JOURNAL OF COMPUTATIONAL PHYSICS 108, 15–26 (1993)

A Tensor Product B-Spline Method for Numerical Grid Generation

JOSEPH W. MANKE*

Boeing Computer Services Co., Seattle, Washington

Received October 11, 1991

We present a numerical grid generation method in which the Cartesian coordinate functions are expanded in tensor product B-spline basis functions and collocation is used to solve the elliptic grid generation equations. The efficiency of the method derives from the fact that the smoothness of the basis functions is exploited to compute fine grids in the physical domain over a coarse set of knots in the computational domain. We formulate the tensor product B-spline method, investigate its computational complexity and compare its performance to the finite difference method for several 2D grids. We show that for comparable grids the computational cost of the tensor product B-spline method is less than the cost of the finite difference method. © 1993 Academic Press, Inc.

1. INTRODUCTION

We present a numerical grid generation method in which the Cartesian coordinate functions are expanded in tensor product B-spline basis functions and collocation is used to solve the elliptic grid generation equations. We describe the implementation of the method in a multi-block grid generation code and compare its performance to that of the usual finite difference method.

The basic idea of the method is to represent the Cartesian coordinate functions for a grid in the physical domain as a sum of tensor product B-spline basis functions defined on a rectangular grid of knots in the computational domain. The tensor product B-spline basis functions are constructed so that the basis functions and their first partials are continuous on the computational domain. 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 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 collocation equations are solved by an iterative solution method analogous to the solution method used for the finite difference method. An important result of the formulation is that the number of collocation equations is equal to the number of (distinct) interior knots for the tensor product B-spline basis functions. This is to be compared to the finite difference method where the number of finite difference equations is equal to the number of interior grid points. Thus it is possible to reduce the computational cost of the tensor product B-spline 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 of knots in the computational domain. The expected reduction in computational cost is partially offset by the increased complexity of the collocation equations versus the finite difference equations.

In the following sections, we formulate the tensor product B-spline method, investigate its computational complexity and compare its performance to the finite difference method for several 2D grids. For the grid generation problems studied, we show that fine grids comparable in smoothness to fine grids computed by the finite difference method can be computed over a coarse set of knots. We show that for comparable grids the computational cost of the tensor product B-spline method is less than the cost of the finite difference method.

2. ELLIPTIC GRID GENERATION

One of the commonly used numerical grid generation techniques is the elliptic grid generation method. It is a differential equation method, where the grid is computed as the discrete solution of an elliptic partial differential equation on the physical domain [3, 4]. Usually, the

* This research was supported in part by Boeing Computer Services Co., Seattle, WA, by AFOSR-86-0154 and by DOE-627815.

0021-9991/93 $5.00

Copyright © 1993 by Academic Press, Inc.
All rights of reproduction in any form reserved.

boundary conditions are formulated to generate boundary conforming grids, where the boundary of the physical domain is composed of coordinate surfaces. For such grids, numerical generation of the grid is most conveniently performed by transforming the generating equation into a rectangular computational domain. In the following discussion, we briefly introduce the generating equations and the finite difference method commonly used for elliptic grid generation. We also establish some notation needed for the formulation of the tensor product B-spline method.

2.1. *Elliptic Grid Generation Equations*

We consider a 3D physical domain D with a curvilinear coordinate map $(\xi^i(\cdot))$ mapping D onto the 3D computational domain $I^3 = \{(\xi^i)\} = [0, 1] \times [0, 1] \times [0, 1]$ such that $(\xi^i(\cdot))$ is continuous and has nonzero Jacobian on the interior of D. The curvilinear coordinate functions $\xi^i(\cdot)$ are the components of $(\xi^i(\cdot))$. Since the Jacobian of $(\xi^i(\cdot))$ is non-zero on the interior of D, the Cartesian coordinate map $(x^l(\cdot))$ is defined on I^3 as the inverse of $(\xi^i(\cdot))$. The Cartesian coordinate functions $x^l(\cdot)$ are the components of $(x^l(\cdot))$.

For a domain D, the commonly used Poisson generating system with control of the grid point distribution, transformed to I^3, is given by

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

In conjunction with the condition that the boundary of I^3 map onto the boundary of D, (1) defines a well-posed elliptic boundary value problem on I^3 for the Cartesian coordinate map $(x^l(\cdot))$

2.2. *Finite Difference Method*

For the coordinate ξ^i, we define n_i equally spaced points $P^i_{\tau_i}$ on the interval $[0, 1]$

$$P^i_{\tau_i} = \frac{\tau_i - 1}{n_i - 1}, \qquad \tau_i = 1, ..., n_i. \tag{2}$$

We define the $n_1 n_2 n_3$ computational grid points $P^I_{\tau_1\tau_2\tau_3}$ for I^3 as the cross-product of the points on the intervals $[0, 1]$ for coordinates ξ^i defined in (2) as

$$P^I_{\tau_1\tau_2\tau_3} = (P^1_{\tau_1}, P^2_{\tau_2}, P^3_{\tau_3}). \tag{3}$$

We define the $n_1 n_2 n_3$ physical grid points $P^D_{\tau_1\tau_2\tau_3}$ for D as the images of the computational grid points $P^I_{\tau_1\tau_2\tau_3}$ under the Cartesian coordinate map $(x^l(\cdot))$ as

$$P^D_{\tau_1\tau_2\tau_3} = (x^1(P^I_{\tau_1\tau_2\tau_3}), x^2(P^I_{\tau_1\tau_2\tau_3}), x^3(P^I_{\tau_1\tau_2\tau_3})). \tag{4}$$

In the following discussion, the notation $[\cdot]_{\tau_1\tau_2\tau_3}$ indicates that the function inside the brackets is evaluated at the computational grid point $P^I_{\tau_1\tau_2\tau_3}$. In particular, the coordinates of the physical grid points $P^D_{\tau_1\tau_2\tau_3}$ are denoted by $[x^l]_{\tau_1\tau_2\tau_3}$.

To derive the finite difference equations, we write the difference formulas for $[x^l_{\xi^i}]_{\tau_1\tau_2\tau_3}$ and $[x^l_{\xi^j\xi^i}]_{\tau_1\tau_2\tau_3}$ at the interior computational grid points in the form (convenient for later comparison with the tensor product B-spline formulation)

$$[x^l_{\xi^i}]_{\tau_1\tau_2\tau_3} = \sum_{\nu_1\nu_2\nu_3} D^i_{\nu_1\nu_2\nu_3} [x^l]_{\sigma_1\sigma_2\sigma_3}, \tag{5}$$

$$[x^l_{\xi^j\xi^i}]_{\tau_1\tau_2\tau_3} = \sum_{\nu_1\nu_2\nu_3} D^{ji}_{\nu_1\nu_2\nu_3} [x^l]_{\sigma_1\sigma_2\sigma_3}, \tag{6}$$

where the indices $\tau_i = 2, ..., n_i - 1$, the indices ν_i range over the values $-1, 0, 1$, the indices $\sigma_i = \tau_i + \nu_i$ and the coefficients $D^i_{\nu_1\nu_2\nu_3}$ and $D^{ji}_{\nu_1\nu_2\nu_3}$ are defined in Fig. 1 and Fig. 2, respectively.

