Automatic elliptic grid generation by an approximate factorisation algorithm

E. Ly[1] D. Norrison[2]

(Received 14 July 2006; revised 01 July 2007)

Abstract

A procedure for automatic numerical generation of a structured grid system with coordinate lines coinciding with all boundaries of a general two-dimensional region containing a body of arbitrary shape is presented. The solution procedure incorporated the method of false transients and the approximate factorisation algorithm, where a sequence of time steps is cycled in a geometric fashion with repeated endpoints, and has a capability for clustering grid lines close to the body. The procedure requires significantly much less computational effort to obtain a converged solution than a point or line successive over-relaxation iterative scheme. Although, the superiority of the presented algorithm has been demonstrated for the grid generation problem, it can be utilised for other problems requiring the solution of a set of elliptic partial differential equations of similar nature.

See http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/48 for this article, Published July 9, 2007. ISSN 1446-8735

Contents

1 Introduction

In order to solve the governing partial differential equations (PDEs) of fluid dynamics numerically, approximations to the partial differentials are introduced. Commonly employed numerical methods for solving the PDEs required all partial derivatives to be converted into finite difference equations (FDEs), which are solved at discrete points within the domain of interest. Therefore, a set of grid points within and on the boundaries of the domain is required to be specified to form a grid system. This process is known as grid generation [2, 8], which is the focus of this article.

We describe one of several common methods for generating smooth grids for odd geometries, namely the elliptic grid generation method, which is used to generate a two-dimensional boundary-fitted coordinate system with quadrilateral elements. This work is confined in two dimensions in the interest of simplicity of the theory, but in principle it can be extended to three dimensions. We let the curvilinear coordinates be the solution of a system of elliptic PDEs with source terms, subjected to Dirichlet boundary conditions

on all boundaries [2, 6, 8]. The source terms provide an option for clustering of grid line(s) or point(s), or even a combination of both, in a specified region of the domain. One coordinate is specified to be constant on each of the boundaries, so that there is a coordinate line coincident with each boundary. A finite difference based method, incorporating the method of false transients and the approximate factorisation (AF) algorithm [1, 3, 4, 5, 9], solves the PDEs in the computational domain for the physical coordinates of the grid points [6, 8]. In closure, grids around a body of irregular shape are generated, and the convergence rate of the proposed numerical solution procedure are compared with that of the point (PSOR) and line (LSOR) successive over-relaxation iterative schemes.

2 Grid generation equations

The mapping process from the physical coordinates $\boldsymbol{r} = (x, z)$ to the computational coordinates $\boldsymbol{\vartheta} = (\xi, \zeta)$ is described by the relation $\boldsymbol{\vartheta} = \boldsymbol{\vartheta}(\boldsymbol{r})$, which is assumed to have continuous derivatives of all orders. In order to generate an applicable grid, the mapping must be one-to-one to ensure the grid lines of the same family do not cross each other, and provides a smooth grid distribution with minimum skewness.

The following system of Poisson's equations is considered,

$$\boldsymbol{\vartheta}_{xx} + \boldsymbol{\vartheta}_{zz} = \mathcal{S}(\boldsymbol{\vartheta}) \,, \tag{1}$$

where $\mathcal{S} = (p, q)$ contains the source terms. As usual, $\boldsymbol{\vartheta}_x$ and $\boldsymbol{\vartheta}_{xx}$ denote $\partial\boldsymbol{\vartheta}/\partial x$ and $\partial^2\boldsymbol{\vartheta}/\partial x^2$, respectively. Grid point clustering is enforced by proper selection of the functions $p(\boldsymbol{\vartheta})$ and $q(\boldsymbol{\vartheta})$, and the selection is based on grid point or line attraction in the vicinity of defined grid line(s) or point(s), or even a combination of both. Since it is more convenient to solve for $\boldsymbol{r}$, where $\boldsymbol{\vartheta}$ is known in the computational domain, the dependent and independent

