Hindawi
International Journal of Mathematics and Mathematical Sciences
Volume 2022, Article ID 7266801, 20 pages
https://doi.org/10.1155/2022/7266801

Research Article

Newton Linearization of the Curvature Operator in Structured Grid Generation with Sample Solutions

Pat Piperni and Maahi M. Talukder

Clarkson University, Potsdam, NY, USA

Correspondence should be addressed to Pat Piperni; ppiperni@clarkson.edu

Received 7 January 2022; Accepted 28 April 2022; Published 8 July 2022

Academic Editor: Nawab Hussain

Elliptic grid generation equations based on the Laplacian operator have the well-known property of clustering the mesh near convex boundaries and declustering it near concave boundaries. In prior work, a new differential operator was derived and presented to address this issue. This new operator retains the strong smoothing properties of the Laplacian without the latter's adverse curvature effects. However, the new operator exhibits slower convergence properties than the Laplacian, which can lead to increased turnaround times and in some cases preclude the achievement of convergence to machine accuracy. In the work presented here, a Newton linearization of the new operator is presented, with the objective of achieving more robust convergence properties. Sample solutions are presented by evaluating a number of solvers and preconditioners and assessing the convergence properties of the solution process. The efficiency of each solution method is demonstrated with applications to two-dimensional airfoil meshes.

1. Introduction

Elliptic grid generation is a powerful tool for optimizing the quality of a mesh in a complex domain. The solution of elliptic equations based on the Laplace operator can be used to control virtually all the characteristics of a given mesh, through the proper specification of control functions. Various forms of these equations have been proposed, based on the early works of Winslow [1], Barfield [2], Chu [3], Godunov and Prokopov [4], Amsden and Hirt [5], and Thompson et al. [6–10].

However, elliptic grid generation equations based on the Laplacian operator have the well-known property of clustering the mesh near convex boundaries and declustering it near concave boundaries. The result of this effect is that the mesh spacing near curved boundaries may not reflect the spacing prescribed by the user. In a prior work [11, 12], a new differential operator was derived to address this issue. This new operator retains the strong smoothing properties of the Laplacian operator without the latter's adverse curvature effects. The capability of this new operator, herein referred to as the curvature operator, is illustrated in Figures 1(a) and 1(b) for a simple mesh over a cylinder with equally-spaced boundary points. It can be seen from Figure 1(b) that the mesh generated with curvature control yields the desired cell sizes near the concave corners at the top and bottom of the cylinder, whereas the mesh generated without curvature control increases the mesh spacing in these areas. The capabilities of this new operator have been demonstrated in numerous applications involving the generation of meshes in both two and three dimensions on complex geometry such as complete aircraft configurations [11].

However, it has been observed that the convergence of the numerical solution of the curvature operator is more problematic and slower than the solution of the basic Laplace operator. The main reason for this is that the curvature operator is inherently more sensitive to mesh curvature, and thus, any spurious or discontinuous curvature distribution arising during the solution process may cause a slowdown in convergence. The approach employed thus far to deal with this issue has been to simply under-relax the solution process. However, this approach leads to increased

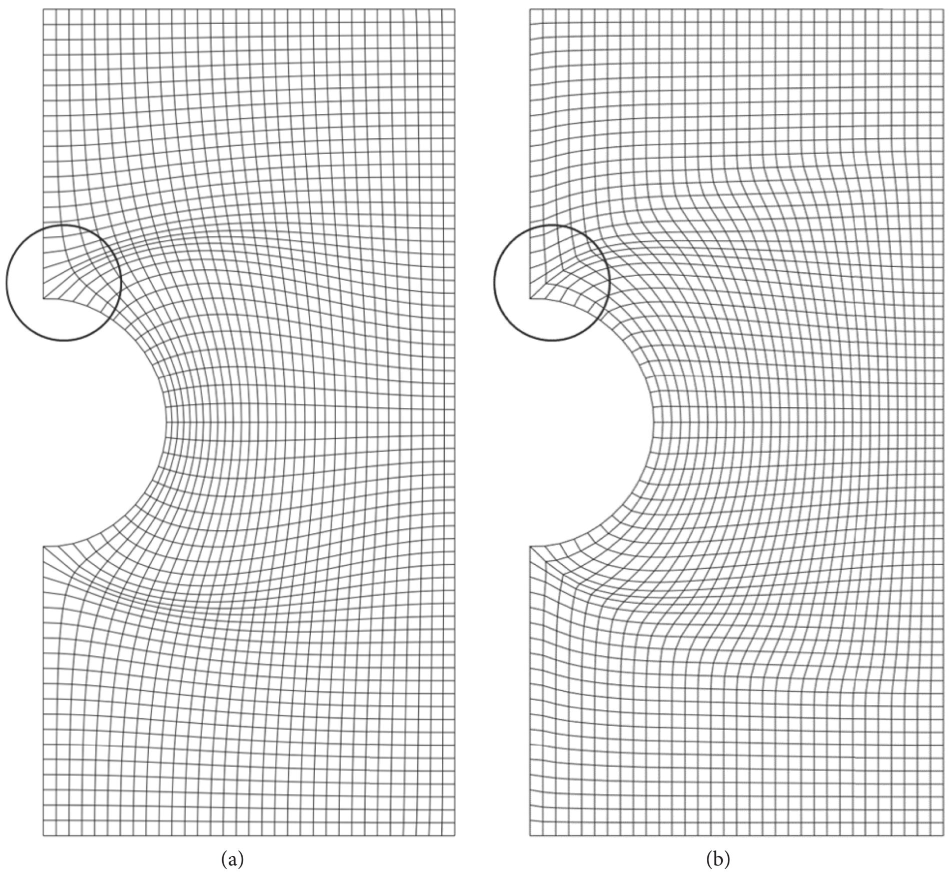

FIGURE 1: Grids generated without and with curvature control, (a) with no curvature control and (b) with curvature control.

turnaround times, and in some cases precludes the achievement of full convergence to machine accuracy.

There is extensive literature on the numerical solution of similar systems of nonlinear partial differential equations. Davis and McCammon [13] showed that the incomplete Cholesky conjugate gradient (ICCG) method is superior to relaxation methods for solving a system of linear equations arising from the finite-difference form of the linearized Poisson–Boltzmann equation. Nicholls and Honig [14] developed an efficient algorithm for the numerical solution of the Poisson–Boltzmann equation by the finite-difference method utilizing successive over-relaxation through the rapid estimation of the optimum relaxation parameter. Cai et al. [15] implemented and optimized seven finite-difference solvers for the full nonlinear Poisson–Boltzmann equation, including four relaxation methods, one conjugate gradient method, and two inexact Newton methods. Wang and Luo [16] assessed five commonly used finite-difference solvers for the Poisson–Boltzmann equation and found that the modified incomplete Cholesky conjugate gradient and geometric multigrid are the most efficient in the diversified test set.

Knoll et al. [17] developed a nonlinear solution method for the nonequilibrium radiation diffusion problem by combining an outer Newton-based iteration with the inner conjugate gradient-like (Krylov) iteration.

Rider et al. [18] developed a robust and scalable solution of the equilibrium radiation diffusion problem using the GMRES Newton–Krylov method preconditioned by a multigrid method resulting from a Picard-type linearization of the governing equation.

Berndta et al. [19] developed two nonlinear solvers based on the Jacobian-Free Newton–Krylov (JFNK) methodology to solve the Laplace–Beltrami system of nonlinear, elliptic, and partial differential equations discretized using a finite-element method. They also developed two different preconditioners, both of which employ existing algebraic multigrid (AMG) methods. These solvers have been demonstrated to be significantly faster than a standard Newton–Krylov approach.

Gupta et al. [20] combined a compact high-order finite-difference approximation with a multigrid V-cycle algorithm to solve the two-dimensional Poisson elliptic differential equations and showed that the nine-point discretization formula, combined with a full-weighting projection operator, is much more accurate than the five-point discretization formula. Zhang [21] employed a fourth-order compact finite-difference scheme (FOS) with the multigrid algorithm to solve the three-dimensional Poisson equation and showed that it yields a fast and high-accuracy 3D Poisson solver.

Ilić et al. [22] showed that the fractional Poisson equation can be approximately solved using a finite-difference discretization of the Laplacian to produce an appropriate matrix representation of the operator and proposed an algorithm based on a Krylov subspace method that could be used efficiently to solve the resulting linear system.

Froese and Oberman [23] developed a fast and accurate finite-difference solver for the elliptic Monge–Ampère equation using a hybrid finite-difference discretization which selects between an accurate standard finite-difference discretization and a stable monotone discretization.

In the work presented herein, the grid generation equations with and without curvature control functions are discretized using finite-differences and a Newton linearization of the discretized equations is then implemented. To solve the resulting linear system, a number of solvers and preconditioners drawn from the Portable Extensible Toolkit for Scientific Computation (PETSc) library are implemented and evaluated.

In what follows, Section 2 provides an overview of the curvature operator and its derivation, followed by a detailed description of the methodology employed to linearize the equations. Section 3 presents the numerical implementation process and setup of the matrix equation, and Section 4 presents applications of the solution process to C-meshes around a two-dimensional airfoil.

2. Mathematical Derivation

2.1. Overview of the Curvature Operator. In this section, a brief overview of the curvature operator and its derivation will be presented followed by a detailed description of the methodology employed to linearize the equations.

2.1.1. Mesh Generation Equations. Denoting ξ^i as the curvilinear coordinates, it was shown in [11, 24] that for a mesh with uniform spacing, the curvature operator can be expressed as follows:

$$\nabla^2 \xi^i = C^i; \quad i = 1, 2, 3, \tag{1}$$

where C^i represents the curvature control functions and is given by the following:

$$C^i = -\left(K_1^{(i)} + K_2^{(i)}\right)\sqrt{g^{ii}}, \tag{2}$$

where

$$K_1^{(i)} + K_2^{(i)} = \frac{1}{g\left(g^{ii}\right)^{3/2}}\left(\Gamma^i_{jj} g_{kk} - 2\Gamma^i_{jk} g_{jk} + \Gamma^i_{kk} g_{jj}\right), \tag{3}$$

and where (i, j, k) are cyclic, with no sum on j and k.

In the above expressions, the g^{ij} are the contravariant metric tensor components, and the Γ^i_{jk} are the Christoffel symbols of the second kind. The physical interpretation of equation (3) can be made by noting that, for a ξ^i = constant surface, $K_1^{(i)} + K_2^{(i)}$ represents the sum of the local principal curvatures of the surface.

As shown in [11], in the general case of a nonuniform mesh, the curvature operator must be expressed in terms of the functions defining the grid clustering, denoted here as s^i $(i = 1, 2, 3)$. The s^i functions are defined by a mapping of the spacing distributions of the physical mesh onto a

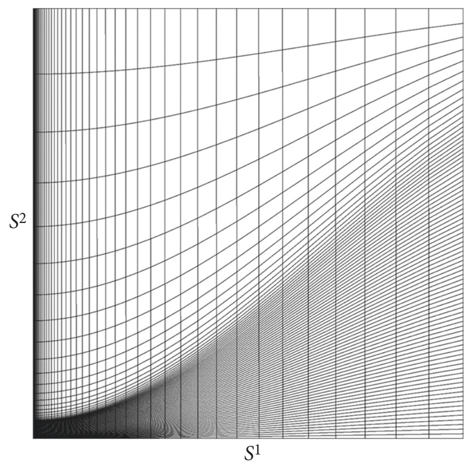

Figure 2: Computational domain defining the s^i stretching functions.

computational space scaled to a unit square, as shown in Figure 2. The points along the boundaries of the s^i space represent the normalized arc length distributions along the boundaries of the physical space, while the points in the interior of the s^i space are interpolated algebraically from the boundaries [25].

Thus, for a nonuniform mesh, equation (1) is applied to the s^i functions, as follows:

$$\nabla^2 s^i = \overline{C}^i; \quad i = 1, 2, 3, \tag{4}$$

where

$$\overline{C}^i = -\left(\overline{K}_1^{(i)} + \overline{K}_2^{(i)}\right)\sqrt{\overline{g}^{ii}}, \tag{5}$$

and

$$\overline{K}_1^{(i)} + \overline{K}_2^{(i)} = \frac{1}{\overline{g}\left(\overline{g}^{ii}\right)^{3/2}}\left(\overline{\Gamma}^i_{jj} \overline{g}_{kk} - 2\overline{\Gamma}^i_{jk} \overline{g}_{jk} + \overline{\Gamma}^i_{kk} \overline{g}_{jj}\right), \tag{6}$$

and where the bar over the terms in equations (4) to (6) denotes that they are functions of s^i, not functions of the curvilinear coordinates ξ^i [11].

The abovementioned relations can be recast in terms of the curvilinear coordinates by expressing the Laplacian of s^i in equation (4) as a function of the curvilinear coordinates, which is given by the following expression:

$$\nabla^2 s^i = g^{jk}\frac{\partial^2 s^i}{\partial \xi^j \partial \xi^k} + \nabla^2 \xi^p \frac{\partial s^i}{\partial \xi^p}, \tag{7}$$

$$= \overline{C}^i. \tag{8}$$

It is to be noted that in equation (7) and similar expressions in the remainder of this work, the summation convention is used whenever repeated upper and lower dummy indices are present, unless otherwise noted. Isolating $\nabla^2 \xi^p$ in equation (7) then leads to the following:

$$\nabla^2 \xi^p = R^p - g^{jk}\frac{\partial \xi^p}{\partial s^i}\frac{\partial^2 s^i}{\partial \xi^j \partial \xi^k}; \quad p = 1, 2, 3, \tag{9}$$

where

$$R^p = \overline{C}^i \frac{\partial \xi^p}{\partial s^i}. \tag{10}$$

In equation (9), the second term on the right-hand side represents the functions controlling the mesh spacing, as derived by [25]. Equation (10) gives the general form of the curvature control functions, R^p, for a mesh with nonuniform spacing. For a mesh with uniform spacing, equation (9) reduces to equation (1).

The mesh generation equations are then built by using the identity $\nabla^2 \mathbf{r} = 0$ (where $\mathbf{r} = [x^1\ x^2\ x^3]^T = [x\ y\ z]^T$) and expressing it in terms of the curvilinear coordinates:

$$\nabla^2 \mathbf{r} = g^{jk} \frac{\partial^2 \mathbf{r}}{\partial \xi^j \partial \xi^k} + \nabla^2 \xi^p \frac{\partial \mathbf{r}}{\partial \xi^p} = 0. \tag{11}$$

The above system of equations is solved to yield the mesh coordinates, with the $\nabla^2 \xi^p$ terms given by equations (9) and (10).

2.1.2. Evaluation of Curvature Control Functions. The numerical generation of the mesh using equation (11) requires the evaluation of the $\nabla^2 \xi^p$ terms using equations (9) and (10) and the evaluation of the $\overline{C}^i$ terms using equations (5) and (6).

The solution process begins by defining the s^i functions, following the methodology described in detail in [25]. These functions are initially calculated based on the mesh clustering on the boundaries and are interpolated into the interior of the domain. During the iterative solution process, the s^i functions can be updated using the Neumann boundary conditions to enforce orthogonality conditions where desired, thereby modifying the s^i functions both on the boundaries and in the domain at every iteration.

