DOE/ER/25061--2

DE89 017440

A Tensor Product b-Spline Method for 3D Multi-Block Elliptic Grid Generation

*Joseph W. Manke**

Technical Report No. 88-8

December 1988

*This research was supported in part by Boeing Computer Servies Co., Seattle,WA, and in part by Airforce Office of Scientific Research Grant No. 86-0154, and in part by Department of Energy Grant No. DE-FG06-88ER25061

A Tensor Product b–Spline Method for 3D Multi–Block Elliptic Grid Generation

Joseph W. Manke *

1 INTRODUCTION

We formulate a tensor product b–spline method for multi–block numerical grid generation. The Cartesian coordinate functions for a block are represented as a sum of tensor product b–spline basis functions defined on a parameter space for the block. The tensor product b–spline basis functions are constructed so that the basis functions and their first partials are continuous on the parameter space. The coordinate functions inherit this smoothness: a grid computed by evaluating the coordinate functions along constant parameter lines leads to smooth grid lines with smoothly varying tangents. The expansion coefficients for the coordinate functions are computed by solving the usual elliptic grid generation equations using simple collocation. This assures that the computed grid has the smoothness and resolution expected for an elliptic grid with appropriate control. The formulation also leads to a solution algorithm analogous to the ADI method with SOR used in elliptic grid generation. An important result of the formulation is that the dimension of the collocation equations is the number of distinct knots for the tensor product b–spline basis functions. Combining this result with the smoothness of the b–spline representation makes it possible to reduce the dimension of the tensor product method with respect to the finite difference method, simply by using fewer knots than grid points. In effect, a fine grid in the physical domain is obtained by constructing a smooth expansion of the coordinate functions on a coarse grid in the parameter space. These properties suggest that the method may prove to be a fast and reliable grid generation technique appropriate for use in interactive and adaptive grid codes in workstation environments.

In Section 2 we formulate the expansion of the Cartesian coordinate functions as a sum of tensor product b–spline basis functions, and then we derive the collocation and boundary condition equations for the usual elliptic grid generation equations. In Section 3 we investigate the structure of the system of equations for the expansion coefficients and then formulate a solution algorithm to compute the coefficients. Finally, in Section 4 we describe the implementation of the method in a 2D multi–block grid code and discuss the performance of the method for several grids.

*Department of Applied Mathematics, University of Washington, Seattle, WA. and on educational leave from Boeing Computer Services Co., Seattle, WA.

2 TENSOR PRODUCT b-SPLINE METHOD

For a single block $\mathcal{B} \subseteq \Re^3$ with Cartesian coordinates x^i, we consider a general coordinate system defined by curvilinear coordinate functions $\xi^i(\cdot)$ mapping $\mathcal{B}$ into a parameter space $I \subseteq \Re^3$, where $I = [0,1] \times [0,1] \times [0,1]$. The functions $\xi^i(\cdot)$ are required to be one–to–one maps from $\mathcal{B}$ onto $[0,1]$ and twice continuously differentiable with non-zero Jacobian on an open set containing $\mathcal{B}$. Since the Jacobian is non-zero on $\mathcal{B}$, the system of equations $\xi^i = \xi^i(x^j)$ on $\mathcal{B}$ may be inverted to obtain the system of equations $x^i = x^i(\xi^j)$ on I for the Cartesian coordinate functions $x^i(\cdot)$ mapping I into $\Re$.

In the grid generation method, we represent the Cartesian coordinate functions $x^i(\cdot)$ as a sum of tensor product b–spline basis functions defined on the parameter space I. The expansion coefficients are computed by solving the usual elliptic grid generation equations on I using simple collocation. The result is a parametric representation of the functions $x^i(\cdot)$, which may be evaluated along constant parameter lines to compute a grid for the block $\mathcal{B}$.

The b–spline basis functions are defined in Section 2.1 and the collocation and boundary condition equations are presented in Sections 2.2 and 2.3.

2.1 Expansion of the Cartesian Coordinate Functions

For a block $\mathcal{B}$, the representation of the Cartesian coordinate functions x^l as a sum of tensor product b–splines defined on the parameter space I is constructed by first defining b–spline basis functions on the interval $[0,1]$ for each of the curvilinear coordinates. For the coordinate ξ^i, we divide the interval $[0,1]$ into l_i intervals of equal length and define the set of $l_i + 1$ equally spaced breakpoints $s^i_{\tau_i}$ as the endpoints of the intervals

$$s^i_{\tau_i} = \frac{\tau_i - 1}{l_i}, \qquad \tau_i = 1, \ldots, l_i + 1. \tag{1}$$

Next, we define the set of knots by placing k_i knots at each interior breakpoint and $k_i + 2$ knots at the initial and final breakpoints, which leads to the set $n_i + m_i$ knots $t^i_{\tau_i}$, where

$$\begin{aligned} n_i &= k_i l_i + 2, & (2)\\ m_i &= k_i + 2, & (3) \end{aligned}$$

and

$$t^i_{\tau_i} = \begin{cases} s^i_1, & (\tau_i = 1, \ldots, m_i), \\ s^i_{1+(\tau_i - m_i + k_i + 1)/k_i}, & (\tau_i = m_i + 1, \ldots, n_i), \\ s^i_{l_i+1}, & (\tau_i = n_i + 1, \ldots, n_i + m_i). \end{cases} \tag{4}$$

Finally, we define the b–spline functions for the coordinate ξ^i as the n_i normalized b–spline basis functions $U^i_{\tau_i}$ of order m_i on the interval $[0,1]$ for the set of knots $t^i_{\tau_i}$ specified in (4), [1]. These b–spline basis functions form a basis for the linear space of piecewise polynomials on the interval $[0,1]$ of degree $m_i - 1$ with $m_i - k_i = 2$ continuity conditions at the interior breakpoints, i.e., the b–spline basis functions and their first derivatives are continuous at the interior breakpoints. This property of the b–spline basis functions will be exploited in the representation of the Cartesian coordinate functions. Another important property of the b–spline basis functions is that they have small support; $U^i_{\tau_i}$ is zero outside the interval $[t^i_{\tau_i}, t^i_{\tau_i+m_i}]$. This property will be used to simplify the collocation equations in Section 2.2.

We now define the $n_1 \times n_2 \times n_3$ tensor product b–spline basis functions on the parameter space I as products of the b–spline basis functions $U^i_{\tau_i}$

$$U_{\tau_1\tau_2\tau_3} = U_{\tau_1\tau_2\tau_3}(\xi^1, \xi^2, \xi^3) = U^1_{\tau_1}(\xi^1)\, U^2_{\tau_2}(\xi^2)\, U^3_{\tau_3}(\xi^3). \tag{5}$$

We may now use the tensor product b–splines $U_{\tau_1\tau_2\tau_3}$ defined in (5) to construct a parametric representation of the Cartesian coordinate functions x^l on the parameter space I

$$x^l = \sum_{\tau_1\tau_2\tau_3} A^l_{\tau_1\tau_2\tau_3} U_{\tau_1\tau_2\tau_3}, \tag{6}$$

where the expansion coefficients $A^l_{\tau_1\tau_2\tau_3}$ are to be determined.

With this parametric representation as a sum of the tensor product b–splines $U_{\tau_1\tau_2\tau_3}$, the Cartesian coordinate functions inherit the smoothness properties of the b–spline basis functions $U^i_{\tau_i}$, i.e., the Cartesian coordinate functions x^l and all their first partial derivatives are continuous on the parameter space I. The result of this fact is that a grid computed by evaluating the Cartesian coordinate functions along constant parameter lines leads to smooth grid lines with smoothly varying tangents.

Another result of the construction is that all the second partials of the Cartesian coordinate functions exist and are continuous on I, except possibly at points that correspond to break points for the b–spline basis functions. Thus all the terms in the usual elliptic grid generation equations may be evaluated on I and collocation may be used to adjust the expansion coefficients $A^l_{\tau_1\tau_2\tau_3}$ in (6) to obtain a solution. The detailed development of the collocation equations is given in the next section.

2.2 Collocation Equations for Elliptic Grid Generation

For a block $\mathcal{B}$, the elliptic grid generation equations may be formulated on the parameter space I, [3]

$$0 = \sum_{i=1}^{3}\sum_{j=1}^{3} g^{ij} x^l_{\xi^j\xi^i} + \sum_{i=1}^{3} g^{ii} P_i x^l_{\xi^i}, \tag{7}$$

where x^l are the Cartesian coordinate functions, ξ^i are the curvilinear coordinates, g^{ij} are the contravariant metric elements and P_i are the control functions.

As pointed out in Section 2.1, all the terms in the elliptic grid generation equations (7) may be evaluated on I and collocation may be used to adjust the expansion coefficients $A^l_{\tau_1\tau_2\tau_3}$ in (6) to obtain a solution. We begin the development of the collocation equations by first defining collocation points on the interval $[0,1]$ for each of the curvilinear coordinates. For the coordinate ξ^i, we assign k_i collocation points to each of the l_i intervals defined in Section 2.1. As suggested in [1], the collocation points are distributed in each interval according to the roots of the k_i-th Legendre polynomial. This construction gives $k_il_i = n_i-2$ collocation points $P^i_{\tau_i}$ for the interval $[0,1]$

$$P^i_{k_i(L_i-1)+K_i} = \left(s^i_{L_i+1} + s^i_{L_i} + \rho_{K_i}\left(s^i_{L_i+1} - s^i_{L_i}\right)\right)/2, \qquad L_i = 1,\ldots,l_i \tag{8}$$

where $\rho_{K_i}, K_i = 1,\ldots,k_i$ are the roots of the $k_i - th$ Legendre polynomial.