variables of Equation (1) are interchanged to provide

$$\left(\alpha \boldsymbol{r}_{\xi\xi} + P\boldsymbol{r}_{\xi}\right) - 2\beta \boldsymbol{r}_{\xi\zeta} + \left(\gamma \boldsymbol{r}_{\zeta\zeta} + Q\boldsymbol{r}_{\zeta}\right) = \boldsymbol{0}\,, \tag{2}$$

where $\boldsymbol{0}$ is a zero vector. The constants in Equation (2) are

$$\alpha = \boldsymbol{r}_{\zeta} \cdot \boldsymbol{r}_{\zeta}\,, \quad \beta = \boldsymbol{r}_{\xi} \cdot \boldsymbol{r}_{\zeta}\,, \quad \gamma = \boldsymbol{r}_{\xi} \cdot \boldsymbol{r}_{\xi}\,, \tag{3}$$

where $\cdot$ represents the dot product of two vectors, and the source terms are

$$P = \frac{p}{J^2}\,, \quad Q = \frac{q}{J^2}\,, \quad J = \frac{1}{x_{\xi} z_{\zeta} - x_{\zeta} z_{\xi}}\,, \tag{4}$$

where J is the Jacobian of transformation. The solution to Equation (2) is periodic in ξ due to the existence of a re-entrant boundary that formed the left and right boundaries of the computational domain. Since the system is quasi-linear, a linearisation process must be used in the numerical solution. For simplicity, a lagging of the coefficients (3) is employed, with the coefficients evaluated at the previous iteration or time level [2, 8].

3 Numerical solution procedure

Here we discuss an efficient AF algorithm, together with the method of false transients, for solving the elliptic PDE system defined by Equation (2). An artificial time dependent term, $\boldsymbol{r}_{\tau}$ (where τ is the artificial time scale), is appended to (2) to incorporate the temporal numerical dissipation, hence resulting in

$$\boldsymbol{r}_{\tau} = \left(\alpha \boldsymbol{r}_{\xi\xi} + P\boldsymbol{r}_{\xi}\right) - 2\beta \boldsymbol{r}_{\xi\zeta} + \left(\gamma \boldsymbol{r}_{\zeta\zeta} + Q\boldsymbol{r}_{\zeta}\right). \tag{5}$$

Since the boundary conditions are time independent for a static grid system, and provided that the numerical solution converges, we anticipate that $\boldsymbol{r}_{\tau}$ approaches zero. The price paid is the loss of a true transient solution, but this

TABLE 1: Well-known time difference rules for Equation (6).

Time difference rule	a	b
Trapezoidal rule	0	1/2
Euler implicit	0	1
Three-point backward	1/2	1
Euler explicit	0	0
Leap frog	−1/2	0

is not significant, since the objective of this work is to develop a numerical scheme that will generate the final solution at an enhanced convergence rate.

Let τ be discretised as $\tau \equiv \tau_n = n\Delta\tau$, where $\Delta\tau$ is a discrete increment of τ and n the iteration or time level, and $\boldsymbol{r}(\tau_n) = \boldsymbol{r}(n\Delta\tau) = \boldsymbol{r}^n$. Here the spatial dependence has been temporarily suppressed. The time derivative is approximated by a general time difference rule of a form suggested by Warming and Beam [9], which includes the rules presented in Table 1,

$$\left(\frac{\partial \boldsymbol{r}}{\partial \tau}\right)^n = \frac{(1+a)\overrightarrow{\Delta}_\tau - a\overleftarrow{\Delta}_\tau}{\Delta\tau\,(1+b\overrightarrow{\Delta}_\tau)}\boldsymbol{r}^n + \left(b-a-\tfrac{1}{2}\right)\mathcal{O}(\Delta\tau) + \mathcal{O}(\Delta\tau)^2\,, \quad (6)$$

where $\overrightarrow{\Delta}_\tau$ and $\overleftarrow{\Delta}_\tau$ are the forward and backward time difference operators.