The s^i functions are expressed in terms of the curvilinear coordinates ξ^i, where $s = [s^1\ s^2\ s^3]^T$. Once the s^i functions have been defined, their first and second derivatives with respect to ξ^i can be computed directly at every point in the domain via finite differences of the coordinate transformation. To evaluate the second term on the right-hand side of equation (9), it is also required to compute the derivatives of the curvilinear coordinates with respect to the s^i functions, and these can be evaluated using the following well-known relations [26]:

$$\frac{\partial \xi^i}{\partial s^j} = \frac{1}{\hat{J}} \left(\frac{\partial s^s}{\partial \xi^r} \frac{\partial s^n}{\partial \xi^m} - \frac{\partial s^s}{\partial \xi^m} \frac{\partial s^n}{\partial \xi^r} \right), \tag{12}$$

where (i, r, m) and (j, s, n) are cyclic, and where $\hat{J}$ is the determinant of the Jacobian matrix of the s^i-to-ξ^i coordinate transformation:

$$\hat{J} = \det \left\{ \frac{\partial s^i}{\partial \xi^j} \right\}. \tag{13}$$

To evaluate the R^p terms in equation (10), the $\overline{C}^i$ control functions must be evaluated with respect to the s^i coordinates. To do this, we must compute the metric tensor components and Christoffel symbols of the Cartesian x^i-to-s^i coordinate transformation. The metric tensors of the transformation can be evaluated using the following well-known relations [27]:

$$\overline{g}_{ij} = \mathbf{r}_{s^i} \cdot \mathbf{r}_{s^j} = \frac{\partial x^p}{\partial s^i} \frac{\partial x^p}{\partial s^j}. \tag{14}$$

$$\overline{g} = \det \left| \overline{g}_{ij} \right|. \tag{15}$$

$$\overline{g}^{ij} = \frac{\left(\overline{g}_{rs} \overline{g}_{mn} - \overline{g}_{rn} \overline{g}_{ms} \right)}{\overline{g}}. \tag{16}$$

where (i, r, m) and (j, s, n) are cyclic. To evaluate equation (14), the derivatives of the Cartesian coordinates with respect to the s^i coordinates are required, and these can be computed using the chain rule involving the ξ^i coordinates:

$$\frac{\partial \mathbf{r}}{\partial s^j} = \frac{\partial \mathbf{r}}{\partial \xi^p} \frac{\partial \xi^p}{\partial s^j}, \tag{17}$$

where the $\partial \xi^p / \partial s^j$ are evaluated using equation (12).

To evaluate the Christoffel symbols with respect to the s^i coordinates in (6), denoted as $\overline{\Gamma}^i_{jk}$, the following expression can be used [27]:

$$\overline{\Gamma}^i_{jk} = \overline{g}^{il} \mathbf{r}_{s^l} \cdot \mathbf{r}_{s^j s^k}, \tag{18}$$

In turn, to evaluate the second derivative of the physical coordinates with respect to the s^i coordinates, it is useful to express the second term on the right-hand side of equation (9) in the following compact form:

$$p^p_{jk} = -\frac{\partial \xi^p}{\partial s^i} \frac{\partial^2 s^i}{\partial \xi^j \partial \xi^k}; \quad p = 1, 2, 3. \tag{19}$$

Substituting (19) back into equation (9) then leads to the following:

$$\nabla^2 \xi^p = R^p + g^{jk} p^p_{jk}; \quad p = 1, 2, 3. \tag{20}$$

From equation (19) and the transformation laws for the Christoffel symbols [27], it can be shown that:

$$\frac{\partial^2 \mathbf{r}}{\partial s^j \partial s^k} = \left(\frac{\partial^2 \mathbf{r}}{\partial \xi^m \partial \xi^n} + p^p_{mn} \frac{\partial \mathbf{r}}{\partial \xi^p} \right) \frac{\partial \xi^m}{\partial s^j} \frac{\partial \xi^n}{\partial s^k}. \tag{21}$$

Expressions (12) to (21) can be used to evaluate all terms required to construct the R^p curvature functions appearing in equations (9) and (10), and the grid generation equation (11) can then be solved numerically to generate the mesh.

2.2. Two-Dimensional Form of the Equations. In this work, the applications of the grid generation equations are limited to two-dimensional domains. To this end, if we let $\xi = \xi^1$ and $\eta = \xi^2$ in equation (11), we can write

$$\nabla^2 \mathbf{r} = g^{11} \mathbf{r}_{\xi\xi} + 2 g^{12} \mathbf{r}_{\xi\eta} + g^{22} \mathbf{r}_{\eta\eta} + \nabla^2 \xi \mathbf{r}_\xi + \nabla^2 \eta \mathbf{r}_\eta = 0. \tag{22}$$

In two dimensions, the transformation relations between the covariant and contravariant metric tensors reduce to the following:

$$g^{11} = \frac{g_{22}}{g}, g^{12} = \frac{-g_{12}}{g}, g^{22} = \frac{g_{11}}{g}. \tag{23}$$

Substituting relations (23) into equation (22), we then obtain the following:

$$g_{22}\mathbf{r}_{\xi\xi} - 2g_{12}\mathbf{r}_{\xi\eta} + g_{11}\mathbf{r}_{\eta\eta} + g\left(\nabla^2\xi\mathbf{r}_\xi + \nabla^2\eta\mathbf{r}_\eta\right) = 0. \tag{24}$$

If we now combine equation (23) with equation (20) and expand, we obtain

$$\nabla^2\xi = R^1 + \frac{\left(g_{22}p_{11}^1 - 2g_{12}p_{12}^1 + g_{11}p_{22}^1\right)}{g}. \tag{25a}$$

$$\nabla^2\eta = R^2 + \frac{\left(g_{22}p_{11}^2 - 2g_{12}p_{12}^2 + g_{11}p_{22}^2\right)}{g}. \tag{25b}$$

If we define the functions P^1 and P^2 as follows,

$$P^1 = g_{22}p_{11}^1 - 2g_{12}p_{12}^1 + g_{11}p_{22}^1, \tag{26a}$$

$$P^2 = g_{22}p_{11}^2 - 2g_{12}p_{12}^2 + g_{11}p_{22}^2, \tag{26b}$$

and insert equations (25) and (26) into equation (24), we obtain

$$g_{22}\mathbf{r}_{\xi\xi} - 2g_{12}\mathbf{r}_{\xi\eta} + g_{11}\mathbf{r}_{\eta\eta} + \left(P^1 + gR^1\right)\mathbf{r}_\xi + \left(P^2 + gR^2\right)\mathbf{r}_\eta = 0. \tag{27}$$

In the above equations, the components of the curvature control functions are represented by the R^1 and R^2 terms. If R^1 and R^2 are set to zero, the standard grid generation equations without curvature control functions are recovered.

2.2.1. Expansion of Curvature Control Functions in Two Dimensions. The general expression for the curvature control functions are given by equations (5) and (10). In two-dimensions, $g_{23} = g_{13} = g^{23} = g^{13} = 0$, $g^{33} = g_{33} = 1$, and the expansion of the Christoffel symbols $(\overline{\Gamma}^i_{jk})$ given by equation (18), in combination with the relations given in (23), lead to the following expressions for the curvature control functions:

$$\overline{C}^1 = -\left(\overline{K}_1^{(1)} + \overline{K}_2^{(1)}\right)\sqrt{\overline{g}^{11}} = -\frac{1}{\overline{g}}\left(\mathbf{r}_s \cdot \mathbf{r}_{tt} - \frac{\overline{g}_{12}}{\overline{g}_{22}}\mathbf{r}_t \cdot \mathbf{r}_{tt}\right), \tag{28a}$$

$$\overline{C}^2 = -\left(\overline{K}_1^{(2)} + \overline{K}_2^{(2)}\right)\sqrt{\overline{g}^{22}} = -\frac{1}{\overline{g}}\left(\mathbf{r}_t \cdot \mathbf{r}_{ss} - \frac{\overline{g}_{12}}{\overline{g}_{11}}\mathbf{r}_s \cdot \mathbf{r}_{ss}\right), \tag{28b}$$

where $(s,t) = (s^1, s^2)$.

It can also be shown that

$$\frac{1}{\overline{g}} = \frac{\widehat{g}}{g} = \frac{\widehat{J}^2}{g}. \tag{29}$$

where g is the determinant of the metric tensor of the $(x,y) \longrightarrow (\xi,\eta)$ transformation, $\overline{g}$ is the determinant of the metric tensor of the $(x,y) \longrightarrow (s,t)$ transformation, $\widehat{g}$ is the determinant of the metric tensor of the $(s,t) \longrightarrow (\xi,\eta)$ transformation, and $\widehat{J}$ is the determinant of the Jacobian matrix of the $(s,t) \longrightarrow (\xi,\eta)$ transformation (13).

Inserting relations (29) into (28a)-(28b) then yields:

$$g\overline{C}^1 = -\widehat{J}^2\left(\mathbf{r}_s \cdot \mathbf{r}_{tt} - \frac{\overline{g}_{12}}{\overline{g}_{22}}\mathbf{r}_t \cdot \mathbf{r}_{tt}\right), \tag{30a}$$

$$g\overline{C}^2 = -\widehat{J}^2\left(\mathbf{r}_t \cdot \mathbf{r}_{ss} - \frac{\overline{g}_{12}}{\overline{g}_{11}}\mathbf{r}_s \cdot \mathbf{r}_{ss}\right), \tag{30b}$$

where the first and second derivatives of $\mathbf{r}$ with respect to the (s,t) coordinates can be computed using equations (17) and (21), respectively. The latter equation can be simplified further by defining the function $\mathbf{R}$ as follows:

$$\mathbf{R}_{\xi^i\xi^j} = \mathbf{r}_{\xi^i\xi^j} + p_{ij}^p\mathbf{r}_{\xi^p}. \tag{31}$$

Equation (21) can then be rewritten in a more compact form:

$$\mathbf{r}_{s^js^k} = \mathbf{R}_{\xi^m\xi^n}\xi^m_{s^j}\xi^n_{s^k}. \tag{32}$$

The curvature functions in equation (27) are then evaluated using (10):

$$gR^p = g\overline{C}^i\xi^p_{s^i}. \tag{33}$$

In this work, numerical solutions of the grid generation equation (27) will be evaluated both with and without the curvature control functions to quantify the impact of the latter on the convergence of the numerical solution.

2.3. Discretization Scheme. The discretization of equation (27) is accomplished using second-order finite-difference expressions for all first and second derivatives given by the following expressions:

$$\begin{aligned}
\mathbf{r}_\xi &= \frac{\left(\mathbf{r}_{i+1,j} - \mathbf{r}_{i-1,j}\right)}{2\Delta\xi} \\
\mathbf{r}_\eta &= \frac{\left(\mathbf{r}_{i,j+1} - \mathbf{r}_{i,j-1}\right)}{2\Delta\eta} \\
\mathbf{r}_{\xi\xi} &= \frac{\left(\mathbf{r}_{i+1,j} - 2\mathbf{r}_{i,j} + \mathbf{r}_{i-1,j}\right)}{\Delta\xi^2} \\
\mathbf{r}_{\eta\eta} &= \frac{\left(\mathbf{r}_{i,j+1} - 2\mathbf{r}_{i,j} + \mathbf{r}_{i,j-1}\right)}{\Delta\eta^2} \\
\mathbf{r}_{\xi\eta} &= \frac{\left(\mathbf{r}_{i+1,j+1} - \mathbf{r}_{i+1,j-1} - \mathbf{r}_{i-1,j+1} + \mathbf{r}_{i-1,j-1}\right)}{4\Delta\xi\Delta\eta}.
\end{aligned} \tag{34}$$

The updated values of the coordinates at iteration $n+1$, denoted as $\mathbf{r}^+_{i,j}$, are expressed in terms of the old values at iteration n plus the increments from iteration n to $n+1$, denoted as follows:

$$\mathbf{r}_{i,j}^{+} = \mathbf{r}_{i,j} + \Delta\mathbf{r}_{i,j}. \tag{35}$$

Substituting finite-difference expressions (34) and (35) into (27), and linearizing the equations by neglecting all second-order terms, $\mathcal{O}(\Delta\mathbf{r}_{i,j}^2)$, leads to a linear system of equations that must be solved at every iteration. Although (27) is a vector equation, because the grid coordinates x^i are coupled through the product of the metric tensor components and the Christoffel symbols, the x^i coordinates must be solved simultaneously through the solution of one matrix equation encompassing the entire system of equations.

The methodology employed to linearize the discretized form of the grid generation equations is described in detail in the next section.

2.4. Linearization of Grid Generation Equations. In this section, the linearization of the grid generation equations is developed in two steps, first without the curvature control functions and then with the latter included. The linearization process involves the discretization of the equations using second-order finite differences followed by a Newton linearization of the discretized equations. As will be shown in the following sections, the addition of the curvature functions significantly increases the complexity of the derivations.

2.4.1. Linearization of Equations without Curvature Functions. The grid generation equations without the curvature functions are obtained by setting $R^1 = 0$ and $R^2 = 0$ in (27). Employing the notation described in Section 2.3, the updated value of equation (27) at iteration $n+1$ can be written as follows:

$$g_{22}^{+}\mathbf{r}_{\xi\xi}^{+} - 2g_{12}^{+}\mathbf{r}_{\xi\eta}^{+} + g_{11}^{+}\mathbf{r}_{\eta\eta}^{+} + P^{1+}\mathbf{r}_{\xi}^{+} + P^{2+}\mathbf{r}_{\eta}^{+} = 0. \tag{36}$$

Expanding the above, we obtain the following:

$$\begin{aligned}&(g_{22} + \Delta g_{22})(\mathbf{r}_{\xi\xi} + \Delta\mathbf{r}_{\xi\xi}) - 2(g_{12} + \Delta g_{12})(\mathbf{r}_{\xi\eta} + \Delta\mathbf{r}_{\xi\eta})\\ &\quad + (g_{11} + \Delta g_{11})(\mathbf{r}_{\eta\eta} + \Delta\mathbf{r}_{\eta\eta}) + (P^1 + \Delta P^1)(\mathbf{r}_{\xi} + \Delta\mathbf{r}_{\xi})\\ &\quad + (P^2 + \Delta P^2)(\mathbf{r}_{\eta} + \Delta\mathbf{r}_{\eta}) = 0.\end{aligned} \tag{37}$$

Multiplying all terms in (37) and neglecting all higher-order terms, the equation reduces to

$$\begin{aligned}&g_{22}\mathbf{r}_{\xi\xi} - 2g_{12}\mathbf{r}_{\xi\eta} + g_{22}\mathbf{r}_{\eta\eta} + P^1\mathbf{r}_{\xi} + P^2\mathbf{r}_{\eta}\\ &\quad + \left[g_{22}\Delta\mathbf{r}_{\xi\xi} - 2g_{12}\Delta\mathbf{r}_{\xi\eta} + g_{11}\Delta\mathbf{r}_{\eta\eta} + P^1\Delta\mathbf{r}_{\xi} + P^2\Delta\mathbf{r}_{\eta}\right]\\ &\quad + \left[\Delta g_{22}\mathbf{r}_{\xi\xi} - 2\Delta g_{12}\mathbf{r}_{\xi\eta} + \Delta g_{11}\mathbf{r}_{\eta\eta} + \Delta P^1\mathbf{r}_{\xi} + \Delta P^2\mathbf{r}_{\eta}\right] = 0.\end{aligned} \tag{38}$$

To simplify the subsequent derivations, the left-hand side of equation (38) is expressed as follows:

$$\mathbf{G}^{+} = \mathbf{G} + \Delta\mathbf{G}^1 + \Delta\mathbf{G}^2 = 0, \tag{39}$$

where

$$\mathbf{G} = g_{22}\mathbf{r}_{\xi\xi} - 2g_{12}\mathbf{r}_{\xi\eta} + g_{11}\mathbf{r}_{\eta\eta} + P^1\mathbf{r}_{\xi} + P^2\mathbf{r}_{\eta}. \tag{40a}$$

$$\Delta\mathbf{G}^1 = g_{22}\Delta\mathbf{r}_{\xi\xi} - 2g_{12}\Delta\mathbf{r}_{\xi\eta} + g_{11}\Delta\mathbf{r}_{\eta\eta} + P^1\Delta\mathbf{r}_{\xi} + P^2\Delta\mathbf{r}_{\eta}. \tag{40b}$$

$$\Delta\mathbf{G}^2 = \Delta g_{22}\mathbf{r}_{\xi\xi} - 2\Delta g_{12}\mathbf{r}_{\xi\eta} + \Delta g_{11}\mathbf{r}_{\eta\eta} + \Delta P^1\mathbf{r}_{\xi} + \Delta P^2\mathbf{r}_{\eta}. \tag{40c}$$

In equation (40c), all terms must be expressed as functions of the grid coordinates, $\mathbf{r}_{ij}$. To this end, expressions for ΔP^1 and ΔP^2 are derived from equations (26a) and (26b) as follows:

$$\Delta P^1 = \Delta g_{22}p_{11}^1 - 2\Delta g_{12}p_{12}^1 + \Delta g_{11}p_{22}^1. \tag{41a}$$

$$\Delta P^2 = \Delta g_{22}p_{11}^2 - 2\Delta g_{12}p_{12}^2 + \Delta g_{11}p_{22}^2. \tag{41b}$$

In the above expressions, the p_{ij}^p terms represent the grid stretching functions, which are only functions of the (s,t) coordinates. Since the latter are held fixed during the solution process, they do not require the calculation of increments.

