Mesh generation using differential methods

Jeff Shragge

ABSTRACT

This paper examines a differential gridding method for generating computational meshes appropriate for solving partial differential equations. Differential methods pose mesh generation as an elliptical boundary value problem within a framework of differential geometry. Generalized Laplacian operators are used to propagate the known coordinate values on the boundary points into the interior in a smooth manner. The methodology allows for the specification of monitor functions that provide mesh regularization and prevent grid clustering. Examples are provided for two seismic imaging applications: wave-equation Green's function generation and wave-equation migration from topography. In both cases, the resulting regularized meshes have minimal convexity and are conformal to the the prescribed boundaries.

INTRODUCTION

Geophysical methodology often requires computing numerical solutions of partial differential equations (PDEs). In many cases, computational efficiency and/or accuracy can be enhanced by posing the physical equations on generalized coordinate meshes rather than Cartesian grids. By following this approach, though, mesh generation becomes a necessary solution process step. An important consideration is ensuring that the developed mesh has no attributes that would cause numerical instability (e.g. grid clustering). This is not a straightforward task because mesh generation is controlled by non-linear geometric coupling. Hence, localized mesh regularization can be difficult to implement and developing new gridding techniques with built-in regularization remains an open and important topic.

Meshing techniques are less advanced in the geophysical community relative to other scientific disciplines (e.g. computer graphics and fluid flow). However, a number of different grid generation approaches have been reported. Alkhalifah (2003) performs anisotropic velocity analysis on structured, non-orthogonal τ-coordinate meshes based on two-way travel times. Sava and Fomel (2005) developed Riemannian wavefield extrapolation that generalizes one-way wave-propagation to structured semi-orthogonal ray-based meshes. Shragge and Sava (2005) examine wave-equation migration from topography in acquisition coordinates using structured locally orthogonal meshes. Rüger and Hale (2006) use a non-structured gridding technique based on tesselation to break subsurface velocity models into logical units.

An important set of structured meshing techniques used in other scientific fields are based on differential methods (Liseikin, 2004). These approaches pose mesh generation as solving an elliptic boundary value problem (BVP) within a framework of differential geometry. Numerical solution of these generalized Laplacian systems is facilitated by recasting the elliptic problem into a set of parabolic equations for which well-developed and efficient solution techniques exist. The steady-state solutions of the parabolic equations are structured non-orthogonal meshes. Importantly, the gridding equations can be manipulated to ensure that meshes exhibit appropriate attributes for numerical solution of PDEs: i) piece-wise smoothness of interior cells; ii) non-propagation of boundary singularities; and iii) non-overlapping neighboring cells. In addition, problematic grid clustering can be controlled by introducing monitor functions that force the mesh to conform locally to minimum geometric standards.

In this paper, I generate structured non-orthogonal meshes appropriate for generalized Riemannian wavefield extrapolation (Shragge, 2006) using a differential method advocated by Liseikin (2004). The goal of this paper is not to simply re-develop Liseikin's method. Rather, it is to demonstrate its relevance to geophysical meshing problems by summarizing the general motivation behind the method and highlighting the essential theory and advocated numerical solution. I begin the paper by reviewing why differential methods form an elliptical BVP and describing how to pose the gridding problem within a N-D differential geometric framework. I then present gridding equations for 2-D geometry and provide meshing examples for two seismic imaging applications: wave-equation generation of Green's function estimates and wave-equation migration from topography. The attached appendix provides the numerical details required to implement the differential meshing technique.

THEORETICAL OVERVIEW

The goal of most grid generation methods is to find the transformation from a regular computational mesh defined by $\{\xi^i\}$ on a domain Ξ^n to an irregular grid defined by $\{x^k\}$ on domain X^{n+l}. Usually, the only *a priori* information is the location of the mesh boundary points (i.e. $\phi^i(x^j)$ on $\partial\Xi^N$). Hence, grid generation is a two-fold task: propagate the boundary values into the interior in a physically consistent manner, and generate meshes with appropriate attributes (e.g. well-formed, smooth and non-singular). Thus, two important questions are by what physical principles are the boundary values propagated into the interior? And by what manner is the mesh regularized to ensure that it has acceptable attributes?

An answer to the first query is found by recasting grid generation as a Dirchelet BVP: given the boundary values of the mesh, solve an elliptic Laplace's equation on the mesh's interior. Mathematically, this requires solving the following system of equations,

$$D^{\xi}[s^j] = 0, \qquad \Gamma[\xi^i] = \xi^i\big|_{\partial S} = \phi^i[s^j], \qquad i,j = 1,n \tag{1}$$

where $D^{\xi}[v]$ is a generalized Laplacian operator acting on field v in coordinates $\{\xi^i\}$, and Γ is a projection operator that maps boundary values $\phi^i[s^j]$ of intermediate domain S^n onto the boundaries of initial domain Ξ^n. More specifically, a BVP is formed for each coordinate component $\{s^j\}$, which requires finding N solutions. Generally, this is an iterative process that continues until the coordinate fields in $\{s^j\}$ converge to those of $\{x^j\}$ to within some level of tolerance.

Division of the coordinate fields into independent BVPs, though, does not permit mesh regularization because coordinate fields are subject to geometric coupling. Because we wish to generate meshes over generalized spaces, we must represent this coupling through differential geometry. Importantly, this provides us with a powerful set of tools for mesh generation because it: i) specifies a metric (literally!) for evaluating mesh characteristics (e.g. extent and orientation of grid clustering); and ii) permits the introduction of monitor metrics that enable local mesh regularization. These two topics are discussed in the following two sections.

Differential N-D Mesh Generation

Generalized Laplacian systems can be solved by many different ways; however, numerical solution often is facilitated through intermediate mappings to meshes exhibiting many attributes of the final grid. This consists of a composite of a transformation - $s^i(\xi^j)$ - from a Cartesian $\{\xi^j\}$ to an intermediate basis $\{s^i\}$, and a transformation - $x^k(s^j)$ - from $\{s^i\}$ to the final coordinate mesh $\{x^k\}$. Notationally, the composite mapping transforms for the N-D problem are,

$$\begin{aligned} s^j(\xi^i) \,:\, \Xi^n \to S^n, \quad \xi^i = \{\xi^1,\xi^2,...,\xi^n\}, \quad s^j = \{s^1,s^2,...,s^n\}, \\ x^k(s^j) \,:\, S^n \to X^{n+l}, \quad s^j = \{s^1,s^2,...,s^n\}, \quad x^k = \{x^1,x^2,...,x^{n+l}\}, \end{aligned} \tag{2}$$

Note that coordinate system $\{x^k\}$ may be of a greater dimension than $\{s^j\}$, which allows for composite mapping operations of $x^k\left[s^j\left(\xi^i\right)\right] \,:\, \Xi^n \to X^{n+k}$ that project a 2-D surface into 3-D space (see figure 1 for an example).