Inserting (6) into (5) for $\boldsymbol{r}_\tau$ at time level τ_n provides

$$\left(1-\Delta\tilde{\tau}\mathcal{L}\right)\overrightarrow{\Delta}_\tau \boldsymbol{r}^n = \tilde{a}\overleftarrow{\Delta}_\tau \boldsymbol{r}^n + \Delta\tilde{\tau}\frac{\omega}{b}\mathcal{R}^n + \left(b-a-\tfrac{1}{2}\right)\mathcal{O}(\Delta\tau)^2 + \mathcal{O}(\Delta\tau)^3\,, \quad (7)$$

where ω is a relaxation factor,

$$\Delta\tilde{\tau} = \frac{b\,\Delta\tau}{1+a}\,, \quad \tilde{a} = \frac{a}{1+a}\,, \quad (8)$$

with $a \neq -1$, and $\mathcal{R}^n$ denotes the residual, which measures how well the FDEs are satisfied by the approximate solution at time level τ_n,

$$\mathcal{R}^n = \mathcal{L}\boldsymbol{r}^n = \left[\left(\alpha^n\frac{\partial^2}{\partial\xi^2} + P^n\frac{\partial}{\partial\xi}\right) - 2\beta^n\frac{\partial^2}{\partial\xi\partial\zeta} + \left(\gamma^n\frac{\partial^2}{\partial\zeta^2} + Q^n\frac{\partial}{\partial\zeta}\right)\right]\boldsymbol{r}^n\,. \quad (9)$$

In general, the operator appearing on the left side of (7) is difficult to invert. Therefore, in the AF algorithm it is chosen as a product of two or more factors, such that it closely resembles the operator $\mathcal{L}$ where only simple matrix operations are required, by neglecting all mixed derivatives and omitting third and higher order terms in $\Delta\tau$,

$$\left[1-\Delta\tilde{\tau}\left(\alpha^n\frac{\partial^2}{\partial\xi^2}+P^n\frac{\partial}{\partial\xi}\right)\right]\left[1-\Delta\tilde{\tau}\left(\gamma^n\frac{\partial^2}{\partial\zeta^2}+Q^n\frac{\partial}{\partial\zeta}\right)\right]\overrightarrow{\Delta}_\tau\boldsymbol{r}^n = \tilde{a}\overleftarrow{\Delta}_\tau\boldsymbol{r}^n+\Delta\tilde{\tau}\frac{\omega}{b}\mathcal{R}^n\,. \quad (10)$$

The factored equation (10) is then solved in an alternating direction manner,

$$\left[1-\Delta\tilde{\tau}\left(\alpha^n\frac{\partial^2}{\partial\xi^2}+P^n\frac{\partial}{\partial\xi}\right)\right]\Delta\boldsymbol{r}^* = \tilde{a}\overleftarrow{\Delta}_\tau\boldsymbol{r}^n+\Delta\tilde{\tau}\frac{\omega}{b}\mathcal{R}^n\,, \quad (11)$$

$$\left[1-\Delta\tilde{\tau}\left(\gamma^n\frac{\partial^2}{\partial\zeta^2}+Q^n\frac{\partial}{\partial\zeta}\right)\right]\overrightarrow{\Delta}_\tau\boldsymbol{r}^n = \Delta\boldsymbol{r}^*, \quad (12)$$

$$\boldsymbol{r}^{n+1} = \boldsymbol{r}^n+\overrightarrow{\Delta}_\tau\boldsymbol{r}^n\,. \quad (13)$$