Substituting equations (41a) and (41b) into equation (40c) and collecting terms, we obtain the following:

$$\begin{aligned}\Delta\mathbf{G}^2 = {}&\Delta g_{22}\left(\mathbf{r}_{\xi\xi} + p_{11}^1\mathbf{r}_{\xi} + p_{11}^2\mathbf{r}_{\eta}\right) - 2\Delta g_{12}\left(\mathbf{r}_{\xi\eta} + p_{12}^1\mathbf{r}_{\xi} + p_{12}^2\mathbf{r}_{\eta}\right)\\ &+ \Delta g_{11}\left(\mathbf{r}_{\eta\eta} + p_{22}^1\mathbf{r}_{\xi} + p_{22}^2\mathbf{r}_{\eta}\right).\end{aligned} \tag{42}$$

If we now substitute equation (31) in equation (42), we can write,

$$\Delta\mathbf{G}^2 = \Delta g_{22}\mathbf{R}_{\xi\xi} - 2\Delta g_{12}\mathbf{R}_{\xi\eta} + \Delta g_{11}\mathbf{R}_{\eta\eta}. \tag{43}$$

We also need to express the change in the metric tensor components in terms of the grid coordinates. The following expressions can be used for this purpose:

$$g_{11} = \mathbf{r}_{\xi}\cdot\mathbf{r}_{\xi}, \quad g_{12} = \mathbf{r}_{\xi}\cdot\mathbf{r}_{\eta}, \quad g_{22} = \mathbf{r}_{\eta}\cdot\mathbf{r}_{\eta}. \tag{44}$$

Differentiating equation (44) then leads to the following:

$$\begin{aligned}\Delta g_{11} &= 2(\mathbf{r}_{\xi}\cdot\Delta\mathbf{r}_{\xi}) = 2(x_{\xi}\Delta x_{\xi} + y_{\xi}\Delta y_{\xi})\\ \Delta g_{12} &= \mathbf{r}_{\xi}\cdot\Delta\mathbf{r}_{\eta} + \Delta\mathbf{r}_{\xi}\cdot\mathbf{r}_{\eta}.\end{aligned} \tag{45a}$$

$$= x_{\xi}\Delta x_{\eta} + y_{\xi}\Delta y_{\eta} + x_{\eta}\Delta x_{\xi} + y_{\eta}\Delta y_{\xi}. \tag{45b}$$

$$\Delta g_{22} = 2(\mathbf{r}_{\eta}\cdot\Delta\mathbf{r}_{\eta}) = 2(x_{\eta}\Delta x_{\eta} + y_{\eta}\Delta y_{\eta}). \tag{45c}$$

Substituting equations (45a)-(45c) into equation (43) and collecting terms, we get an expression for $\Delta\mathbf{G}^2$ in terms of the increments in the derivatives of the grid coordinates:

$$\begin{aligned}\Delta\mathbf{G}^2 = 2\big\{&(\mathbf{R}_{\eta\eta}x_{\xi} - \mathbf{R}_{\xi\eta}x_{\eta})\Delta x_{\xi} + (\mathbf{R}_{\xi\xi}x_{\eta} - \mathbf{R}_{\xi\eta}x_{\xi})\Delta x_{\eta}\\ &+ (\mathbf{R}_{\eta\eta}y_{\xi} - \mathbf{R}_{\xi\eta}y_{\eta})\Delta y_{\xi} + (\mathbf{R}_{\xi\xi}y_{\eta} - \mathbf{R}_{\xi\eta}y_{\xi})\Delta y_{\eta}\big\}.\end{aligned} \tag{46}$$

Inserting equations (46), (40a) and (40b) into (39), we then obtain the following:

$$\begin{aligned} g_{22}\Delta \mathbf{r}_{\xi\xi} - 2g_{12}\Delta \mathbf{r}_{\xi\eta} + g_{11}\Delta \mathbf{r}_{\eta\eta} + P^1\Delta \mathbf{r}_\xi + P^2\Delta \mathbf{r}_\eta \\ + 2\left(\mathbf{R}_{\eta\eta}x_\xi - \mathbf{R}_{\xi\eta}x_\eta\right)\Delta x_\xi \\ + 2\left(\mathbf{R}_{\xi\xi}x_\eta - \mathbf{R}_{\xi\eta}x_\xi\right)\Delta x_\eta \\ + 2\left(\mathbf{R}_{\eta\eta}y_\xi - \mathbf{R}_{\xi\eta}y_\eta\right)\Delta y_\xi \\ + 2\left(\mathbf{R}_{\xi\xi}y_\eta - \mathbf{R}_{\xi\eta}y_\xi\right)\Delta y_\eta = -\mathbf{G}. \end{aligned} \quad (47)$$

Equation (47) is a vector equation in which $\mathbf{r} = [x\, y]^T$, $\mathbf{R} = [X\, Y]^T$, and $\mathbf{G} = [G_x\, G_y]^T$. To separate this equation into its x and y components, we note that from equation (31), we have the following:

$$X_{\xi^i\xi^j} = x_{\xi^i\xi^j} + p^p_{ij}x_{\xi^p}. \quad (48a)$$

$$Y_{\xi^i\xi^j} = y_{\xi^i\xi^j} + p^p_{ij}y_{\xi^p}. \quad (48b)$$

Therefore, the x and y components of equation (47) can be written as

$$\begin{bmatrix} g_{22}\Delta x_{\xi\xi} - 2g_{12}\Delta x_{\xi\eta} + g_{11}\Delta x_{\eta\eta} \\ +\left\{P^1 + 2\left(X_{\eta\eta}x_\xi - X_{\xi\eta}x_\eta\right)\right\}\Delta x_\xi \\ +\left\{P^2 + 2\left(X_{\xi\xi}x_\eta - X_{\xi\eta}x_\xi\right)\right\}\Delta x_\eta \\ +2\left(X_{\eta\eta}y_\xi - X_{\xi\eta}y_\eta\right)\Delta y_\xi \\ +2\left(X_{\xi\xi}y_\eta - X_{\xi\eta}y_\xi\right)\Delta y_\eta \end{bmatrix} = -G_x. \quad (49)$$

$$\begin{bmatrix} g_{22}\Delta y_{\xi\xi} - 2g_{12}\Delta y_{\xi\eta} + g_{11}\Delta y_{\eta\eta} \\ +2\left(Y_{\eta\eta}x_\xi - Y_{\xi\eta}x_\eta\right)\Delta x_\xi \\ +2\left(Y_{\xi\xi}x_\eta - Y_{\xi\eta}x_\xi\right)\Delta x_\eta \\ +\left\{P^1 + 2\left(Y_{\eta\eta}y_\xi - Y_{\xi\eta}y_\eta\right)\right\}\Delta y_\xi \\ +\left\{P^2 + 2\left(Y_{\xi\xi}y_\eta - Y_{\xi\eta}y_\xi\right)\right\}\Delta y_\eta \end{bmatrix} = -G_y. \quad (50)$$

Equations (49) and (50) are the fully linearized versions of the grid generation equations without curvature control functions. In these equations, all the derivatives are computed using finite differences involving $x_{i,j}$ and $y_{i,j}$, using the expressions given in (34). However, only the updates in the derivatives in these equations, i.e., the Δ terms, involve $x^+_{i,j}$ and $y^+_{i,j}$, while all other terms involve old values of $x_{i,j}$ and $y_{i,j}$ from the previous iteration. Thus, the discretized form of these equations leads to a system of linear equations that can be solved for $x^+_{i,j}$ and $y^+_{i,j}$ at every grid point. The numerical procedure to assemble the linear system of equations based on (49) and (50) is described in detail in Section 3.

2.4.2. Linearization of Curvature Control Functions. The curvature control functions in equation (27) are given by equation (33). The updated values of these terms at iteration $n+1$, denoted as $(gR^p)^+$, are expressed in terms of the old values at iteration n plus the increments from iteration n to $n+1$:

$$(gR^p)^+ = gR^p + \Delta(gR^p). \quad (51)$$

In equation (33), the stretching function derivative terms $(\xi^p_{s^i})$ are held constant during the solution process, and thus $\Delta(gR^p)$ is given by the following:

$$\Delta(gR^p) = \Delta\left(g\overline{C}^i\right)\xi^p_{s^i}, \quad (52)$$

where the $(g\overline{C}^i)$ terms are given by equations (30a) and (30b). In the latter equations, $\hat{J}$ is the Jacobian of the $(s,t) \longrightarrow (\xi,\eta)$ coordinate transformation, and is held constant during the solution process. However, all the other terms, involving derivatives of $\mathbf{r}$ with respect to the (s,t) coordinates, or metric tensor components of the $(x,y) \longrightarrow (s,t)$ coordinate transformation, will vary at every iteration. Therefore, using the chain rule for differentiation, $\Delta(g\overline{C}^1)$ can be expressed as

$$\begin{aligned} \Delta\left(g\overline{C}^1\right) = -\hat{J}^2\Bigg\{\Delta\mathbf{r}_s\cdot\mathbf{r}_{tt} + \mathbf{r}_s\cdot\Delta\mathbf{r}_{tt} - \Delta\left(\frac{\overline{g}_{12}}{\overline{g}_{22}}\right)\mathbf{r}_t\cdot\mathbf{r}_{tt} \\ -\frac{\overline{g}_{12}}{\overline{g}_{22}}\left(\Delta\mathbf{r}_t\cdot\mathbf{r}_{tt} + \mathbf{r}_t\cdot\Delta\mathbf{r}_{tt}\right)\Bigg\}, \end{aligned} \quad (53)$$

and $\Delta(g\overline{C}^2)$ is given by the following:

$$\begin{aligned} \Delta\left(g\overline{C}^2\right) = -\hat{J}^2\Bigg\{\Delta\mathbf{r}_t\cdot\mathbf{r}_{ss} + \mathbf{r}_t\cdot\Delta\mathbf{r}_{ss} - \Delta\left(\frac{\overline{g}_{12}}{\overline{g}_{11}}\right)\mathbf{r}_s\cdot\mathbf{r}_{ss} \\ -\frac{\overline{g}_{12}}{\overline{g}_{11}}\left(\Delta\mathbf{r}_s\cdot\mathbf{r}_{ss} + \mathbf{r}_s\cdot\Delta\mathbf{r}_{ss}\right)\Bigg\}. \end{aligned} \quad (54)$$

To evaluate equations (53) and (54), every term involving an update in the grid coordinates, i.e., every term preceded by a "Δ," must be expanded further. We begin by evaluating $\Delta\mathbf{r}_{s^i}$. From equation (17), $\mathbf{r}_{s^k}$ can be written as follows:

$$\mathbf{r}_{s^i} = \mathbf{r}_{\xi^m}\xi^m_{s^i}. \quad (55)$$

Assuming that the stretching function derivative terms $\xi^m_{s^i}$ are held constant, $\Delta\mathbf{r}_{s^i}$ can thus be written as

$$\Delta\mathbf{r}_{s^i} = \Delta\mathbf{r}_{\xi^m}\xi^m_{s^i}. \quad (56)$$

Therefore, expanding the vector dot product and collecting terms, $\Delta\mathbf{r}_{s^i}\cdot\mathbf{r}_{s^j}$ can be expressed as follows:

$$\begin{aligned} \Delta\mathbf{r}_{s^i}\cdot\mathbf{r}_{s^j} &= \left(\Delta\mathbf{r}_{\xi^m}\xi^m_{s^i}\right)\cdot\mathbf{r}_{s^j} \\ &= \left(\Delta x^l_{\xi^m}\xi^m_{s^i}\right)x^l_{s^j} \\ &= \left(\xi^m_{s^i}x^l_{s^j}\right)\Delta x^l_{\xi^m}. \end{aligned} \quad (57)$$

Similarly, $\Delta\mathbf{r}_{s^i}\cdot\mathbf{r}_{s^is^j}$ can be written as

$$\Delta\mathbf{r}_{s^i}\cdot\mathbf{r}_{s^is^j} = \left(\xi^m_{s^i}x^l_{s^is^j}\right)\Delta x^l_{\xi^m}. \quad (58)$$

To evaluate the $\mathbf{r}_{s^i} \cdot \Delta\mathbf{r}_{s^i s^j}$ terms, we assume again that the stretching function derivatives are held constant. Therefore, using equation (32), $\Delta\mathbf{r}_{s^i s^j}$ can be expressed as follows:

$$\Delta\mathbf{r}_{s^i s^j} = \Delta\mathbf{R}_{\xi^m \xi^n} \xi^m_{s^i} \xi^n_{s^j}. \tag{59}$$

Similarly, evaluating $\Delta\mathbf{R}$ using equation (31) and substituting it into equation (59), we obtain the following:

$$\Delta\mathbf{r}_{s^i s^j} = \left(\Delta\mathbf{r}_{\xi^m \xi^n} + p^k_{mn}\Delta\mathbf{r}_{\xi^k}\right)\xi^m_{s^i}\xi^n_{s^j}. \tag{60}$$

Equating the vector components on both sides of equation (60), we can write

$$\Delta x^l_{s^i s^j} = \left(\Delta x^l_{\xi^m \xi^n} + p^k_{mn}\Delta x^l_{\xi^k}\right)\xi^m_{s^i}\xi^n_{s^j}. \tag{61}$$

Similarly, using equation (55), the components of $\mathbf{r}_{s^i}$ can be expressed as follows:

$$x^l_{s^i} = x^l_{\xi^m}\xi^m_{s^i}. \tag{62}$$

Therefore, taking the dot product of $\mathbf{r}_{s^i}$ and $\Delta\mathbf{r}_{s^i s^j}$ and substituting the expressions for $\Delta x^l_{s^i s^j}$ and $x^l_{s^i}$ given by equations (61) and (62), $\mathbf{r}_{s^i} \cdot \Delta\mathbf{r}_{s^i s^j}$ can be expressed as

$$\begin{aligned}
\mathbf{r}_{s^i} \cdot \Delta\mathbf{r}_{s^i s^j} &= \left(x^l_{s^i}\right)\left(\Delta x^l_{s^i s^j}\right) \\
&= x^l_{s^i}\left(\Delta x^l_{\xi^m \xi^n} + p^k_{mn}\Delta x^l_{\xi^k}\right)\xi^m_{s^i}\xi^n_{s^j} \\
&= \left[x^l_{s^i}\xi^m_{s^i}\xi^n_{s^j}\right]\Delta x^l_{\xi^m \xi^n} + \left[x^l_{s^i}\xi^m_{s^i}\xi^n_{s^j}p^k_{mn}\right]\Delta x^l_{\xi^k}.
\end{aligned} \tag{63}$$