We may now define the $(n_1-2)\times(n_2-2)\times(n_3-2)$ collocation points $P_{\tau_1\tau_2\tau_3}$ for the parameter space I as the cross-product of the collocation points on the intervals $[0,1]$ for coordinates ξ^i defined in (8)

$$P_{\tau_1\tau_2\tau_3} = \left(P^1_{\tau_1}, P^2_{\tau_2}, P^3_{\tau_3}\right). \tag{9}$$

To derive the collocation equations, we first compute the Cartesian coordinate functions and their partial derivatives at the collocation points $P_{\tau_1\tau_2\tau_3}$ and then substitute into the grid generation equations (7); the details of the derivation are presented below. Notice that a final interchange of the order of summation leads to a discrete approximation of the generating equation at the collocation points. (The notation $[\cdot]_{\tau_1\tau_2\tau_3}$ used in the equations below indicates that the function inside the brackets is evaluated at the collocation point $P_{\tau_1\tau_2\tau_3}$.)

$$\begin{aligned}
0 &= \sum_{i=1}^{3}\sum_{j=1}^{3}\left[g^{ij}\right]_{\tau_1\tau_2\tau_3}\left[x^l_{\xi^j\xi^i}\right]_{\tau_1\tau_2\tau_3} + \sum_{i=1}^{3}\left[g^{ii}P_i\right]_{\tau_1\tau_2\tau_3}\left[x^l_{\xi^i}\right]_{\tau_1\tau_2\tau_3}, && (10)\\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}\left[g^{ij}\right]_{\tau_1\tau_2\tau_3}\sum_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3}\left[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i\xi^j}\right]_{\tau_1\tau_2\tau_3}\\
&\qquad + \sum_{i=1}^{3}\left[g^{ii}P_i\right]_{\tau_1\tau_2\tau_3}\sum_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3}\left[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i}\right]_{\tau_1\tau_2\tau_3}, && (11)\\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{\sigma_1\sigma_2\sigma_3}\left[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i\xi^j}\right]_{\tau_1\tau_2\tau_3}\left[g^{ij}\right]_{\tau_1\tau_2\tau_3} A^l_{\sigma_1\sigma_2\sigma_3}\\
&\qquad + \sum_{i=1}^{3}\sum_{\sigma_1\sigma_2\sigma_3}\left[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i}\right]_{\tau_1\tau_2\tau_3}\left[g^{ii}P_i\right]_{\tau_1\tau_2\tau_3} A^l_{\sigma_1\sigma_2\sigma_3}, && (12)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{\sigma_1\sigma_2\sigma_3} \left(\sum_{i=1}^{3}\sum_{j=1}^{3} \left[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i\xi^j}\right]_{\tau_1\tau_2\tau_3} \left[g^{ij}\right]_{\tau_1\tau_2\tau_3} \right. \\
&\qquad \left. + \sum_{i=1}^{3} \left[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i}\right]_{\tau_1\tau_2\tau_3} \left[g^{ii}P_i\right]_{\tau_1\tau_2\tau_3} \right) A^l_{\sigma_1\sigma_2\sigma_3}, \qquad (13)\\
&= \sum_{\sigma_1\sigma_2\sigma_3} [B]^{\tau_1\tau_2\tau_3}_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3}, \qquad (14)
\end{aligned}
$$

where

$$
\begin{aligned}
[B]^{\tau_1\tau_2\tau_3}_{\sigma_1\sigma_2\sigma_3} &= \sum_{i=1}^{3}\sum_{j=1}^{3} \left[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i\xi^j}\right]_{\tau_1\tau_2\tau_3} \left[g^{ij}\right]_{\tau_1\tau_2\tau_3} \\
&\qquad + \sum_{i=1}^{3} \left[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i}\right]_{\tau_1\tau_2\tau_3} \left[g^{ii}P_i\right]_{\tau_1\tau_2\tau_3}. \qquad (15)
\end{aligned}
$$

This derivation leads to $(n_1-2)\times(n_2-2)\times(n_3-2)$ collocation equations for a block $\mathcal{B}$, but there are $n_1 \times n_2 \times n_3$ expansion coefficients $A^l_{\tau_1\tau_2\tau_3}$ in the representation of the Cartesian coordinate functions. The additional equations needed to complete the formulation of the method for a block $\mathcal{B}$ are the boundary condition equations developed in the next section.

2.3 Boundary Conditions for the Coordinate Functions

For a block $\mathcal{B}$, we obtain the boundary condition equations for the coefficients $A^l_{\tau_1\tau_2\tau_3}$ in (6) by first interpolating the coordinate data for the six faces of $\mathcal{B}$. We use the b-spline basis functions defined in Section 2.1 to form a tensor product b-spline expansion for the faces of $\mathcal{B}$ on the faces of the parameter space I and interpolate the coordinate data for the faces to compute the expansion coefficients.

$$
\begin{aligned}
F^{l,1}\left(\xi^2,\xi^3\right) &= \sum_{\tau_2\tau_3} f^{l,1}_{\tau_2\tau_3} U^2_{\tau_2}\left(\xi^2\right) U^3_{\tau_3}\left(\xi^3\right), \qquad (16)\\
F^{l,2}\left(\xi^2,\xi^3\right) &= \sum_{\tau_2\tau_3} f^{l,2}_{\tau_2\tau_3} U^2_{\tau_2}\left(\xi^2\right) U^3_{\tau_3}\left(\xi^3\right), \qquad (17)\\
F^{l,3}\left(\xi^3,\xi^1\right) &= \sum_{\tau_3\tau_1} f^{l,3}_{\tau_3\tau_1} U^3_{\tau_3}\left(\xi^3\right) U^1_{\tau_1}\left(\xi^1\right), \qquad (18)\\
F^{l,4}\left(\xi^3,\xi^1\right) &= \sum_{\tau_3\tau_1} f^{l,4}_{\tau_3\tau_1} U^3_{\tau_3}\left(\xi^3\right) U^1_{\tau_1}\left(\xi^1\right), \qquad (19)\\
F^{l,5}\left(\xi^1,\xi^2\right) &= \sum_{\tau_1\tau_2} f^{l,5}_{\tau_1\tau_2} U^1_{\tau_1}\left(\xi^1\right) U^2_{\tau_2}\left(\xi^2\right), \qquad (20)\\
F^{l,6}\left(\xi^1,\xi^2\right) &= \sum_{\tau_1\tau_2} f^{l,6}_{\tau_1\tau_2} U^1_{\tau_1}\left(\xi^1\right) U^2_{\tau_2}\left(\xi^2\right). \qquad (21)
\end{aligned}
$$

We note that $2\times n_2\times n_3 + 2\times n_3\times n_1 + 2\times n_1\times n_2$ coefficients appear in the specification of the b-spline representation of the faces in (16) to (21). However, these coefficients are not independent because the faces must be consistent along the block edges and at block corners; there are $4 \times (n_1 - 2) + 4 \times (n_2 - 2) + 4 \times (n_3 - 2) + 8$ consistency conditions.

Next, we consider the boundary conditions for the elliptic grid generation equations for a block $\mathcal{B}$, which require that the Cartesian coordinate functions x^l map the boundary of the parameter space I one–to–one onto the boundary of $\mathcal{B}$. When the Cartesian coordinate functions have the parametric representation (6) on I, this boundary condition defines parametric representations of the faces of $\mathcal{B}$. The faces are defined by fixing one of the curvilinear coordinates ξ^i to be 0 or 1

$$\begin{aligned}
x^l\left(0,\xi^2,\xi^3\right) &= \sum_{\tau_1\tau_2\tau_3} A^l_{\tau_1\tau_2\tau_3} U^1_{\tau_1}(0)\, U^2_{\tau_2}\left(\xi^2\right) U^3_{\tau_3}\left(\xi^3\right), && (22)\\
x^l\left(1,\xi^2,\xi^3\right) &= \sum_{\tau_1\tau_2\tau_3} A^l_{\tau_1\tau_2\tau_3} U^1_{\tau_1}(1)\, U^2_{\tau_2}\left(\xi^2\right) U^3_{\tau_3}\left(\xi^3\right), && (23)\\
x^l\left(\xi^1,0,\xi^3\right) &= \sum_{\tau_1\tau_2\tau_3} A^l_{\tau_1\tau_2\tau_3} U^1_{\tau_1}\left(\xi^1\right) U^2_{\tau_2}(0)\, U^3_{\tau_3}\left(\xi^3\right), && (24)\\
x^l\left(\xi^1,1,\xi^3\right) &= \sum_{\tau_1\tau_2\tau_3} A^l_{\tau_1\tau_2\tau_3} U^1_{\tau_1}\left(\xi^1\right) U^2_{\tau_2}(1)\, U^3_{\tau_3}\left(\xi^3\right), && (25)\\
x^l\left(\xi^1,\xi^2,0\right) &= \sum_{\tau_1\tau_2\tau_3} A^l_{\tau_1\tau_2\tau_3} U^1_{\tau_1}\left(\xi^1\right) U^2_{\tau_2}\left(\xi^2\right) U^3_{\tau_3}(0), && (26)\\
x^l\left(\xi^1,\xi^2,1\right) &= \sum_{\tau_1\tau_2\tau_3} A^l_{\tau_1\tau_2\tau_3} U^1_{\tau_1}\left(\xi^1\right) U^2_{\tau_2}\left(\xi^2\right) U^3_{\tau_3}(1). && (27)
\end{aligned}$$

