 2 [6] is used to illustrate the enhanced detection capabilities of the weight function resulting due to incorporation of Boolean sum in Eqs. (10) and (11). Fig. 2 shows a comparison of the weight functions evaluated without using the Boolean sum, in the lower half plane, and the Boolean sum, in the upper half plane. It can be observed that both weight functions clearly detect the primary shock. It can also be seen that the expansion fan, the boundary layer, and the reflected shocks are much more clearly represented within the weight function using the Boolean summation.

Adapted grids using both weight function formulations are presented in Fig. 3. The high gradient regions of the expansion region are only reflected in the adapted grid using the weight function using the Boolean summation. The reflected shock is also much sharper. Fig. 4 shows a comparison of the solution obtained using the adaption procedure with the solution obtained using the original, unadapted grid. The enhanced resolution of the flow features is clearly evident.

The effect of the modified weight function (given by Eqs. (13) and (14)) on adaptation is demonstrated in Fig. 5. This case represents a chemically reacting, supersonic flow through a convergent channel. The initial

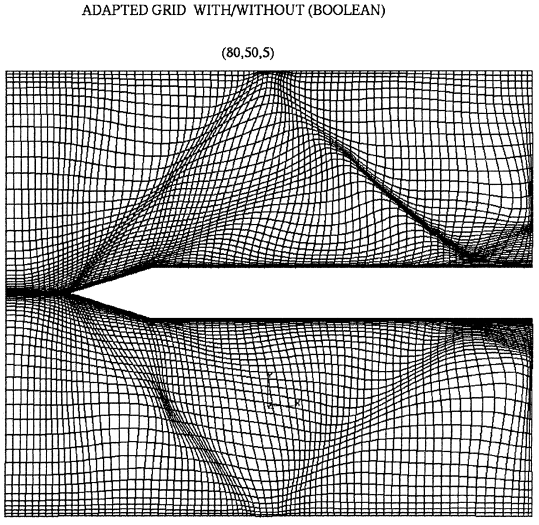

Fig. 3. Comparison of adapted grids.

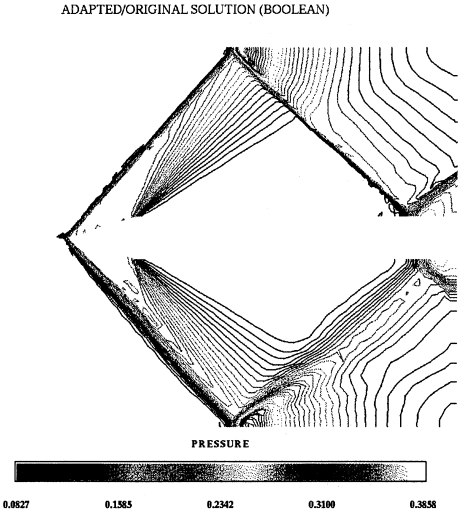

Fig. 4. Comparison of solutions using adapted grid.