Substituting (5) and (6) into the elliptic grid generation equations (1), we obtain the finite difference equations

$$\begin{aligned} 0 &= \sum_{i=1}^{3} \sum_{j=1}^{3} [g^{ij}]_{\tau_1\tau_2\tau_3} [x^l_{\xi^j\xi^i}]_{\tau_1\tau_2\tau_3} \\ &\quad + \sum_{i=1}^{3} [g^{ii} Q^i]_{\tau_1\tau_2\tau_3} [x^l_{\xi^i}]_{\tau_1\tau_2\tau_3}, && (7) \\ &= \sum_{\nu_1\nu_2\nu_3} [B]^{\tau_1\tau_2\tau_3}_{\nu_1\nu_2\nu_3} [x^l]_{\sigma_1\sigma_2\sigma_3}, && (8) \end{aligned}$$

where

$$[B]^{\tau_1\tau_2\tau_3}_{\nu_1\nu_2\nu_3} = \sum_{i=1}^{3} \sum_{j=1}^{3} [g^{ij}]_{\tau_1\tau_2\tau_3} D^{ji}_{\nu_1\nu_2\nu_3} + \sum_{i=1}^{3} [g^{ii} Q^i]_{\tau_1\tau_2\tau_3} D^i_{\nu_1\nu_2\nu_3}, \tag{9}$$

the indices $\tau_i = 2, ..., n_i - 1$, the indices ν_i range over the values $-1, 0, 1$, the indices $\sigma_i = \tau_i + \nu_i$ and the coefficients $D^i_{\nu_1\nu_2\nu_3}$ and $D^{ji}_{\nu_1\nu_2\nu_3}$ are defined in Fig. 1 and Fig. 2, respectively.

$D^i_{\nu_1\nu_2\nu_3}$
$D^1_{-1,0,0} = -1/(2h_1)$
$D^1_{+1,0,0} = +1/(2h_1)$
$D^2_{0,-1,0} = -1/(2h_2)$
$D^2_{0,+1,0} = +1/(2h_2)$
$D^3_{0,0,-1} = -1/(2h_3)$
$D^3_{0,0,+1} = +1/(2h_3)$
$D^i_{\nu_1\nu_2\nu_3} = 0$, otherwise

FIG. 1. Coefficients $D^i_{\nu_1\nu_2\nu_3}$ for first partials.

$D^{ji}_{\nu_1\nu_2\nu_3}$
$D^{11}_{0,0,0} = -2/(h_1h_1)$
$D^{11}_{-1,0,0} = D^{11}_{+1,0,0} = +1/(h_1h_1)$
$D^{22}_{0,0,0} = -2/(h_2h_2)$
$D^{22}_{0,-1,0} = D^{22}_{0,+1,0} = +1/(h_2h_2)$
$D^{33}_{0,0,0} = -2/(h_3h_3)$
$D^{33}_{0,0,-1} = D^{33}_{0,0,+1} = +1/(h_3h_3)$
$D^{12}_{+1,-1,0} = D^{12}_{-1,+1,0} = D^{21}_{+1,-1,0} = D^{21}_{-1,+1,0} = -1/(4h_1h_2)$
$D^{12}_{+1,+1,0} = D^{12}_{-1,-1,0} = D^{21}_{+1,+1,0} = D^{21}_{-1,-1,0} = +1/(4h_1h_2)$
$D^{23}_{0,+1,-1} = D^{23}_{0,-1,+1} = D^{32}_{0,+1,-1} = D^{32}_{0,-1,+1} = -1/(4h_2h_3)$
$D^{23}_{0,+1,+1} = D^{23}_{0,-1,-1} = D^{32}_{0,+1,+1} = D^{32}_{0,-1,-1} = +1/(4h_2h_3)$
$D^{31}_{+1,0,-1} = D^{31}_{-1,0,+1} = D^{13}_{+1,0,-1} = D^{13}_{-1,0,+1} = -1/(4h_3h_1)$
$D^{31}_{+1,0,+1} = D^{31}_{-1,0,-1} = D^{13}_{+1,0,+1} = D^{13}_{-1,0,-1} = +1/(4h_3h_1)$
$D^{ji}_{\nu_1\nu_2\nu_3} = 0$, otherwise

FIG. 2. Coefficients $D^{ji}_{\nu_1\nu_2\nu_3}$ for second partials.

This derivation leads to $(n_1-2)(n_2-2)(n_3-2)$ finite difference equations on I^3 for the $(n_1-2)(n_2-2)(n_3-2)$ unknown coordinates of the physical grid points in the interior of D. An iterative method is required to solve the system of finite difference equations because the equations are quasi-linear in the coordinates of the interior physical grid points. The solution method commonly used may be described as an ADI method with line SOR, which sweeps over lines of computational grid points for each coordinate direction and updates the coordinates of the corresponding physical grid points. The corrections to the coordinates are computed by solving a system of equations derived from the finite difference equations for the line of computational grid points. We briefly describe the calculations for a sweep over τ_1-lines of computational grid points.

A τ_1-line consists of a line of computational grid points on which τ_1 varies with $2 \leqslant \tau_1 \leqslant n_1 - 1$, τ_2 is fixed with $2 \leqslant \tau_2 \leqslant n_2 - 1$ and τ_3 is fixed with $2 \leqslant \tau_3 \leqslant n_3 - 1$. A sweep over τ_1-lines of computational grid points consists of all the τ_1-lines where the fixed values of τ_2 and τ_3 vary over their indicated ranges. For each line in the sweep, the coordinates of the physical grid points corresponding to the line are corrected as

$$[x^l]_{\tau_1\tau_2\tau_3} = [x^l]_{\tau_1\tau_2\tau_3} + [\delta x^l]_{\tau_1\tau_2\tau_3}, \tag{10}$$

where the correction terms $[\delta x^l]_{\tau_1\tau_2\tau_3}$ satisfy

$$\sum_{\nu_1} [B]^{\tau_1\tau_2\tau_3}_{\nu_1 00} [\delta x^l]_{\sigma_1\tau_2\tau_3} = -\sum_{\nu_1\nu_2\nu_3} [B]^{\tau_1\tau_2\tau_3}_{\nu_1\nu_2\nu_3} [x^l]^{(*)}_{\sigma_1\sigma_2\sigma_3}, \tag{11}$$

the indices τ_i are the indices of a point on the τ_1-line, the indices ν_i range over the values -1, 0, 1, the indices $\sigma_i = \tau_i + \nu_i$, and the superscript $(*)$ indicates that the most recent value of the coordinates are used in the sum.

The resulting system of $n_1 - 2$ linear equations defining the $n_1 - 2$ correction terms for a τ_1-line has tridiagonal structure and is well conditioned. It may be solved for the correction terms with a standard tridiagonal solver such as the LINPACK subroutine SGTSL.

3. TENSOR PRODUCT B-SPLINE METHOD

In the following discussion we define the B-spline basis functions on I^3 and use them to specify the representation of the Cartesian coordinate functions on I^3. Next we show how the boundary conditions for the elliptic grid generation equations are applied on I^3. Then we derive the collocation equations by requiring that the B-spline representation of the Cartesian coordinate functions satisfy the elliptic grid generation equations at collocation points defined on I^3. Finally, we present the ADI method used to solve the collocation equations.

3.1. *Tensor Product B-Spline Representation*

To define the B-spline basis functions 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 $s^i_{\alpha_i}$ as the endpoints of the intervals

$$s^i_{\alpha_i} = \frac{\alpha_i - 1}{l_i}, \qquad \alpha_i = 1, ..., l_i + 1. \tag{12}$$

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 $t^i_{\beta_i}$, where

$$n_i = k_i l_i + 2, \tag{13}$$

$$m_i = k_i + 2, \tag{14}$$

and

$$t^i_{\beta_i} = \begin{cases} s^i_1 & (\beta_i = 1, ..., m_i), \\ s^i_{1 + \lfloor(\beta_i - m_i + k_i + 1)/k_i\rfloor} & (\beta_i = m_i + 1, ..., n_i), \\ s^i_{l_i + 1} & (\beta_i = n_i + 1, ..., n_i + m_i). \end{cases} \tag{15}$$

We define the B-spline basis functions for the coordinate ξ^i as the n_i normalized B-spline basis functions $U^i_{\sigma_i}$ of order m_i on the interval $[0, 1]$ for the set of knots $t^i_{\beta_i}$ specified in (15) [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. The B-spline basis functions also have small support, i.e., $U^i_{\sigma_i}$ is zero outside the interval $[t^i_{\sigma_i}, t^i_{\sigma_i+m_i}]$.

