ELSEVIER

Comput. Methods Appl. Mech. Engrg. 160 (1998) 359–382

Computer methods in applied mechanics and engineering

A monotone time relaxation matrix procedure for improved convergence to steady-state for CFD algorithms

Subrata Roy [a,1], A.J. Baker [b,*]

[a] *Computational Mechanics Corporation, Knoxville, TN 37919-3382, USA*
[b] *University of Tennessee, Knoxville, TN 37996-2030, USA*

Received 21 January 1997; revised 5 August 1997; accepted 28 August 1997

Abstract

For a time independent formulation, temporal schemes are traditionally employed for stable convergence to steady state. However, the pseudo-transient solution process may become slow as it suffers heavily from oscillations at large timesteps. This paper develops a theory, involving a scaling parameter embedded in the statically-condensed weak statement finite element time-term (mass) matrix, leading to monotone convergence to steady state using significantly fewer timesteps. Verification tests for a heat transfer and a compressible flow problem in 1-D and 2-D document the developed procedure.

1. Introduction

Consider the unsteady conservation law system

$$\mathcal{L}(q) = \frac{\partial q}{\partial t} + \nabla \cdot (\boldsymbol{f} - \boldsymbol{f}^{v}) - s = 0, \quad \text{on } \Omega \times t \subset \Re^{d} \times \Re^{+} \tag{1}$$

where q is the state variable, $\boldsymbol{f} = f_1(\boldsymbol{u}, q)$ is the kinetic flux vector with convection velocity vector $\boldsymbol{u}$, $\boldsymbol{f}^{v} = f_2(\epsilon \nabla q)$ is the dissipative flux vector with diffusion coefficient ϵ, and s is the source term. The boundary of the d-dimensional problem domain Ω for (1) is the *union* of piecewise convex segments $\partial\Omega_i$ upon which appropriate Dirichlet–Neumann constraints are imposed.

Discrete approximate algorithms for this transient CFD problem class characteristically exhibit a phase dispersion error that distorts the approximation solution evolution process. This is especially true using the relatively 'higher-order accurate' FE-generated non-diagonal mass matrix coupling the state variable time derivative. The spatial discretization-induced '$2\Delta x$ wave' zero phase velocity produces a short wavelength dispersive error mode that cascades to smaller wave numbers as time advances, as illustrated in Fig. 1 for an elementary transient heat conduction problem.

The traditional, and theoretically esthetic remedy is to increase the density of degrees of freedom, e.g. mesh nodalization. The less-pleasing approach is to introduce an artificial diffusion mechanism, either explicitly via an added dissipation term [1] or implicitly through flux vector upwinding [2]. Time-

*Corresponding author.
[1] Present address: Case Corporation, Technology Center, Burr Ridge, IL 60521-6975, USA. email: rsubrata@casecorp.com

0045-7825/98/$19.00
PII: S0045-7825(97)00294-6

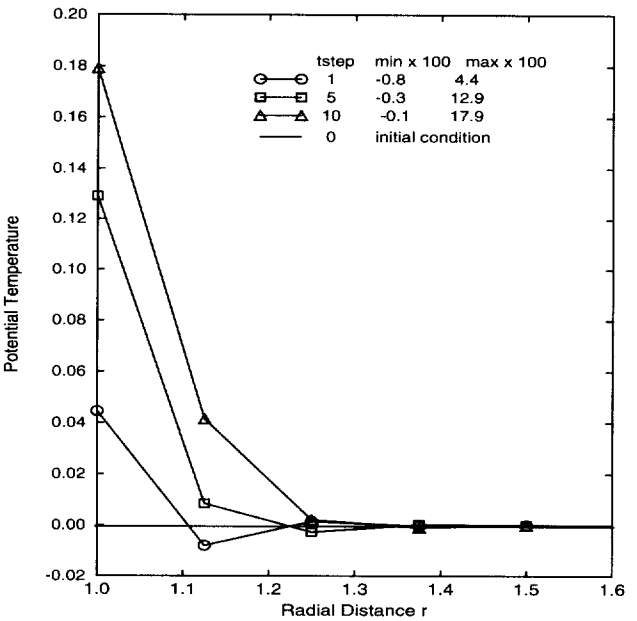


Fig. 1. Discrete approximate solution dispersive oscillation example, 1-D unsteady pure diffusion.

relaxation procedures like ADI [3], or subiteration procedures, e.g. NPARC [4], are available to induce faster convergence to steady state in a time-independent formulation.