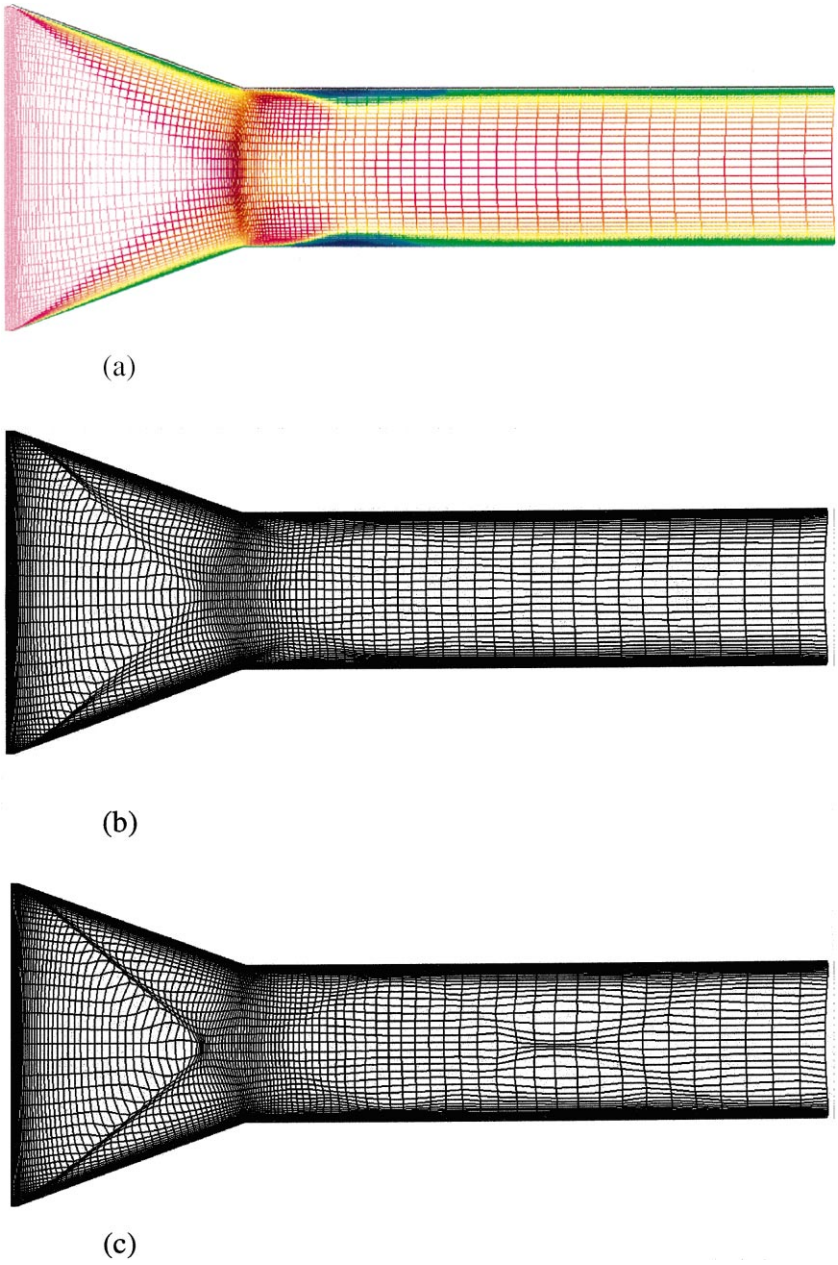

Fig. 5. Effect of g_{kk} terms in the adaptation (chemically reacting flows): (a) initial grid and U component of velocity; (b) adapted grid without g_{kk} terms (Eq. (10)); (c) adapted grid with g_{kk} terms (Eq. (13)).

grid and the U velocity component are shown in Fig. 5(a). The adaptation is based on the U,V components of velocity, pressure, temperature and Mach number. Since there was insufficient grid clustering present in the near the shock region in the initial grid, the adaptation was not effective in the vicinity of the shock (see Fig. 5(b)). By introducing the metric terms in the weight function, the shock is captured properly (Fig. 5(c)) and other solution features are evident in the resulting adapted grid.

8.2. Algebraic method

The incompressible flow over a cylinder at a Reynolds number of 500 is the first example [23] of the grid adaption strategy using algebraic distribution of points. The grid dimension used for this simulation is

181×121. The two-dimensional velocity components were used in the weight function. After several time steps, flow separation occurred behind the cylinder and mesh points were redistributed in response to the solution as shown in the Fig. 6. As the time progressed, the shed vortices were convected downstream of the cylinder. Fig. 6 clearly shows the redistribution of the grid points in the wake in response to the shed vortices.

An unsteady shock movement through a generic inlet is used to demonstrate grid adaptation for unsteady flows [23]. The grid used for this problem is of size 200×40 and the inlet Mach number is 3. The initial condition is taken as the converged supersonic flow through the entire inlet and the grid is adapted based on these conditions. A pressure increase was then imposed at the exit plane resulting in the formation of a normal shock that begins to move upstream. The algebraic adaptive grid redistribution algorithm automatically adjusts the mesh to resolve this shock, as can be seen in Fig. 7.

The next case considered is the inviscid, hypersonic, real gas flow over a blunt nose [23]. The freestream conditions, corresponding to an altitude of 10 km, were: pressure $= 26.5$ kPa, density $= 0.414$ kg/m^3, and temperature $= 223$ K. The freestream Mach number is taken as 10 at zero angle of attack. A five-species air model is used for this simulation. Temperature was used as the variable in the weight function. The initial