In each iteration, a new approximation to the solution is found by systemically solving Equation (11) for the dummy temporal differences $\Delta\boldsymbol{r}^*$, Equation (12) for the unknown vector $\overrightarrow{\Delta}_\tau\boldsymbol{r}^n$, and finally, applying relation (13) to update the solution vector $\boldsymbol{r}^{n+1}$. When the solution converges, $\overrightarrow{\Delta}_\tau\boldsymbol{r}^n$ approach zero, and the numerical solution is $\boldsymbol{r}\approx\boldsymbol{r}^n$. This method is potentially fast since the solution process is fully vectorised, and variable time stepping is incorporated. Note that for an accurate factorisation, and to ensure that each linear system is strongly diagonally dominant, the time steps must be small relative to the spatial grid spacings. Ly and Gear [3, 4] observed that large errors can occur at the extreme ends of the frequency range, and suggested that this unfavourable behaviour be eliminated by cycling the time steps in a geometric sense, but with repeated endpoints ($\Delta\tilde{\tau}_1=\Delta\tilde{\tau}_2$ and $\Delta\tilde{\tau}_{M-1}=\Delta\tilde{\tau}_M$), where M is the number of time steps per cycle.

In the finite difference method, all the spatial derivatives of (11) and (12) are discretised using standard second order accurate central difference rules. The first spatial derivatives of coefficients (3) are discretised using standard second order accurate forward, central and backward difference rules, depending on whether the concerned grid point is on the body boundary, an interior grid point, or on the far-field boundary. This forms two sets of two linear tridiagonal equation systems, instead of tridiagonal block systems of equations as would occur if unfactored equation (7) is used. Equation (10) reduces a formidable matrix inversion problem to a series of small bandwidth matrix inversion problems, where efficient solution algorithms are utilised, by reducing the two dimensional matrix inversion problem to two one dimensional problems. Equation (11) forms a so-called cyclic tridiagonal system (due to the periodic boundary conditions at the re-entrant boundary), which is inverted by applying the Sherman–Morrison formula [7] together with a standard tridiagonal system solver, treating the system as tridiagonal plus a small correction. Equation (12) forms a pure tridiagonal system that can be efficiently inverted by any robust tridiagonal solver. When $a \neq 0$, Equations (11) and (12) form a three time level iterative scheme, which requires no additional computation and no additional storage as compared to the two time level scheme (when $a = 0$) that requires two levels of data, namely $\boldsymbol{r}^n$ and $\overrightarrow{\Delta}_\tau \boldsymbol{r}^n$. At the start of an iteration, $\boldsymbol{r}^n$ and $\overleftarrow{\Delta}_\tau \boldsymbol{r}^n$ are both known subsequent to advancing the solution from time level τ_n to τ_{n+1} using (11) to (13). After $\Delta \boldsymbol{r}^*$ has been computed from (11) along a ζ-constant line, $\overleftarrow{\Delta}_\tau \boldsymbol{r}^n$ along this same line is no longer required, and is over written with the $\Delta \boldsymbol{r}^*$ values. Similarly, as $\overrightarrow{\Delta}_\tau \boldsymbol{r}^n$ is computed from (12) along a ξ-constant line, the new result is written in the storage space containing the $\Delta \boldsymbol{r}^*$ values.

4 Computational examples

The AF algorithm is employed with an Euler implicit time difference rule ($a = 0$, $b = 1$) to generate an O-type grid system around a body of irregular

TABLE 2: Comparison of computational times.

Algorithm	PSOR	LSOR	AF1	AF2
Iterations	189	603	31	37
Error, E	$4.980(10^{-5})$	$4.975(10^{-5})$	$4.981(10^{-5})$	$4.988(10^{-5})$
Total CPU time	0.291 secs	1.963 secs	0.181 secs	0.271 secs