Comparing these two representations of the boundary of $\mathcal{B}$, we observe that the expansion coefficients $A^l_{\tau_1\tau_2\tau_3}$ in (6) must satisfy the boundary condition equations

$$\begin{aligned}
\sum_{\tau_1} A^l_{\tau_1\tau_2\tau_3} U^1_{\tau_1}(0) &= f^{l,1}_{\tau_2\tau_3}, && (28)\\
\sum_{\tau_1} A^l_{\tau_1\tau_2\tau_3} U^1_{\tau_1}(1) &= f^{l,2}_{\tau_2\tau_3}, && (29)\\
\sum_{\tau_2} A^l_{\tau_1\tau_2\tau_3} U^2_{\tau_2}(0) &= f^{l,3}_{\tau_3\tau_1}, && (30)\\
\sum_{\tau_2} A^l_{\tau_1\tau_2\tau_3} U^2_{\tau_2}(1) &= f^{l,4}_{\tau_3\tau_1}, && (31)\\
\sum_{\tau_3} A^l_{\tau_1\tau_2\tau_3} U^3_{\tau_3}(0) &= f^{l,5}_{\tau_1\tau_2}, && (32)\\
\sum_{\tau_3} A^l_{\tau_1\tau_2\tau_3} U^3_{\tau_3}(1) &= f^{l,6}_{\tau_1\tau_2}. && (33)
\end{aligned}$$

The above set of equations is redundant because the faces of $\mathcal{B}$ must satisfy the corner and edge consistency conditions. Thus, we must eliminate $4\times(n_1-2)+4\times(n_2-2)+4\times(n_3-2)+8$ equations. It is convenient to do this by replacing pairs of equations along an edge with a single edge boundary condition equation, and similarly, triples of equations at corners with a single corner boundary condition equation. We do not present the details, but the procedure leads to a set of $2 \times (n_2 - 2) \times (n_3 - 2) + 2 \times (n_3 - 2) \times (n_1 - 2) + 2 \times (n_1 - 2) \times (n_2 - 2)$ face boundary condition equations, $4 \times (n_1 - 2) + 4 \times (n_2 - 2) + 4 \times (n_3 - 2)$ edge boundary condition equations and 8 corner boundary condition equations.

When combined with the set of $(n_1 - 2) \times (n_2 - 2) \times (n_3 - 2)$ collocation equations derived in Section 2.2, the set of boundary condition equations constructed above completes the set of equations needed to formulate the tensor product b–spline method for a block $\mathcal{B}$. The full set consists of $n_1 \times n_2 \times n_3$ equations for the $n_1 \times n_2 \times n_3$ expansion coefficients $A^l_{\tau_1\tau_2\tau_3}$ in (6).

3 APPLICATION of the METHOD

We have developed a block solver for the tensor product b–spline method for application in a multi–block grid code. The solution algorithm is designed to solve the system of equations derived in Sections 2.2 and 2.3 for the expansion coefficients $A^l_{\tau_1\tau_2\tau_3}$ in (6). The formulation of the solver is analogous to the ADI method with SOR commonly used for elliptic grid generation. The parametric representations of the block faces, needed to define the boundary condition equations for each block, are computed by interpolating coordinate data for the faces obtained from the block interface conditions.

In Section 3.1 we investigate the structure of the system of equations for the expansion coefficients for a block, and then in Section 3.2 we describe the block solution algorithm.

3.1 Structure of System Equations

To formulate a solution algorithm for the tensor product b–spline method, we first investigate the structure of the system of equations derived in Sections 2.2 and 2.3 for the expansion coefficients $A^l_{\tau_1\tau_2\tau_3}$ in (6). The structure of the collocation equations may be identified by noting that the small support of the b–spline basis functions makes it possible to reduce the number of terms in (14). We consider a particular collocation point $P_{\tau_1\tau_2\tau_3}$, where L_i and K_i are fixed in the intervals

$$1 \leq \; L_i \; \leq l_i, \tag{34}$$

$$1 \leq \; K_i \; \leq k_i, \tag{35}$$

and

$$\tau_i \;=\; k_i(L_i - 1) + K_i, \tag{36}$$

and observe that the tensor product b–spline basis function $U_{\sigma_1\sigma_2\sigma_3}$ is nonzero at $P_{\tau_1\tau_2\tau_3}$ only if

$$k_i(L_i - 1) + 1 \leq \sigma_i \leq k_i(L_i - 1) + m_i. \tag{37}$$

Thus, the collocation equation (14) has only $m_1 \times m_2 \times m_3$ nonzero terms for each collocation point, and we may reduce (14) to the form

$$0 = \sum_{\nu_1=1}^{m_1} \sum_{\nu_2=1}^{m_2} \sum_{\nu_3=1}^{m_3} [B]^{\tau_1\tau_2\tau_3}_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3}, \tag{38}$$

where

$$\sigma_i = k_i(L_i - 1) + \nu_i. \tag{39}$$

In a similar fashion, we observe that the boundary condition equations (28) and (29) have only $m_1 - 1$ nonzero terms, (30) and (31) have only $m_2 - 1$ nonzero terms, and (32) and (33) have only $m_3 - 1$ nonzero terms.

The result of these observations is that the system of equations for the expansion coefficients $A^l_{\tau_1\tau_2\tau_3}$ has a familiar block, banded structure. When $k_1 = k_2 = k_3 = 1$, the b–spline basis functions are quadratics and the structure is similar to the structure of the finite difference equations for elliptic grid generation when 2-rd order centered differences are used.

3.2 Solution Algorithm for System Equations

Using the structure of the system equations identified in Section 3.1, we have formulated a solution algorithm for the tensor product b–spline method that is analogous to the ADI method with SOR commonly used for elliptic grid generation. The algorithm sweeps over lines of collocation points and updates corresponding lines of expansion coefficients. The corrections to the expansion coefficients are computed by solving a system of equations derived from the collocation and boundary condition equations for the line.

We begin by establishing a correspondence between collocation points and expansion coefficients. For a collocation point $P_{\tau_1\tau_2\tau_3}$, where L_i and K_i are fixed in the intervals

$$1 \leq L_i \leq l_i, \tag{40}$$

$$1 \leq K_i \leq k_i, \tag{41}$$

and

$$\tau_i = k_i(L_i - 1) + K_i, \tag{42}$$

we specify that the corresponding expansion coefficient $A^l_{\sigma_1\sigma_2\sigma_3}$ has the indices

$$\sigma_i = k_i(L_i - 1) + K_i + 1. \tag{43}$$

For the sweep over τ_1-lines (τ_1 varies, τ_2 and τ_3 are fixed) of collocation points, we define the corrections for the corresponding sil-line (σ_1 varies, σ_2 and σ_3 are fixed) of expansion coefficients

$$A^{l,(n+1)}_{\sigma_1\sigma_2\sigma_3} = A^{l,(n)}_{\sigma_1\sigma_2\sigma_3} + \delta A^{l,(n)}_{\sigma_1\sigma_2\sigma_3}, \tag{44}$$

where the indices $(\sigma_1\sigma_2\sigma_3)$ correspond to the indices $(\tau_1\tau_2\tau_3)$ as defined in (43). The superscript (n) in (44) refers to the iteration number. Lines already processed in the sweep over τ_1-lines of collocation points are at iteration $(n+1)$; the current and succeeding lines are at iteration (n).

Finally, we derive the equations defining the correction terms by substituting (44) into the collocation equation (39) for the τ_1-line of collocation points $P_{\tau_1\tau_2\tau_3}$, where L_1 and K_1 vary and L_2, K_2, L_3 and K_3 are fixed in the intervals

$$1 \le L_i \le l_i, \tag{45}$$
$$1 \le K_i \le k_i. \tag{46}$$

and rearranging terms to obtain

$$\sum_{\nu_1=1}^{m_1} [B]^{\tau_1\tau_2\tau_3}_{\sigma_1\tilde{\sigma}_2\tilde{\sigma}_3} \delta A^{l,(n)}_{\sigma_1\tilde{\sigma}_2\tilde{\sigma}_3}, = -\sum_{\nu_1=1}^{m_1}\sum_{\nu_2=1}^{m_2}\sum_{\nu_3=1}^{m_3} [B]^{\tau_1\tau_2\tau_3}_{\sigma_1\sigma_2\sigma_3} A^{l,(*)}_{\sigma_1\sigma_2\sigma_3}, \tag{47}$$

where

$$\tau_i = k_i(L_i - 1) + K_i \tag{48}$$
$$\tilde{\sigma}_i = k_i(L_i - 1) + K_i + 1 \tag{49}$$
$$\sigma_i = k_i(L_i - 1) + \nu_i. \tag{50}$$

The superscript $(*)$ in (47) indicates that the most recent iterate of the expansion coefficients are used in the sum.

This derivation leads to $n_1 - 2$ equations for the n_1 correction terms $\delta A^{l,(n)}_{\sigma_1\tilde{\sigma}_2\tilde{\sigma}_3}$ for the σ_1-line of expansion coefficients. The two additional equations needed to complete the system of equations for the correction terms are derived using the boundary condition equations for $A^l_{1\tilde{\sigma}_2\tilde{\sigma}_3}$ and $A^l_{n_1\tilde{\sigma}_2\tilde{\sigma}_3}$.