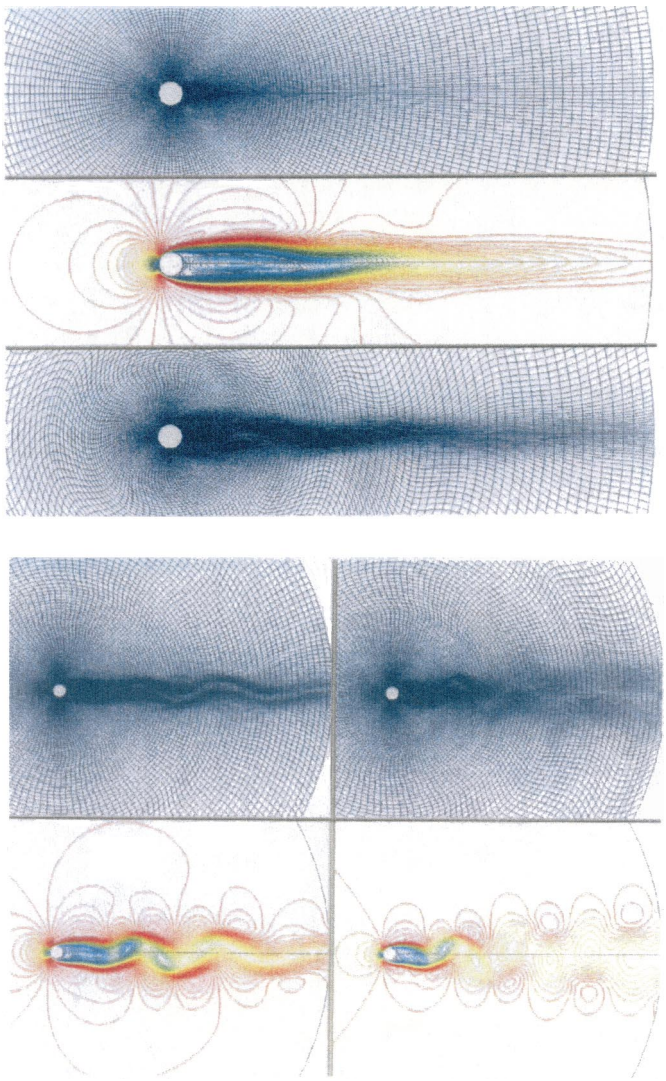

Fig. 6. Incompressible flow over a cylinder: vortex shedding and the corresponding adapted grid.

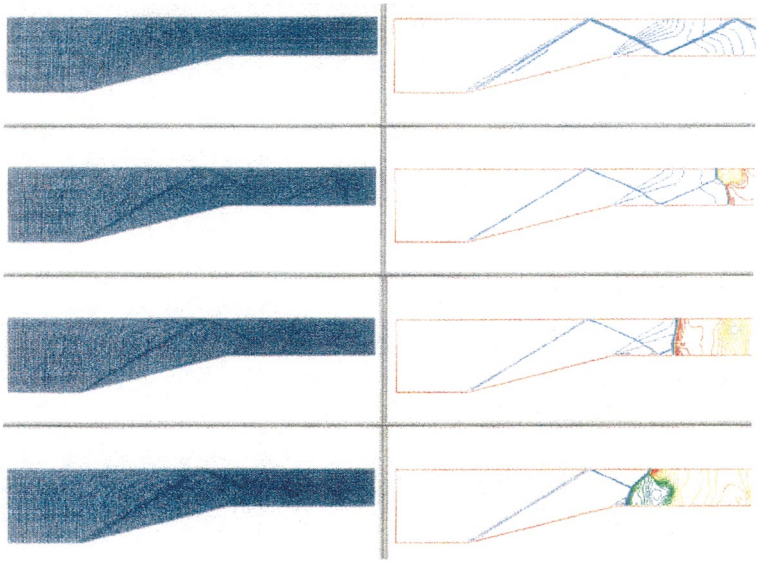

Fig. 7. Grid adaptation for unsteady moving shocks in a convergent inlet.

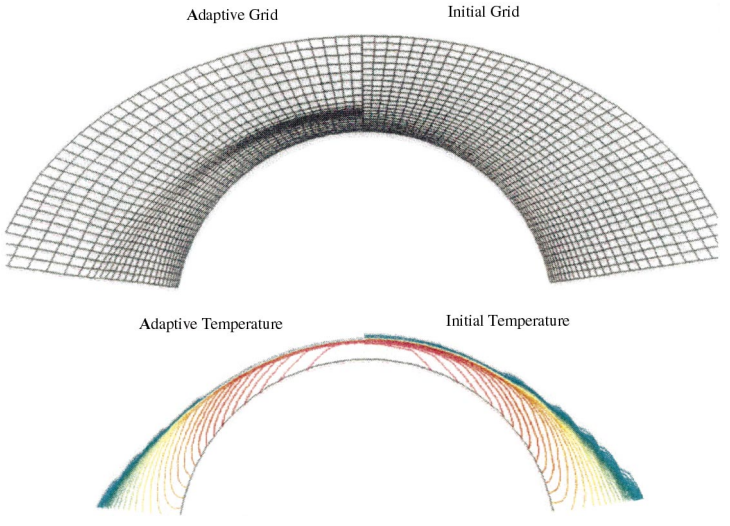

Fig. 8. Grid adaptation for hypersonic, five species air, flow over a blunt nose.

grid, adapted grid, and the corresponding temperature contours are shown in Fig. 8. It can be seen from the temperature contours that grid adaptation results in a much thinner and cleaner shock.

8.3. Parabolic method

The application demonstrating the parabolic adaptive grid technique is an inviscid supersonic (Mach 4) channel flow. The lower wall consists of a horizontal segment followed by a 20° ramp. A 20° expansion then returns the flow to the horizontal. The upper wall (or symmetry boundary) is parallel to the horizontal lower wall. An oblique shock forms at the ramp on the lower wall. The shock reflects off the upper wall and interacts with the expansion that occurs because the flow is turning back to the horizontal. The solution variables used in the weight function were density, pressure, and Mach number.

The solution adaptive grid generated for the inviscid channel flow is shown in Fig. 9. The adaptation procedure does a good job of clustering grid points in the regions near the initial and reflected shocks as well as the expansion. Points on the lower and upper boundaries were held fixed during the grid generation process. The grid dimension was 51×31.