In this paper, we derive, implement and document performance of a theoretical innovation for improving steady-state evolution stability via static condensation of an augmented time (mass) matrix of a finite element discretized Galerkin weak statement (GWS). The performance of the monotone time acceleration (MTA) method, easily incorporated with implicit time methodology, is theoretically characterized via a Fourier modal analysis. Numerical solutions are presented for 1-D and 2-D test problems, confirming theoretical expectations for algorithm performance for appropriate problems in heat transfer and compressible inviscid flow.

2. Weak statement algorithm

Independent of the dimension d of Ω, and for general variation of physical properties in the flux vectors (1), the semi-discrete finite element (FE) implementation of a weak form with associated trial and test space function sets, produces the so-called 'weak statement WS^h' for (1), which always yields an ordinary differential equation (ODE) system of the form [5]

$$WS^h = [M]\{Q(t)\}' + \{R\} = \{0\} \tag{2a}$$

where

$$[M] = S_e[M_k]_e \tag{2b}$$

$$\{R\} = S_e\left(([U_k]_e + [D_k]_e)\{Q(t)\}_e - \{b(t)\}_e\right) \tag{2c}$$

and S_e symbolizes the 'assembly operation' carrying local (element) matrix coefficients (subscript e) into the global arrays. $\{Q(t)\}$ is the time-dependent discrete approximation nodal degree-of-freedom (DOF) coefficient set. $\{Q(t)\}'$ denotes $\mathrm{d}\{Q\}/\mathrm{d}t$ utilized within any ODE algorithm, e.g. θ-implicit, n-step Runge–Kutta, etc. to convert (2a) to a computable algebraic statement.

On each finite element domain Ω_e, where $\cup\Omega_e = \Omega^h$ is the discretization of Ω, $[M]_e$ is the mass matrix associated with interpolation, $[U]_e$ carries the convective information from $\boldsymbol{f}$, $[D]_e$ is the diffusion

matrix resulting from genuine (f^v) and/or artificial diffusion effects, and $\{b\}_e$ contains all given data. The subscript k in (2b,c) denotes that each element-DOF rank matrix is dependent on the polynomial degree k of the selected FE basis.

Element-level matrices are always easy to form, e.g. the linear ($k = 1$) basis one dimensional ($d = 1$) GWS matrices for uniform fluid speed U and diffusion coefficient ϵ are [5],

$$[M_1]_e = \int_{\Omega_e} \{N_1\}\{N_1\}^{\mathrm{T}}\,\mathrm{d}x = \frac{\ell_e}{3}\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}_{(2,2)}$$

$$[U_1]_e = \mathrm{U}\int_{\Omega_e} \{N_1\}\frac{\mathrm{d}\{N_1\}^{\mathrm{T}}}{\mathrm{d}x}\,\mathrm{d}x = \frac{\mathrm{U}}{2}\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix}_{(2,2)} \tag{3}$$

$$[D_1]_e = \epsilon\int_{\Omega_e} \frac{\mathrm{d}\{N_1\}}{\mathrm{d}x}\frac{\mathrm{d}\{N_1\}^{\mathrm{T}}}{\mathrm{d}x}\,\mathrm{d}x = \frac{\epsilon}{2\ell_e}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}_{(2,2)}$$

where ℓ_e is the element measure, equal to half the finite element span between vertex (end-point) nodes. The subscripts in parenthesis denote matrix rank. Conversely, for the quadratic ($k = 2$) basis on $d = 1$ [5],

$$[M_2]_e = \int_{\Omega_e} \{N_2\}\{N_2\}^{\mathrm{T}}\,\mathrm{d}x = \frac{\ell_e}{15}\begin{bmatrix}4 & 2 & -1\\ 2 & 16 & 2\\ -1 & 2 & 4\end{bmatrix}_{(3,3)} \tag{4}$$