We define the $n_1 n_2 n_3$ tensor product B-spline basis functions on I^3 as products of the B-spline basis functions $U^i_{\sigma_i}$

$$U_{\sigma_1\sigma_2\sigma_3} = U_{\sigma_1\sigma_2\sigma_3}(\xi^1, \xi^2, \xi^3) = U^1_{\sigma_1}(\xi^1)\, U^2_{\sigma_2}(\xi^2)\, U^3_{\sigma_3}(\xi^3). \tag{16}$$

We use the tensor product B-splines $U_{\sigma_1\sigma_2\sigma_3}$ defined in (16) to construct a parametric representation of the Cartesian coordinate functions x^l on the computational domain I^3,

$$x^l = \sum_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3} U_{\sigma_1\sigma_2\sigma_3}, \tag{17}$$

where the indices $\sigma_i = 1, ..., n_i$ and the $n_1 n_2 n_3$ expansion parameters $A^l_{\sigma_1\sigma_2\sigma_3}$ are to be determined.

With this representation, the Cartesian coordinate functions inherit the smoothness properties of the B-spline basis functions $U^i_{\sigma_i}$; i.e., the Cartesian coordinate functions x^l and all their first partial derivatives are continuous on I^3. 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^3, except possibly at points that correspond to break points for the B-spline basis functions. Thus all the terms in the elliptic grid generation equations (1) may be evaluated on I^3 and collocation may be used to adjust the expansion coefficients $A^l_{\sigma_1\sigma_2\sigma_3}$ in (17) to obtain an approximate solution.

In the following discussion, we will refer to the expansion parameters for which $2 \leqslant \tau_i \leqslant n_i - 1$ as interior expansion parameters and all the rest as boundary expansion parameters.

3.2. *Boundary Condition Equations*

The parametric representation of the Cartesian coordinate functions defined in (17) must agree with the specification the Cartesian coordinate functions on the faces of I^3 (defined by fixing one of the curvilinear coordinates ξ^i to be zero or one). We obtain the boundary condition equations

$$\sum_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3} U^1_{\sigma_1}(0)\, U^2_{\sigma_2}(\xi^2)\, U^3_{\sigma_3}(\xi^3) = \sum_{\sigma_2\sigma_3} f^{l,1}_{\sigma_2\sigma_3} U^2_{\sigma_2}(\xi^2)\, U^3_{\sigma_3}(\xi^3), \tag{18}$$

$$\sum_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3} U^1_{\sigma_1}(1)\, U^2_{\sigma_2}(\xi^2)\, U^3_{\sigma_3}(\xi^3) = \sum_{\sigma_2\sigma_3} f^{l,2}_{\sigma_2\sigma_3} U^2_{\sigma_2}(\xi^2)\, U^3_{\sigma_3}(\xi^3), \tag{19}$$

$$\sum_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3} U^1_{\sigma_1}(\xi^1)\, U^2_{\sigma_2}(0)\, U^3_{\sigma_3}(\xi^3) = \sum_{\sigma_3\sigma_1} f^{l,3}_{\sigma_3\sigma_1} U^3_{\sigma_3}(\xi^3)\, U^1_{\sigma_1}(\xi^1), \tag{20}$$

$$\sum_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3} U^1_{\sigma_1}(\xi^1)\, U^2_{\sigma_2}(1)\, U^3_{\sigma_3}(\xi^3) = \sum_{\sigma_3\sigma_1} f^{l,4}_{\sigma_3\sigma_1} U^3_{\sigma_3}(\xi^3)\, U^1_{\sigma_1}(\xi^1) \tag{21}$$

$$\sum_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3} U^1_{\sigma_1}(\xi^1)\, U^2_{\sigma_2}(\xi^2)\, U^3_{\sigma_3}(0) = \sum_{\sigma_1\sigma_2} f^{l,5}_{\sigma_1\sigma_2} U^1_{\sigma_1}(\xi^1)\, U^2_{\sigma_2}(\xi^2), \tag{22}$$

$$\sum_{\sigma_1\sigma_2\sigma_3} A^l_{\sigma_1\sigma_2\sigma_3} U^1_{\sigma_1}(\xi^1)\, U^2_{\sigma_2}(\xi^2)\, U^3_{\sigma_3}(1) = \sum_{\sigma_1\sigma_2} f^{l,6}_{\sigma_1\sigma_2} U^1_{\sigma_1}(\xi^1)\, U^2_{\sigma_2}(\xi^2), \tag{23}$$

where the indices $\sigma_i = 1, ..., n_i$ and the boundary interpolation parameters $f^{l,1}_{\sigma_2\sigma_3}$, $f^{l,2}_{\sigma_2\sigma_3}$, $f^{l,3}_{\sigma_3\sigma_1}$, $f^{l,4}_{\sigma_3\sigma_1}$, $f^{l,5}_{\sigma_1\sigma_2}$, $f^{l,6}_{\sigma_1\sigma_2}$ are computed by using the B-spline basis functions $U^i_{\sigma_i}$ to interpolate the functions specifying the Cartesian coordinate functions x^l on the faces of I^3.

We note that $2n_2n_3 + 2n_3n_1 + 2n_1n_2$ boundary interpolation parameters appear in the specification of the representation of the block faces in (18) to (23). However, these boundary interpolation parameters are not independent because the faces must be consistent along the block edges and at block corners. Thus the boundary interpolation parameters satisfy $4n_1 + 4n_2 + 4n_3 - 8$ consistency conditions.

We can further reduce the boundary condition equations derived above by applying first the orthogonality properties and then the small support of the B-spline basis functions $U_{\sigma_1\sigma_2\sigma_3}$ to obtain

$$\sum_{\tilde\sigma_1} A^l_{\tilde\sigma_1\sigma_2\sigma_3} U^1_{\tilde\sigma_1}(0) = A^l_{1,\sigma_2,\sigma_3} U^1_1(0) = f^{l,1}_{\sigma_2\sigma_3}, \tag{24}$$

$$\sum_{\tilde\sigma_1} A^l_{\tilde\sigma_1\sigma_2\sigma_3} U^1_{\tilde\sigma_1}(1) = A^l_{n_1,\sigma_2,\sigma_3} U^1_{n_1}(1) = f^{l,2}_{\sigma_2\sigma_3}, \tag{25}$$

$$\sum_{\tilde\sigma_2} A^l_{\sigma_1\tilde\sigma_2\sigma_3} U^2_{\tilde\sigma_2}(0) = A^l_{\sigma_1,1,\sigma_3} U^2_1(0) = f^{l,3}_{\sigma_3\sigma_1}, \tag{26}$$

$$\sum_{\tilde\sigma_2} A^l_{\sigma_1\tilde\sigma_2\sigma_3} U^2_{\tilde\sigma_2}(1) = A^l_{\sigma_1,n_2,\sigma_3} U^2_{n_2}(1) = f^{l,4}_{\sigma_3\sigma_1}, \tag{27}$$

$$\sum_{\tilde\sigma_3} A^l_{\sigma_1\sigma_2\tilde\sigma_3} U^3_{\tilde\sigma_3}(0) = A^l_{\sigma_1,\sigma_2,1} U^3_1(0) f^{l,5}_{\sigma_1\sigma_2}, \tag{28}$$

$$\sum_{\tilde\sigma_3} A^l_{\sigma_1\sigma_2\tilde\sigma_3} U^3_{\tilde\sigma_3}(1) = A^l_{\sigma_1,\sigma_2,n_3} U^3_{n_3}(1) f^{l,6}_{\sigma_1\sigma_2}, \tag{29}$$

where the indices $\sigma_i = 1, ..., n_i$ and the indices $\tilde\sigma_i = 1, ..., n_i$.

This derivation leads to $2n_2n_3 + 2n_3n_1 + 2n_1n_2$ linear boundary conditions equations on the computational domain I^3 for the $2n_2n_3+2n_3n_1+2n_1n_2-4n_1-4n_2-4n_3+8$ boundary expansion parameters. However, this redundant set of equations is consistent because the boundary interpolation parameters satisfy the $4n_1+4n_2+4n_3-8$ consistency conditions discussed above.