From the expression for the metric tensor in equation (14), $\Delta\overline{g}_{ij}$ can be expressed as follows:

$$\Delta\overline{g}_{ij} = \Delta\mathbf{r}_{s^i} \cdot \mathbf{r}_{s^j} + \Delta\mathbf{r}_{s^j} \cdot \mathbf{r}_{s^i}. \tag{64}$$

Therefore, the change in the covariant metric tensor components for the s^i coordinates can be expressed as follows:

$$\Delta\overline{g}_{12} = \Delta\mathbf{r}_s \cdot \mathbf{r}_t + \mathbf{r}_s \cdot \Delta\mathbf{r}_t. \tag{65a}$$

$$\Delta\overline{g}_{11} = 2\mathbf{r}_s \cdot \Delta\mathbf{r}_s. \tag{65b}$$

$$\Delta\overline{g}_{22} = 2\mathbf{r}_t \cdot \Delta\mathbf{r}_t. \tag{65c}$$

If we then expand the dot products in equation (64) and rearrange, $\Delta\overline{g}_{ij}$ can be expressed as follows:

$$\begin{aligned}
\Delta\overline{g}_{ij} &= \Delta x^l_{s^i}x^l_{s^j} + x^l_{s^i}\Delta x^l_{s^j} \\
&= \left(\Delta x^l_{\xi^m}\xi^m_{s^i}\right)x^l_{s^j} + x^l_{s^i}\left(\Delta x^l_{\xi^m}\xi^m_{s^j}\right) \\
&= \left(\xi^m_{s^i}x^l_{s^j} + \xi^m_{s^j}x^l_{s^i}\right)\Delta x^l_{\xi^m}.
\end{aligned} \tag{66}$$

From the chain rule applied to the differentiation of quotients, we can write the following:

$$\Delta\left(\frac{\overline{g}_{12}}{\overline{g}_{22}}\right) = \frac{\Delta\overline{g}_{12}}{\overline{g}_{22}} - \frac{\overline{g}_{12}}{\overline{g}_{22}}\frac{\Delta\overline{g}_{22}}{\overline{g}_{22}}. \tag{67a}$$

$$\Delta\left(\frac{\overline{g}_{12}}{\overline{g}_{11}}\right) = \frac{\Delta\overline{g}_{12}}{\overline{g}_{11}} - \frac{\overline{g}_{12}}{\overline{g}_{11}}\frac{\Delta\overline{g}_{11}}{\overline{g}_{11}}. \tag{67b}$$

Substituting equation (66) into equations (67a)-(67b), we then have the following

$$\Delta\left(\frac{\overline{g}_{12}}{\overline{g}_{22}}\right) = \frac{1}{\overline{g}_{22}}\left(\xi^m_s x^l_t + \xi^m_t x^l_s\right)\Delta x^l_{\xi^m} - 2\frac{\overline{g}_{12}}{\overline{g}^2_{22}}\left(\xi^m_t x^l_t\right)\Delta x^l_{\xi^m}. \tag{68a}$$

$$\Delta\left(\frac{\overline{g}_{12}}{\overline{g}_{11}}\right) = \frac{1}{\overline{g}_{11}}\left(\xi^m_s x^l_t + \xi^m_t x^l_s\right)\Delta x^l_{\xi^m} - 2\frac{\overline{g}_{12}}{\overline{g}^2_{11}}\left(\xi^m_s x^l_s\right)\Delta x^l_{\xi^m}. \tag{68b}$$

Now, combining equations (58), (63), (67a) and (68a) into equation (53) and rearranging, we get the following expression for $\Delta(g\overline{C}^1)$:

$$\begin{aligned}
\Delta\left(g\overline{C}^1\right) = -\widehat{J}^2\Bigg\{&\left(\xi^m_s x^l_{tt}\right)\Delta x^l_{\xi^m} + \left(x^l_s\xi^m_t\xi^n_t\right)\Delta x^l_{\xi^m\xi^n} \\
&+ \left(x^l_s\xi^m_t\xi^n_t p^k_{mn}\right)\Delta x^l_{\xi^k} \\
&- \frac{(\mathbf{r}_t \cdot \mathbf{r}_{tt})}{\overline{g}_{22}}\left[\left(\xi^m_s x^l_t + \xi^m_t x^l_s\right) - 2\frac{\overline{g}_{12}}{\overline{g}_{22}}\left(\xi^m_t x^l_t\right)\right]\Delta x^l_{\xi^m} \\
&- \frac{\overline{g}_{12}}{\overline{g}_{22}}\left(\xi^m_t x^l_{tt}\right)\Delta x^l_{\xi^m} \\
&- \frac{\overline{g}_{12}}{\overline{g}_{22}}\left[\left(x^l_t\xi^m_t\xi^n_t\right)\Delta x^l_{\xi^m\xi^n} + \left(x^l_t\xi^m_t\xi^n_t p^k_{mn}\right)\Delta x^l_{\xi^k}\right]\Bigg\}.
\end{aligned} \tag{69}$$

After rearranging and collecting terms, we can finally express $\Delta(g\overline{C}^1)$ as follows:

$$\Delta\left(g\overline{C}^1\right) = A^1_{lmn}\Delta x^l_{\xi^m\xi^n} + B^1_{lk}\Delta x^l_{\xi^k}, \tag{70}$$

where

$$A^1_{lmn} = -\widehat{J}^2\xi^m_t\xi^n_t\left(x^l_s - \frac{\overline{g}_{12}}{\overline{g}_{22}}x^l_t\right), \tag{71}$$

and

$$\begin{aligned}
B^1_{lk} = -\widehat{J}^2\Bigg\{&x^l_{tt}\left(\xi^k_s - \frac{\overline{g}_{12}}{\overline{g}_{22}}\xi^k_t\right) \\
&+ \left(\xi^m_t\xi^n_t p^k_{mn}\right)\left(x^l_s - \frac{\overline{g}_{12}}{\overline{g}_{22}}x^l_t\right) \\
&- \frac{(\mathbf{r}_t \cdot \mathbf{r}_{tt})}{\overline{g}_{22}}\left[\xi^k_t\left(x^l_s - 2\frac{\overline{g}_{12}}{\overline{g}_{22}}x^l_t\right) + \xi^k_s x^l_t\right]\Bigg\}.
\end{aligned} \tag{72}$$

Similarly, if we combine equations (58), (63), (67b) and (68b) into equation (54) and rearrange, we get the following expression for $\Delta(g\overline{C}^2)$:

$$\begin{aligned}\Delta\left(g\overline{C}^2\right) = -\widehat{J}^2 \Big\{ &\left(\xi_t^k x_{ss}^l\right)\Delta x_{\xi^k}^l + \left(x_t^l \xi_s^m \xi_s^n\right)\Delta x_{\xi^m\xi^n}^l \\ &+ \left(x_t^l \xi_s^m \xi_s^n p_{mn}^k\right)\Delta x_{\xi^k}^l \\ &- \frac{(\mathbf{r}_s\cdot\mathbf{r}_{ss})}{\overline{g}_{11}}\left[\left(\xi_s^k x_t^l + \xi_t^k x_s^l\right) - 2\frac{\overline{g}_{12}}{\overline{g}_{11}}\left(\xi_s^k x_s^l\right)\right]\Delta x_{\xi^k}^l \\ &- \frac{\overline{g}_{12}}{\overline{g}_{11}}\left(\xi_s^k x_{ss}^l\right)\Delta x_{\xi^k}^l \\ &- \frac{\overline{g}_{12}}{\overline{g}_{11}}\left[\left(x_s^l \xi_s^m \xi_s^n\right)\Delta x_{\xi^m\xi^n}^l + \left(x_s^l \xi_s^m \xi_s^n p_{mn}^k\right)\Delta x_{\xi^k}^l\right]\Big\}.\end{aligned} \tag{73}$$

Rearranging and collecting terms in (73), we can then write $\Delta(g\overline{C}^2)$ as follows:

$$\Delta\left(g\overline{C}^2\right) = A_{lmn}^2 \Delta x_{\xi^m\xi^n}^l + B_{lk}^2 \Delta x_{\xi^k}^l, \tag{74}$$

where

$$A_{lmn}^2 = -\widehat{J}^2 \xi_s^m \xi_s^n \left(x_t^l - \frac{\overline{g}_{12}}{\overline{g}_{11}} x_s^l\right), \tag{75}$$

and

$$\begin{aligned}B_{lk}^2 = -\widehat{J}^2 \Bigg\{ &x_{ss}^l \left(\xi_t^k - \frac{\overline{g}_{12}}{\overline{g}_{11}}\xi_s^k\right) \\ &+ \left(\xi_s^m \xi_s^n p_{mn}^k\right)\left(x_t^l - \frac{\overline{g}_{12}}{\overline{g}_{11}} x_s^l\right) \\ &- \frac{(\mathbf{r}_s\cdot\mathbf{r}_{ss})}{\overline{g}_{11}}\left[\xi_s^k\left(x_t^l - 2\frac{\overline{g}_{12}}{\overline{g}_{11}} x_s^l\right) + \xi_t^k x_s^l\right]\Bigg\}.\end{aligned} \tag{76}$$

Using expressions for $\Delta(g\overline{C}^1)$ and $\Delta(g\overline{C}^2)$ from equations (70) and (74), respectively, $\Delta(gR^p)$ in equation (52) can now be expressed as

$$\begin{aligned}\Delta\left(gR^p\right) &= \Delta\left(g\overline{C}^1\right)\xi_s^p + \Delta\left(g\overline{C}^2\right)\xi_t^p \\ &= \left(A_{lmn}^1 \xi_s^p + A_{lmn}^2 \xi_t^p\right)\Delta x_{\xi^m\xi^n}^l \\ &\quad + \left(B_{lk}^1 \xi_s^p + B_{lk}^2 \xi_t^p\right)\Delta x_{\xi^k}^l.\end{aligned} \tag{77}$$

We now have all the terms necessary to evaluate $(gR^p)^+$ in equation (51).

In Section 2.4.1, the derivation of the linearized grid generation equations ((49) and (50)) excluded the curvature terms (gR^p). Adding these terms back into equation (27) and applying the linearization process described in Section 2.4.1, the following additional terms must be added to the left-hand side of equation (38):

$$g\left(R^1\mathbf{r}_\xi + R^2\mathbf{r}_\eta + R^1\Delta\mathbf{r}_\xi + R^2\Delta\mathbf{r}_\eta\right) + \Delta\left(gR^1\right)\mathbf{r}_\xi + \Delta\left(gR^2\right)\mathbf{r}_\eta. \tag{78}$$

Separating equation (78) into its x and y components then yields the following expressions:

$$g\left(R^1 x_\xi + R^2 x_\eta + R^1\Delta x_\xi + R^2\Delta x_\eta\right) + \Delta\left(gR^1\right)x_\xi + \Delta\left(gR^2\right)x_\eta, \tag{79a}$$

and

$$g\left(R^1 y_\xi + R^2 y_\eta + R^1\Delta y_\xi + R^2\Delta y_\eta\right) + \Delta\left(gR^1\right)y_\xi + \Delta\left(gR^2\right)y_\eta. \tag{79b}$$

If we now add expression (79a) to the left-hand side of equation (49) and expression (79b) to left-hand side of equation (50), and then substitute the expressions for $\Delta(g\overline{C}^1)$ and $\Delta(g\overline{C}^2)$ given by equation (77), we obtain the following complete linearized grid generation equations with curvature control functions:

$$\begin{aligned}&g_{22}\Delta x_{\xi\xi} - 2g_{12}\Delta x_{\xi\eta} + g_{11}\Delta x_{\xi\xi} \\ &+\left[P^1 + 2\left(X_{\eta\eta}x_\xi - X_{\xi\eta}x_\eta\right) + gR^1\right]\Delta x_\xi \\ &+\left[P^2 + 2\left(X_{\xi\xi}x_\eta - X_{\xi\eta}x_\xi\right) + gR^2\right]\Delta x_\eta \\ &+2\left(X_{\eta\eta}y_\xi - X_{\xi\eta}y_\eta\right)\Delta y_\xi \\ &+2\left(X_{\xi\xi}y_\eta - X_{\xi\eta}y_\xi\right)\Delta y_\eta \\ &+\left(A_{lmn}^1\xi_s + A_{lmn}^2\xi_t\right)x_\xi\Delta x_{\xi^m\xi^n}^l + \left(B_{lk}^1\xi_s + B_{lk}^2\xi_t\right)x_\xi\Delta x_{\xi^k}^l \\ &+\left(A_{lmn}^1\eta_s + A_{lmn}^2\eta_t\right)x_\eta\Delta x_{\xi^m\xi^n}^l + \left(B_{lk}^1\eta_s + B_{lk}^2\eta_t\right)x_\eta\Delta x_{\xi^k}^l = -G_x - gR^1 x_\xi - gR^2 x_\eta.\end{aligned} \tag{80}$$

and

$$\begin{aligned}&g_{22}\Delta y_{\xi\xi} - 2g_{12}\Delta y_{\xi\eta} + g_{11}\Delta y_{\xi\xi} \\ &+2\left(Y_{\eta\eta}x_\xi - Y_{\xi\eta}x_\eta\right)\Delta x_\xi \\ &+2\left(Y_{\xi\xi}x_\eta - Y_{\xi\eta}x_\xi\right)\Delta x_\eta \\ &+\left[P^1 + 2\left(Y_{\eta\eta}y_\xi - Y_{\xi\eta}y_\eta\right) + gR^1\right]\Delta y_\xi \\ &+\left[P^2 + 2\left(Y_{\xi\xi}y_\eta - Y_{\xi\eta}y_\xi\right) + gR^2\right]\Delta y_\eta \\ &+\left(A_{lmn}^1\xi_s + A_{lmn}^2\xi_t\right)y_\xi\Delta x_{\xi^m\xi^n}^l + \left(B_{lk}^1\xi_s + B_{lk}^2\xi_t\right)y_\xi\Delta x_{\xi^k}^l \\ &+\left(A_{lmn}^1\eta_s + A_{lmn}^2\eta_t\right)y_\eta\Delta x_{\xi^m\xi^n}^l + \left(B_{lk}^1\eta_s + B_{lk}^2\eta_t\right)y_\eta\Delta x_{\xi^k}^l = -G_y - gR^1 y_\xi - gR^2 y_\eta.\end{aligned} \tag{81}$$