For $d = 2$ quadrilateral elements, using transformation coordinates $\eta_1, \eta_2 \in (-1, 1)$, the companion GWS Lagrange bilinear ($k = 1$) and biquadratic ($k = 2$) mass matrices are, respectively

$$[M_1]_e = \frac{\det_e}{3^d}\begin{bmatrix}4 & 2 & 1 & 2\\ 2 & 4 & 2 & 1\\ 1 & 2 & 4 & 2\\ 2 & 1 & 2 & 4\end{bmatrix}_{(4,4)} \tag{5}$$

and

$$[M_2]_e = \frac{\det_e}{15^d}\begin{bmatrix}16 & -4 & 1 & -4 & 8 & -2 & -2 & 8 & 4\\ -4 & 16 & -4 & 1 & 8 & 8 & -2 & -2 & 4\\ 1 & -4 & 16 & -4 & -2 & 8 & 8 & -2 & 4\\ -4 & 1 & -4 & 16 & -2 & -2 & 8 & 8 & 4\\ 8 & 8 & -2 & -2 & 64 & 4 & -16 & 4 & 32\\ -2 & 8 & 8 & -2 & 4 & 64 & 4 & -16 & 32\\ -2 & -2 & 8 & 8 & -16 & 4 & 64 & 4 & 32\\ 8 & -2 & -2 & 8 & 4 & -16 & 4 & 64 & 32\\ 4 & 4 & 4 & 4 & 32 & 32 & 32 & 32 & 256\end{bmatrix}_{(9,9)} \tag{6}$$

where the 'measure' $\det_e$ is the determinant of the transformation matrix to the local coordinate system intrinsic to Ω_e [5]. For Ω_e on $d = 3$, $\eta_1, \eta_2, \eta_3 \in (-1, 1)$, the GWS Lagrange tri-linear ($k = 1$) tensor product basis mass matrix $[M_1]_e$ is

$$[M_1]_e = \frac{\det_e}{3^d}\begin{bmatrix}8 & 4 & 2 & 4 & 4 & 2 & 1 & 2\\ 4 & 8 & 4 & 2 & 2 & 4 & 2 & 1\\ 2 & 4 & 8 & 4 & 1 & 2 & 4 & 2\\ 4 & 2 & 4 & 8 & 2 & 1 & 2 & 4\\ 4 & 2 & 1 & 2 & 8 & 4 & 2 & 4\\ 2 & 4 & 2 & 1 & 4 & 8 & 4 & 2\\ 1 & 2 & 4 & 2 & 2 & 4 & 8 & 4\\ 2 & 1 & 2 & 4 & 4 & 2 & 4 & 8\end{bmatrix}_{(8,8)} \tag{7}$$

The $d = 3$ quadratic ($k = 2$) tensor product basis mass matrix is of rank 27 and possesses a coefficient distribution character analogous to (6).

The assembled (global) form of (2a) provides the time derivatives necessary to evaluate temporal terms in an ODE algorithm Taylor series. Selecting the θ-implicit one-step Euler family as the example, where subscript n denotes time level, then

$$\{Q(t_{n+1})\} \equiv \{Q\}_{n+1} = \{Q\}_n + \Delta t[\theta\{Q\}'_{n+1} + (1-\theta)\{Q\}'_n] + \mathcal{O}\left(\Delta t^{f(\theta)}\right)$$

$$= \{Q\}_n - \Delta t[M]^{-1}(\theta\{R\}_{n+1} + (1-\theta)\{R\}_n) + .. \tag{8}$$

upon substitution of (2a). Clearing $[M]^{-1}$ and collecting terms into a homogeneous form yields the $\mathrm{WS}^h + \theta$ Taylor series algorithm terminal matrix algebra statement as

$$\{FQ\} = [M]\{Q_{n+1} - Q_n\} + \Delta t(\theta\{R\}_{n+1} + (1-\theta)\{R\}_n) = \{0\} \tag{9}$$

The Newton algorithm for solution of (9) is

$$[J]\{\Delta Q\}_{n+1} = -\Delta t\{R\}_n, \quad \text{linear} \tag{10a}$$

$$[J]\{\delta Q\}^{p+1} = -\{FQ\}^p_{n+1}, \quad \text{nonlinear} \tag{10b}$$

where for iteration index p

$$\{Q\}^{p+1}_{n+1} \equiv \{Q\}^p_{n+1} + \{\delta Q\}^{p+1} = \{Q\}_n + \sum_{i=0}^{p}\{\delta Q\}^{i+1} \tag{11}$$