3.3. *Collocation Equations*

For the coordinate ξ^i, we assign k_i points to each of the l_i intervals defined in Section 3.1. As suggested in [1], the points are distributed in each interval according to the roots of the k_ith Legendre polynomial. This construction gives $k_il_i=n_i-2$ points $P^i_{\tau_i}$ for the interval [0, 1]

$$P^i_{\tau_i}=(s^i_{\alpha_i+1}+s^i_{\alpha_i}+\rho^i_{\kappa_i}(s^i_{\alpha_i+1}-s^i_{\alpha_i}))/2, \tag{30}$$

where

$$\tau_i=k_i(\alpha_i-1)+\kappa_i, \qquad 1\leqslant\alpha_i\leqslant l_i, \quad 1\leqslant\kappa_i\leqslant k_i, \tag{31}$$

and the $\rho^i_{\kappa_i}$ are the roots of the k_ith Legendre polynomial.

We define the $(n_1-2)(n_2-2)(n_3-2)$ collocation points $P^l_{\tau_1\tau_2\tau_3}$ on I^3 as the cross-product of the points on the intervals [0, 1] for the coordinates ξ^i defined in (30)

$$P^l_{\tau_1\tau_2\tau_3}=(P^1_{\tau_1}, P^2_{\tau_2}, P^3_{\tau_3}). \tag{32}$$

In the following discussion, the notation $[\cdot]_{\tau_1\tau_2\tau_3}$ indicates that the function inside the brackets is evaluated at the collocation point $P^l_{\tau_1\tau_2\tau_3}$. In particular, the Cartesian coordinates of the collocation points $P^l_{\tau_1\tau_2\tau_3}$ are denoted by $[x^l]_{\tau_1\tau_2\tau_3}$. To derive the collocation equations, we first differentiate (17) to write formulas for $[x^l_{\xi^i}]_{\tau_1\tau_2\tau_3}$ and $[x^l_{\xi^j\xi^i}]_{\tau_1\tau_2\tau_3}$ at the collocation points

$$[x^l_{\xi^i}]_{\tau_1\tau_2\tau_3}=\sum_{\nu_1\nu_2\nu_3} A^l_{\sigma_1\sigma_2\sigma_3}[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i}]_{\tau_1\tau_2\tau_3}, \tag{33}$$

$$[x^l_{\xi^j\xi^i}]_{\tau_1\tau_2\tau_3}=\sum_{\nu_1\nu_2\nu_3} A^l_{\sigma_1\sigma_2\sigma_3}[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i\xi^j}]_{\tau_1\tau_2\tau_3}, \tag{34}$$

where the indices $\tau_i=1, ..., n_i-2$, the indices $\nu_i=1, ..., m_i$ and the indices $\sigma_i=k_i(\alpha_i-1)+\nu_i$ for $\tau_i=k_i(\alpha_i-1)+\kappa_i$ with $1\leqslant\alpha_i\leqslant l_i$ and $1\leqslant\kappa_i\leqslant k_i$. We have used the small support of the B-spline basis functions $U_{\sigma_1\sigma_2\sigma_3}$ to reduce the number of terms in the sums in (33) and (34) from $n_1n_2n_3$ to $m_1m_2m_3$.

Substituting (33) and (34) into the elliptic grid generation equations (1), we obtain the collocation equations

$$0=\sum_{i=1}^{3}\sum_{j=1}^{3}[g^{ij}]_{\tau_1\tau_2\tau_3}[x^l_{\xi^j\xi^i}]_{\tau_1\tau_2\tau_3} + \sum_{i=1}^{3}[g^{ii}Q^i]_{\tau_1\tau_2\tau_3}[x^l_{\xi^i}]_{\tau_1\tau_2\tau_3}, \tag{35}$$

$$=\sum_{\nu_1\nu_2\nu_3}[B]^{\tau_1\tau_2\tau_3}_{\nu_1\nu_2\nu_3}A^l_{\sigma_1\sigma_2\sigma_3}, \tag{36}$$

where

$$[B]^{\tau_1\tau_2\tau_3}_{\nu_1\nu_2\nu_3}=\sum_{i=1}^{3}\sum_{j=1}^{3}[g^{ij}]_{\tau_1\tau_2\tau_3}[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i\xi^j}]_{\tau_1\tau_2\tau_3} + \sum_{i=1}^{3}[g^{ii}Q^i]_{\tau_1\tau_2\tau_3}[(U_{\sigma_1\sigma_2\sigma_3})_{\xi^i}]_{\tau_1\tau_2\tau_3}, \tag{37}$$

the indices $\tau_i=1, ..., n_i-2$, the indices $\nu_i=1, ..., m_i$ and the indices $\sigma_i=k_i(\alpha_i-1)+\nu_i$ for $\tau_i=k_i(\alpha_i-1)+\kappa_i$ with $1\leqslant\alpha_i\leqslant l_i$ and $1\leqslant\kappa_i\leqslant k_i$.

This derivation leads to $(n_1-2)(n_2-2)(n_3-2)$ collocation equations on I^3 for the $(n_1-2)(n_2-2)(n_3-2)$ unknown interior expansion parameters. An iterative method is required to solve the system of collocation equations because the equations are quasi-linear in the interior expansion parameters. The solution method we use, which is similar to the ADI method with line SOR described in Section 2.2 for the finite difference method, sweeps over lines of collocation points for each coordinate direction an updates corresponding lines of interior expansion parameters. The corrections to the expansion parameters are computed by solving a system of equations derived from the collocation equations for the line of collocation points. We briefly describe the calculations for a sweep over τ_1-lines of collocation points.

A τ_1-line consists of a line of collocation points on which τ_1 varies with $1\leqslant\tau_1\leqslant n_1-2$, τ_2 is fixed with $1\leqslant\tau_2\leqslant n_2-2$ and τ_3 is fixed with $1\leqslant\tau_3\leqslant n_3-2$, and the corresponding σ_1-line consists of a line of expansion parameters on which σ_1 varies with $2\leqslant\sigma_1\leqslant n_1-1$, $\sigma_2=\tau_2+1$, is fixed, and $\sigma_3=\tau_3+1$ is fixed. A sweep over τ_1-lines of collocation points consists of all the τ_1-lines, where the fixed values of τ_2 and τ_3 vary over their indicated ranges. For each line in the sweep over τ_1-lines of collocation points, the corresponding σ_1-line of expansion parameters is corrected as

$$A^l_{\sigma_1\sigma_2\sigma_3}=A^l_{\sigma_1\sigma_2\sigma_3}+\delta A^l_{\sigma_1\sigma_2\sigma_3}, \tag{38}$$

where the correction terms $\delta A^l_{\sigma_1\sigma_2\sigma_3}$ satisfy

$$\sum_{\nu_1}[B]^{\tau_1\tau_2\tau_3}_{\nu_1,\kappa_2+1,\kappa_3+1}\delta A^l_{\sigma_1,\tau_2+1,\tau_3+1} = -\sum_{\nu_1\nu_2\nu_3}[B]^{\tau_1\tau_2\tau_3}_{\nu_1\nu_2\nu_3}A^{l(*)}_{\sigma_1\sigma_2\sigma_3}, \tag{39}$$

the indices τ_i are the indices of a point on the τ_1-line, the indices $\nu_i = 1, ..., m_i$, the indices $\sigma_i = k_i(\alpha_i - 1) + \nu_i$ for $\tau_i = k_i(\alpha_i - 1) + \kappa_i$ with $1 \leqslant \alpha_i \leqslant l_i$ and $1 \leqslant \kappa_i \leqslant k_i$, and the superscript $(*)$ indicates that the most recent value of the interior expansion parameters are used in the sum.

The resulting system of $n_1 - 2$ linear equations defining the $n_1 - 2$ correction terms for the σ_1-line of interior expansion parameters corresponding to a τ_1-line of collocation point has a banded structure, with bandwidth k_i, and is well conditioned. It may be solved for the correction terms with a standard band solver such as the LINPACK subroutines SGBCO or SGBFA. We note that for $k_i = 1$ the system is tridiagonal and can be solved with a standard tridiagonal solver such as the LINPACK subroutine SGTSL.