The resulting system of n_1 equations for the n_1 correction terms for the σ_1-line of expansion coefficients has a banded structure; the lower and upper bands are of width k_1. We use a band solver to compute the correction terms and then use SOR to apply the corrections to the expansion coefficients.

4 RESULTS of the RESEARCH

To test the tensor product b–spline method, we have implemented the block solver described in Section 3.2 in a 2D multi–block grid code previously developed by the author, [2]. In the modified grid code, the elliptic grid generation equations are solved using either the original block solver for the finite difference method or the block solver for the tensor producr b–spline method. With either solver, the generation equations are solved block by block, while maintaining the block interface conditions. The solution for each block is iterated using the ADI method with SOR commonly used for elliptic grid generation. Having similar implementations of the finite difference and tensor product b–spline formulations in the modified grid code made it possible to compare the performance of the two methods.

To demonstrate the tensor product b–spline method and compare its performance to the finite difference method we selected two grid generation problems: a C–grid for the NACA–0012 Airfoil and an O–grid for a cross–section of the Model 350 Fighter. The studies conducted with these test cases are reviewed in Sections 4.1 and 4.2.

The results of the studies indicate that the tensor product b–spline method may be applied in a multi–block grid code to compute smooth elliptic grids for practical grid generation problems. The expansion of the coordinate functions in smooth b–spline basis functions makes it possible to compute fine grids in the physical domain using coarse grids in the parameter domain. For the problems studied, this reduction in the dimension of the problem and the corresponding increase in convergence rate, made the method as fast or faster than the finite difference method.

4.1 C–Grid for NACA–0012 Airfoil

Figure 1–a shows a C–grid for the NACA–0012 airfoil computed with the modified multi–block grid code using three blocks and the block solver for the finite difference method. Figure 1–b shows the block boundaries in the computed grid and indicates the number of grid points for each coordinate direction. The total number of grid points for all blocks is 4160. The converged grid was computed with 70 outer iterations over the three blocks with an inner iteration of one line sweep for each coordinate direction for each block. As shown in Figure 1–a the computed grid is a smooth elliptic grid with high resolution near the the airfoil surface.

Figures 2 and 3 illustrate the first study with the tensor product b–spline method for the NACA–0012 airfoil, in which we considered the task of generating a C–grid for the airfoil. Figure 2–a shows a C–grid computed with the modified multi–block grid code using three blocks and the block solver for the tensor product b–spline method. The grid lines in the figure are lines on which one coordinate has a constant value equal to the value of a collocation point for the coordinate. Thus the grid points in the figure are the points at which the elliptic grid generation equations were solved by the tensor product b–spline method. Figure 2–b shows the block boundaries in the computed grid and indicates the number of intervals for each coordinate direction. Quadratic b–splines ($k_i = 1$) were used for all blocks. The total number of b–spline basis functions for all blocks is 256, which is approximately 6.2% of the total number of grid points in the finite difference grid. The converged grid was computed with 10 outer iterations over the three blocks with an inner iteration of one line sweep for each coordinate direction for each block. Figure 3–a shows the C–grid computed by evaluating the tensor product b–spline solution at equally spaced parameter values to obtain the same number of grid points as in the finite difference grid. Figure 3–b shows an expanded view of the C–grid near the airfoil surface. Notice that the fine grid computed from the coarse b–spline grid is as smooth and well resolved as the finite difference grid. The total cost of computing the fine grid with the tensor product b–spline method is 12.7% of the total cost of the finite difference method. The solution phase cost of computing the fine grid with the tensor product b–spline method is 7.6% of the solution phase cost of the finite difference method.

Figures 4 to 7 illustrate the second study with the tensor product b–spline method for the NACA–0012 airfoil, in which we considered the task of generating a fine inner C–grid and a coarse outer C–grid for the airfoil. Figures 4–a and 4–b show the finite difference grid and block boundaries computed for a six block C–grid. The total number of grid points for all blocks is 4290. The converged grid was computed with 70 outer iterations over the six blocks with an inner iteration of one line sweep for each coordinate direction for each block. As shown in Figure 4–a the computed grid is a smooth elliptic grid with high resolution near the the airfoil surface and is nearly identical with the C–grid obtained with three blocks in the first study. Figures 5–a and 5–b show the b–spline grid and block boundaries computed for the six block C–grid. Quadratic b–splines ($k_i = 1$) were used for all blocks.The total number of b–spline basis functions for all blocks is 512, which is approximately 11.9% of the total number of grid points in the finite difference grid. The converged grid was computed with 15 outer iterations over the six blocks with an inner iteration of one line sweep for each coordinate direction for each block. Figure 6–a shows the C–grid computed by evaluating the tensor product b–spline solution at equally spaced parameter values to obtain the same number of grid points as in the finite difference grid. Figure 6–b shows an expanded view of the C–grid near the airfoil surface. Notice that the fine grid computed from the coarse b–spline grid is as smooth and well resolved as the finite difference grid. The total cost of computing the fine grid with the tensor product b–spline method is 33.6% of the total cost of the finite difference method. The solution phase cost of computing the fine grid with the tensor product b–spline method is 27.0% of the solution phase cost of the finite difference method. Figure 7–a shows the C–grid computed by evaluating the tensor product b–spline solution at different parameter spacings in the inner and outer blocks. For the inner blocks, the tensor product b–spline solution is evaluated at equally spaced parameter values to obtain the same number of grid points as in the finite difference grid. For the outer blocks the parameter spacing is doubled to obtain one–half the number of grid points as in the finite difference grid. Figure 7–b shows an expanded view of the C–grid near the airfoil surface.

4.2 O–Grid for Model 350 Fighter

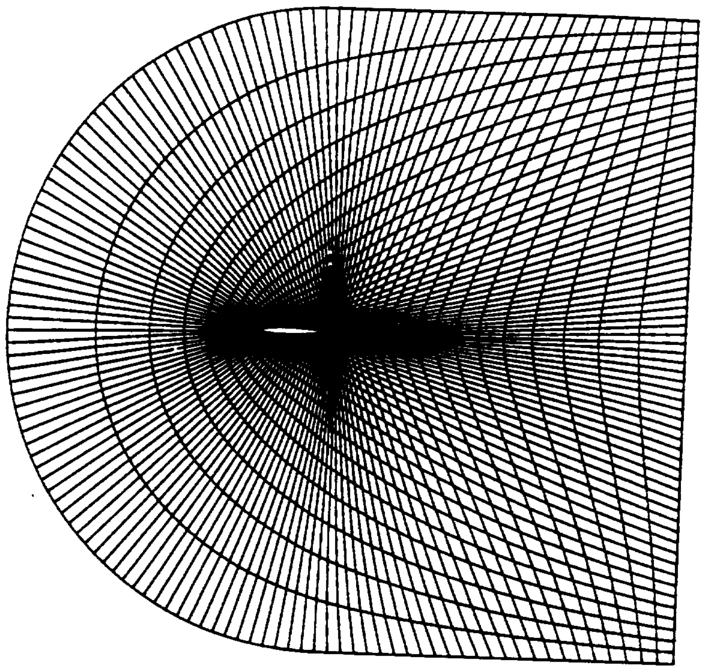
Figures 8 to 11 illustrate the study of a cross–section of the Model 350 Fighter with the tensor product b–spline method, in which we considered the task of generating an O–grid for the fighter. Since the boundary of the cross–section is complex, we used an inner blocking with a fine b–spline grid for accuracy near the boundary and an outer blocking with a coarse b–spline grid for the far-field. Figures 8–a and 8–b show the finite difference grid and block boundaries computed for a four block O–grid for the fighter. The total number of grid points for all blocks is 11132. The convergered grid was computed with 100 outer iterations over the four blocks with an inner iteration of one line sweep for each coordinate direction for each block. As shown in Figure 8–a the computed grid is a smooth elliptic grid with high resolution near the surface of the fighter. Figures 9–a and 9–b show the b–spline grid and block boundaries computed for the four block O–grid. A detailed representation of the cross–section boundary was obtained by using a large number of intervals for the coordinate of the inner block which follows the cross–section boundary. Quadratic b–splines ($k_i = 1$) were used for all blocks. The total number of b–spline basis functions for all blocks is 1920, which is approximately 17.2% of the total number of grid points in the finite difference grid. The converged grid was computed with 40 outer iterations over the four blocks with an inner iteration of one line sweep for each coordinate direction for each block. Figure 10–a shows the O–grid computed by evaluating the tensor product b–spline solution at equally spaced parameter values to obtain the same number of grid points as in the finite difference grid. Figure 10–b shows an expanded view of the O–grid near the fighter surface. Notice that the fine grid computed from the coarse b–spline grid is as smooth and well resolved as the finite difference grid and gives an accurate representation of the fighter boundary. The total cost of computing the fine grid with the tensor product b–spline method is 58.7% of the total cost of the finite difference method. The solution phase cost of computing the fine grid with the tensor product b–spline method is 55.2% of the solution phase cost of the finite difference method. Figure 11–a shows the O–grid computed by evaluating the tensor product b–spline solution at different parameter spacings in the inner and outer blocks. For the inner blocks, the tensor product b–spline solution is evaluated at equally spaced parameter values to obtain the same number of grid points as in the finite difference grid. For the outer blocks the parameter spacing is doubled to obtain one–half the number of grid points as in the finite difference grid. Figure 11–b shows an expanded view of the O–grid near the fighter surface.