8.4. Point movement

An example of the grid adaptation using the weighted Laplacian approach is shown in Fig. 10. An unstructured grid for a geometry representative of a scramjet engine is considered for this example. The inlet Mach number is taken as 3 and the resultant pressure distribution together with initial grid is shown in Fig. 10(a). The weight function is based on the four conserved variables and is plotted in Fig. 10(b). The adapted grid and the solution on the adapted grid are shown in Fig. 10(c) and (d). It can be seen from the pictures that the shocks and expansion fans are captured more distinctly in comparison to the original unadapted grid.

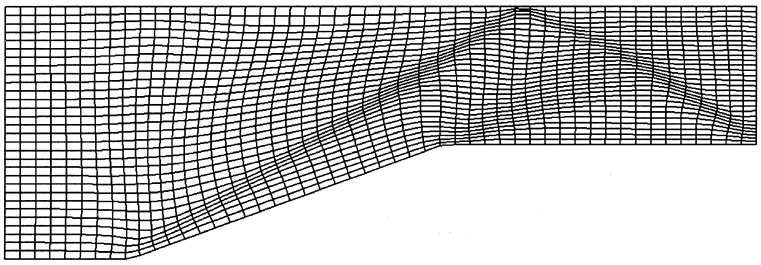

Fig. 9. Parabolic adaptive grid for supersonic channel flow.

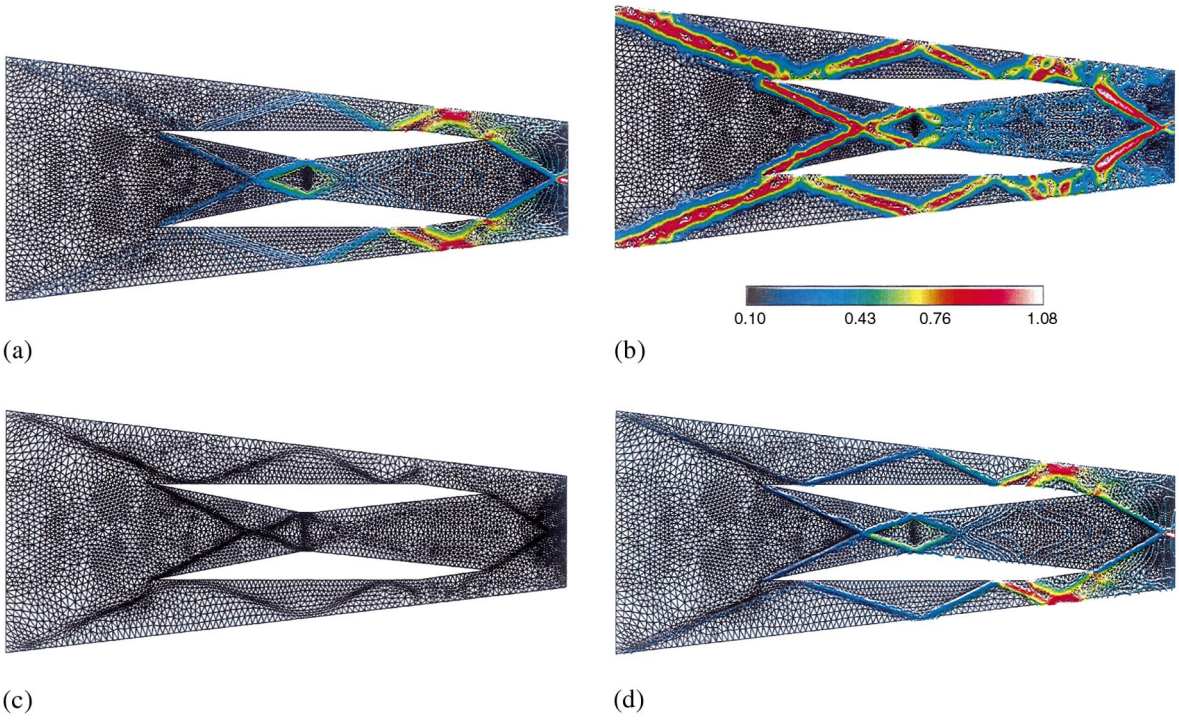

Fig. 10. Grid adaptation for unstructured grid using weighted Laplacian approach: (a) initial grid and pressure distribution; (b) initial grid and weight function; (c) adapted grid; (d) pressure distribution on adapted grid.

8.5. PMAG system

The multiblock grid adaptation capability based on the elliptic redistribution scheme is included in the PMAG system. The ability of PMAG to generate grids appropriate for computing complex, three-dimensional flow fields is demonstrated by simulating a supersonic flow around a tangentogive cylinder at a Mach number of 1.45 and an angle of attack of 14°. The CFD systems, NPARC [5] and CFL3D [10] are utilized for this simulation. Detailed descriptions of the simulations are reported in [35]. The initial grid and the solution are presented in Fig. 11(a)–(b). The weight functions evaluated using the formulation presented in Eqs. (10) and (11) is displayed in Fig. 11(c). It can be seen that the solution features presented in Fig. 11(b) are captured by the weight function. Fig. 11(d) shows the adapted grid. The adapted grid shows a concentration of grid points in the primary and secondary shock regions as well as in the boundary layer regions. The concentration in the vortex core is also pronounced.