4. APPLICATION TO 2D GRIDS

To test the tensor product B-spline method and compare its performance to the finite difference method, we implemented the tensor product B-spline method in the author's 2D multi-block grid code [2]. The code uses a two-stage iteration procedure to compute the grid block by block so that the grid for each block satisfies the Poisson generating system (1) and prescribed block interface conditions. Interface blocks, which overlap pairs of grid blocks, are introduced to enforce the block interface conditions. In the inner iteration, the boundaries of all grid blocks and interface blocks are fixed and their grids are updated using the ADI method described above for either the finite difference method or the tensor product B-spline method. In the outer iteration, all grid block and interface block boundaries are updated to satisfy the block interface conditions. This procedure assures smooth block interfaces and generally results in smooth grid lines across block interfaces, although slope discontinuities are possible, e.g., Fig. 6a. The control terms Q^i in (1) are initialized block by block by first setting the control terms on the boundary of each block to values consistent with the initial grid points specified on the block boundary and then extending the control terms into the interior of the block by transfinite interpolation

a

Method	N Points	N Iter	Grid CPU	Block CPU	Block CPU/pt
FDE	3720	70	85.95	83.77	.00032
B-Spline	256	10	05.77	03.97	.00155

b

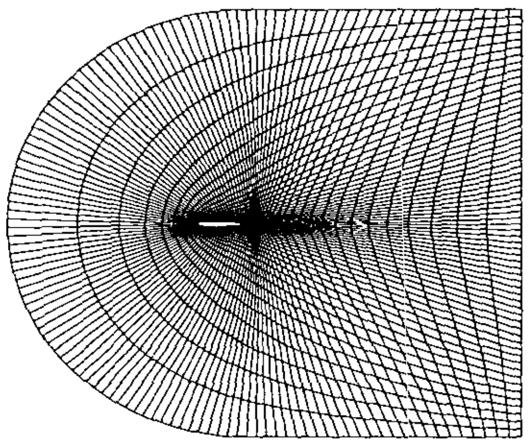
Method	N Points	N Iter	Grid CPU	Block CPU	Block CPU/pt
FDE	3596	70	84.05	78.50	.00031
B-Spline	512	15	20.67	11.76	.00153

c

Method	N Points	N Iter	Grid CPU	Block CPU	Block CPU/pt
FDE	9996	100	326.46	315.54	.00032
B-Spline	1920	40	144.24	119.31	.00155

FIG. 3. Timing data for tensor product B-spline method: (a) C-grid for NACA-0012 Airfoil—3 blocks; (b) C-grid for NACA-0012 Airfoil—6 blocks; (c) O-grid for Model 350 Fighter—4 blocks.

a



b

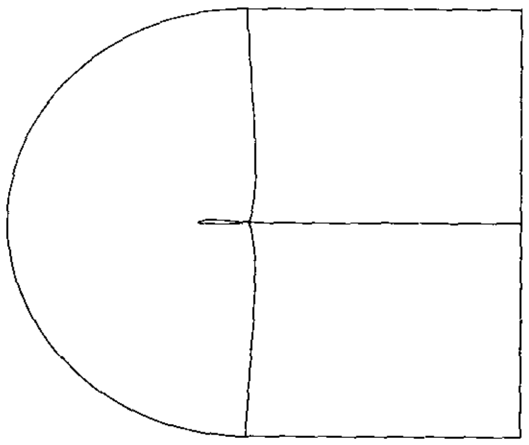


FIG. 4. C-grid for NACA-0012 Airfoil: (a) Three block finite difference grid; (b) block boundaries.

[3, p. 226]. Setting the control terms in this manner leads to multi-block grids which are dense near domain boundaries that are highly curved. The grid code does not provide for any iterative adjustment of the control terms during the calculation of the grid to control such grid features as the orthogonality of grid lines at the domain boundary or the skewness of grid cells, e.g., Fig. 6b. Although the author's grid code has some limitations, it provided a test bed where we could compare the performance of similar implementations of the finite difference and tensor product B-spline methods. We selected three grid generation problems for the comparisons: a three block C-grid for the NACA-0012 Airfoil, a six block C-grid for the NACA-0012 Airfoil, and a four block O-grid for a cross section of the Model 350 Fighter.

The results of the three tests are summarized in Fig. 3, which compares the computational cost of obtaining elliptic grids of similar smoothness and resolution with the two methods. The first column indicates the method, the second column gives the number of interior grid points for the finite difference method and the number of collocation points for the tensor product B-spline method, the third column gives the number of iterations required to generate the final grids, the fourth column gives the total CPU seconds required to generate the final grids, the fifth column gives the total CPU seconds spent in solving the finite difference or collocation equations and the sixth column gives the average CPU seconds per interior grid point per iteration for the finite difference method and the average CPU seconds per collocation point per iteration for the tensor product B-spline method. Using the data in Figure 3, we can compare the finite difference and tensor product B-spline methods, as implemented in the author's 2D multi-block grid code, with respect to computational complexity, block

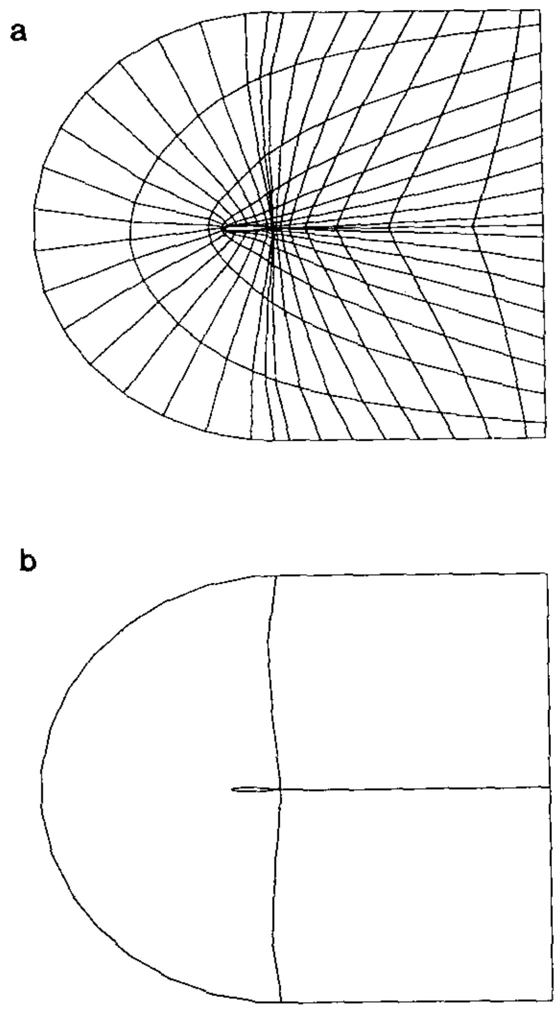


FIG. 5. C-grid for NACA-0012 Airfoil: (a) Three block tensor product B-spline grid; (b) Block boundaries.