References

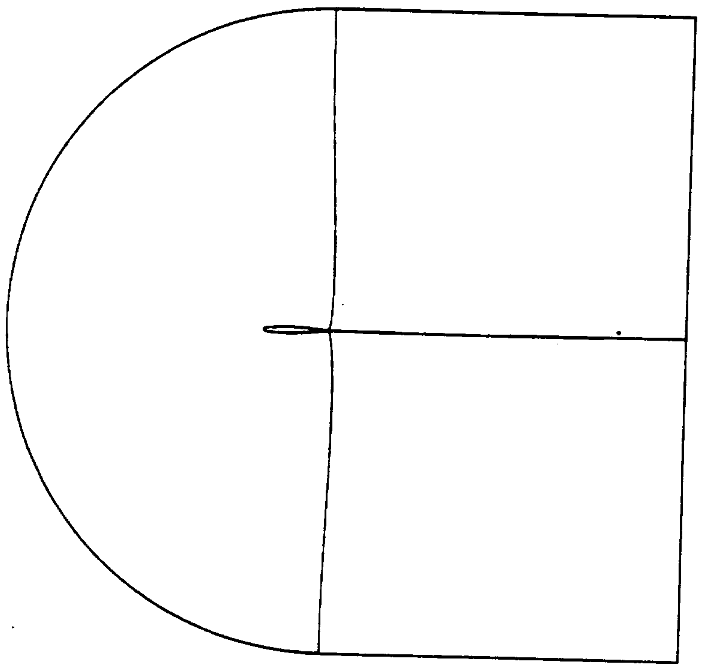
[1] C. de Boor. *A Practical Guide to Splines.* Volume 27 of *Applied Mathematical Sciences,* Springer–Verlag, 1978. [ref72].

[2] J. W. Manke. *Demonstration of a Multi–Block Grid Generation Code.* Technical Report ETA–TR–66, Boeing Computer Services Co., December 1987. [ref71].

[3] J. F. Thompson, editor. *Numerical Grid Generation.* North–Holland, 1982. [ref11].