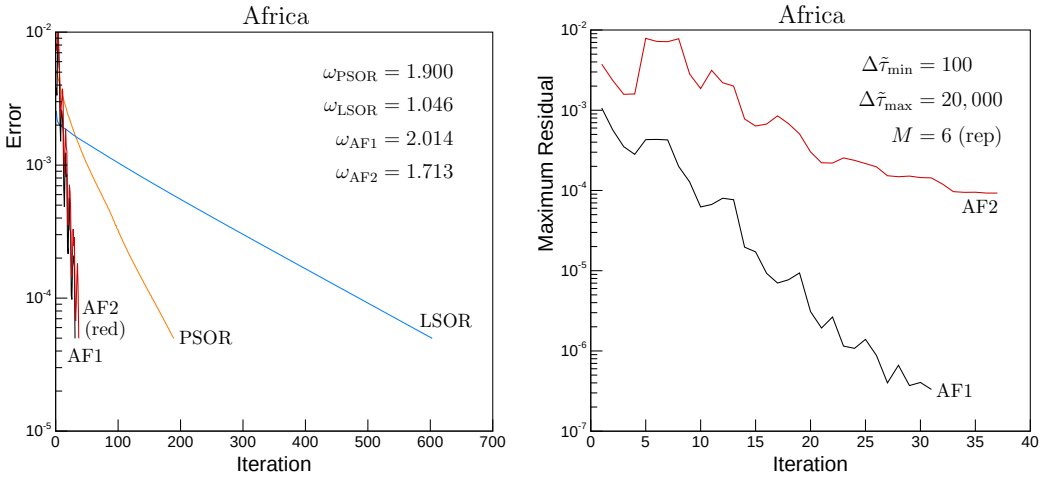

FIGURE 1: Convergence histories.

shape represented by the continent Africa, with 87 points around the body contour and 31 points in the radial direction. It was found, through numerical experiments, that the AF algorithm became unstable if the Euler explicit and leap frog time difference rules were used. A cycle of six time steps with values between $\Delta\tilde{\tau}_{\min} = 100$ and $\Delta\tilde{\tau}_{\max} = 2 \times 10^4$ is used. The grids have been generated without the source terms (designated as AF1 algorithm, essentially solving a system of Laplace's equations) and with source terms of the following form (AF2 algorithm),

$$p(\xi, \zeta) = 0 \quad \text{and} \quad q(\xi, \zeta) = \sum_{j=1}^{J} \mu_j \operatorname{sign}(\zeta_j - \zeta) \exp\left(-\kappa_j \left|\zeta_j - \zeta\right|\right) . \qquad (14)$$

Functions (14) enforce the ζ-lines clustered toward a group of specified grid lines ζ_j, where μ_j and κ_j are the amplification and decay factor, respectively, for grid line ζ_j. For the purposes of comparison, the PSOR and LSOR schemes are used to generate the same set of grids, but without the source terms. Optimal ω values were obtained by numerical experiments and were used in all the schemes, see Figure 1. It was surprising that ω for the AF1 algorithm is greater than two, and very close to one for the LSOR scheme, which probably helps to explain why the LSOR scheme is much slower than the PSOR scheme in reaching convergence (see Table 2). All schemes are terminated when the error (representing the total changes in the solution) per grid point, E, is less than 5×10^{-5}. Figure 1 compares the convergence history of all schemes, and shows that the maximum residual of the AF algorithms reduce substantially whenever the algorithms completed one cycle of time steps. Overall, the errors decrease in a logarithmical manner, and the AF algorithms require the least number of iterations for convergence.

Since a single AF iteration requires more computational (CPU) time than a single PSOR or LSOR iteration, a comparison of the number of iterations required for convergence is not meaningful. Therefore, the actual CPU times required for convergence on an IBM ThinkPad T22 model notebook with an Intel Pentium III processor of speed 995 MHz, running the Microsoft Windows XP Professional operating system with 384 MB of physical memory and