Coordinate system transformations - $s^j(\xi^i)$ and $x^k(s^j)$ - are described in differential geometry through metric tensor, g_{ij}, which relates the geometry of a coordinate system $\{s^j\}$ to that of $\{\xi^i\}$ (Guggenheimer, 1977). The metric tensor is symmetric (i.e. $g_{ij} = g_{ji}$) and has elements given by,

$$g^{\xi}_{ij} = \frac{\partial s^k}{\partial \xi_i}\frac{\partial s^k}{\partial \xi_j} \quad \text{and} \quad g^{s}_{ij} = \frac{\partial x^k}{\partial s_i}\frac{\partial x^k}{\partial s_j}, \tag{3}$$

where the metric tensor superscript specifies the coordinate system in which the operator is defined. (Note that summation notation - $g_{ii} = g_{11} + g_{22} + g_{33}$ - is implicit for any repeated indicies found in the paper.) The associated metric tensor g^{ij} is related to the metric tensor through $g^{ij} = g_{ij}/|g^s|$ where g^s is the metric tensor determinant. Through use of this differential geometric framework, the governing set of differential gridding equations (Liseikin, 2004) become,

$$D^{\xi}[s^j] = \frac{1}{\sqrt{g_s}}\frac{\partial}{\partial \xi_i}\left(\sqrt{g_s}\, g_s^{jk}\frac{\partial s^j}{\partial \xi_k}\right) = 0 \quad i,j,k = 1,n, \tag{4}$$

$$\Gamma\left[\xi^i\right] \;\equiv \xi^i\big|_{\partial S^n} = \phi^i\left[s^j\right], \quad i,j = 1,n. \tag{5}$$

Equations 4 represent the N generalized Laplace's equations acting on coordinate fields $\{s^j\}$, and equations 5 map the boundary values of each coordinate field $\phi^i\left[s^j\right]$ to the boundary of domain Ξ^n. As posed, equations 4 and 5 provide no guarantee that generated grids will exhibit appropriate characteristics because no mesh regularization has yet been enforced.

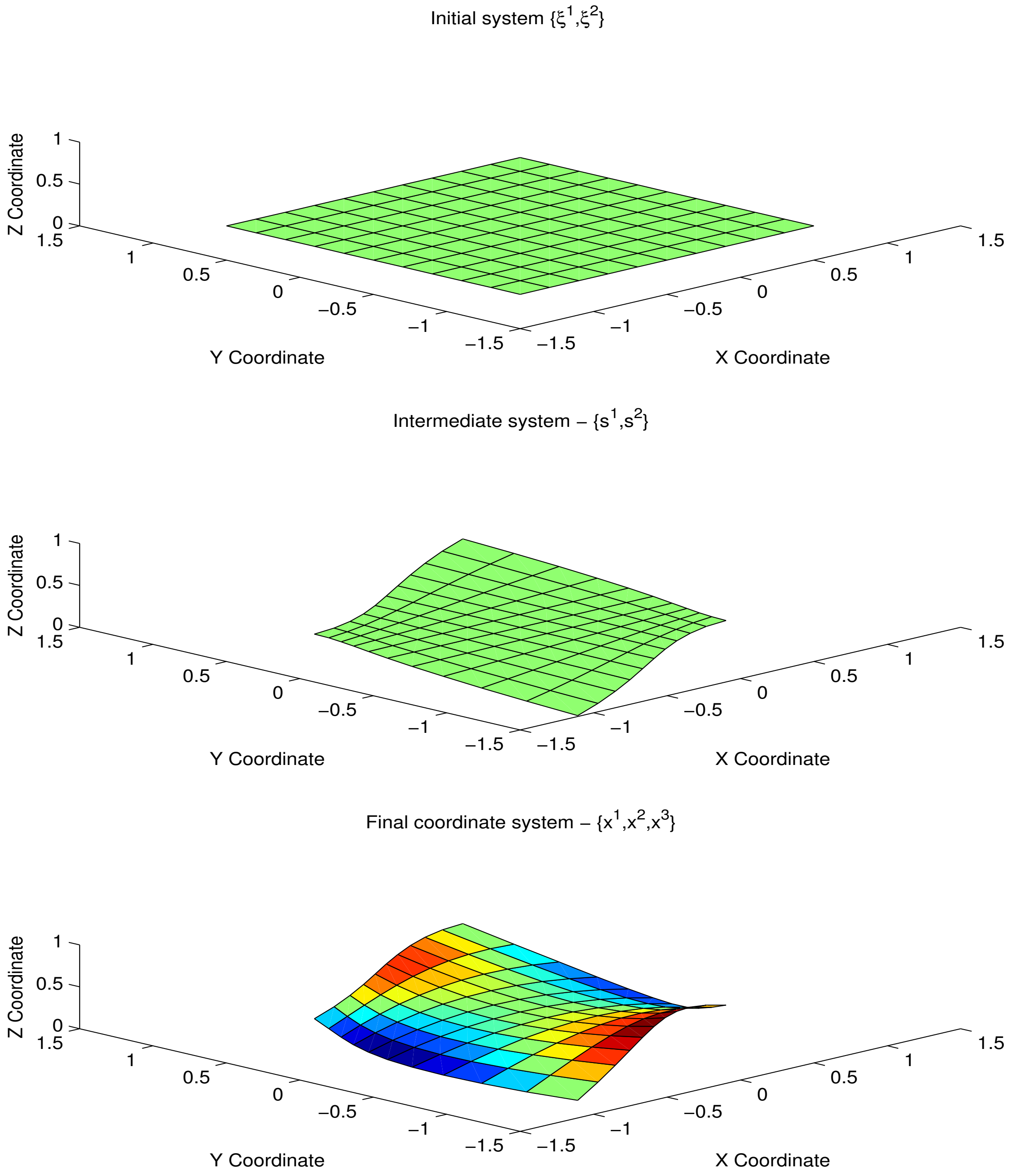

Figure 1: Meshing example for mapping a 2-D Cartesian domain to a surface in a 3-D volume. Top panel: Regular Cartesian mesh $\{\xi^1,\xi^2\}$; Middle panel: Intermediate transformation domain $\{s^1,s^2\}$; and Bottom panel: Surface in middle panel projected onto 3-D surface $\{x^1,x^2,x^3\}$ where increasing grey scale intensity represents increasing height. jeff2-Example [NR]