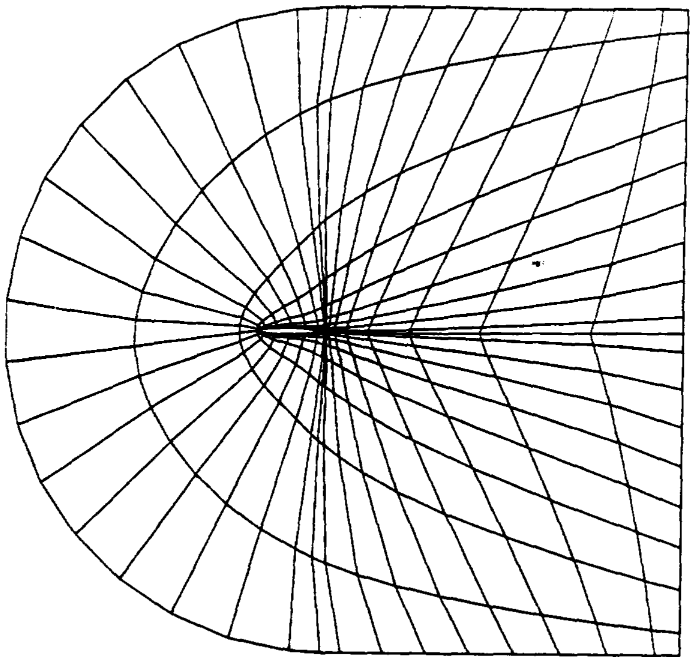


a. Three Block Finite Difference Grid

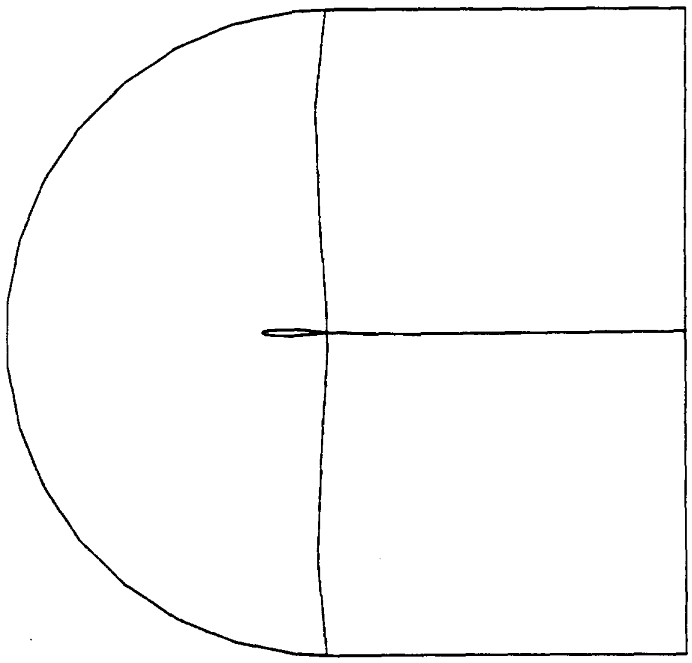


b. Block Boundaries

Figure 1: C–Grid for NACA–0012 Airfoil

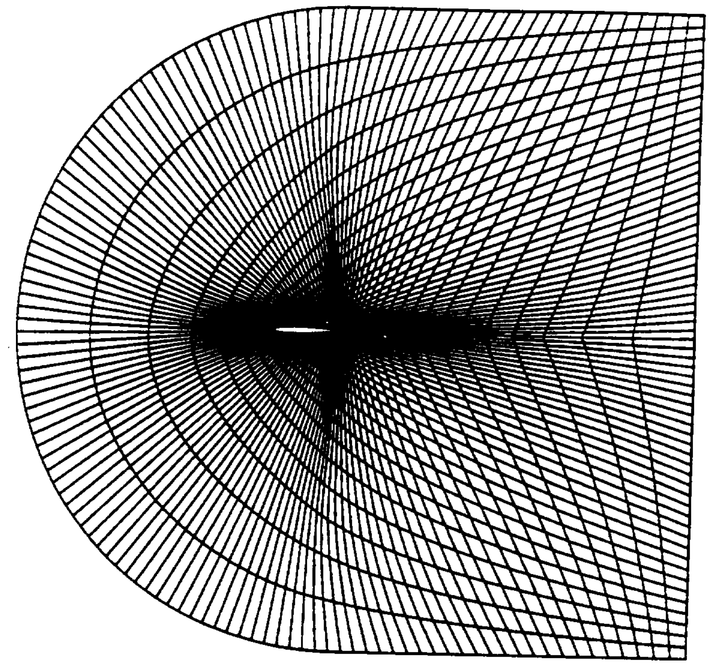


a. Three Block Tensor Product b–Spline Grid

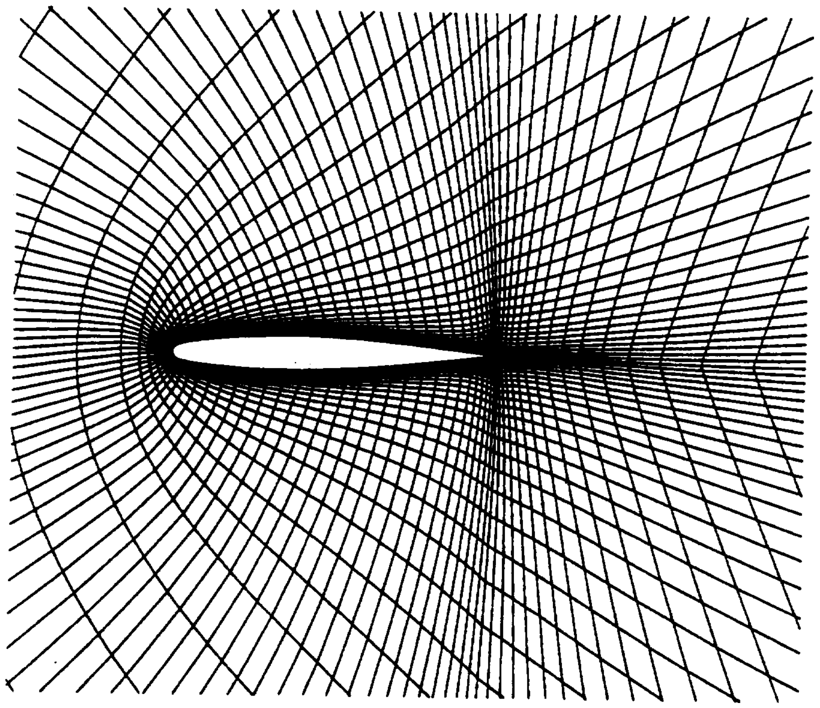


b. Block Boundaries

Figure 2: C–Grid for NACA–0012 Airfoil

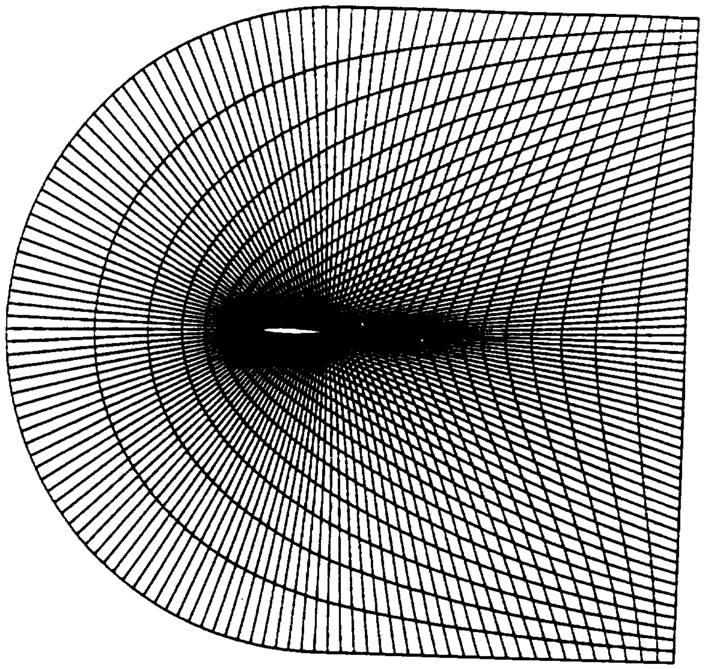


a. Fine Grid Computed from Tensor Product b–Spline Grid



b. Detail of Fine Grid

Figure 3: C–Grid for NACA–0012 Airfoil


a. Six Block Finite Difference Grid

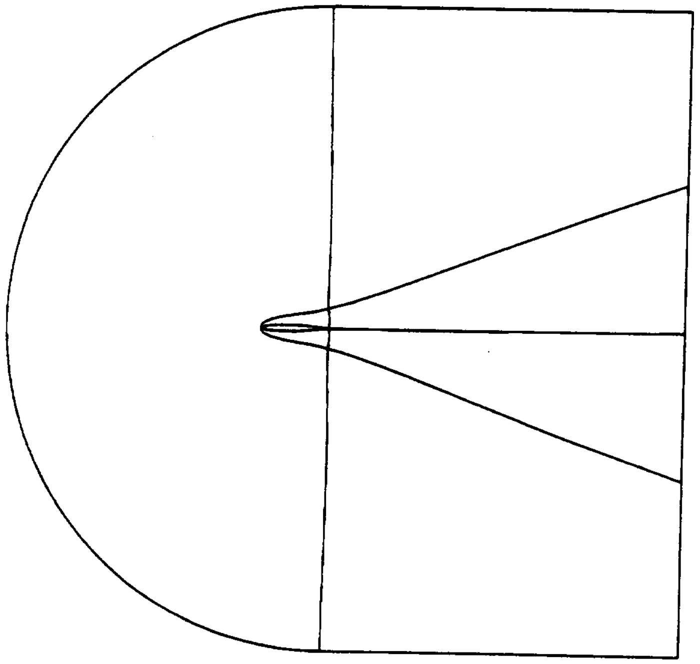
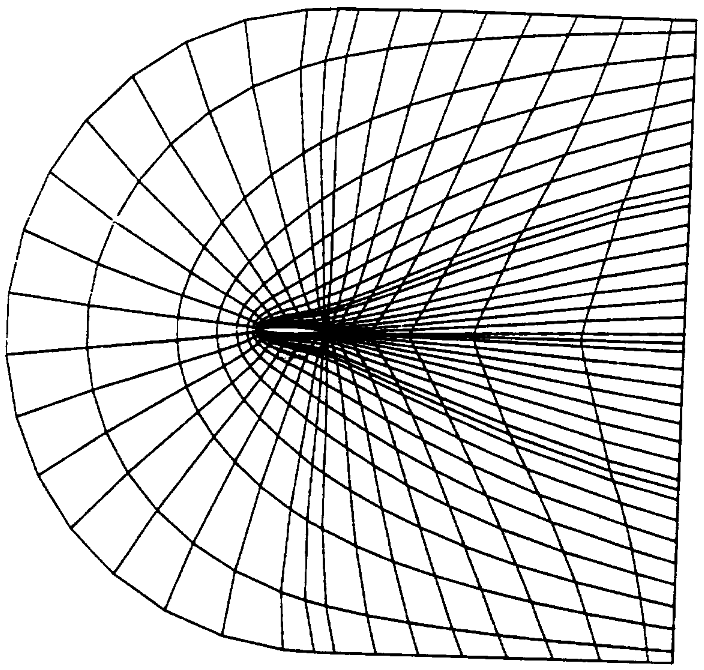

b. Block Boundaries

Figure 4: C–Grid for NACA–0012 Airfoil


a. Six Block Tensor Product b–Spline Grid

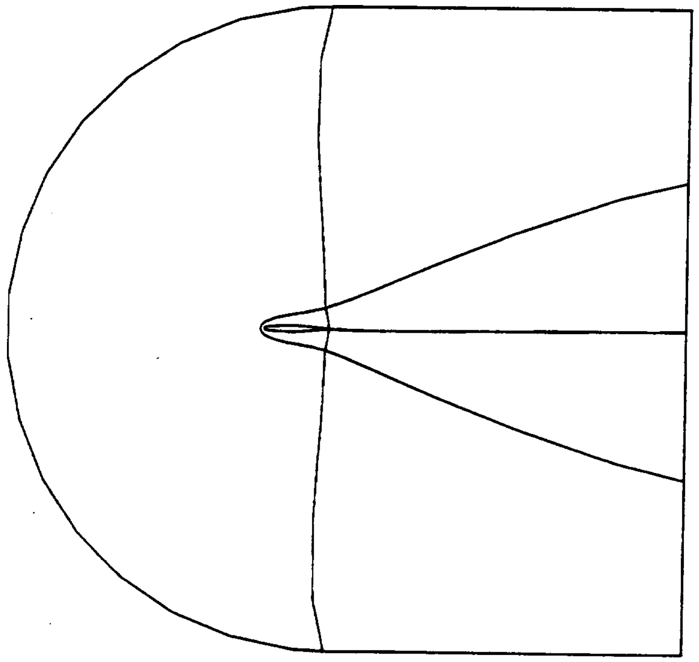
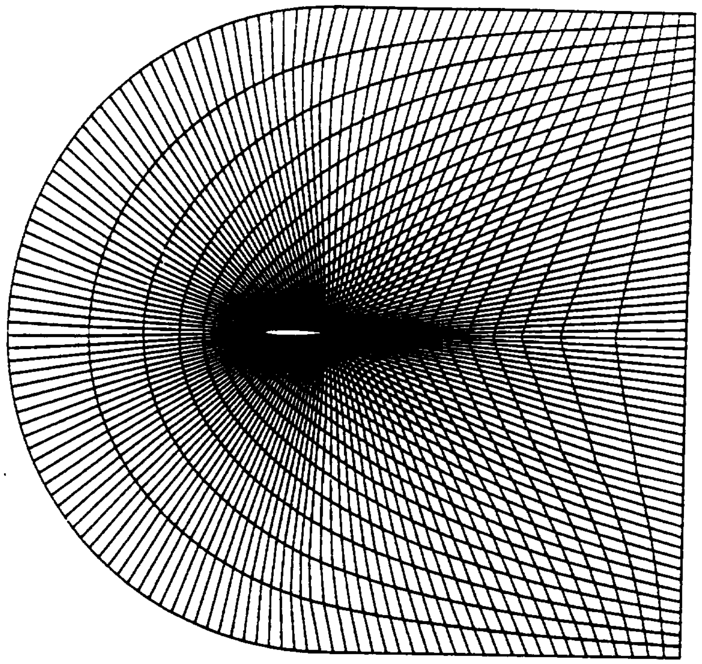
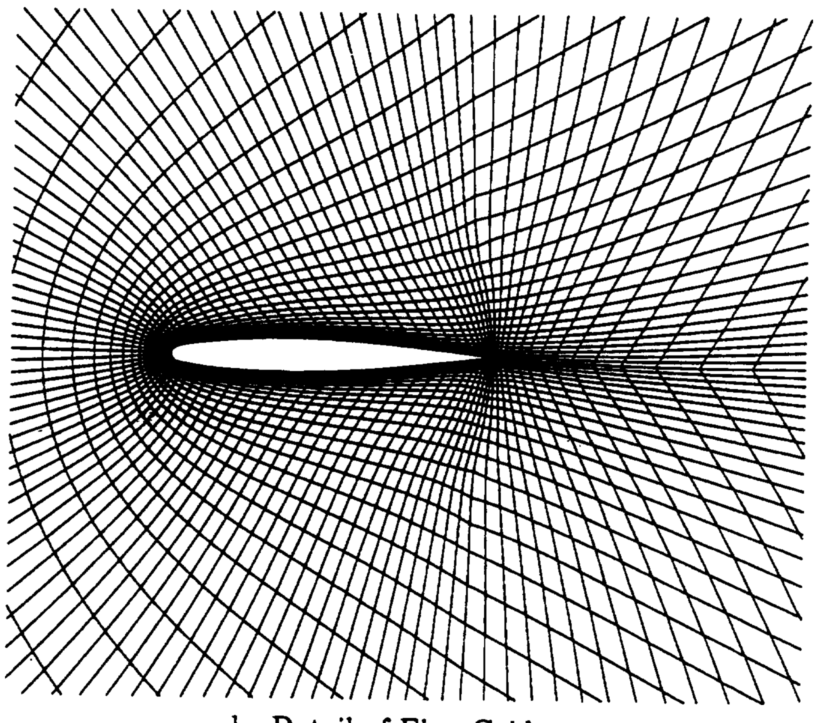