For the linear problem statement (1), (10a) converges in a single step, hence $\{Q\}_{n+1} = \{Q\}_n + \{\Delta Q\}_{n+1}$. The Newton–Jacobian definition is

$$[J] = \frac{\partial\{FQ\}}{\partial\{Q\}} = [M] + \theta\Delta t\left(\frac{\partial\{R\}}{\partial\{Q\}}\right) \tag{12}$$

3. Static condensation

A basic ingredient of the MTA procedure, for implementation and for efficiency, is static condensation, [5,6]. As the mesh is refined, and with Lagrange FE basis functions $\{N_k\}, k > 1$, the global system DOF increases rapidly in (9), especially for $d > 1$. This rank escallation transcends directly to significantly increased computer resource requirements for solving (9)–(12).

Static condensation is a FE methodology that can contain escalation of matrix rank via element-level matrix rank reduction prior to assembly. The global matrix equation (10a), is always formed as the assembly of the corresponding element contributions. Rearrange as necessary the representative element-level Jacobian matrix $[J]_e$ of rank m into the partitioned form

$$[J]_{e\,(m,m)} = \begin{bmatrix} [A]_{(\alpha,\alpha)} & [B]_{(\alpha,\beta)} \\ [C]_{(\beta,\alpha)} & [D]_{(\beta,\beta)} \end{bmatrix}_{e\,(m,m)} \tag{13}$$

Letting $\{\chi\}$ denote appropriate entries in $\{\Delta Q\}_e$ or $\{\delta Q\}_e$, the corresponding element-level form of (10a) or (10b) is

$$[J]_{e\,(m,m)}\{\chi\}_{e\,(m)} \equiv -\begin{Bmatrix} b_{(\alpha)} \\ b_{(\beta)} \end{Bmatrix}_{e\,(m)} \equiv \begin{bmatrix} [A]_{(\alpha,\alpha)} & [B]_{(\alpha,\beta)} \\ [C]_{(\beta,\alpha)} & [D]_{(\beta,\beta)} \end{bmatrix} \begin{Bmatrix} \chi_{(\alpha)} \\ \chi_{(\beta)} \end{Bmatrix}_{e\,(m)} \tag{14}$$

For the lower partition $[D]_{(\beta,\beta)}$ in (14) assumed invertible, the DOF contained in the partition $\{\chi_{(\beta)}\}$, $\beta = m - \alpha$ can be eliminated from explicit appearance via static condensation. This operation produces the reduced-rank (superscript R), element-level statement for (14) as

$$[J]^{\mathrm{R}}_{e\,(\alpha,\alpha)}\{\chi_{(\alpha)}\} = -\{b_{(\alpha)}\}^{\mathrm{R}} \tag{15}$$

with definitions

$$[J]^{\mathrm{R}}_{e\,(\alpha,\alpha)} = [A]_{(\alpha,\alpha)} - [B]_{(\alpha,\beta)}[D]^{-1}_{(\beta,\beta)}[C]_{(\beta,\alpha)} \tag{16a}$$

$$\{b\}^{\mathrm{R}}_{(\alpha)} = \{b_{(\alpha)}\} - [B]_{(\alpha,\beta)}[D]^{-1}_{(\beta,\beta)}\{b_{(\beta)}\} \tag{16b}$$

Thereby, β DOF for the FE domain Ω_e have been locally 'eliminated', and assembly of (15) to the global form (10a,b) will reflect this DOF reduction.

The MTA algorithm focusses on the time (mass) matrix contribution contained in (10a), hence (16a). For the $k = 2$ Lagrange mass matrix (4) on $d = 1$, static condensation of the non-vertex node DOF yields

$$[M_2]^{\mathrm{R}}_e = \frac{\ell_e}{12}\begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}_{(2,2)} = \frac{\det_e}{12^d}\begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}_{(2,2)} \tag{17}$$

Correspondingly, the (9,9) rank matrix $[M_2]_e$ for $d = 2$, recall (6), with all (five) non-vertex node DOF statically condensed, produces the linear basis rank (4,4) reduced mass matrix

$$[M_2]^{\mathrm{R}}_e = \frac{\det_e}{12^d}\begin{bmatrix} 9 & -3 & 1 & -3 \\ -3 & 9 & -3 & 1 \\ 1 & -3 & 9 & -3 \\ -3 & 1 & -3 & 9 \end{bmatrix}_{(4,4)} \tag{18}$$