Regularization through Monitor Functions

Monitor functions are a useful tool for regularizing meshes because they can establish a metric of minimal quality that prevents problematic grid clustering. (A simple scalar analog is preventing division by zero by adding a small number to the denominator.) Monitor functions can be introduced into equations 4 and 5 by adding additional components to the metric tensor,

$$g_{ij}^{s} = z(\mathbf{s})\, g_{ij}^{xs} + v^k(\mathbf{s})\, f^k(\mathbf{s}) \quad i,j = 1,n, \quad k = 1,l, \tag{6}$$

where g_{ij}^{s} is the regularized metric tensor, g_{ij}^{xs} is the unregularized metric tensor calculated by equations 3, $f^k(\mathbf{s})$ are functions of coordinate $\{s^i\}$ that provide metric stabilization, and $z(\mathbf{s})$ and $v^k(\mathbf{s})$ are weighting functions. Hence, where g_{ij}^{xs} tends to zero, functions $f^k(\mathbf{s})$ are set to non-zero values. Note that the functions and their corresponding weights are specified on a point-by-point basis allowing for localized mesh regularization, and that the functions should have zero values and derivatives on the boundary so as to not regularize the boundary geometry.

Incorporating monitor functions into the Laplacian equation framework requires altering equations 4. Accordingly, the N-D differential method gridding equations incorporating monitor functions are given by,

$$D^{\xi}[s^j] = -D^{\xi}[f^k]\frac{\partial f^k}{\partial s^j}, \quad i,j = 1,n, \quad k = 1,l, \tag{7}$$

$$\Gamma\left[\xi^i\right] \equiv \xi^i\big|_{\partial S^n} = \phi^i\,[\mathbf{s}], \qquad i = 1,n, \tag{8}$$

where D^{ξ} is specified in equation 4 above. Liseikin (2004) provides theoretical justification of a number of different approaches to control grid clustering through the manipulation of monitor functions. In this paper, I use a fairly basic approach where the monitoring function is specified by a scaled spatially variant metric determinant.

DIFFERENTIAL 2-D MESH GENERATION

Generating a 2-D coordinate system through differential methods requires solving for coordinates $\{x^1, x^2\}$ within domain X^2. Incorporating l monitor functions for grid regularization expands the dimensionality of the mapping to,

$$\mathbf{x}(\mathbf{s}) \,:\, S^2 \rightarrow X^{2+l}, \qquad \mathbf{x}(\mathbf{s}) = \{s^1, s^2, f^1(\mathbf{s}), ..., f^l(\mathbf{s})\}. \tag{9}$$

Coordinate system $\{s^j\}$ is related to an underlying Cartesian grid, which is chosen to be a unit square defined by $\Xi^2 = \{0 \le \xi^1, \xi^2 \le 1\}$. Transformation $s^j(\xi^i)$ is assumed to be piece-wise smooth and known on the boundary of Ξ^2 such that: $\phi(\xi) \,:\, \partial\Xi^2 \rightarrow \partial S^2, \quad \phi = (\phi^1, \phi^2)$. Within this framework, the 2-D gridding equations become,

$$D^{\xi}[s^j] = -D^{\xi}[f^k]\frac{\partial f^k}{\partial s^j}, \quad \mathbf{s}(\xi)|_{\partial\Xi^2} = \phi(\xi), \quad j = 1,2, \quad k = 1,l, \tag{10}$$

where Laplacian operator $D^{\xi}[\cdot]$ and metric tensor g_{ij} are written explicitly (Liseikin, 2004),

$$D^{\xi}[v] = g^{\xi}_{22}\frac{\partial^2 v}{\partial \xi^1 \partial \xi^1} - 2\, g^{\xi}_{12}\frac{\partial^2 v}{\partial \xi^1}\partial \xi^2 + g^{\xi}_{11}\frac{\partial^2 v}{\partial \xi^2 \partial \xi^2}, \tag{11}$$

$$g^{\xi}_{ij} = \frac{\partial s^k}{\partial \xi^i}\frac{\partial s^k}{\partial \xi^j} + \frac{\partial f^m[\mathbf{s}(\xi)]}{\partial \xi^i}\frac{\partial f^m[\mathbf{s}(\xi)]}{\partial \xi^j}, \quad i,j,k = 1,2, \quad m = 1,l. \tag{12}$$

One convenient way to solve the set of elliptical equations 10 is by transforming them to a set of parabolic equations (i.e. include time-dependence) that have a common steady-state solution. Thus, equations 10 are reformulated to include time-dependence - $\{s^i(\xi^1,\xi^2,t)\}$ - leading to the six governing equations,

$$\frac{\partial s^1}{\partial t} = D[s^1] + D[f^k]\frac{\partial f^k}{\partial s^1}, \quad k = 1,l \tag{13}$$

$$\frac{\partial s^2}{\partial t} = D[s^2] + D[f^k]\frac{\partial f^k}{\partial s^2}, \quad k = 1,l \tag{14}$$

$$s^1(\xi^1,\xi^2,t) = \phi^1(\xi^1,\xi^2), \quad t \geq 0 \tag{15}$$

$$s^2(\xi^1,\xi^2,t) = \phi^2(\xi^1,\xi^2), \quad t \geq 0 \tag{16}$$

$$s^1(\xi^1,\xi^2,0) = s^1_0(\xi^1,\xi^2), \quad t = 0 \tag{17}$$

$$s^2(\xi^1,\xi^2,0) = s^2_0(\xi^1,\xi^2), \quad t = 0 \tag{18}$$

Solutions $s^1(\xi^1,\xi^2,t)$ and $s^2(\xi^1,\xi^2,t)$ satisfying equations 13-18 will converge to the solutions $\{x^1,x^2\}$ of equations 10 as $t \rightarrow \infty$. Hence, the answer to within tolerance factor ϵ occurs at some T_n. Details of an iterative scheme and an algorithm for computation to solve equations 13 are provided in Appendix A.

NUMERICAL EXAMPLES

Wave-equation generated Green's Functions

Generating high-quality source Green's function estimates is an important component of seismic imaging. Because wave-equation methods naturally handle wavefield triplication, Green's functions developed using corresponding operators should normally be superior to those generated by other methods. However, implementing the approximations required to handle propagation through laterally variant velocity profiles often can lead to inferior Green's function estimates. These errors become increasingly more apparent, both kinematically and dynamically, as wavefield propagation directions become increasingly oblique to the extrapolation axis.