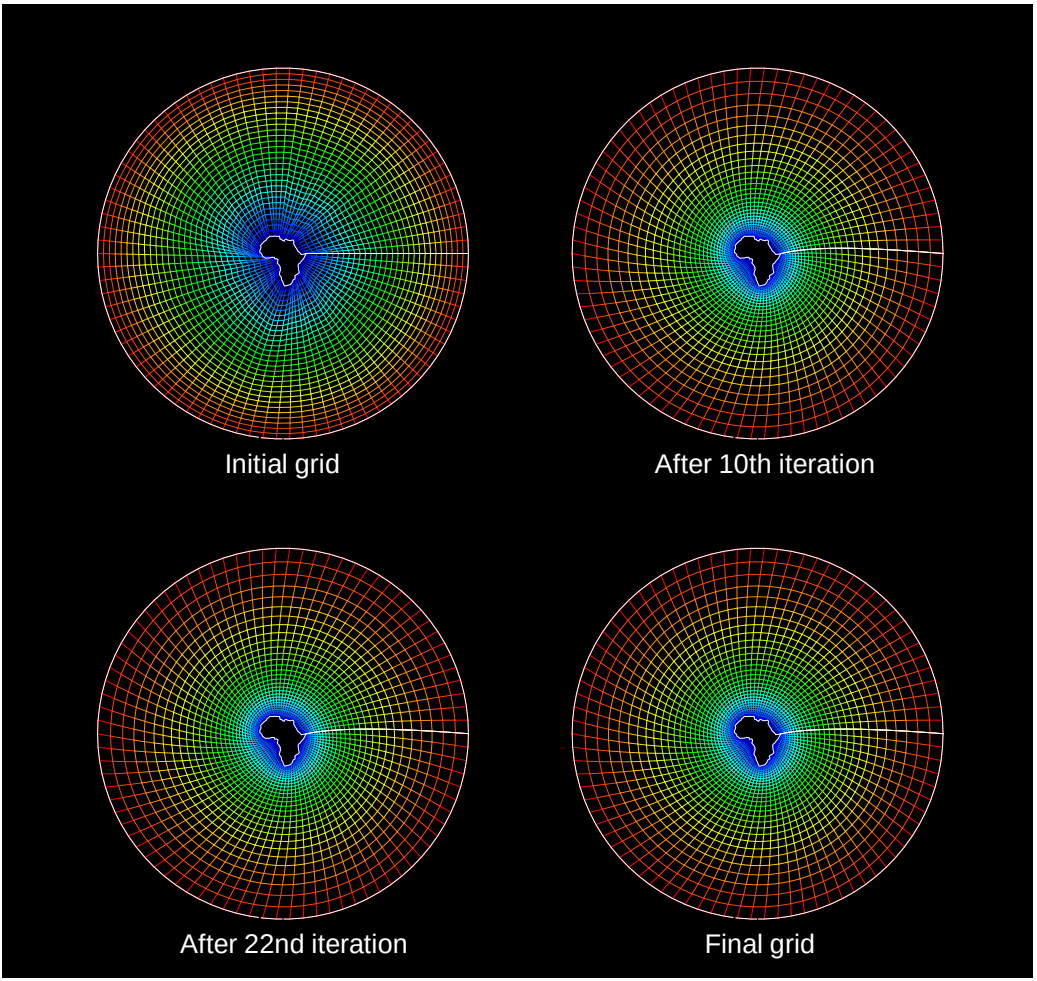

FIGURE 2: Development of a grid system during the solution process.