Equations (80) and (81) are the fully linearized equations that must be solved numerically at each grid point to compute the x and y coordinates. To facilitate the discretization of these equations, we now recast them in terms of the coefficients multiplying each of the increments in the derivative terms, i.e., the coefficients of $\Delta x_{\xi^m}^l$ and $\Delta x_{\xi^m\xi^n}^l$, leading to equations of the following form:

$$\begin{aligned}&C_{x_{ij}}^1\Delta x_\xi + C_{x_{ij}}^2\Delta x_\eta + C_{x_{ij}}^3\Delta y_\xi + C_{x_{ij}}^4\Delta y_\eta + C_{x_{ij}}^5\Delta x_{\xi\xi} \\ &+C_{x_{ij}}^6\Delta x_{\xi\eta} + C_{x_{ij}}^7\Delta x_{\eta\eta} + C_{x_{ij}}^8\Delta y_{\xi\xi} + C_{x_{ij}}^9\Delta y_{\xi\eta} + C_{x_{ij}}^{10}\Delta y_{\eta\eta} = Rh_{x_{ij}}.\end{aligned} \tag{82}$$

$$\begin{aligned}&C_{y_{ij}}^1\Delta x_\xi + C_{y_{ij}}^2\Delta x_\eta + C_{y_{ij}}^3\Delta y_\xi + C_{y_{ij}}^4\Delta y_\eta + C_{y_{ij}}^5\Delta x_{\xi\xi} \\ &+C_{y_{ij}}^6\Delta x_{\xi\eta} + C_{y_{ij}}^7\Delta x_{\eta\eta} + C_{y_{ij}}^8\Delta y_{\xi\xi} + C_{y_{ij}}^9\Delta y_{\xi\eta} + C_{y_{ij}}^{10}\Delta y_{\eta\eta} = Rh_{y_{ij}}.\end{aligned} \tag{83}$$

where

$$C^1_{x_{ij}} = P^1 + 2\left(X_{\eta\eta}x_\xi - X_{\xi\eta}x_\eta\right) + gR^1 + B^1_{11}x_s + B^2_{11}x_t. \tag{84a}$$

$$C^2_{x_{ij}} = P^2 + 2\left(X_{\xi\xi}x_\eta - X_{\xi\eta}x_\xi\right) + gR^2 + B^1_{12}x_s + B^2_{12}x_t. \tag{84b}$$

$$C^3_{x_{ij}} = 2\left(X_{\eta\eta}y_\xi - X_{\xi\eta}y_\eta\right) + B^1_{21}x_s + B^2_{21}x_t. \tag{84c}$$

$$C^4_{x_{ij}} = 2\left(X_{\xi\xi}y_\eta - X_{\xi\eta}y_\xi\right) + B^1_{22}x_s + B^2_{22}x_t. \tag{84d}$$

$$C^5_{x_{ij}} = g_{22} + A^1_{111}x_s + A^2_{111}x_t. \tag{84e}$$

$$C^6_{x_{ij}} = -2g_{12} + A^1_{112}x_s + A^2_{112}x_t. \tag{84f}$$

$$C^7_{x_{ij}} = g_{11} + A^1_{122}x_s + A^2_{122}x_t. \tag{84g}$$

$$C^8_{x_{ij}} = A^1_{211}x_s + A^2_{211}x_t. \tag{84h}$$

$$C^9_{x_{ij}} = A^1_{212}x_s + A^2_{212}x_t. \tag{84i}$$

$$C^{10}_{x_{ij}} = A^1_{222}x_s + A^2_{222}x_t, \tag{84j}$$

and

$$C^1_{y_{ij}} = 2\left(Y_{\eta\eta}x_\xi - Y_{\xi\eta}x_\eta\right) + B^1_{11}y_s + B^2_{11}y_t. \tag{85a}$$

$$C^2_{y_{ij}} = 2\left(Y_{\xi\xi}x_\eta - Y_{\xi\eta}x_\xi\right) + B^1_{12}y_s + B^2_{12}y_t. \tag{85b}$$

$$C^3_{y_{ij}} = P^1 + 2\left(Y_{\eta\eta}y_\xi - Y_{\xi\eta}y_\eta\right) + gR^1 + B^1_{21}y_s + B^2_{21}y_t. \tag{85c}$$

$$C^4_{y_{ij}} = P^2 + 2\left(Y_{\xi\xi}y_\eta - Y_{\xi\eta}y_\xi\right) + gR^2 + B^1_{22}y_s + B^2_{22}y_t. \tag{85d}$$

$$C^5_{y_{ij}} = A^1_{111}y_s + A^2_{111}y_t. \tag{85e}$$

$$C^6_{y_{ij}} = A^1_{112}y_s + A^2_{112}y_t. \tag{85f}$$

$$C^7_{y_{ij}} = A^1_{122}y_s + A^2_{122}y_t. \tag{85g}$$

$$C^8_{y_{ij}} = g_{22} + A^1_{211}y_s + A^2_{211}y_t. \tag{85h}$$

$$C^9_{y_{ij}} = -2g_{12} + A^1_{212}y_s + A^2_{212}y_t. \tag{85i}$$

$$C^{10}_{y_{ij}} = g_{11} + A^1_{222}y_s + A^2_{222}y_t, \tag{85j}$$

and where

$$Rh_{x_{ij}} = -G_x - gR^1x_\xi - gR^2x_\eta. \tag{86a}$$

$$Rh_{y_{ij}} = -G_y - gR^1y_\xi - gR^2y_\eta. \tag{86b}$$

In equations (84a)–(84j) to (86a)–(86b), the curvature control functions can be deactivated by setting all the R^p, A^p_{lmn} and B^p_{lk} terms to zero. The numerical implementation of equations (82) and (83) is described in detail in the next section.

3. Numerical Implementation

3.1. Setup of the Matrix Equation. Writing the finite-difference expressions in (34) in terms of the change in x and y coordinates from iteration n to $n+1$, we have

$$\Delta x_\xi = \frac{\left(\Delta x_{i+1,j} - \Delta x_{i-1,j}\right)}{2\Delta\xi}. \tag{87a}$$

$$\Delta y_\xi = \frac{\left(\Delta y_{i+1,j} - \Delta y_{i-1,j}\right)}{2\Delta\xi}. \tag{87b}$$

$$\Delta x_\eta = \frac{\left(\Delta x_{i,j+1} - \Delta x_{i,j-1}\right)}{2\Delta\eta}. \tag{87c}$$

$$\Delta y_\eta = \frac{\left(\Delta y_{i,j+1} - \Delta y_{i,j-1}\right)}{2\Delta\eta}. \tag{87d}$$

$$\Delta x_{\xi\xi} = \frac{\left(\Delta x_{i+1,j} - 2\Delta x_{i,j} + \Delta x_{i-1,j}\right)}{\Delta\xi^2}. \tag{87e}$$

$$\Delta y_{\xi\xi} = \frac{\left(\Delta y_{i+1,j} - 2\Delta y_{i,j} + \Delta y_{i-1,j}\right)}{\Delta\xi^2}. \tag{87f}$$

$$\Delta x_{\eta\eta} = \frac{\left(\Delta x_{i,j+1} - 2\Delta x_{i,j} + \Delta x_{i,j-1}\right)}{\Delta\eta^2}. \tag{87g}$$

$$\Delta y_{\eta\eta} = \frac{\left(\Delta y_{i,j+1} - 2\Delta y_{i,j} + \Delta y_{i,j-1}\right)}{\Delta\eta^2}. \tag{87h}$$

$$\Delta x_{\xi\eta} = \frac{\left(\Delta x_{i+1,j+1} - \Delta x_{i+1,j-1} - \Delta x_{i-1,j+1} + \Delta x_{i-1,j-1}\right)}{4\Delta\xi\Delta\eta}. \tag{87i}$$

$$\Delta y_{\xi\eta} = \frac{\left(\Delta y_{i+1,j+1} - \Delta y_{i+1,j-1} - \Delta y_{i-1,j+1} + \Delta y_{i-1,j-1}\right)}{4\Delta\xi\Delta\eta}. \tag{87j}$$

In the mesh generation process, the curvilinear coordinates, ξ and η, are the independent variables, and for convenience, they are set equal to the mesh indices i and j, respectively. This implies that ξ and η vary from 1 to *imax* and *jmax*, respectively, and thus $\Delta\xi = \Delta\eta = 1$ everywhere in the mesh. Therefore, substituting the finite-difference expressions from equations (87a)–(87j) into equations (82) and (83), yields the following:

$$\begin{aligned}
&-\frac{1}{4}C^6_{x_{ij}}\Delta x_{i+1,j-1} - \left(\frac{1}{2}C^2_{x_{ij}} - C^7_{x_{ij}}\right)\Delta x_{i,j-1} + \frac{1}{4}C^6_{x_{ij}}\Delta x_{i-1,j-1}\\
&+\left(\frac{1}{2}C^1_{x_{ij}} + C^5_{x_{ij}}\right)\Delta x_{i+1,j} - 2\left(C^5_{x_{ij}} + C^7_{x_{ij}}\right)\Delta x_{i,j} - \left(\frac{1}{2}C^1_{xij} - C^5_{x_{ij}}\right)\Delta x_{i-1,j}\\
&+\frac{1}{4}C^6_{x_{ij}}\Delta x_{i+1,j+1} + \left(\frac{1}{2}C^2_{x_{ij}} + C^7_{x_{ij}}\right)\Delta x_{i,j+1} - \frac{1}{4}C^6_{x_{ij}}\Delta x_{i-1,j+1}\\
&-\frac{1}{4}C^9_{x_{ij}}\Delta y_{i+1,j-1} - \left(\frac{1}{2}C^4_{x_{ij}} - C^{10}_{x_{ij}}\right)\Delta y_{i,j-1} + \frac{1}{4}C^9_{x_{ij}}\Delta y_{i-1,j-1}\\
&+\left(\frac{1}{2}C^3_{x_{ij}} + C^8_{x_{ij}}\right)\Delta y_{i+1,j} - 2\left(C^8_{x_{ij}} + C^{10}_{x_{ij}}\right)\Delta y_{i,j} - \left(\frac{1}{2}C^3_{x_{ij}} - C^8_{x_{ij}}\right)\Delta y_{i-1,j}\\
&+\frac{1}{4}C^9_{x_{ij}}\Delta y_{i+1,j+1} + \left(\frac{1}{2}C^4_{x_{ij}} + C^{10}_{x_{ij}}\right)\Delta y_{i,j+1} - \frac{1}{4}C^9_{x_{ij}}\Delta y_{i-1,j+1} = Rh_{x_{ij}},
\end{aligned} \tag{88}$$

and

$$\begin{aligned}
&-\frac{1}{4}C^6_{y_{ij}}\Delta x_{i+1,j-1} - \left(\frac{1}{2}C^2_{y_{ij}} - C^7_{y_{ij}}\right)\Delta x_{i,j-1} + \frac{1}{4}C^6_{y_{ij}}\Delta x_{i-1,j-1}\\
&+\left(\frac{1}{2}C^1_{y_{ij}} + C^5_{y_{ij}}\right)\Delta x_{i+1,j} - 2\left(C^5_{y_{ij}} + C^7_{y_{ij}}\right)\Delta x_{i,j} - \left(\frac{1}{2}C^1_{y_{ij}} - C^5_{y_{ij}}\right)\Delta x_{i-1,j}\\
&+\frac{1}{4}C^6_{y_{ij}}\Delta x_{i+1,j+1} + \left(\frac{1}{2}C^2_{y_{ij}} + C^7_{y_{ij}}\right)\Delta x_{i,j+1} - \frac{1}{4}C^6_{y_{ij}}\Delta x_{i-1,j+1}\\
&-\frac{1}{4}C^9_{y_{ij}}\Delta y_{i+1,j-1} - \left(\frac{1}{2}C^4_{y_{ij}} - C^{10}_{y_{ij}}\right)\Delta y_{i,j-1} + \frac{1}{4}C^9_{y_{ij}}\Delta y_{i-1,j-1}\\
&+\left(\frac{1}{2}C^3_{y_{ij}} + C^8_{y_{ij}}\right)\Delta y_{i+1,j} - 2\left(C^8_{y_{ij}} + C^{10}_{y_{ij}}\right)\Delta y_{i,j} - \left(\frac{1}{2}C^3_{y_{ij}} - C^8_{y_{ij}}\right)\Delta y_{i-1,j}\\
&+\frac{1}{4}C^9_{y_{ij}}\Delta y_{i+1,j+1} + \left(\frac{1}{2}C^4_{y_{ij}} + C^{10}_{y_{ij}}\right)\Delta y_{i,j+1} - \frac{1}{4}C^9_{y_{ij}}\Delta y_{i-1,j+1} = Rh_{y_{ij}}.
\end{aligned} \tag{89}$$

For the ease of presentation, we introduce the parameters $J^k_{x_{ij}}$ and $J^k_{y_{ij}}$ to represent the coefficients in equations (88) and (89) as follows:

$$\begin{aligned}
&J^1_{x_{ij}} = \frac{1}{4}C^6_{x_{ij}}; \quad J^2_{x_{ij}} = -\left(\frac{1}{2}C^2_{x_{ij}} - C^7_{x_{ij}}\right); \quad J^3_{x_{ij}} = \left(\frac{1}{2}C^1_{x_{ij}} + C^5_{x_{ij}}\right)\\
&J^4_{x_{ij}} = -2\left(C^5_{x_{ij}} + C^7_{x_{ij}}\right); \quad J^5_{x_{ij}} = -\left(\frac{1}{2}C^1_{x_{ij}} - C^5_{x_{ij}}\right); \quad J^6_{x_{ij}} = \left(\frac{1}{2}C^2_{x_{ij}} + C^7_{x_{ij}}\right)\\
&J^7_{x_{ij}} = \frac{1}{4}C^9_{x_{ij}}; \quad J^8_{x_{ij}} = -\left(\frac{1}{2}C^4_{x_{ij}} - C^{10}_{x_{ij}}\right); \quad J^9_{x_{ij}} = \left(\frac{1}{2}C^3_{x_{ij}} + C^8_{x_{ij}}\right)\\
&J^{10}_{x_{ij}} = -2\left(C^8_{x_{ij}} + C^{10}_{x_{ij}}\right); \quad J^{11}_{x_{ij}} = -\left(\frac{1}{2}C^3_{x_{ij}} - C^8_{x_{ij}}\right); \quad J^{12}_{x_{ij}} = \left(\frac{1}{2}C^4_{x_{ij}} + C^{10}_{x_{ij}}\right),
\end{aligned} \tag{90}$$

and

$$\begin{aligned}
&J^1_{y_{ij}} = \frac{1}{4}C^6_{y_{ij}}; \quad J^2_{y_{ij}} = -\left(\frac{1}{2}C^2_{y_{ij}} - C^7_{y_{ij}}\right); \quad J^3_{y_{ij}} = \left(\frac{1}{2}C^1_{y_{ij}} + C^5_{y_{ij}}\right)\\
&J^4_{y_{ij}} = -2\left(C^5_{y_{ij}} + C^7_{y_{ij}}\right); \quad J^5_{y_{ij}} = -\left(\frac{1}{2}C^1_{y_{ij}} - C^5_{y_{ij}}\right); \quad J^6_{y_{ij}} = \left(\frac{1}{2}C^2_{y_{ij}} + C^7_{y_{ij}}\right)\\
&J^7_{y_{ij}} = \frac{1}{4}C^9_{y_{ij}}; \quad J^8_{y_{ij}} = -\left(\frac{1}{2}C^4_{y_{ij}} - C^{10}_{y_{ij}}\right); \quad J^9_{y_{ij}} = \left(\frac{1}{2}C^3_{y_{ij}} + C^8_{y_{ij}}\right)\\
&J^{10}_{y_{ij}} = -2\left(C^8_{y_{ij}} + C^{10}_{y_{ij}}\right); \quad J^{11}_{y_{ij}} = -\left(\frac{1}{2}C^3_{y_{ij}} - C^8_{y_{ij}}\right); \quad J^{12}_{y_{ij}} = \left(\frac{1}{2}C^4_{y_{ij}} + C^{10}_{y_{ij}}\right).
\end{aligned} \tag{91}$$

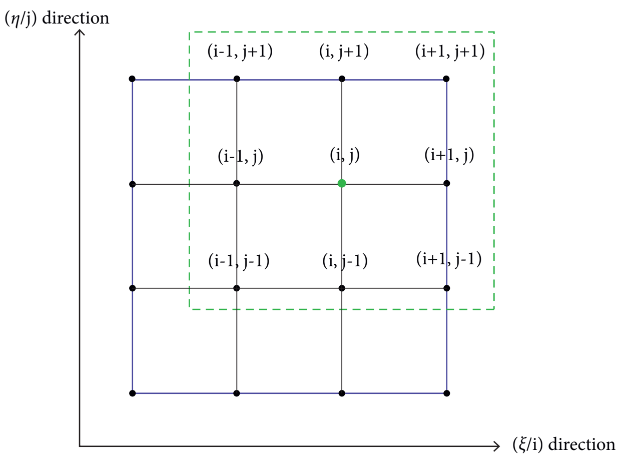

Figure 3: 4×4 grid with a 9-point stencil.