b. Block Boundaries

Figure 5: C–Grid for NACA–0012 Airfoil

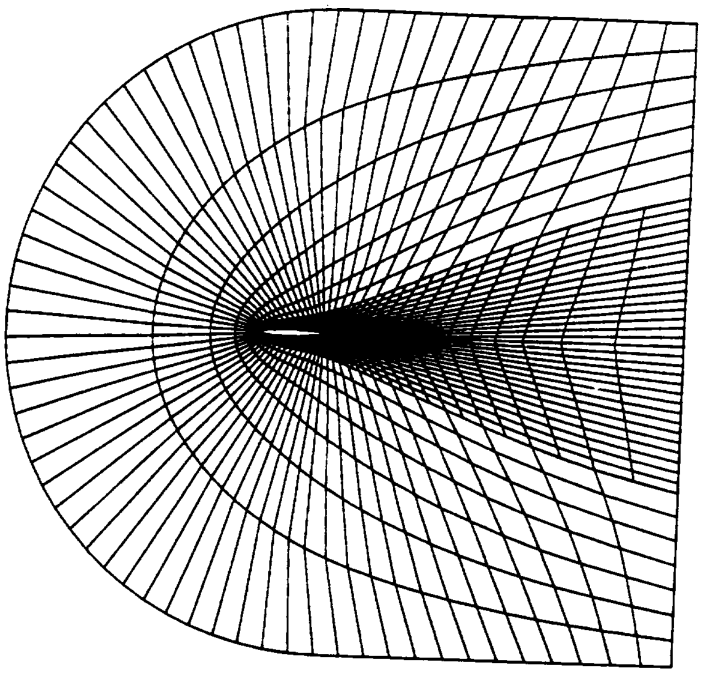


a. Fine Grid Computed from Tensor Product b–Spline Grid

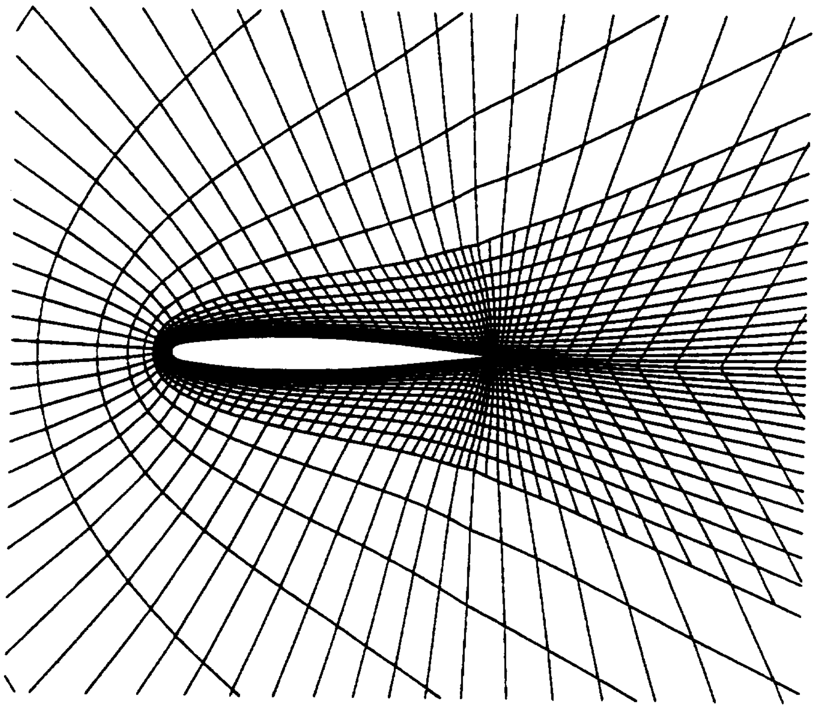


b. Detail of Fine Grid

Figure 6: C–Grid for NACA–0012 Airfoil

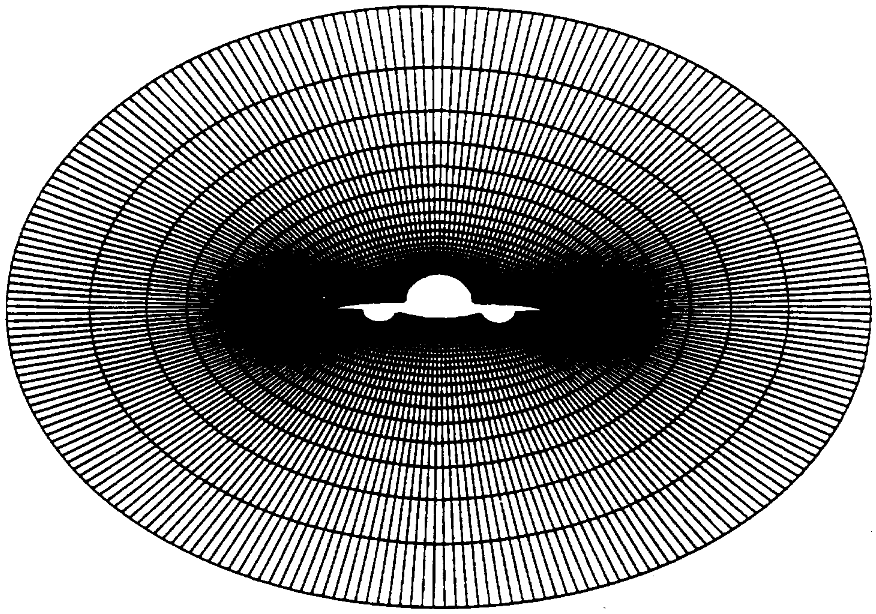


a. Fine Inner Grid, Coarse Outer Grid

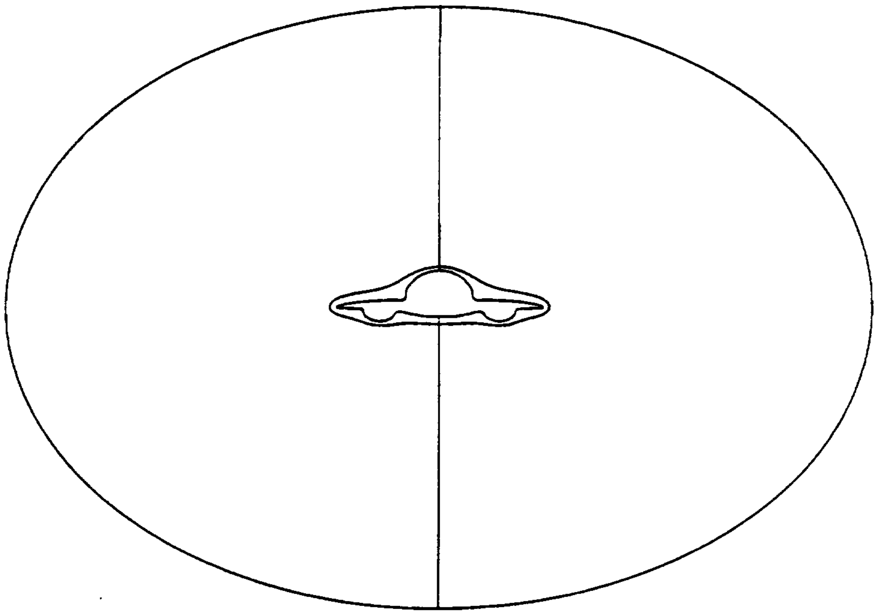


b. Detail of Inner Grid

Figure 7: C–Grid for NACA–0012 Airfoil

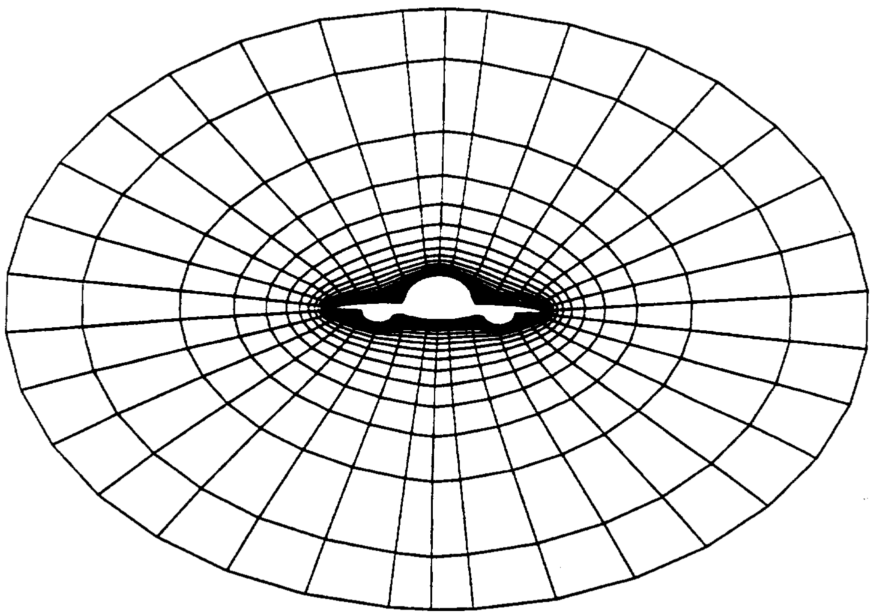


a. Four Block Finite Difference Grid

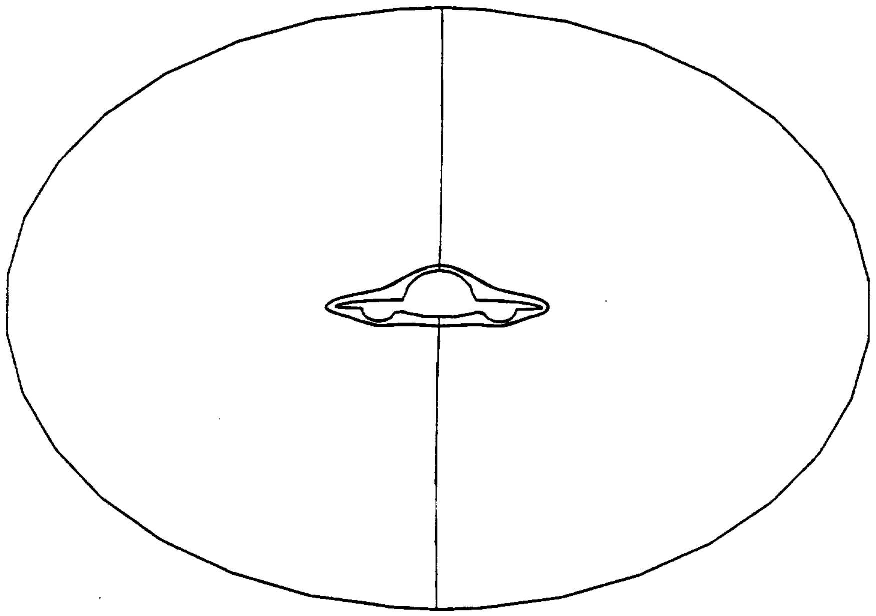