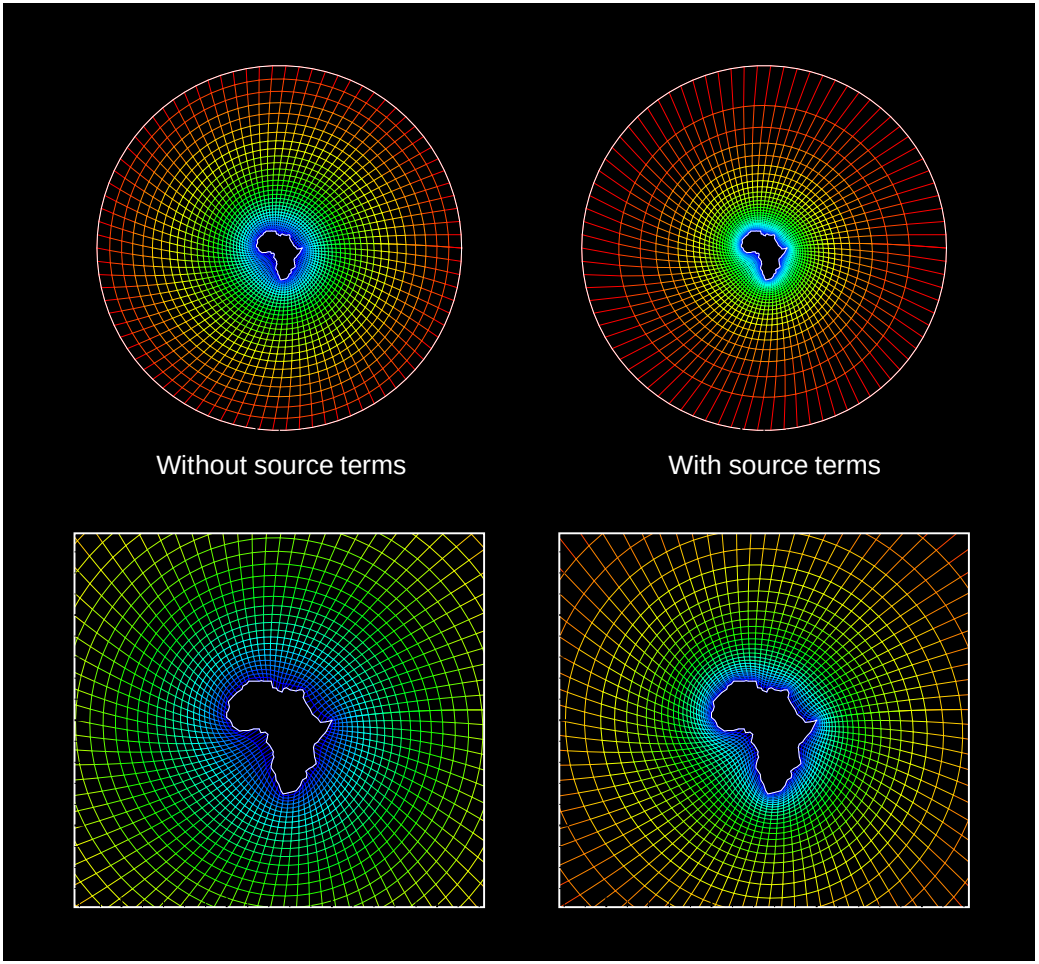

FIGURE 3: Generating grid systems without (left plots) and with (right plots) grid lines clustering towards ζ_1- to ζ_4-line with $\mu = (100, 100, 90, 90)$ and $\kappa = (0.2, 0.2, 0.25, 0.3)$.

576 MB of virtual memory are compared in Table 2. The comparisons show that the AF1 algorithm is about twice as fast as the PSOR scheme, and ten times faster than the LSOR scheme. We observed that the speed at which the scheme converges mainly depended on the manner in which the shape of the branch-cuts are changing from one iteration to the next. In the AF algorithm, the grid points along the branch-cut are included implicitly into the equation system, and so, the branch-cut points are updated with all other interior points simultaneously (as illustrated in Figure 2). Whereas for the PSOR and LSOR schemes, these grid points are computed after each iteration, hence the branch-cuts are updated with one iteration level lapsed. When the source terms are included, the number of iterations required for convergence by the AF2 algorithm increases by 20 %, and the computational time increases by 50 %.