Thus, inserting equations (90) and (91) into equations (88) and (89), respectively, we obtain the following

$$\begin{aligned}
&J^5_{x_{ij}}\Delta x_{i-1,j} - J^1_{x_{ij}}\Delta x_{i+1,j-1} + J^2_{x_{ij}}\Delta x_{i,j-1} + J^1_{x_{ij}}\Delta x_{i-1,j-1} + J^3_{x_{ij}}\Delta x_{i+1,j} + J^4_{x_{ij}}\Delta x_{i,j}\\
&\quad + J^1_{x_{ij}}\Delta x_{i+1,j+1} + J^6_{x_{ij}}\Delta x_{i,j+1} - J^1_{x_{ij}}\Delta x_{i-1,j+1}\\
&\quad - J^7_{x_{ij}}\Delta y_{i+1,j-1} + J^8_{x_{ij}}\Delta y_{i,j-1} + J^7_{x_{ij}}\Delta y_{i-1,j-1} + J^9_{x_{ij}}\Delta y_{i+1,j} + J^{10}_{x_{ij}}\Delta y_{i,j}\\
&\quad + J^{11}_{x_{ij}}\Delta y_{i-1,j} + J^7_{x_{ij}}\Delta y_{i+1,j+1} + J^{12}_{x_{ij}}\Delta y_{i,j+1} - J^7_{x_{ij}}\Delta y_{i-1,j+1} = Rh_{x_{ij}},
\end{aligned} \tag{92}$$

and

$$\begin{aligned}
&J^5_{y_{ij}}\Delta x_{i-1,j} - J^1_{y_{ij}}\Delta x_{i+1,j-1} + J^2_{y_{ij}}\Delta x_{i,j-1} + J^1_{y_{ij}}\Delta x_{i-1,j-1} + J^3_{y_{ij}}\Delta x_{i+1,j} + J^4_{y_{ij}}\Delta x_{i,j}\\
&\quad + J^1_{y_{ij}}\Delta x_{i+1,j+1} + J^6_{y_{ij}}\Delta x_{i,j+1} - J^1_{y_{ij}}\Delta x_{i-1,j+1}\\
&\quad - J^7_{y_{ij}}\Delta y_{i+1,j-1} + J^8_{y_{ij}}\Delta y_{i,j-1} + J^7_{y_{ij}}\Delta y_{i-1,j-1} + J^9_{y_{ij}}\Delta y_{i+1,j} + J^{10}_{y_{ij}}\Delta y_{i,j}\\
&\quad + J^{11}_{y_{ij}}\Delta y_{i-1,j} + J^7_{y_{ij}}\Delta y_{i+1,j+1} + J^{12}_{y_{ij}}\Delta y_{i,j+1} - J^7_{y_{ij}}\Delta y_{i-1,j+1} = Rh_{y_{ij}}.
\end{aligned} \tag{93}$$

Equations (92) and (93) are the complete discretized forms of the grid generation equations that must be solved at each grid point of the mesh. As can be seen from these equations, the second-order finite difference expressions result in a 9-point stencil surrounding each grid point (i, j), as illustrated in Figure 3 for a 4×4 grid.

Equations (92) and (93) yield a linear system of equations involving a nonsymmetric Jacobian matrix, with entries given by the $J^k_{x_{ij}}$ and $J^k_{y_{ij}}$ terms. This matrix is populated by sweeping the mesh one row at a time along the i direction, and placing the coefficients multiplying the x and y coordinates of every point in adjacent rows. This ordering of the mesh nodes yields the matrix with the smallest bandwidth of nonzero entries about the diagonal, as illustrated by the matrix sparsity pattern shown in Figure 4. As shown, the matrix consists of three diagonal bands, each consisting of a block tridiagonal band of 2×2 blocks. The distance of the upper and lower bands from the diagonal is only 2(*jmax* - 5) columns, where *jmax* is the upper limit of the smaller of the two mesh indices, (i, j). As shown in Figure 4(a), the total bandwidth of the

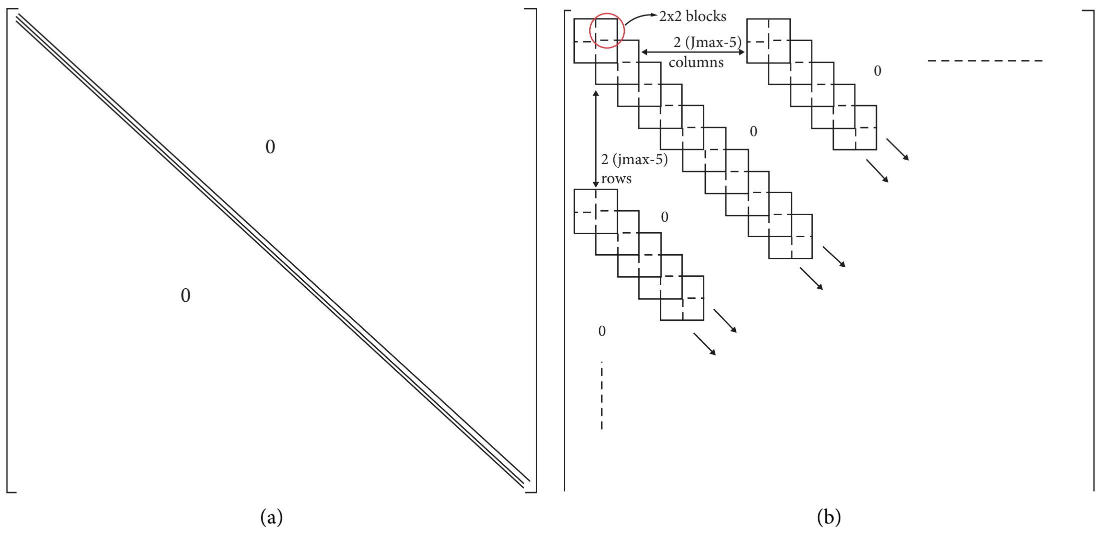

FIGURE 4: Matrix sparsity pattern. (a) Sparsity pattern. (b) Details of upper-left corner.

nonzero elements is very small relative to the overall size of the matrix.

To solve this system, a number of solvers and preconditioners, drawn from the Portable Extensible Toolkit for Scientific Computation (PETSc) library [28], are employed. The solution vector obtained from the PETSc solution gives the changes in the coordinates, $\Delta x_{i,j}$ and $\Delta y_{i,j}$, at each grid point. The updated values of the coordinates at iteration $n+1$ are then obtained as follows:

$$\begin{aligned} x_{i,j}^{n+1} &= x_{i,j}^{n} + \Delta x_{i,j}^{n+1}, \\ y_{i,j}^{n+1} &= y_{i,j}^{n} + \Delta y_{i,j}^{n+1}. \end{aligned} \qquad (94)$$

3.2. Evaluation of Solvers and Preconditioners. The PETSc library is a collection of data structures and routines for the solution of scientific and engineering problems formulated by partial differential equations. It consists of a number of libraries which contain modules for creating both sequential and parallel vectors and matrices as well as modules dealing with linear and nonlinear solvers. It is structured such that each module can be invoked by a set of calling sequences facilitating the solution of PDE's [29]. Once the interface to the PETSc library is established, all solvers and preconditioners are accessible. This approach enables the evaluation of various solvers and preconditioners for a particular problem to determine the best combination for tackling the problem at hand.

3.2.1. Selected Solvers and Preconditioners. In the applications presented Section 4, virtually all methods available in the PETSc library were evaluated, and the set of preconditioners and solvers that turned out to be the most effective were identified as follows:

Preconditioners:

ILU: Incomplete lower-upper (LU) factorization.
SOR: Successive over-relaxation variant of Gauss-Seidel.
EISENSTAT: SSOR (symmetric successive over-relaxation, symmetric Gauss-Seidel) incorporating Eisenstat's "trick" to reduce computation.

Solvers:

LU: Direct solver based on lower-upper factorization.
BiCGSTAB: Stabilized version of Biconjugate Gradient method.
GMRES: Generalized Minimal Residual method, optionally with accelerated restart.
PIPELINED GCR: Pipelined Generalized Conjugate Residual method.

3.3. Implementation in Fortran. The numerical solution of the grid generation equations with curvature functions was implemented in a fortran code. The simplified algorithm describing the implementation of the solution method is summarized in Algorithm 1 below.

4. Applications

The solution process developed herein was applied to the generation of two C-meshes around a two-dimensional airfoil. The first mesh is suitable for the solution of the Euler equations, while the second is a more refined mesh suitable for the Navier–Stokes equations, in which the height of the first layer of cells on the airfoil boundary is set to $10^{-6} \times c$, where c is the airfoil chord. The Euler mesh consists of 321×49 (15,729) points, yielding a matrix with dimensions $[29986 \times 29986]$, while the Navier–Stokes mesh contains 321×97 (31,137) points, resulting in a matrix with dimensions $[60610 \times 60610]$. These matrices are very sparse, with the number of nonzero entries being equal to 530,980 for the Euler mesh, and 1,081,060 for the Navier-Stokes mesh, such that only approximately 0.06% and 0.03% of the matrix entries are nonzero, respectively. Figure 5 shows the initial algebraic mesh and domain boundaries. The far field regions of the Euler and Navier–Stokes meshes are virtually

```
(1) Read the initial algebraic mesh
(2) Initialize various PETSc variables
(3) **For** $j = 2 \longrightarrow (jmax - 1)$ **do**
(4) **For** $i = 2 \longrightarrow (imax - 1)$ **do**
(5) Set the exact no. of nonzeros in each row of the matrix depending on various conditions
(6) **End**
(7) **End**
(8) Set up the PETSc matrix and vector
(9) **For** $iter = 1 \longrightarrow niter$ **do**
(10) **For** $j = 2 \longrightarrow jmax - 1$ **do**
(11) **For** $i = 2 \longrightarrow imax - 1$ **do**
(12) (1) Calculate derivatives of $x$ and $y$ w.r.t. to $\xi$ and $\eta$ (equations (87a)–(87j))
(13) (2) Evaluate the metric tensor components (equations (14)–(16))
(14) (3) Calculate the control functions (stretching and orthogonality) (equations (19) and (26a)-(26b))
(15) (4) Evaluate the curvature control terms (equations (10) and (28a)-(28b))
(16) (5) Evaluate the RHS vector components for $x$ and $y$ (equations (86a)-(86b))
(17) (6) Evaluate the coefficients of the linearized equations (equations (92)-(93))
(18) (7) Populate the matrix entries in the nonzero locations
(19) **End**
(20) **End**
(21) (1) Assemble the PETSc vector and matrix
(22) (2) Solve the linear system using any of the KSP methods
(23) (3) Extract the solution vector
(24) (4) Update the values of $x$ and $y$
(25) **End**
(26) (1) Delete the PETSc objects
(27) (2) Write the output file
```

Algorithm 1: Algorithm for the implementation in Fortran.

identical, as the meshes only differ in the region close to the airfoil surface. Figures 6(a) and 6(b) show the two meshes in the vicinity of the airfoil boundary. The initial meshes are generated using transfinite interpolation from the boundaries.

Both the Euler and Navier–Stokes meshes were generated with and without curvature control functions, in order to compare the properties of the meshes as well as the convergence characteristics of the solution process. In addition, for the meshes generated without curvature functions, optimizations were done with and without orthogonality functions. The reason for this is that meshes generated with curvature functions are inherently more orthogonal than meshes generated without them, and for many applications, including the ones presented here, orthogonality functions are not required. Therefore, in the following figures the effects of turning on and off both the curvature functions and the orthogonality functions are presented and discussed.

The first optimizations were performed on the Euler meshes and the results are shown in Figures 7 to 10. Figure 7 shows the far field of the Euler meshes optimized without curvature functions, where 7(a) and 7(b) show the meshes optimized without and with orthogonality functions, respectively. Closeup views of these meshes are provided in Figure 8. It is clear from these figures that orthogonality functions are required to obtain a high-quality mesh in this case, as is virtually always the case when using a Laplace-based operator. To highlight the impact of the curvature functions on orthogonality, Figures 9 and 10 show the Euler mesh optimized with curvature functions, but without orthogonality functions. These figures can be compared to Figures 7(a) and 8(a), respectively, to see the impact of the curvature functions on the mesh. As can be seen in these figures, the mesh optimized with curvature functions is fully orthogonal on the airfoil boundary, without the need for dedicated orthogonality functions, in contrast to the mesh optimized without curvature functions. Also, evident in

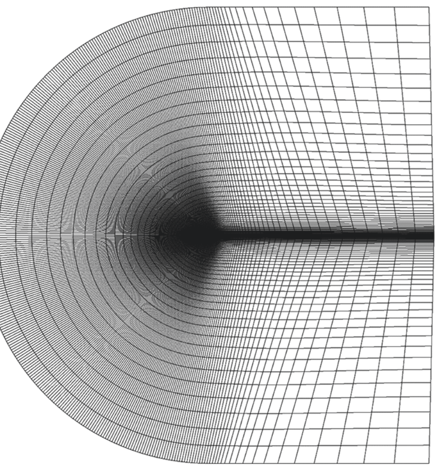
Figure 5: Far-field of initial algebraic Euler and Navier–Stokes meshes.

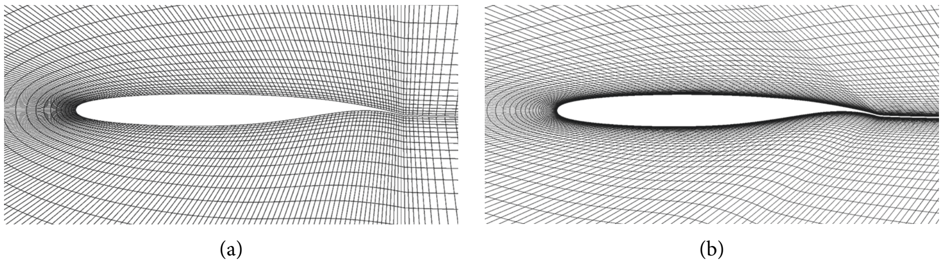

Figure 6: Closeup of initial algebraic meshes. (a) Euler mesh. (b) Navier–Stokes mesh.

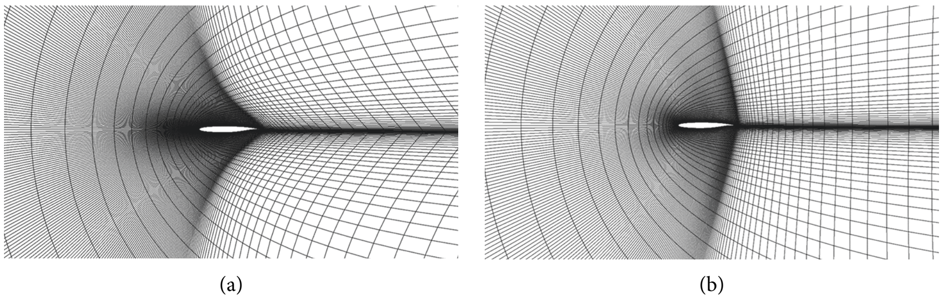

Figure 7: Far-field of Euler meshes optimized without curvature functions. (a) Optimized without orthogonality functions. (b) Optimized with orthogonality functions.

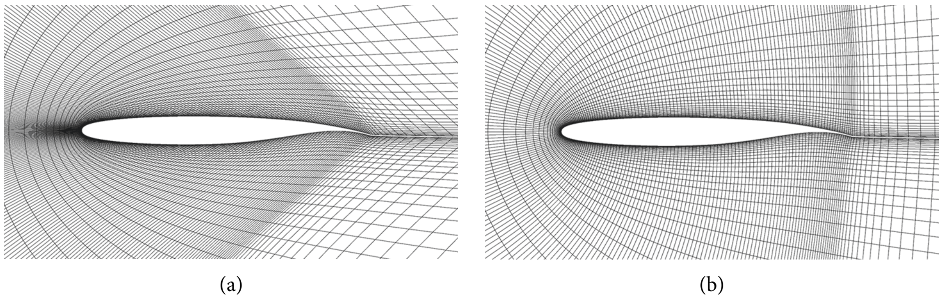

Figure 8: Closeup of Euler meshes optimized without curvature functions. (a) Optimized without orthogonality functions. (b) Optimized with orthogonality functions.