Fig. 12 shows two different longitudinal locations, one displaying the solution and the weight function and the second displaying the solution and the adapted grid. The flow features corresponding to the vortex and the feeding sheet are clearly visible in the weight function as well as in the adapted grid. Axial and

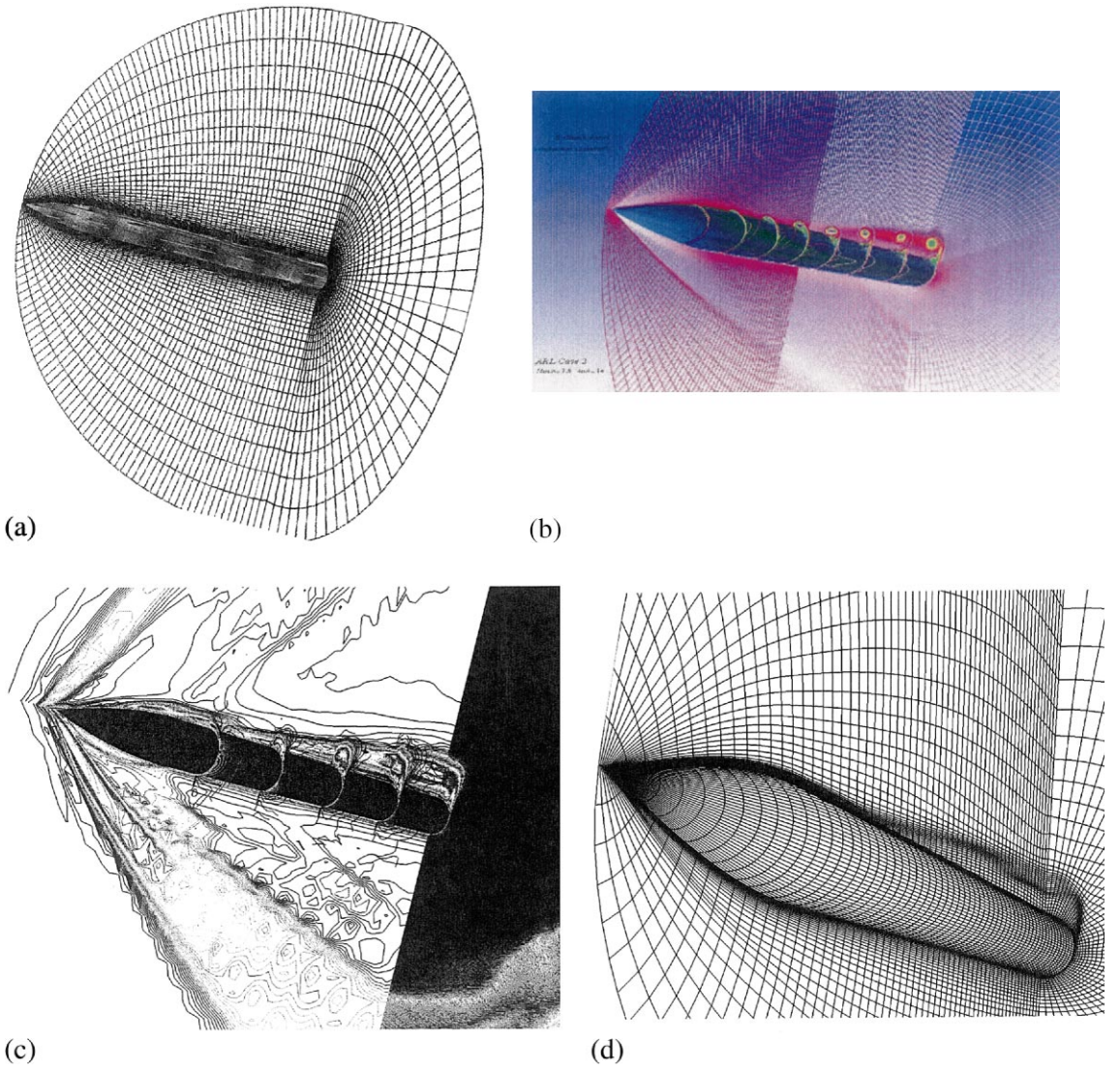

Fig. 11. Grid adaptation for massively separated flows: (a) initial grid; (b) initial solution; (c) weight function; (d) adapted grid.