Careful examination of the plots in Figure 2 (where the grids shown after the 10th, 22nd and final iterations are visually indistinguishable), and Figure 3 clearly indicate a smooth distribution of the grid points within the domain, in particular noting the adjustment of the grid points on the branch-cut. The natural clustering of grid points due to the elliptic nature of the governing PDEs, and enforced clustering of grid lines due to the inclusion of the source terms toward the body can be seen. The final grid is correctly generated, even though the initial grid is badly distorted with grid lines crossing and highly skewed in some regions near the body, which makes the process suitable for automatic grid generation computer code.

5 Concluding remarks

A system of Poisson equations is solved in the computational domain by a finite difference method to generate a structured grid around a single body of arbitrary shape in two dimensions. In the AF algorithm, the method of false transients is incorporated, in which a sequence of time steps is cycled

in a geometric fashion with repeated endpoints. We found that the proposed AF algorithm (with and without the source terms for clustering of grid lines towards the body) is significantly faster in reaching convergence to the user's required accuracy than both the PSOR and LSOR schemes. We observed that if the employed numerical scheme is numerically stable and converged, a correct final grid system can always be obtained independent of the form of its initial grid system, and hence, make it suitable for automatic grid generation computer code. No restrictions are enforced on the shape of the boundaries, which may even be time dependent, and the approach can be extended to generating grids for body of arbitrary shapes in three dimensions. Although, the superiority of the AF algorithm has been demonstrated for the automatic grid generation problem, it can be utilised for other problems requiring the solution of a set of elliptic PDEs of similar nature.

References

[1] Catherall, D., Optimum Approximate-Factorization Schemes for Two-Dimensional Steady Potential Flows, *AIAA Journal*, **20**, 8, 1982, pp. 1057–1063. C205

[2] Hoffmann, K. A., *Computational Fluid Dynamics for Engineers*, Engineering Educational System, Texas, USA, 1989. C204, C205, C206

[3] Ly, E., Improved Approximate Factorisation Algorithm for the Steady Subsonic and Transonic Flow over an Aircraft Wing, in *Proceedings of the 21st Congress of the International Council of the Aeronautical Sciences (ICAS98)*, AIAA and ICAS, Melbourne, Australia, Sep. 1998, Paper A98-31699. C205, C208

[4] Ly, E., and Gear, J. A., Time-Linearized Transonic Computations Including Shock Wave Motion Effects, *Journal of Aircraft*, **39**, 6, Nov./Dec. 2002, pp. 964–972. C205, C208

[5] Ly, E., and Nakamichi, J., Time-Linearised Transonic Computations Including Entropy, Vorticity and Shock Wave Motion Effects, *The Aeronautical Journal*, Nov. 2003, pp. 687–695. C205

[6] Mathur, J. S., and Chakrabartty, S. K., An Approximate Factorization Scheme for Elliptic Grid Generation with Control Functions, *Numerical Methods for Partial Differential Equations*, **10**, 6, 1994, pp. 703–713. C205

[7] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, Second Edition, Cambridge University Press, USA, 1994. C209

[8] Thompson, J. F., Thames, F. C., and Mastin, C. W., *Boundary-Fitted Curvilinear Coordinate Systems for Solution of Partial Differential Equations on Fields Containing any Number of Arbitrary Two-Dimensional Bodies*, NASA Contractor Report CR-2729, Washington DC, USA, July 1977, 253 pages. C204, C205, C206

[9] Warming, R. F., and Beam, R. M., On the Construction and Application of Implicit Factored Schemes for Conservation Laws, in *SIAM-AMS Proceedings*, **11**, USA, 1978, pp. 85–129. C205, C207

Author addresses

1. **E. Ly**, School of Mathematical and Geospatial Sciences, SET Portfolio, RMIT University, Melbourne, Victoria 3001, Australia.
mailto:eddie.ly@rmit.edu.au

2. **D. Norrison**, School of Mathematical and Geospatial Sciences, SET Portfolio, RMIT University, Melbourne, Victoria 3001, Australia.