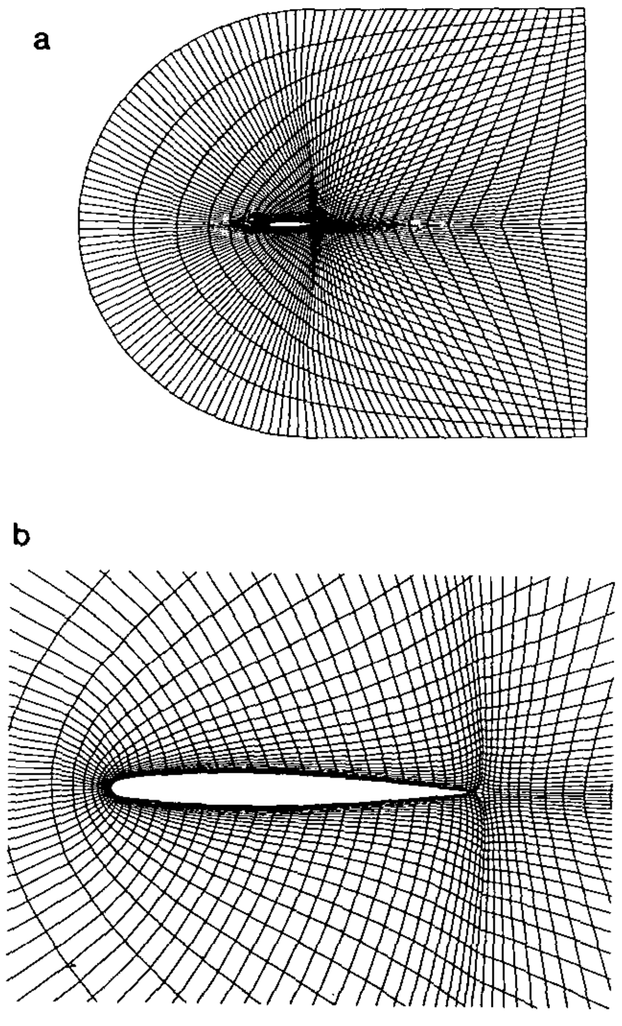


FIG. 6. C-grid for NACA-0012 Airfoil: (a) Fine grid computed from tensor product B-spline grid; (b) Detail of fine grid.

interface processing overhead and overall computational cost of obtaining elliptic grids of similar smoothness and resolution.

The data in the last column, which measures the computational complexity of the two methods, shows that on average the tensor product B-spline method (using quadratic B-splines) is 4.84 times as expensive as the finite difference method.

The difference between the CPU times given in columns four and five is the amount of CPU time spent processing block interfaces. For the finite difference method, where the number of grid points per block was large in the three tests, the cost of computing the interface grids was a small part of the total cost of computing the final grid. For the tensor product B-spline method, however, where the number of collocation points per block was low in the three tests, the cost of computing the interface grids was a significant fraction of the total cost of computing the final grid.

The CPU times in the fourth column show that an overall speedup of the tensor product B-spline method with respect the finite difference method was obtained in all three tests, in spite of the tensor product B-spline method's higher computational complexity and block interface overhead costs. The best speedup of 14.9 was obtained in the first test, where the number of collocation points was smallest with respect to the number of interior grid points. More modest speedups of 4.1 and 2.3, respectively, were obtained in the second and third tests, where additional collocation points were used to adequately represent complex domain boundaries. The results of the three grid generation problems are described more fully in the following paragraphs.

4.1. *C-Grid for NACA-*0012 *Airfoil; Three Blocks*

Figures 4 to 6 illustrate the first study in which we focused solely with the speed of the tensor product B-spline method.

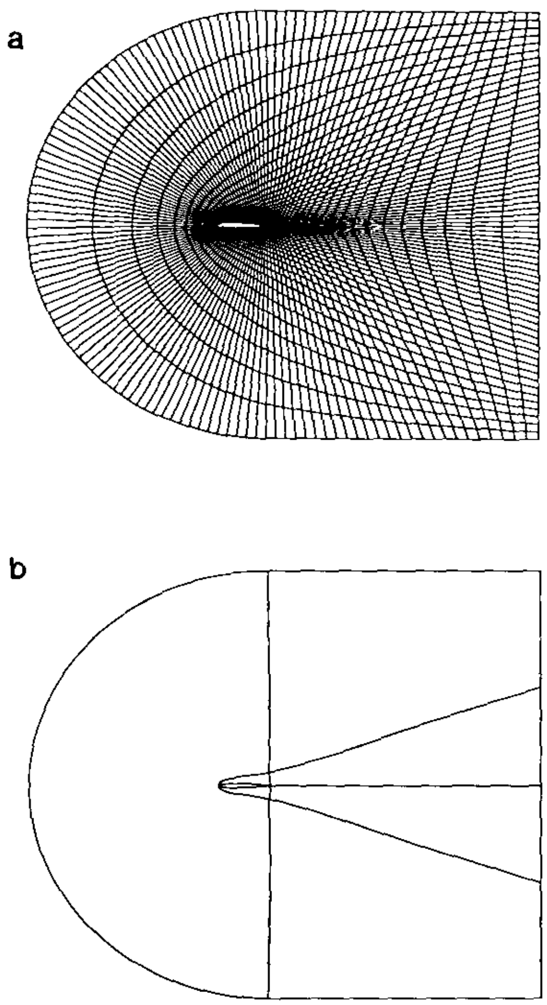


FIG. 7. C-grid for NACA-0012 Airfoil: (a) Six block finite difference grid; (b) Block boundaries.

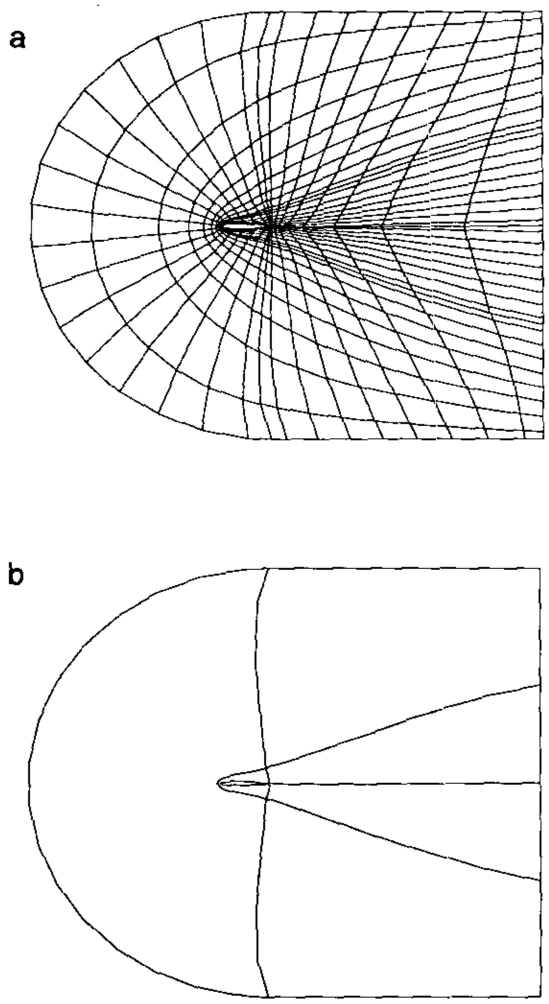


FIG. 8. C-grid for NACA-0012 Airfoil: (a) Six block tensor product B-spline grid; (b) Block boundaries.

The objective of the test was to compute a C-grid for the NACA-0012 Airfoil using the tensor product B-spline method with the coarsest possible grid of knots that gives a fine grid comparable to a fine grid computed by the finite difference method.

Figure 4a shows a C-grid for the NACA-0012 airfoil computed with our multi-block grid code using three blocks and the finite difference method. The computed grid is a smooth elliptic grid with high resolution near the airfoil surface. Figure 4b shows the block boundaries in the computed grid.

Figure 5a shows a C-grid computed with our multi-block grid code using three blocks and the tensor product B-spline method. Quadratic B-splines ($k_i = 1$) were used for all blocks. 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; we call this the B-spline grid. Figure 5b shows the block boundaries in the computed grid.

Figure 6a 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 6b shows an expanded view of the C-grid near the airfoil surface. Note that the fine grid computed from the coarse B-spline grid is as smooth and well resolved as the finite difference grid.