b. Block Boundaries

Figure 8: O–Grid for Model 350 Fighter

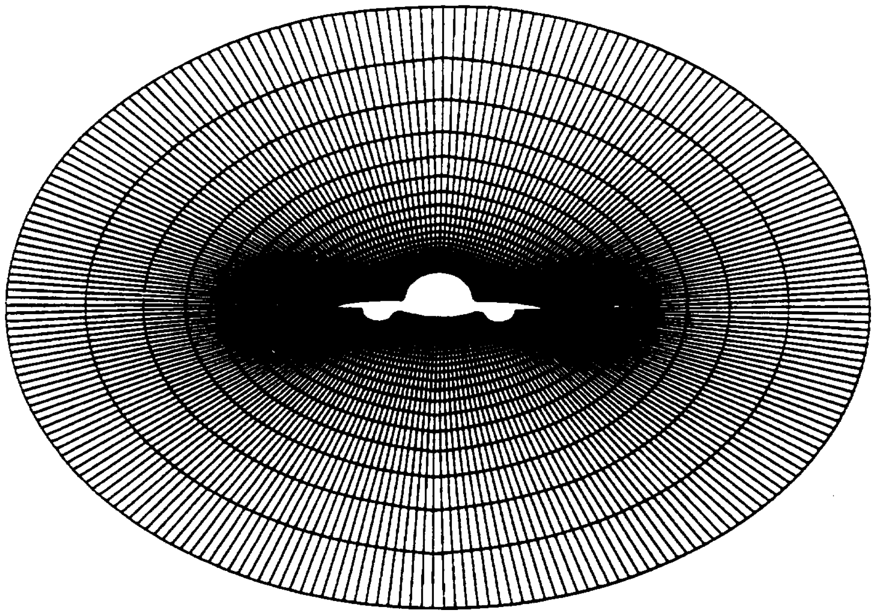


a. Four Block Tensor Product b–Spline Grid

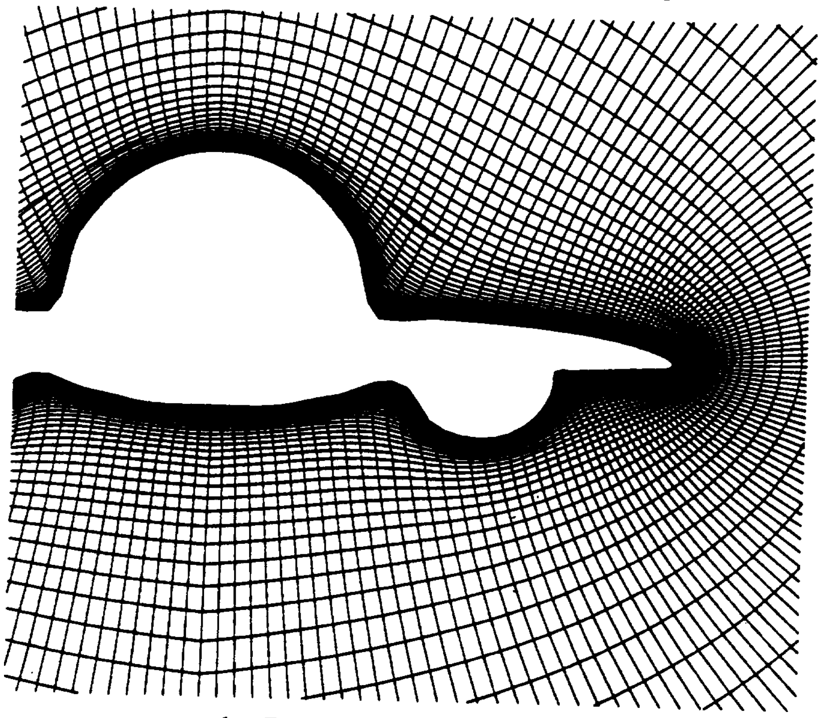


b. Block Boundaries

Figure 9: O–Grid for Model 350 Fighter

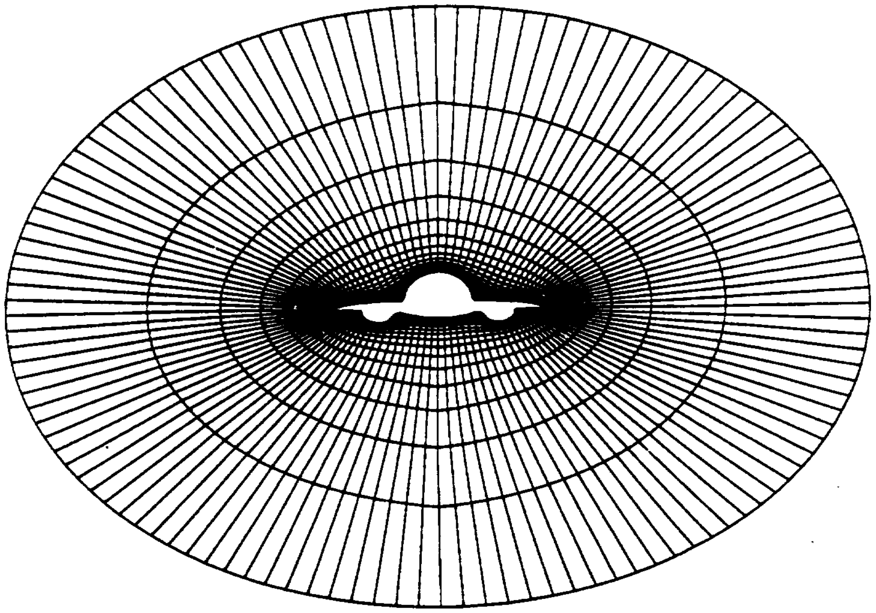


a. Fine Grid Computed from Tensor Product b–Spline Grid

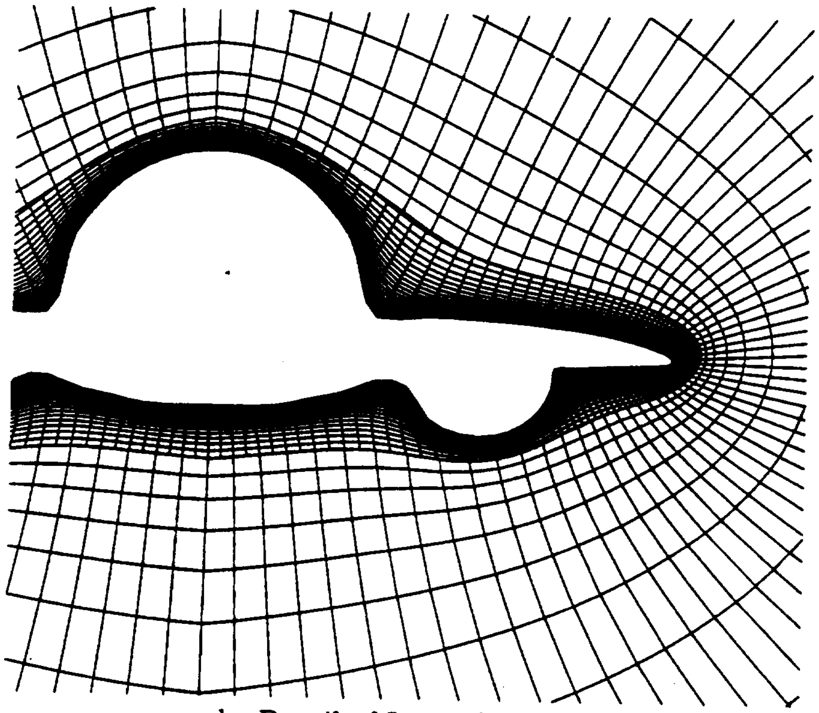


b. Detail of Fine Grid

Figure 10: O–Grid for Model 350 Fighter



a. Fine Inner Grid, Coarse Outer Grid



b. Detail of Inner Grid

Figure 11: O–Grid for Model 350 Fighter