One solution is to pose wavefield extrapolation in a ray-based coordinate system defined by an extrapolation axis oriented along the axis of increasing travel time and additional coordinates represented by shooting angles (Sava and Fomel, 2005). However, grids thus specified exhibit attributes that depend intrinsically on the chosen ray-tracing method. For example, ray-coordinate systems generated by Huygen's ray-tracing (Sava and Fomel, 2001) may triplicate and cause numerical instability during wavefield extrapolation. Hence, care must be taken to ensure that ray-coordinate systems have the appropriate attributes.

One method for calculating singular-valued travel times is with a fast-marching Eikonal equation solver, which provides a travel-time map to each subsurface model location for a given shot point. A travel-time map example is shown in the upper left panel of figure 2 for a velocity slice of the SEG-EAGE salt data set. A ray-based coordinate system can be formed by choosing two isochrons that represent the initial and maximal extrapolation times. The coordinate system is fully defined by connecting the two isochrons with the extremal rays. Blending functions can then be used to specify an intermediate geometry $\{s^1, s^2\}$. (see upper right panel of figure 2).

Coordinate systems generated with this approach, though, are not guaranteed to be smooth and generally will require mesh regularization. The bottom right panel shows the output of the differential grid generation algorithm $\{x^1, x^2\}$ after 20 smoothing iterations. Note that kinks visible in the upper right panel have disappeared leaving a significantly smoother mesh. A qualitatively test of coordinate system smoothness is to examine the smoothness of the underlying velocity model in the transform domain. The salt body example (lower left panel) indicates that the velocity model is fairly smooth and should not create significant problems for generalized coordinate wavefield extrapolation.

Wave-equation Migration from Topography

Seismic data acquired on topography usually require significant preprocessing before any imaging technique can be applied successfully. Performing wave-equation migration, though, usually requires that data exist on regularly sampled meshes before a wavefield extrapolation procedure can begin. Usually, this involves a data regularization step implemented as either pre-migration datuming or through a wavefield injection plus interpolation migration strategy.

An alternative to these standard approaches is discussed in Shragge and Sava (2005), who pose seismic imaging directly in acquisition coordinates and use Riemannian wavefield extrapolation to propagate wavefields. Initially, a conformal mapping approach was used to generate structured, locally orthogonal coordinate meshes (see top panel of figure 3). However, the ensuing grid clustering and rarefaction demanded by orthogonality led to significant spatial variance in metric tensor coefficients. Importantly, this variance caused artifacts in the resulting image, which remains the main drawback of this migration approach.

The extension of RWE to non-orthogonal coordinate systems (Shragge, 2006) allows for greater flexibility in coordinate system design. The middle panel in figure 3 represents a blended coordinate system $\{s^1, s^2\}$ that forms the input to the differential mesh algorithm. Lines predominantly in the horizontal direction mimic topography in the near-surface and slowly heal to form an evenly sampled wavefield at depth. The bottom panel shows the coordinate system output $\{x^1, x^2\}$ from the gridding algorithm after 15 iterations. Grid irregularities now heal more rapidly and the mesh becomes very regular after a few extrapolation steps into the subsurface.

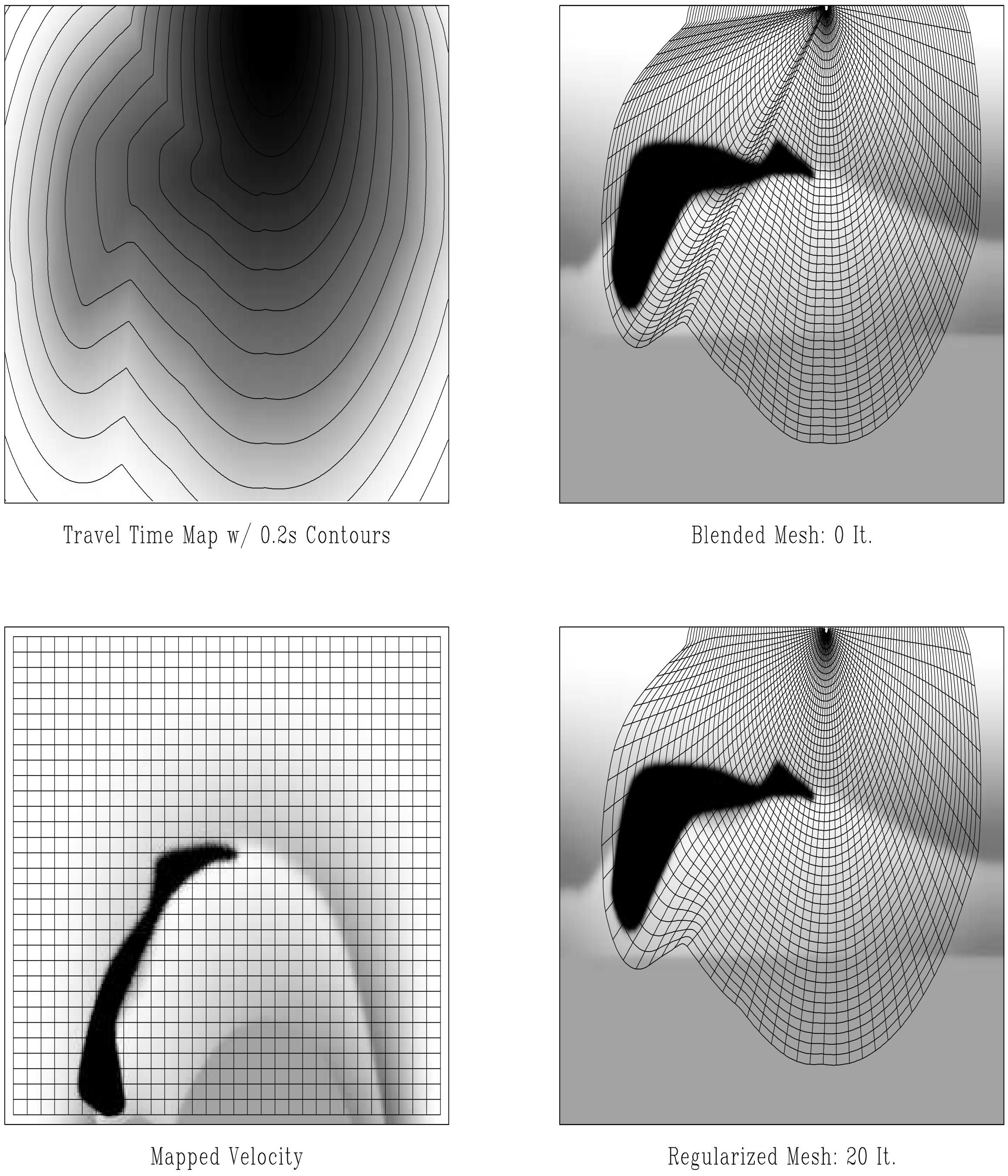

Figure 2: Wave-equation generated Green's functions example. Top left: Travel-time map generated by fast marching eikonal (FMEikonal) equation solver for a slice of the SEG-EAGE salt model overlain by 0.2 s time contours. Top right: A blended coordinate system developed using time contours as the interior and exterior surfaces. This allows for the the coordinate system to be fairly conformal with the orientation of the propagating wavefield. Bottom right: Coordinate mesh output from the differential mesh algorithm after 20 iterations. Note that the kinks in the model have been significantly reduced. Bottom left: Underlying velocity model mapped into the ray-coordinate shown in bottom right. $\boxed{\text{jeff2-Shotpoint}}$ [ER]