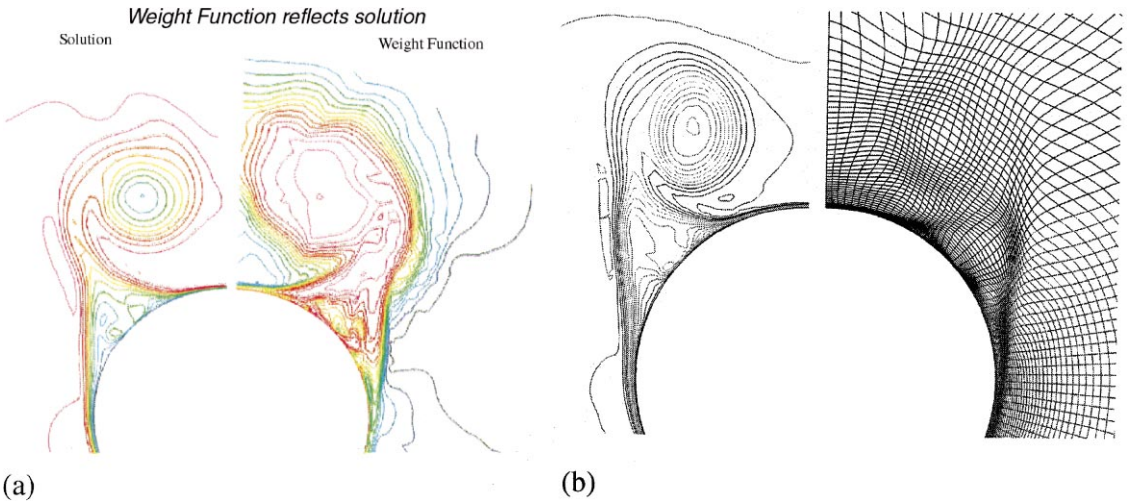

Fig. 12. Grid adaptation showing the capturing of the vortex core: (a) solution and weight function; (b) solution and adapted grid.

Table 1
Comparison of forces and moments

Forces	Experimental data	Unadapted NPARC (BL)	Adapted NPARC (BL)
Axial	0.1957	0.3307	0.3309
Normal	1.9100	1.8855	1.9052
Moment	10.2417	10.0314	10.2060

normal forces and moments were computed from solutions obtained from the adapted and unadapted grids and are compared with the experimental data in Table 1. The axial force determined from the experimental data is misleading due to a discrepancy during testing [35]. However, the normal force and moment for the adapted grid show better agreement with the experimental data than the normal force and moment for the unadapted grid.

8.6. Shock wave propagation after nuclear collision

One example of the efficacy of grid adaptation outside of the field of aerodynamics is in the field of nuclear science. Grid adaptation is utilized in the simulation of the impact of two nuclei to track the propagation of the shock wave resulting from the impact [24]. The time evolution of a head-on collision of a Uranium 238 nucleus with a Silicon 28 together with the gradient and the adaptive grid is shown in Fig. 13. The density is used in the weight function. It can be seen from the figure that the adaptation is able to track the propagation of the shock wave due to the impact of the nuclei.

9. Summary and conclusions

Several methodologies for redistributing a fixed number of points to construct solution adaptive grids were described. Examples covering a wide range of flow conditions were presented to demonstrate the influence of the weight function on the quality of the solution adaptive grid as well as to validate several redistribution methods. The weight function incorporating the Boolean summation of properly scaled, directional weights was shown to be effective in resolving complex flow features. The results presented in Fig. 11 demonstrate the capability of the weight function to detect shocks of varying strengths, primary and secondary vortices, and shear layers. Further research needs to be done to determine the appropriateness of using weight functions of this type in adaptive schemes using refinement and derefinement. Based on the