4.2 *C-Grid for NACA-0012 Airfoil; Six Blocks*

Figures 7 to 10 illustrate the second study in which we considered the resolution of the airfoil geometry in addition to the speed of the method. The blocking strategy we adopted to satisfy this additional requirement was to use an inner and outer set of blocks. A sufficiently fine grid of knots was used in the inner blocks to resolve the airfoil geometry

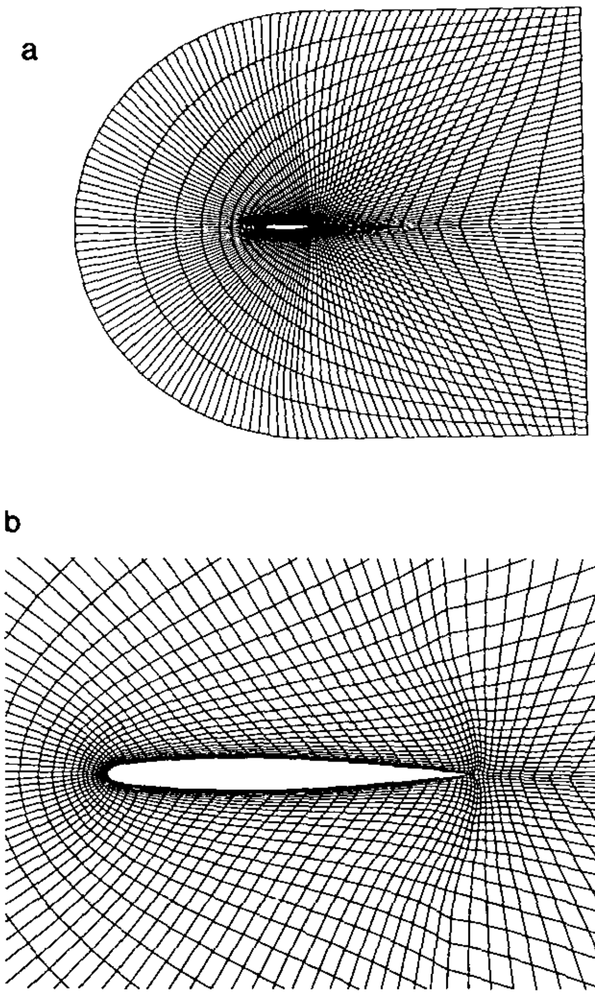


FIG. 9. C-grid for NACA-0012 Airfoil: (a) Fine grid computed from tensor product B-spline grid; (b) Detail of fine grid.

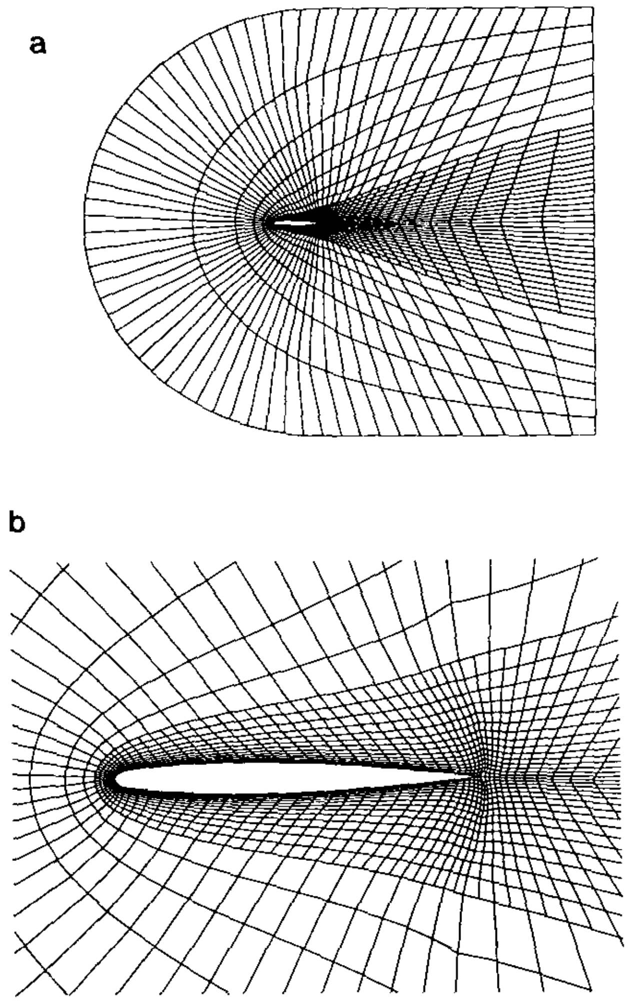


FIG. 10. C-grid for NACA-0012 Airfoil: (a) Fine inner grid, coarse outer grid; (b) Detail of inner grid.

and a coarse grid of knots was used in the outer blocks. The objective of the test was to compute a C-grid for the NACA-0012 Airfoil using the tensor product B-spline method with the coarsest possible grid of knots that gives a fine grid comparable to a fine grid computed by the finite difference method and resolves the airfoil geometry.

Figure 7a shows a C-grid for the NACA-0012 airfoil computed with our multi-block grid code using six blocks and the finite difference method. The computed grid is a smooth elliptic grid with high resolution near the airfoil surface and is nearly identical with the C-grid obtained with three blocks in the first study. Figure 7b shows the block boundaries in the computed grid.

Figure 8a shows the B-spline grid computed with our multi-block grid code using six blocks and the tensor product B-spline method. Quadratic B-splines ($k_i = 1$) were used for all blocks. Figure 8b shows the block boundaries in the computed grid.

Figure 9a 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 9b shows an expanded view of the C-grid near the airfoil surface. Note that the fine grid computed from the coarse B-spline grid is as smooth and well resolved as the finite difference grid.

Figure 10a 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 10b shows an expanded view of the C-grid near the airfoil surface. These figures illustrate an important advantage the tensor product B-spline method has over the finite difference method. Different grid densities can be obtained in the blocks defining a grid simply by evaluating the

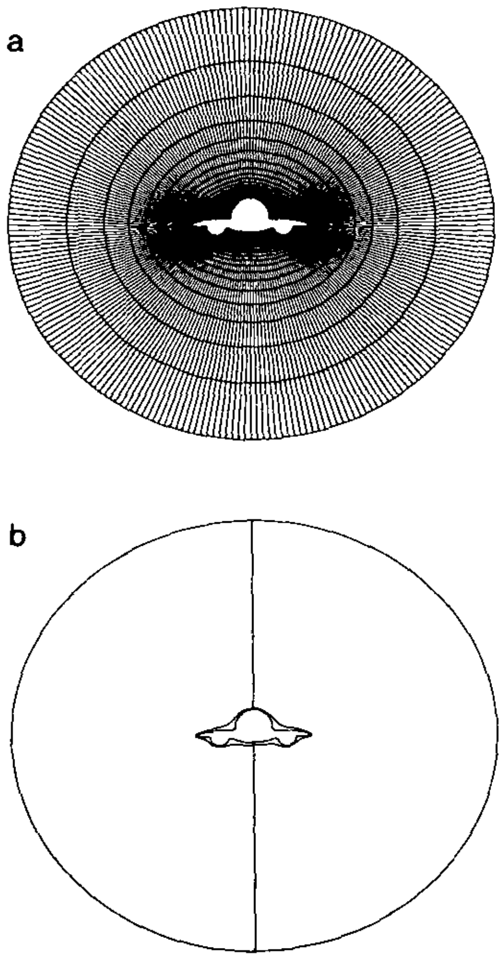


FIG. 11. O-grid for Model 350 Fighter: (a) Four block finite difference grid; (b) Block boundaries.