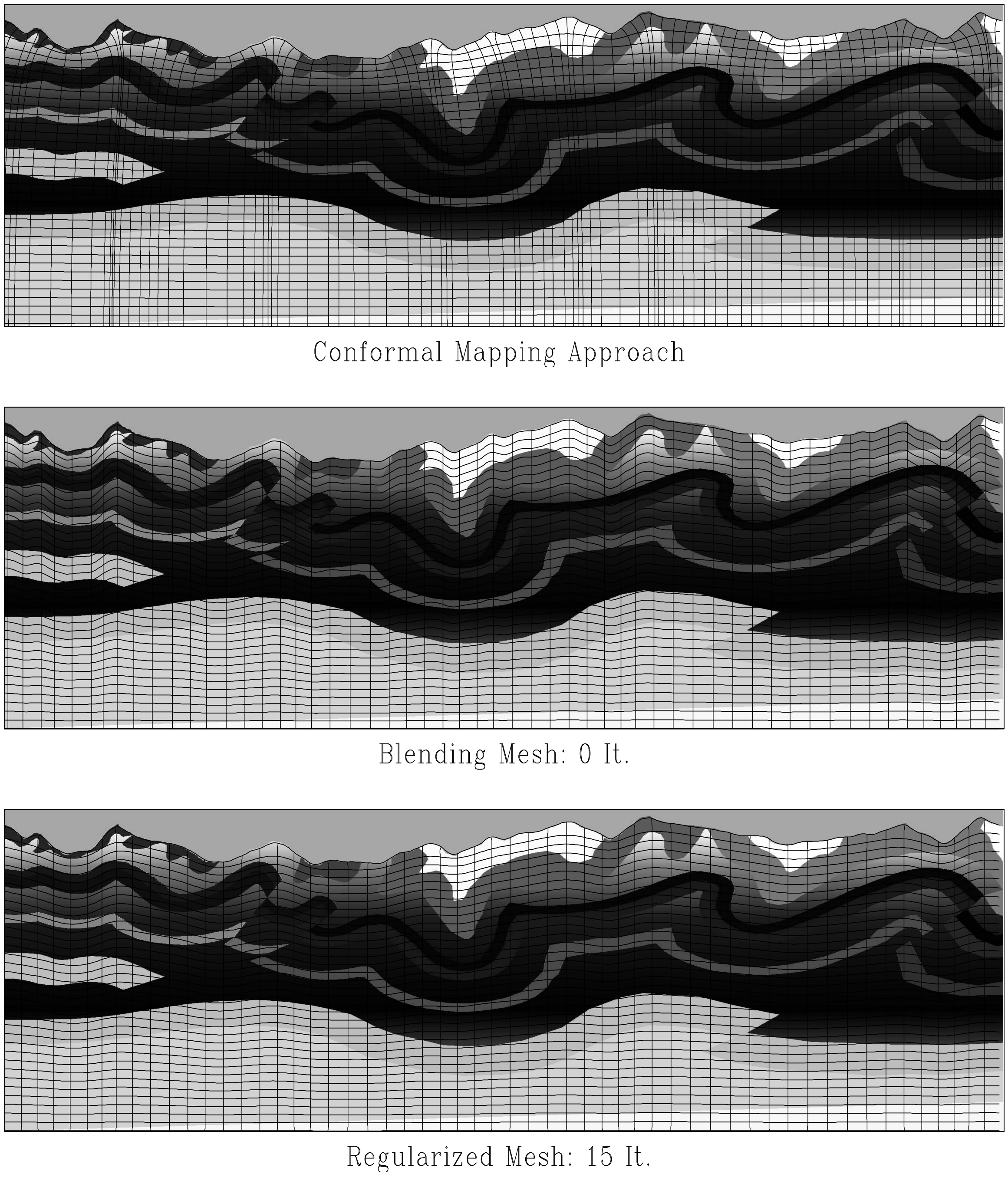

Figure 3: Meshing example for the wave-equation migration from topography application. Top panel: Locally orthogonal coordinate system calculated by conformal mapping (Shragge and Sava, 2005). Note the severe amount of grid clustering indicating a need for meshing regularization. Middle panel: Blended coordinate system $\{s^1, s^2\}$ forming the differential gridding algorithm input. Bottom panel: Regularized mesh $\{x^1, x^2\}$ after 15 iterations. $\boxed{\text{jeff2-Topography}}$ [ER]

CONCLUDING REMARKS

Coordinate systems generated using the approach advocated by Liseikin (2004) offer significant improvements in overall mesh smoothness. Accordingly, they should be good underlying meshes on which to numerical compute PDE solutions. Importantly, because the numerical solution process splits into finite differences along single coordinate directions, this approach easily extends to 3-D meshing or projecting 2-D surfaces into 3-D volumes. The results for this paper were generated using fairly basic monitor functions. Further work is required to better understand how other choices would potentially impact geophysical meshing solutions.

ACKNOWLEDGEMENTS

Thanks to Tom Dickens for drawing my attention to the Liseikin monograph and to Brad Artman for helpful discussions.

REFERENCES

Alkhalifah, T., 2003, Tau migration and velocity analysis: Theory and synthetic examples: Geophysics, **68**, 1331–1339.

Guggenheimer, H., 1977, Differential Geometry: Dover Publications, Inc., New York.

Liseikin, V., 2004, A Computational Differential Geometry Approach to Grid Generation: Springer-Verlag, Berlin.

Rüger, A. and D. Hale, 2006, Meshing for velocity modeling and ray tracing in complex velocity fields: Geophysics, **71**, U1–U11.

Sava, P. and S. Fomel, 2001, 3-D traveltime computation using Huygens wavefront tracing: Geophysics, **66**, 883–889.

Sava, P. and S. Fomel, 2005, Riemannian wavefield extrapolation: Geophysics, **70**, T45–T56.