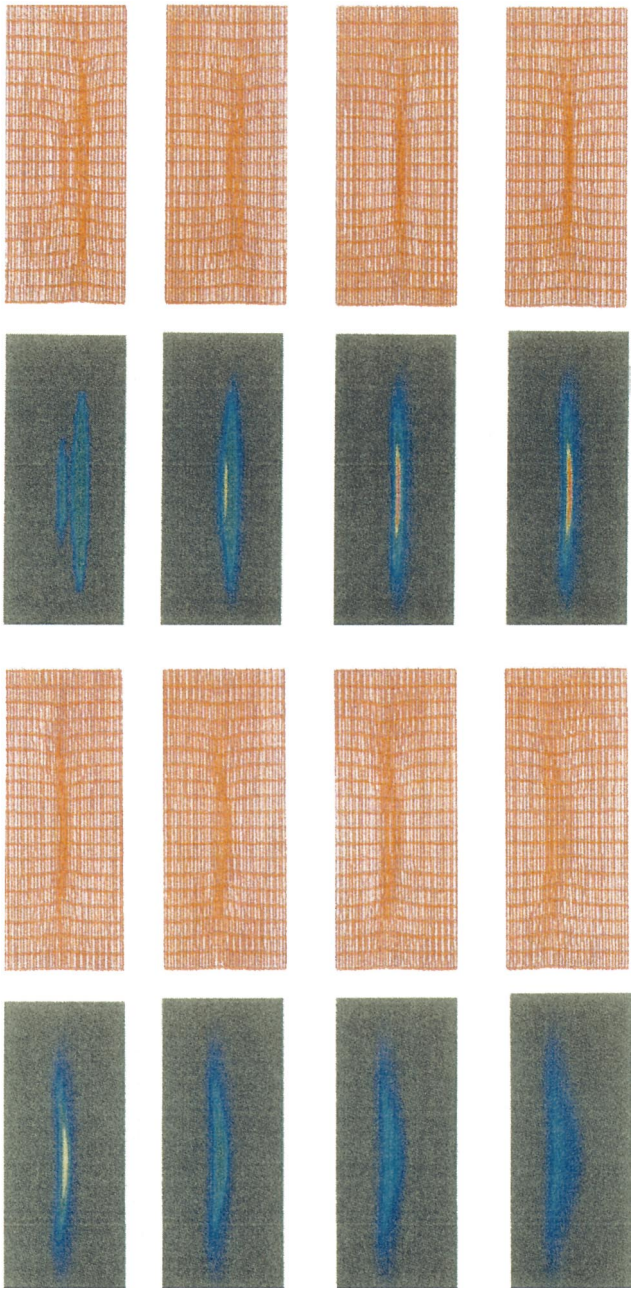

Fig. 13. Different stages of head on collision of uranium 238 and silicon 28 nuclei and the corresponding adapted grid.

results shown in this paper, the construction of the weight function is a key component of any adaptive grid system.

Also demonstrated were several types of redistribution methods. The efficacy of the algebraic and elliptic redistribution methods for unsteady/moving grid problems was demonstrated via computational examples. The algebraic method for redistribution of grid points was demonstrated for unsteady flows involving simple geometries. The PMAG system, which utilizes an elliptic redistribution scheme, was demonstrated for a complex three-dimensional flow field with multiple shocks of varying strengths and complex vortical structures. The applicability of the PMAG system to topologically unstructured, multiple block grid systems is being explored. Also shown demonstrated was the potential of utilizing a parabolic marching

algorithm to generate a high quality adaptive grids for simple configurations. Additionally, the point movement method, which is valid for generalized grids, was presented.

The application of NURBS to represent computational domain boundaries was described for surface/volume grid point redistribution, for maintaining fidelity of solid geometrical components, and for treatment of block interfaces.

Application of these methods to more complex multiblock structured grids and to generalized grids as well as the applicability of different redistribution strategies are the topics of future research.

Based on results included in the paper, here are a few general recommendations:

1. The grid adaption algorithm must be closely coupled with the flow solver, especially for unsteady problems or problems with moving grids.
2. The weight functions should be constructed using consistent scalings and the various contributions should be combined using a Boolean summation.
3. For multiblock structured grids, elliptic redistribution schemes prove to be effective methods and are preferable to other schemes.
4. For structured grids involving simple geometries but with unsteady flow phenomena, algebraic methods were shown to be an economical alternative to elliptic schemes.
5. The parabolic grid redistribution scheme was shown to have much promise and should be explored further.
6. The point movement scheme, while not attractive for structured grids, appears attractive for generalized and unstructured grids. It should be noted that the selection of the relaxation factor is extremely important in the point movement scheme.
7. The NURBS representation of surfaces is a critical factor in maintaining geometric fidelity on domain boundaries where point redistribution occurs.

References

[1] J.F. Thompson, Z.U.A. Warsi, C.W. Mastin, Numerical Grid Generation: Foundations and Applications, North-Holland, Amsterdam, 1985.
[2] P.R. Eiseman, Alternating direction adaptive grid generation, AIAA paper, 1983, pp. 83–1937.
[3] B.K. Soni, Structured grid generation in computational fluid dynamics, in: R. Vichnevetsky, D. Knight, G. Richter (Eds.), Advances in Computer Methods for Partial Differential Equations VII, Rutgers University, June 1992, pp. 689–695.
[4] P.G. Buning, J.L. Steger, Graphics and flow visualization in computational fluid dynamics, AIAA paper 85-1507-CP, in: Proceedings of the AIAA, Seventh, Computational Fluid Dynamics Conference, 1985.
[5] NASA LeRC and USAF AEDC, NPARC 1.0 User notes, June 1993.
[6] K.N. Ghia, U. Ghia, C.T. Shin, D.R. Reddy, Multigrid simulation of asymptotic curved-duct flows using a semi-implicit numerical technique, computers in flow prediction and fluid dynamics experiments, ASME Publication, New York, 1981.
[7] B.K. Soni, J.C. Yang, General Purpose adaptive grid generation system, AIAA-92-0664, 30th, Aerospace Sciences Meeting, Reno, NV, 6–9 January 1992.
[8] H.J. Thornburg, B.K. Soni, Weight Functions in grid adaption, in: Proceedings of the Fourth International Conference in Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Swansea, Wales, 6–8 April 1994.
[9] B.K. Soni, J.F. Thompson, M.L.M.H. Stokes, GENIE++, EAGLE view and TIGER: general and special purpose interactive grid systems, AIAA-92-0071, 30th, Aerospace Sciences Meeting, Reno, NV, 6–9 January 1992.
[10] NASA LaRC, User Document for CFL3D/CFL3DE, Version 1.0, 1993.
[11] J.F. Thompson, A survey of dynamically-adaptive grids in numerical solution of partial differential equations, Applied Numerical Mathematics 1 (1985) 3–27.
[12] D.A. Anderson, Adaptive grid methods for partial differential equations, Advances in Grid Generation, ASME Publication, New York, 1983, pp. 1–15.
[13] B.K. Soni, J.C. Yang, General purpose adaptive grid generation system, AIAA-92-0664, 30th, Aerospace Sciences Meeting, Reno, NV, 6–9 January 1992.
[14] B.K. Soni, N.P. Weatherill J.F. Thompson, Grid adaptive strategies in CFD, International Conference on Hydo Science and Engineering, Washington, DC, 7–11 June 1993.
[15] H.A. Dwyer, Grid adaption for problems in fluid dynamics, AIAA Journal 22 (12) (1984) 1705–1712.
[16] P. Niederdrenk, Solution adaptive grid generation by hyperbolic/parabolized P.D.E.s, in: A.S. Arcilla, J. Hauser, P.R. Eiseman, J.F. Thompson (Eds.), Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Elsevier, New York, 1991.
[17] P. Niederdrenk, Grid adaption to multiple auto-scaled solution features, in: N.P. Weatherill, P.R. Eiseman, J. Hauser, J.F. Thompson (Eds.), Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Pineridge, Swansea, UK, 1994.