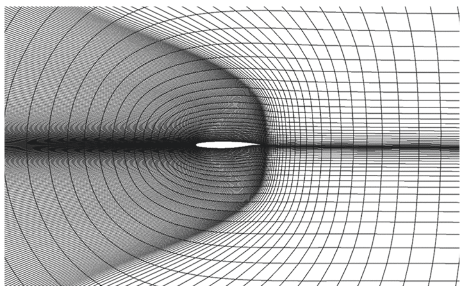

Figure 9: Far-field of Euler mesh optimized with curvature control functions but without orthogonality functions, showing the impact of curvature functions when compared to Figure 7(a).

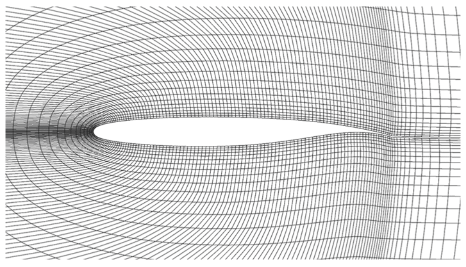

Figure 10: Closeup of Euler mesh optimized with curvature control functions but without orthogonality functions, showing the impact of curvature functions when compared to Figure 8(a).

Figure 9 is that the mesh optimized with curvature functions retains the global curvature of the C-mesh topology, as well as the mesh clustering prescribed via the boundary point distributions. In contrast, the mesh optimized without curvature functions concentrates the mesh points near the airfoil leading edge. This tendency of Laplace-based operators can also be seen when comparing Figures 8(a) to 10. In the latter figure, it can be seen that for the mesh optimized with curvature functions the height of the first layer of cells around the airfoil is consistent with the point distribution

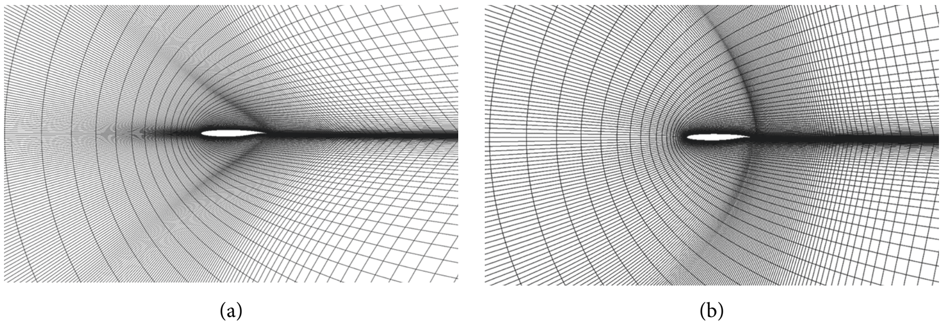

Figure 11: Far-field of the Navier–Stokes meshes optimized without curvature functions. (a) Optimized without orthogonality functions. (b) Optimized with orthogonality functions.

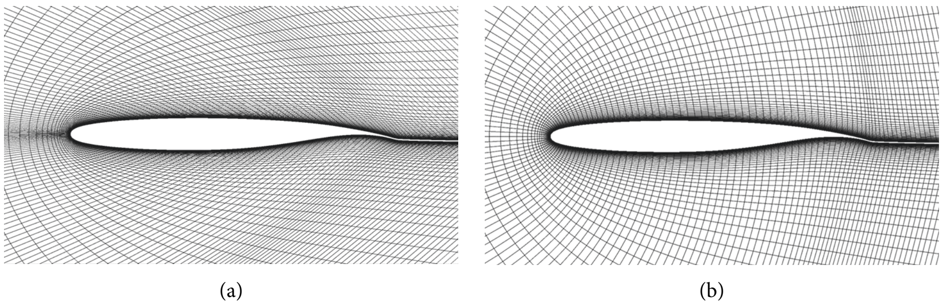

Figure 12: Closeup of the Navier–Stokes meshes optimized without curvature functions. (a) Optimized without orthogonality functions. (b) Optimized with orthogonality functions.

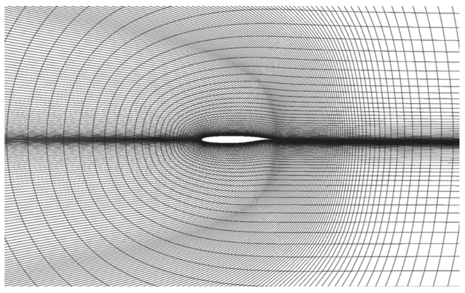

Figure 13: Far-field of the Navier–Stokes mesh optimized with curvature control functions but without orthogonality functions, showing the impact of curvature functions when compared to Figure 11(a).

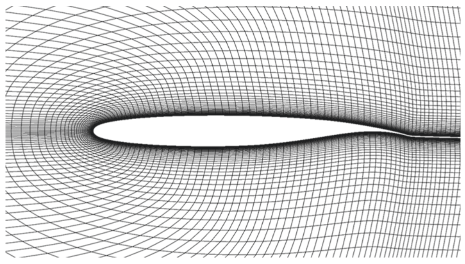

Figure 14: Closeup of the Navier–Stokes mesh optimized with curvature control functions but without orthogonality functions, showing the impact of curvature functions when compared to Figure 12(a).

prescribed on the boundaries (as shown in Figure 6(a)), whereas for the mesh optimized without curvature functions this spacing is significantly altered.

Similar conclusions can be drawn for the optimizations of the Navier–Stokes mesh, as shown in Figures 11 to 14. Figure 11 shows the far field of the Navier-–Stokes meshes optimized without curvature functions, with and without orthogonality functions. Closeup views of these meshes are provided in Figure 12. Figures 13 and 14 show the Navier–Stokes mesh optimized with curvature functions, but without orthogonality functions. These figures can be compared to Figures 11(a) and 12(a), respectively, to see the impact of the curvature functions on the mesh. Again, the mesh optimized with curvature functions is fully orthogonal on the airfoil boundary and retains the global curvature of the C-mesh in both the near field and far field regions.

4.1. Convergence of the Solution Process. The purpose of the sample solutions presented herein is to validate the Newton linearization of the equations and the setup of the system matrix. The results presented do not constitute an exhaustive evaluation of the various solvers and preconditioners that can be used to solve the system, but rather provide a range of results illustrating the efficiency of the solution process for two-dimensional applications. Most available combinations of preconditioners and solvers in the PETSc library were tested for the cases presented here; however, only the more effective combinations are included in the figures. In all cases, all internal settings of the various PETSc preconditioners and solvers were set to their default values, except for the relative tolerance for the stopping criterion for solver subiterations, which was set to 10^{-4}. For more information on the detailed PETSc settings for the various solvers, the reader if referred to the PETSc manual [29].

The convergence curves for the generation of the Euler mesh with curvature control functions are shown in Figures 15 and 16. Figure 15 shows the convergence as a function of iteration, while Figure 16 shows the convergence as a function of time. As shown in Figure 15, no reduction in the maximum residual was achieved in the first 5 iterations, due to the fact that the curvature functions were activated progressively in these early iterations, reaching their full value at iteration 5. In all cases presented here, a gradual activation of the curvature functions was required to smooth out any spurious curvatures in the initial algebraic mesh and ensure convergence. As shown in the convergence plots, once the curvature functions are fully activated, the solution converges rapidly.

As shown in Figure 16, the most efficient solver for the generation of the Euler mesh with curvature functions is the direct "LU" solver. This solver achieved machine accuracy in only 18 iterations, with an elapsed time of 3 seconds, running on a single Intel i7-7700K, 4.2 GHz core.

The convergence curves for the generation of the Navier–Stokes mesh with curvature functions are shown in Figures 17 and 18. Figure 17 shows the convergence as a function of iteration, and Figure 18 shows the convergence as a function of time. Here again, the curvature functions were activated progressively in the first 5 iterations. As shown in Figure 18, the most efficient solver in this case is again the "LU" solver, achieving machine accuracy in 26 iterations, with an elapsed time of 16 seconds. Also interesting in this figure is the performance of the "EISENSTAT" preconditioner with the "PIPELINED GCR" solver; although this combination did not achieve machine accuracy, it achieved what can be considered to be sufficient "engineering" accuracy, and did so in significantly less elapsed time than the other preconditioner-solver combinations.

Based on these results, the most effective overall solver for the solution of the grid generation equations with curvature control functions is the "LU" solver, as it performed well for both the Euler and Navier–Stokes meshes. Hence, using this solver, the meshes were also generated without curvature control functions to illustrate the impact of the latter on convergence rates. Figures 19 and 20 show the comparison of the convergence curves for both the Euler and Navier–Stokes meshes with and without curvature functions. Figure 19 shows the convergence as a function of iteration, and Figure 20 shows the convergence as a function of time. As shown, the inclusion of curvature functions reduces the convergence rate and increases the elapsed time to achieve machine accuracy. However, for practical applications, where a reduction of 10^{-5} in the residual is typically sufficient, the addition of curvature functions does not significantly increase the elapsed time.

It is perhaps not surprising that the "LU" solver outperforms the other solvers in most cases presented here, given that it is a direct solver. However, it is important to note that this will not necessarily be the case for large three-dimensional applications, for which the cost of a direct solver may be too high. What is also observed in the applications presented herein is that the curvature functions

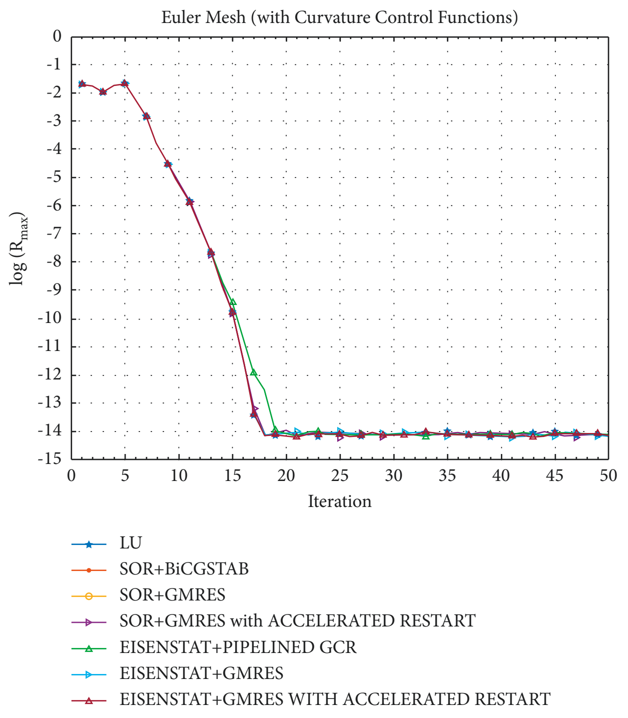

FIGURE 15: Convergence for the Euler mesh vs. iteration, with curvature functions.

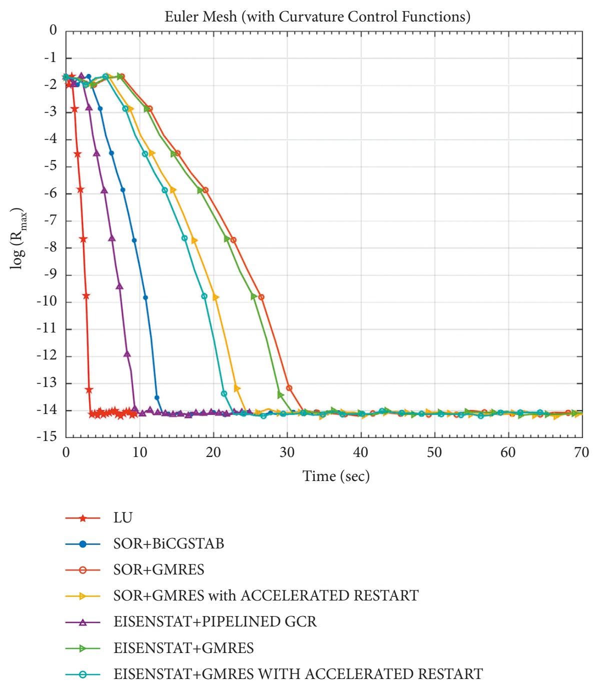

FIGURE 16: Convergence for the Euler mesh vs. time, with curvature functions.

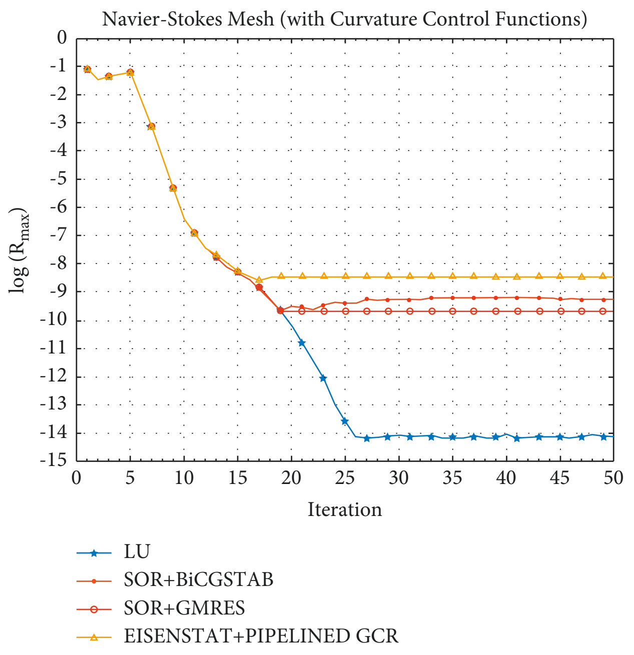

Figure 17: Convergence of the Navier–Stokes mesh vs. iteration, with curvature functions.

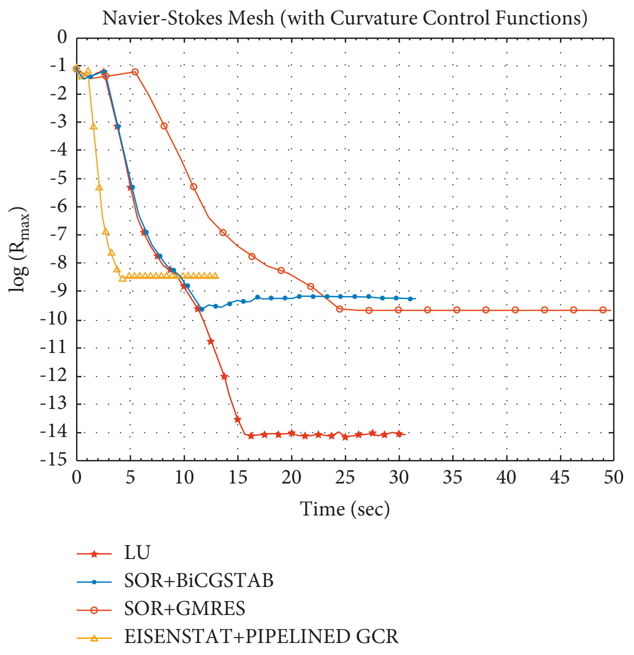

Figure 18: Convergence of the Navier–Stokes mesh vs. time, with curvature functions.