Shragge, J. and P. Sava, 2005, Wave-equation migration from topography: 75st Ann. Internat. Mtg., SEG Technical Program Expanded Abstracts, 1842–1845.

Shragge, J., 2006, Generalized riemannian wavefield extrapolation: SEP–**124**.

APPENDIX A

This appendix details a numerical scheme for solving the differential gridding equations discussed in Liseikin (2004). The set of parabolic equations to solve are,

$$\frac{\partial s^1}{\partial t} = D^\xi[s^1] + D^\xi[f^k]\frac{\partial f^k}{\partial s^1}, \quad k=1,l \tag{A-1}$$

$$\frac{\partial s^2}{\partial t} = D^\xi[s^2] + D^\xi[f^k]\frac{\partial f^k}{\partial s^2}, \quad k=1,l \tag{A-2}$$

$$s^1(\xi^1,\xi^2,t) = \phi^1(\xi^1,\xi^2), \quad t \geq 0 \tag{A-3}$$

$$s^2(\xi^1,\xi^2,t) = \phi^2(\xi^1,\xi^2), \quad t \geq 0 \tag{A-4}$$

$$s^1(\xi^1,\xi^2,0) = s_0^1(\xi^1,\xi^2), \quad t = 0 \tag{A-5}$$

$$s^2(\xi^1,\xi^2,0) = s_0^2(\xi^1,\xi^2), \quad t = 0 \tag{A-6}$$

where,

$$D^\xi[v] = \qquad g^\xi_{22}\frac{\partial^2 v}{\partial\xi^1\partial\xi^1} - 2g^\xi_{12}\frac{\partial^2 v}{\partial\xi^1}\partial\xi^2 + g^\xi_{11}\frac{\partial^2 v}{\partial\xi^2\partial\xi^2}, \tag{A-7}$$

$$g^\xi_{ij} = \quad \frac{\partial s^k}{\partial\xi^i}\frac{\partial s^k}{\partial\xi^j} + \frac{\partial f^m[\mathbf{s}(\xi)]}{\partial\xi^i}\frac{\partial f^m[\mathbf{s}(\xi)]}{\partial\xi^j}, \quad i,j,k=1,2, \quad m=1,l. \tag{A-8}$$

Computational domain Ξ^2 is the unit square divided into N intervals equally spaced in the $\{\xi^1,\xi^2\}$ directions. The first transformation $\Xi^2 \to S^2$ interrelates the known coordinate values on boundaries of domains S^2 and Ξ^2,

$$\Phi(\xi^j) = \left[\phi^1(\xi^j),\phi^2(\xi^j)\right], \quad j=1,2. \tag{A-9}$$

The interior points of S^2 are generated using blending functions, $\alpha^i_{ij}(s)$, $0 \leq s \leq 1$, where α^u_{ij} is linear function defined by,

$$\alpha^i_{0j}(s) = 1-s, \quad \alpha^i_{1j} = s. \tag{A-10}$$

Blended coordinates $\{s^1,s^2\}$ are generated on S^2 with,

$$s^1(\xi^1,\xi^2,0) = F^1(\xi^1,\xi^2) + (1-\xi^2)\left[\phi^1(\xi^1,0) - F^1(\xi^1,0)\right] + \xi^2\left[\phi^1(\xi^1,1) - F^1(\xi^1,1)\right],$$
$$s^2(\xi^1,\xi^2,0) = F^2(\xi^1,\xi^2) + (1-\xi^2)\left[\phi^2(\xi^1,0) - F^2(\xi^1,0)\right] + \xi^2\left[\phi^i(\xi^1,1) - F^2(\xi^1,1)\right] \tag{A-11}$$

where,

$$F^i(\xi^1,\xi^2) = (1-\xi^1)\phi^i(0,\xi^2) + \xi^1\phi^i(1,\xi^2). \tag{A-12}$$

Equations A-1A-6 can be solved using finite difference approximations that march forward in time. To simplify notation coordinates s^1 and s^2 are redefined as $u = s^1$ and $v = s^2$. The finite difference solution is split into a two-stage process along the different coordinate axes. The first stage calculates solutions $u^{0+\frac{1}{2}}$ and $v^{0+\frac{1}{2}}$ for a step in the u direction at time $t = 0+\frac{1}{2}$ using the values u^0 and v^0. The second stage calculates solutions u^{0+1} and v^{0+1} for a step in the v direction at time $t = 0+1$ using intermediate values $u^{0+\frac{1}{2}}$ and $v^{0+\frac{1}{2}}$.

Explicitly, the four equations comprising the finite difference scheme for u^n_{ij} and v^n_{ij}, $0 \le i,j \le N$, $0 \le n$ on uniform grid $(ih, jh, n\tau)$ are,

$$u^{n+1/2}_{ij} - u^n_{ij} = \frac{\tau}{h^2}\left[g_{22}(\mathbf{s}^n_{ij})\,\mathbf{L}_{ij}(u^{n+\frac{1}{2}}) - 2\,g_{12}(\mathbf{s}^n_{ij})\,\mathbf{M}_{ij}(u^n) + g_{11}(\mathbf{s}^n_{ij})\,\mathbf{S}_{ij}(u^n) + \mathbf{P}_{ij}(u^n)\right] \quad 1 \le i,j \le N-1, n \ge 0 \tag{A-13}$$

$$v^{n+1/2}_{ij} - v^n_{ij} = \frac{\tau}{h^2}\left[g_{22}(\mathbf{s}^n_{ij})\,\mathbf{L}_{ij}(v^{n+\frac{1}{2}}) - 2\,g_{12}(\mathbf{s}^n_{ij})\,\mathbf{M}_{ij}(v^n) + g_{11}(\mathbf{s}^n_{ij})\,\mathbf{S}_{ij}(v^n) + \mathbf{Q}_{ij}(u^n)\right] \quad 1 \le i,j \le N-1, n \ge 0 \tag{A-14}$$

$$u^{n+1}_{ij} - u^{n+1/2}_{ij} = \frac{\tau}{h^2} g_{11}(\mathbf{s}^n_{ij})\,\mathbf{S}_{ij}(u^{n+1} - u^n) \quad 1 \le i,j \le N-1, n \ge 0 \tag{A-15}$$

$$v^{n+1}_{ij} - v^{n+1/2}_{ij} = \frac{\tau}{h^2} g_{11}(\mathbf{s}^n_{ij})\,\mathbf{S}_{ij}(v^{n+1} - v^n) \quad 1 \le i,j \le N-1, n \ge 0 \tag{A-16}$$