[18] S. Nakamura, Noniterative grid generation using parabolic partial differential equations, in: J.F. Thompson (Ed.), Numerical Grid Generation, Elsevier, New York, 1982.
[19] W. Chan, Enhancements of a three-dimensional hyperbolic grid generation scheme, Applied Mathematics and Computation 51 (1992) 181–205 1992.
[20] R.W. Noack, D.A. Anderson, Solution-adaptive grid generation using parabolic partial differential equations, AIAA J. 28 (1990) 1016–1023.
[21] R.W. Noack, I.H. Parpia, Solution adaptive parabolic grid generation in two and three dimensions, in: A.S.Arcilla, J. Hauser, P.R. Eiseman, J.F. Thompson (Eds.), Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Elsevier, New York, 1991.
[22] J.F. Thompson, B.K. Soni, N.P. Weatherill, Handbook of Grid Generation, CRC Press, Boca Raton, 1999, pp. 1–26.
[23] J.C. Yang, General purpose adaptive grid generation system, Ph.D. Dissertation, Department of Computational Engineering, Mississippi State University, July 1993.
[24] H. Tleimat, Computational simulation of high-energy heavy-ion collisions, Ph.D. Dissertation, Department of Physics and Astronomy, December 1996.
[25] J.F. Thompson, B.K. Soni, N.P. Weatherill (Eds.), HandBook of Grid Generation, CRC Press, Boca Raton, 1999.
[26] B.K. Soni, N.P. Weatherill, Geometry-grid generation, in: Allen B. Tucker (Ed.), Computer Science and Engineering Hand Book, CRC Press, Boca Raton, 1997, pp. 791–819.
[27] N.P. Weatherill, Mixed structured-unstructured meshes for aerodynamic flow simulation, The Aeronautical Journal 94 (934) (1990) 111–123.
[28] Tzu-Yi Yu, CAGI: Computer aided grid interface, Ph.D. Dissertation, Department of Computational Engineering, Mississippi State University, 1996.
[29] M. Snir, S.W. Otto, Huss-Ledermann, S. Walker, J. Dongarra, MPI: The Complete Reference, MIT Press, Cambridge, MA, 1996.
[30] W. Gropp, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming With the Message-passing Interface, MIT Press, Cambridge, MA, 1994.
[31] M.S. Apte, Parallel adaptive grid generation for structured multiblock domain, Master's Theses, Depatment of Computational Engineering, Mississippi State University, 1997.
[32] S.D. McRae, K.R. Laflin, Dynamic Grid Adaptation and Grid Quality, in: J.F. Thompson, B.K. Soni, N.P. Weatherill (Eds.), HandBook of Grid Generation, CRC Press, Boca Raton, 1999.
[33] O.B. Khairullina, A.F. Sidorov, O.V. Ushakova, Variational methods of construction of optimal grids, in: J.F. Thompson, B.K. Soni, N.P. Weatherill (Eds.), Hand Book of Grid Generation, CRC Press, Boca Raton, 1999.
[34] J.S. Brackbill, Application and generalization of variational methods for generating adaptive meshes, in: J.F. Thompson, Numerical Grid Generation, Elsevier, Amsterdam, 1985.
[35] B.S. Walter, B. Trevor, Lauzon, Marc, C. Housh, J. Manter, E. Josyula, B.K. Soni, The application of CFD to the prediction of missile body vortices, AIAA 97-0637, 35th Aerospace Sciences Meeting and Exhibit, 6–10 January, Reno, NV, 1997.