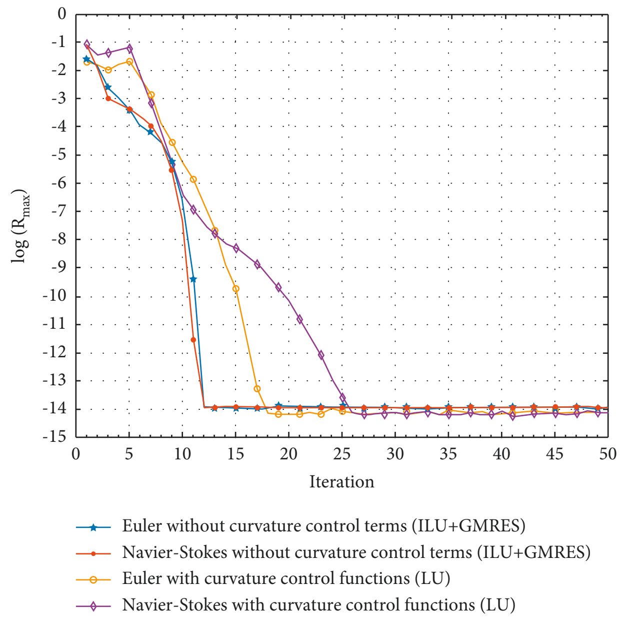

Figure 19: Convergence rates vs. iteration, with and without curvature functions.

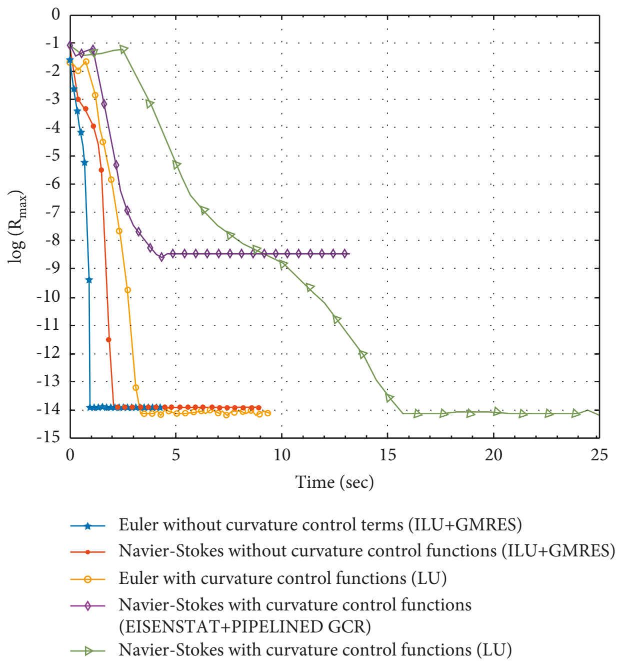

Figure 20: Convergence rates vs. time, with and without curvature functions.

degrade the conditioning of the system matrix, as evidenced by an increase in the condition number. For both the Euler and Navier–Stokes meshes generated without curvature functions, the condition number of the matrices does not exceed $\mathcal{O}(10^4)$ throughout the iterative solution process. In contrast, for Euler meshes generated with curvature control functions, the condition number of the matrices increases to $\mathcal{O}(10^5)$, and for Navier–Stokes meshes the conditions numbers reach $\mathcal{O}(10^6)$ in some cases. This indicates that the very high cell aspect ratios in the boundary layer of the Navier–Stokes meshes have a more detrimental impact on the conditioning of the matrix when curvature functions are included. This is likely the reason that the "SOR + BiCGSTAB", "SOR + GMRES", and "EISENSTAT + PIPELINED GCR" combinations did not achieve machine accuracy for the Navier–Stokes meshes with curvature functions, as shown in Figures 17, 18 and 20.

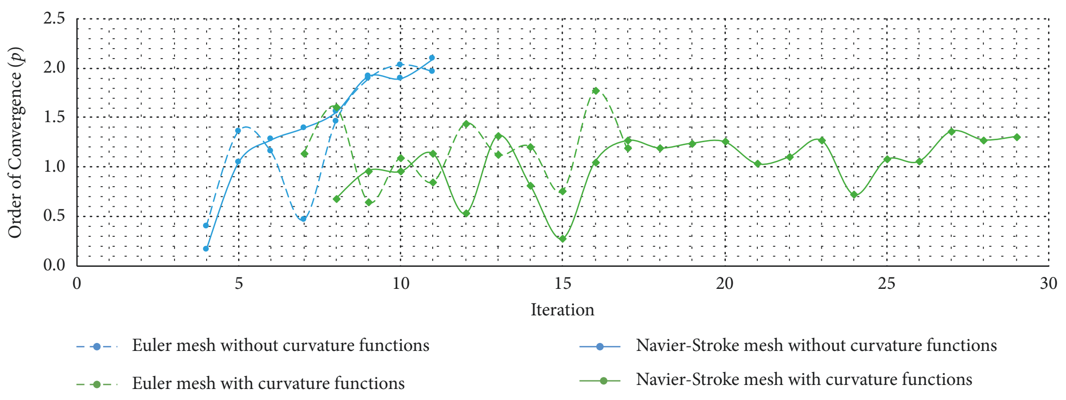

Figure 21: Order of convergence for the Euler and Navier–Stokes meshes, with and without curvature functions.

Figure 21 shows the order of convergence achieved for the Euler and Navier–Stokes meshes, with and without curvatures functions. The order of convergence, p, is calculated as follows:

$$p = \lim_{n\longrightarrow\infty} \frac{\log\left|R^{n+1}_{\max}/R^{n}_{\max}\right|}{\log\left|R^{n}_{\max}/R^{n-1}_{\max}\right|}, \tag{95}$$

where $R^{n}_{\max}$ is the maximum residual anywhere in the mesh at iteration n.

As shown, the numerical solutions without curvature functions achieve quadratic $(p = 2)$ convergence for both the Euler and Navier–Stokes meshes. This indicates that the very high cell aspect-ratios of $\mathcal{O}(10^5)$ in the boundary layer of the Navier–Stokes mesh do not significantly impact the convergence of the solution. This can also be observed in Figure 19, where the converge rates for the Euler and Navier–Stokes meshes generated without curvature functions are virtually identical. In contrast, the solutions for the meshes generated with curvature functions achieve orders of convergence of only 1.3 to 1.8, with the Euler mesh peaking at 1.77 and the Navier–Stokes mesh peaking at 1.36.

It should be noted that the convergence curves shown in Figures 15 to 21 pertain to meshes generated without orthogonality functions. The reason for this is that at the time of this writing, the implementation of orthogonality functions was not fully integrated with the newly implemented solution process, and thus the convergence rates for optimizations with orthogonality functions could not be compared to those obtained without orthogonality functions. Furthermore, since the optimizations with curvature functions did not require orthogonality functions, the latter were not included in the comparisons of convergence rates in order to isolate the impact of the curvature functions on convergence. Thus, the differences in convergence rates shown in Figures 19 and 20 are due solely to the addition of curvature functions. The proper integration of the orthogonality functions with the newly implemented PETSc-based solution process will be part of future work.

5. Conclusions and Recommendations

In the work presented herein, the grid generation equations with and without curvature control functions were discretized using second-order finite differences and linearized using Newton's method. To solve the resulting linear system, a number of solvers and preconditioners available in the PETSc library were implemented and evaluated. The efficiency of the solution process was demonstrated with applications to two-dimensional airfoil meshes, and the impact of the curvature functions on the overall convergence was investigated and quantified.

In future work, an investigation will be undertaken to shed light on the underlying characteristics impacting the conditioning of the system to further enhance the solution process. An efficient methodology will also be developed to properly integrate the orthogonality functions with the newly-implemented solution process.

Ultimately the methodology presented in this work must be generalized to three-dimensional meshes. The challenges arising from the latter, in particular the substantial computing resources required to generate large meshes, will need to be addressed. To this end, a full parallelization of the solution process must also be implemented, and other solution techniques not included herein, such as multigrid methods, will need to be evaluated.

Nomenclature

C^i:	Grid control term for curvilinear coordinates $(i = 1, 2, 3)$
$\overline{C}^i$:	Grid control term for s^i coordinates $(i = 1, 2, 3)$
g^{ij}:	Contravariant metric tensor components for ξ^i coordinates
g_{ij}:	Covariant metric tensor components for ξ^i coordinates
g:	Determinant of the covariant metric tensor for ξ^i coordinates
$\overline{g}^{ij}$:	Contravariant metric tensor components for s^i coordinates

$\overline{g}_{ij}$:	Covariant metric tensor components for s^i coordinates
$\overline{g}$:	Determinant of the covariant metric tensor for s^i coordinates
$\mathbf{G}$:	Function defined in equation (40a), where $\mathbf{G} = [G_x\, G_y\, G_z]^T$
i, j, k:	Coordinate indices
$i_{\max}, j_{\max}, k_{\max}$:	Maximum values of coordinate indices
$\widehat{J}$:	Determinant of Jacobian matrix of s^i-to-ξ^i coordinate transformation
$K_1^{(i)}, K_2^{(i)}$:	Local principal curvatures of $\xi^i = constant$ surface $(i = 1, 2, 3)$
$\overline{K}_1^{(i)}, \overline{K}_2^{(i)}$:	Local principal curvatures of $s^i = constant$ surface $(i = 1, 2, 3)$
P^1, P^2:	Functions defined in equations (26a) and (26b), respectively
R^p:	General curvature control functions for a mesh with non-uniform spacing $(p = 1, 2, 3)$
$\mathbf{R}$:	Function defined in equation (31), where $\mathbf{R} = [X\, Y\, Z]^T$
$\mathbf{r}$:	Vector of physical mesh coordinates, $\mathbf{r} = [x\, y\, z]^T = [x^1\, x^2\, x^3]^T$
s^i:	Coordinate functions controlling the mesh spacing distribution $(i = 1, 2, 3)$
x^i:	Physical mesh coordinates $(i = 1, 2, 3)$
X, Y, Z:	Components of $\mathbf{R}$ function
x, y, z:	Physical mesh coordinates
Γ^i_{jk}:	Christoffel symbols of the second kind for ξ^i coordinates
$\overline{\Gamma}^i_{jk}$:	Christoffel symbols of the second kind for s^i coordinates
ξ^i:	Curvilinear coordinates $(i = 1, 2, 3)$
A^1_{lmn}, B^1_{lk}:	Functions defined in equations (71) and (72), respectively
A^2_{lmn}, B^2_{lk}:	Functions defined in equations (75) and (76), respectively
$C^k_{x^{ij}}, C^k_{y^{ij}}$:	Coefficients defined in equations (84) and (85), respectively
$J^k_{x_{ij}}, J^k_{y_{ij}}$:	Coefficients defined in equations (90) and (91), respectively.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors acknowledge the support of Bombardier Aviation, Montreal, Canada, for funding the original development of the grid generation equations used herein.

References

[1] A. M. Winslow, "Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh," *Journal of Computational Physics*, vol. 1, no. 2, pp. 149–172, 1966.

[2] W. D. Barfield, "An optimal mesh generator for Lagrangian hydrodynamic calculations in two space dimensions," *Journal of Computational Physics*, vol. 6, no. 3, pp. 417–429, 1970.

[3] W.-H. Chu, "Development of a general finite difference approximation for a general domain Part I: machine transformation," *Journal of Computational Physics*, vol. 8, no. 3, pp. 392–408, 1971.

[4] S. K. Godunov and G. P. Prokopov, "The use of moving meshes in gas-dynamical computations," *USSR Computational Mathematics and Mathematical Physics*, vol. 12, no. 2, pp. 182–195, 1972.

[5] A. A. Amsden and C. W. Hirt, "A simple scheme for generating general curvilinear grids," *Journal of Computational Physics*, vol. 11, no. 3, pp. 348–359, 1973.

[6] J. F. Thompson, F. C. Thames, and C. W. Mastin, "Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies," *Journal of Computational Physics*, vol. 15, no. 3, pp. 299–319, 1974.

[7] J. F. Thompson, F. C. Thames, and C. Wayne Mastin, "Tomcat - a code for numerical generation of boundary-fitted curvilinear coordinate systems on fields containing any number of arbitrary two-dimensional bodies," *Journal of Computational Physics*, vol. 24, no. 3, pp. 274–302, 1977.

[8] J. F. Thompson, *Numerical Grid Generation*, North Holland, 1982.

[9] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation, Foundations and Applications*, ACM, North Holland, New York, 1985.

[10] J. F. Thompson, B. K. Soni, and N. P. Weatherill, Eds., *Handbook of Grid Generation*, CRC Press, Florida, United States, 1998.

[11] P. Piperni, "Differential operator for the modeling of curvature in structured grid generation," *AIAA Journal*, vol. 56, no. 11, pp. 4453–4462, 2018.

[12] P. Piperni, "On the modeling of curvature in general curviliner coordinates," in *Proceedings of the 2018 Fluid Dynamics Conference*, June 2018.

[13] M. E. Davis and J. A. McCammon, "Solving the finite difference linearized Poisson-Boltzmann equation: a comparison of relaxation and conjugate gradient methods," *Journal of Computational Chemistry*, vol. 10, no. 3, pp. 386–391, 1989.

[14] A. Nicholls and B. Honig, "A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation," *Journal of Computational Chemistry*, vol. 12, no. 4, pp. 435–445, 1991.

[15] Q. Cai, M.-J. Hsieh, J. Wang, and R. Luo, "Performance of nonlinear finite-difference Poisson–Boltzmann solvers," *Journal of Chemical Theory and Computation*, vol. 6, no. 1, pp. 203–211, 2010.

[16] J. Wang and R. Luo, "Assessment of linear finite-difference Poissonâ "Boltzmann solvers"," *Journal of Computational Chemistry*, vol. 31, no. 8, p. NA, 2010.

[17] D. A. Knoll, W. J. Rider, and G. L. Olson, "An efficient nonlinear solution method for non-equilibrium radiation diffusion," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 63, no. 1, pp. 15–29, 1999.

[18] W. J. Rider, D. A. Knoll, and G. L. Olson, "A multigrid Newton-Krylov method for multimaterial equilibrium radiation diffusion," *Journal of Computational Physics*, vol. 152, no. 1, pp. 164–191, 1999.

[19] M. Berndt, J. David Moulton, and G. Hansen, "Efficient nonlinear solvers for Laplace-Beltrami smoothing of three-dimensional unstructured grids," *Computers & Mathematics with Applications*, vol. 55, no. 12, pp. 2791–2806, 2008.

[20] M. M. Gupta, J. Kouatchou, and J. Zhang, "Comparison of second- and fourth-order discretizations for multigrid Poisson solvers," *Journal of Computational Physics*, vol. 132, no. 2, pp. 226–232, 1997.

[21] J. Zhang, "Fast and high accuracy multigrid solution of the three dimensional Poisson equation," *Journal of Computational Physics*, vol. 143, no. 2, pp. 449–461, 1998.

[22] M. Ilić, I. W. Turner, and V. Anh, "A numerical solution using an adaptively preconditioned lanczos method for a class of linear systems related with the fractional Poisson equation," *Journal of Applied Mathematics and Stochastic Analysis*, vol. 2008, pp. 1–26, 2008.

[23] B. D. Froese and A. M. Oberman, "Fast finite difference solvers for singular solutions of the elliptic Monge-Ampère equation," *Journal of Computational Physics*, vol. 230, no. 3, pp. 818–834, 2011.

[24] P. Piperni, *A Fundamental Approach to the Problem of Domain Decomposition in Structured Grid Generation*, Ph.D. thesis, École Polytechnique de Montréal, Montreal, Canada, September 2003.

[25] S. P. Spekreijse, "Elliptic grid generation based on Laplace equations and algebraic transformations," *Journal of Computational Physics*, vol. 118, no. 1, pp. 38–61, 1995.

[26] V. D. Liseikin, *Grid Generation Methods*, Springer International Publishing, Berlin, Germany, 2017.

[27] Z. U. A. Warsi, "Tensors and differential geometry applied to analytic and numerical coordinate generation," Tech. Rep. Report MSSU-EIRS-ASE-81-1, Mississippi State University, Starkville, Mississippi, 1981.

[28] 2021, http://www.mcs.anl.gov/petsc.

[29] S. Balay, S. Abhyankar, M. Adams et al., "PETSc users manual," 2021, https://publications.anl.gov/anlpubs/2016/05/127241.pdf.