where,

$$g_{11}\left(\mathbf{s}^n_{ij}\right) = \left(\frac{u^n_{i+1\,j} - u^n_{i-1\,j}}{2h}\right)^2 + \left(\frac{v^n_{i+1\,j} - v^n_{i-1\,j}}{2h}\right)^2 + \frac{\mathbf{f}\left(\mathbf{s}^n_{i+1j}\right) - \mathbf{f}\left(\mathbf{s}^n_{i-1j}\right)}{2h} \cdot \frac{\mathbf{f}\left(\mathbf{s}^n_{i+1j}\right) - \mathbf{f}\left(\mathbf{s}^n_{i-1j}\right)}{2h} \quad 1 \le i,j \le N-1, \tag{A-17}$$

$$g_{12}\left(\mathbf{s}^n_{ij}\right) = \frac{u^n_{i+1\,j} - u^n_{i-1\,j}}{2h} \cdot \frac{u^n_{i\,j+1} - u^n_{i\,j-1}}{2h} + \frac{v^n_{i+1\,j} - v^n_{i-1\,j}}{2h} \cdot \frac{v^n_{i\,j+1} - v^n_{i\,j-1}}{2h} + + \frac{\mathbf{f}\left(\mathbf{s}^n_{i+1j}\right) - \mathbf{f}\left(\mathbf{s}^n_{i-1j}\right)}{2h} \cdot \frac{\mathbf{f}\left(\mathbf{s}^n_{ij+1}\right) - \mathbf{f}\left(\mathbf{s}^n_{ij-1}\right)}{2h} \quad 1 \le i,j \le N-1, \tag{A-18}$$

$$g_{22}\left(\mathbf{s}^n_{ij}\right) = \left(\frac{u^n_{i\,j+1} - u^n_{i\,j-1}}{2h}\right)^2 + \left(\frac{v^n_{i\,j+1} - v^n_{i\,j-1}}{2h}\right)^2 + \frac{\mathbf{f}\left(\mathbf{s}^n_{ij-1}\right) - \mathbf{f}\left(\mathbf{s}^n_{ij+1}\right)}{2h} \cdot \frac{\mathbf{f}\left(\mathbf{s}^n_{ij-1}\right) - \mathbf{f}\left(\mathbf{s}^n_{ij+1}\right)}{2h} \quad 1 \le i,j \le N-1,$$

$$\mathbf{L}_{ij}(w) = w_{i+1\,j} - 2\,w_{i\,j} + w_{i-1\,j}, \qquad 1 \le i,j \le N-1 \tag{A-19}$$

$$\mathbf{M}_{ij}(w) = \frac{1}{4}\left[w_{i+1\,j+1} - w_{i-1\,j+1} - w_{i+1\,j-1} + w_{i-1\,j-1}\right] \quad 1 \le i,j \le N-1 \tag{A-20}$$

$$\mathbf{S}_{ij}(w) = w_{i\,j+1} - 2\,w_{i\,j} + w_{i\,j-1}, \qquad 1 \le i,j \le N-1 \tag{A-21}$$

$$\mathbf{P}_{ij}\left(\mathbf{s}^n_{ij}\right) = b^k_{ij}(\mathbf{s}^n_{ij})\frac{\partial f^k}{\partial s_1} \qquad k = 1,l \qquad 1 \le i,j \le N-1 \tag{A-22}$$

$$\mathbf{Q}_{ij}\left(\mathbf{s}^n_{ij}\right) = b^k_{ij}(\mathbf{s}^n_{ij})\frac{\partial f^k}{\partial s_2} \qquad k = 1,l \qquad 1 \le i,j \le N-1 \tag{A-23}$$

$$b^k_{ij}\left(\mathbf{s}^n_{ij}\right) = g_{22}\left(\mathbf{s}^n_{ij}\right)\,\mathbf{L}_{ij}\left[f^k(\mathbf{s}^m)\right] - 2\,g_{12}\left(\mathbf{s}^n_{ij}\right)\,\mathbf{M}_{ij}\left[f^k(\mathbf{s}^m)\right] + g_{11}\left(\mathbf{s}^n_{ij}\right)\,\mathbf{S}_{ij}\left[f^k(\mathbf{s}^m)\right] \quad k = 1,l \qquad 1 \le i,j \le N-1 \tag{A-24}$$

Algorithm for Computation

The solution to the finite difference scheme is obtained through recursive back-substitution.

Equation A-13 - is solved by,

$$A_{ij}^{n+\frac{1}{2}} = g_{22}(\mathbf{s}_{ij}^n) \quad C_{ij}^{n+\frac{1}{2}} = 2\, g_{22}(\mathbf{s}_{ij}^n) + \theta \quad B_{ij}^{n+\frac{1}{2}} = g_{22}(\mathbf{s}_{ij}^n)$$
$$F_{ij}^n = \theta\, u_{ij}^n - 2\, g_{12}(\mathbf{s}_{ij}^n)\, \mathbf{M}_{ij}(u^n) + g_{11}(\mathbf{s}_{ij}^n)\, \mathbf{S}_{ij}(u^n) \; + \; \mathbf{P}_{ij}(u^n). \tag{A-25}$$

The solution for a fixed number j in range $1 \le j \le N-1$ obtained through,

$$u_{ij}^{n+1/2} = \alpha_{i+1j}^{n+1/2}\, u_{i+1j}^{n+\frac{1}{2}} + \beta_{i+1j}^{n+1/2}, \qquad i = 1, N-1, \qquad u_{Nj}^{n+\frac{1}{2}} = \phi_1(1, jh), \tag{A-26}$$

where,

$$\alpha_{i+1j}^{n+1/2} = \frac{B_{ij}^{n+\frac{1}{2}}}{C_{ij}^{n+\frac{1}{2}} - \alpha_{ij}^{n+1/2} A_{ij}^{n+\frac{1}{2}}}, \qquad i = 1, N-1, \qquad \alpha_{0j}^{n+\frac{1}{2}} = 0$$
$$\beta_{i+1j}^{n+1/2} = \frac{A_{ij}^{n+\frac{1}{2}} \beta_{ij}^{n+1/2} + F_{ij}^n}{C_{ij}^{n+\frac{1}{2}} - \alpha_{ij}^{n+1/2} A_{ij}^{n+\frac{1}{2}}} \qquad i = 1, N-1, \qquad \beta_{0j}^{n+\frac{1}{2}} = \phi_1(0, jh).$$