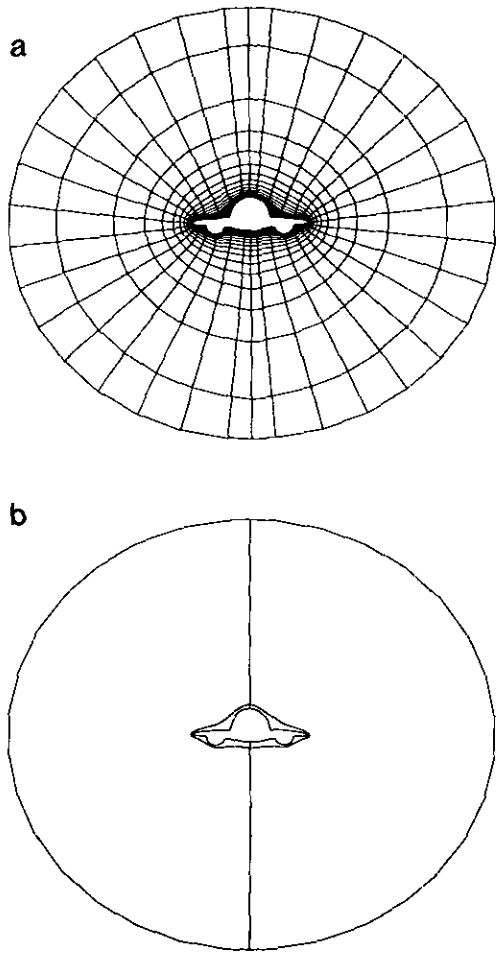


FIG. 12. O-grid for Model 350 Fighter: (a) Four block tensor product B-spline grid; (b) Block boundaries.

coordinate functions at different densities in the blocks' computational domains. As shown in the figure smooth block boundaries are still obtained.

4.3. *O-Grid for Model* 350 *Fighter; Four Blocks*

Figures 11 to 14 illustrate the third study in which we considered the resolution of a very complex domain boundary in addition to the speed of the method. We adopted a blocking strategy similar to that for the airfoil case. A sufficiently fine grid of knots was used in the inner blocks to resolve the fighter geometry and a coarse grid of knots was used in the outer blocks. The objective of the test was to compute an O-grid for the Model 350 Fighter using the tensor product B-spline method with the coarsest possible grid of knots that gives a fine grid comparable to a fine grid computed by the finite difference method and resolves the fighter geometry.

Figure 11 shows an O-grid for the Model 350 Fighter computed with our multi-block grid code using four blocks and the finite difference method. The computed grid is a smooth elliptic grid with high resolution near the surface of the fighter. Figure 11b shows the block boundaries in the computed grid.

Figure 12a shows the B-spline grid computed with our multi-block grid code using four blocks and the tensor product B-spline method. 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. Figure 12b shows the block boundaries in the computed grid.

Figure 13a 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 13b shows an expanded view of the O-grid near the fighter surface. Note that the fine grid computed from the coarse B-spline grid is

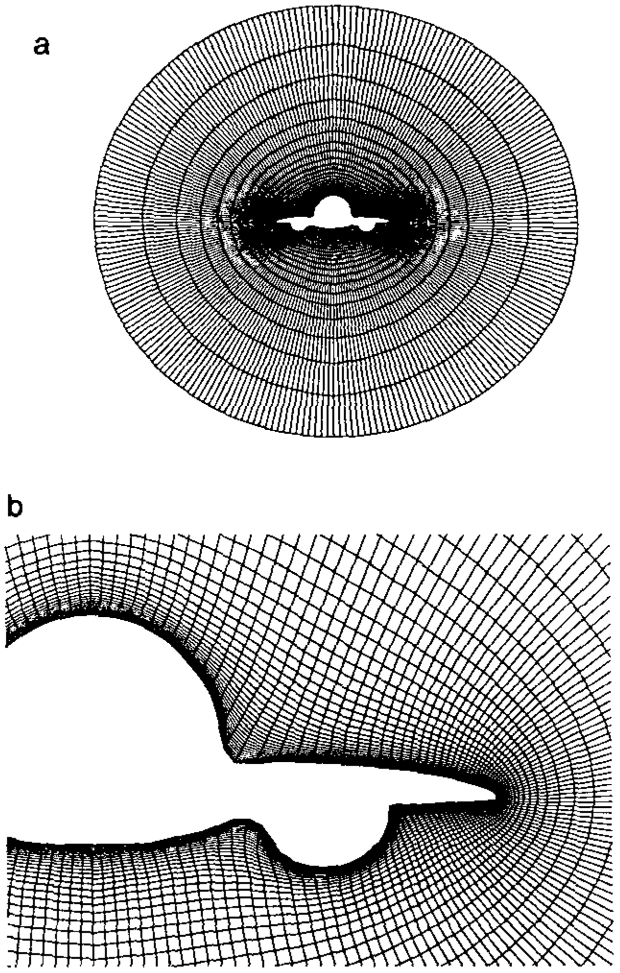


FIG. 13. O-grid for Model 350 Fighter: (a) Fine grid computed from tensor product B-spline grid; (b) Detail of fine grid.

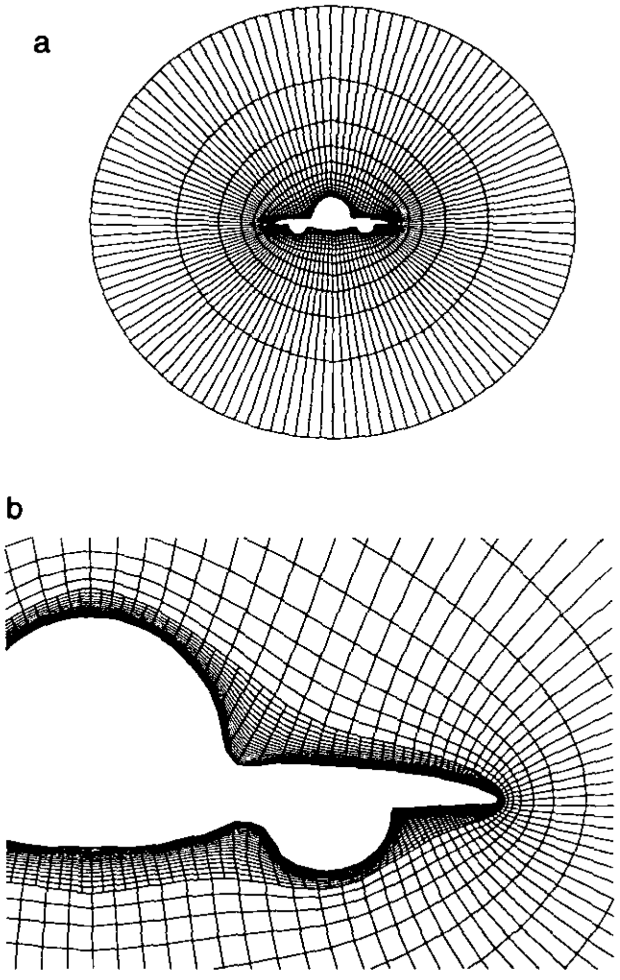


FIG. 14. O-grid for Model 350 Fighter: (a) Fine inner grid, coarse outer grid; (b) Detail of inner grid.

as smooth and well resolved as the finite difference grid and gives an accurate representation of the fighter boundary.

Figure 14a 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 14b shows an expanded view of the O-grid near the fighter surface. These figures illustrate again that different grid densities can be obtained in the blocks defining a grid simply by evaluating the coordinate functions at different densities in the blocks' computational domains. As shown in the figure smooth block boundaries are still obtained, even when knot densities change across the boundaries.

5. CONCLUSIONS

We have presented a tensor product B-spline method for elliptic grid generation and compared its performance to the finite difference method in the context of a multi-block grid generation code. We verified that fine grids can be computed with the tensor product B-spline method over a coarse set of knots and that the resulting grids have resolution and smoothness comparable to fine grids computed with the finite difference method. We showed that for such grids, the tensor product B-spline method was up to 15 times faster than the finite difference method. We demonstrated an effective strategy for using the tensor product B-spline method for domains with complex boundaries. We demonstrated that smooth block boundaries can be obtained even when knot densities change across block boundaries.

REFERENCES

1. C. de Boor, "A Practical Guide to Splines," in *Applied Mathematical Sciences, Vol. 27* (Springer-Verlag, New York/Berlin, 1978).
2. J. W. Manke, Technical Report ETA-TR-66, Boeing Computer Services Co., December 1987 (unpublished).
3. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation, Foundations and Applications* (North-Holland, Amsterdam, 1985).
4. Z. U. A. Warsi, Technical Report MSSU-EIRS-81-1, Mississippi State University, 1981 (unpublished).