Equation A-14 - is solved by,

$$A_{ij}^{n+\frac{1}{2}} = g_{22}(\mathbf{s}_{ij}^n) \quad C_{ij}^{n+\frac{1}{2}} = 2\, g_{22}(\mathbf{s}_{ij}^n) + \theta \quad B_{ij}^{n+\frac{1}{2}} = g_{22}(\mathbf{s}_{ij}^n)$$
$$F_{ij}^n = \theta\, v_{ij}^n - 2\, g_{12}(\mathbf{s}_{ij}^n)\, \mathbf{M}_{ij}(v^n) + g_{11}(\mathbf{s}_{ij}^n)\, \mathbf{S}_{ij}(v^n) \; + \; \mathbf{P}_{ij}(v^n). \tag{A-27}$$

The solution for a fixed j in range $1 \le j \le N-1$ is obtained through,

$$v_{ij}^{n+1/2} = \alpha_{i+1j}^{n+1/2}\, v_{i+1j}^{n+\frac{1}{2}} + \beta_{i+1j}^{n+1/2}, \qquad i = 1, N-1, \qquad v_{Nj}^{n+\frac{1}{2}} = \phi_2(1, jh), \tag{A-28}$$

where,

$$\alpha_{i+1j}^{n+1/2} = \frac{B_{ij}^{n+\frac{1}{2}}}{C_{ij}^{n+\frac{1}{2}} - \alpha_{ij}^{n+1/2} A_{ij}^{n+\frac{1}{2}}}, \qquad i = 1, N-1, \qquad \alpha_{0j}^{n+\frac{1}{2}} = 0$$
$$\beta_{i+1j}^{n+1/2} = \frac{A_{ij}^{n+\frac{1}{2}} \beta_{ij}^{n+1/2} + F_{ij}^n}{C_{ij}^{n+\frac{1}{2}} - \alpha_{ij}^{n+1/2} A_{ij}^{n+\frac{1}{2}}} \qquad i = 1, N-1, \qquad \beta_{0j}^{n+\frac{1}{2}} = \phi_2(0, jh)$$

Equation A-15 - is solved by,

$$A_{ij}^{n+\frac{1}{2}} = g_{11}(\mathbf{s}_{ij}^n) \quad C_{ij}^{n+\frac{1}{2}} = 2\, g_{11}(\mathbf{s}_{ij}^n) + \theta \quad B_{ij}^{n+\frac{1}{2}} = g_{11}(\mathbf{s}_{ij}^n)$$
$$F_{ij}^{n+\frac{1}{2}} = \theta\, u_{ij}^{n+1/2} - g_{11}(\mathbf{s}_{ij}^n)\, \mathbf{S}_{ij}(u^n). \tag{A-29}$$

The solution for a fixed i in the range $1 \le i \le N-1$ is obtained through,

$$u_{ij}^{n+1} = \alpha_{ij+1}^{n+1}\, u_{ij+1}^{n+1} + \beta_{ij+1}^{n+1}, \qquad i = 1, N-1, \qquad u_{iN}^{n+1} = \phi_1(ih, 1), \tag{A-30}$$

where,

$$\alpha_{i+1j}^{n+1} = \frac{B_{ij}^{n+1}}{C_{ij}^{n+1} - \alpha_{ij}^{n+1} A_{ij}^{n+1}}, \qquad j = 1, N-1, \qquad \alpha_{i0}^{n+1} = 0$$

$$\beta_{i+1j}^{n+1} = \frac{A_{ij}^{n+1} \beta_{ij}^{n+1} + F_{ij}^{n+\frac{1}{2}}}{C_{ij}^{n+1} - \alpha_{ij}^{n+1} A_{ij}^{n+1}} \qquad j = 1, N-1, \qquad \beta_{i0}^{n+1} = \phi_1(ih, 0)$$

Equation A-16 - is solved by,

$$A_{ij}^{n+\frac{1}{2}} = g_{11}(\mathbf{s}_{ij}^{n}) \quad C_{ij}^{n+\frac{1}{2}} = 2\, g_{11}(\mathbf{s}_{ij}^{n}) + \theta \quad B_{ij}^{n+\frac{1}{2}} = g_{11}(\mathbf{s}_{ij}^{n})$$

$$F_{ij}^{n+\frac{1}{2}} = \theta\, v_{ij}^{n+1/2} - g_{11}(\mathbf{s}_{ij}^{n})\, \mathbf{S}_{ij}(v^n). \tag{A-31}$$

The solution for a fixed i in the range $1 \le j \le N-1$ is obtained through,

$$v_{ij}^{n+1} = \alpha_{ij+1}^{n+1}\, v_{ij+1}^{n+1} + \beta_{ij+1}^{n+1}, \qquad i = 1, N-1, \qquad v_{iN}^{n+1} = \phi_2(ih, 0), \tag{A-32}$$

where,

$$\alpha_{i+1j}^{n+1} = \frac{B_{ij}^{n+1}}{C_{ij}^{n+1} - \alpha_{ij}^{n+1} A_{ij}^{n+1}}, \qquad j = 1, N-1, \qquad \alpha_{i0}^{n+1} = 0$$

$$\beta_{i+1j}^{n+1} = \frac{A_{ij}^{n+1} \beta_{ij}^{n+1} + F_{ij}^{n+\frac{1}{2}}}{C_{ij}^{n+1} - \alpha_{ij}^{n+1} A_{ij}^{n+1}} \qquad j = 1, N-1, \qquad \beta_{i0}^{n+1} = \phi_1(ih, 0).$$

An approximate solution of the equations A-1 will be found at some large time T_N that satisfies,

$$\max_{0 \le i, j, \le N} \frac{1}{\tau} \sqrt{(u_{ij}^{n+1} - u_{ij}^{n})^2 + (v_{ij}^{n+1} - v_{ij}^{n})^2} \le \epsilon. \tag{A-33}$$

Step-by-Step Algorithm

The iterative solution method can be broken down into a number of distinct steps: i) define initial grid using blending functions in equation A-11; ii) compute functions $g_{11}(\mathbf{s_{ij}^0})$, $g_{12}(\mathbf{s_{ij}^0})$, $g_{22}(\mathbf{s_{ij}^0})$, $g(\mathbf{s_{ij}^0})$, $L_{ij}(u^0)$, $L_{ij}(v^0)$, $M_{ij}(u^0)$, $M_{ij}(v^0)$, $P_{ij}(\mathbf{s}^0)$, $Q_{ij}(\mathbf{s}^0)$, and $b_{ij}^{l}(\mathbf{s}^0)$; iii) compute $u_{ij}^{0+1/2}$ by solving equation A-13; iv) compute $v_{ij}^{0+1/2}$ by solving equation A-14; v) compute u_{ij}^{0+1} by solving equation A-15; vi) compute v_{ij}^{0+1} by solving equation A-16; vii) perform tolerance test in equation A-33; and viii) repeat steps 2-7 until tolerance is reached.