Finally, on $d = 3$, the rank 27 Lagrange $k = 2$ basis mass matrix $[M_2]_e$, with all (nineteen) non-vertex nodal DOF condensed, yields the linear basis rank mass matrix $[M_2]^{\mathrm{R}}_e$

$$[M_2]^{R}_e = \frac{\det_e}{12^d}\begin{bmatrix} 27 & -9 & 3 & -9 & -1 & 3 & -9 & 3 \\ -9 & 27 & -9 & 3 & 3 & -1 & 3 & -9 \\ 3 & -9 & 27 & -9 & -9 & 3 & -1 & 3 \\ -9 & 3 & -9 & 27 & 3 & -9 & 3 & -1 \\ -1 & 3 & -9 & 3 & 27 & -9 & 3 & -9 \\ 3 & -1 & 3 & -9 & -9 & 27 & -9 & 3 \\ -9 & 3 & -1 & 3 & 3 & -9 & 27 & -9 \\ 3 & -9 & 3 & -1 & -9 & 3 & -9 & 27 \end{bmatrix}_{(8,8)} \tag{19}$$

4. Monotone time acceleration (MTA) algorithm

The relatively higher-order accuracy of the (non-diagonal) FE mass matrix $[M]_e$ in (8) (cf [5]) adversely affects monotonicity of a transient solution process. The goal of the MTA algorithm is to capitalize on the non-diagonal character to accelerate convergence to steady-state.

The process for accomplishing this involves replacing $[M_1]_e$ with $[M_2]^{\mathrm{R}}_e$ for all d, [7]. Note that the condensed mass matrices (17), (18) and (19) are not normalized, i.e. $\sum_{i=1,j=1}^{n} m_{ij}$ does not equal 2^d, which is an attribute of all $[M_1]_e$ matrices for all d. Therefore, introduce the scalar $\phi > 0$ such that the MTA-adjusted matrix $[\tilde{M}_2]^{\mathrm{R}}_e$ for $d = 1, 2$ is the form

$$[\tilde{M}_2]^{\mathrm{R}}_e = \frac{\ell_e}{(12\phi)^d}\begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}, \qquad [\tilde{M}_2]^{\mathrm{R}}_e = \frac{\det_e}{(12\phi)^d}\begin{bmatrix} 9 & -3 & 1 & -3 \\ -3 & 9 & -3 & 1 \\ 1 & -3 & 9 & -3 \\ -3 & 1 & -3 & 9 \end{bmatrix}, \tag{20}$$

where ϕ is a time relaxation parameter. The analogous form for (19) is

$$[\tilde{M}_2]_e^{\mathrm{R}} = \frac{\det_e}{(12\phi)^d} \begin{bmatrix} 27 & -9 & 3 & -9 & -1 & 3 & -9 & 3 \\ -9 & 27 & -9 & 3 & 3 & -1 & 3 & -9 \\ 3 & -9 & 27 & -9 & -9 & 3 & -1 & 3 \\ -9 & 3 & -9 & 27 & 3 & -9 & 3 & -1 \\ -1 & 3 & -9 & 3 & 27 & -9 & 3 & -9 \\ 3 & -1 & 3 & -9 & -9 & 27 & -9 & 3 \\ -9 & 3 & -1 & 3 & 3 & -9 & 27 & -9 \\ 3 & -9 & 3 & -1 & -9 & 3 & -9 & 27 \end{bmatrix}_{(8,8)} \tag{21}$$

and $\phi = 1/6$ normalizes all matrices $[\tilde{M}_2]_e^{\mathrm{R}}$ for $1 \leqslant d \leqslant 3$. Note that $\phi > 1/6$ over-relaxes the solution evolution while $\phi < 1/6$ imposes an under-relaxation process.

A criterion for an alternative definition for ϕ not equal to one-sixth (1/6) can be established from a stability analysis. The assembly of (10a,b) for $d = 1$ at the generic vertex node $x_j = j\ell_e \equiv j\ell$ of a uniform mesh on $\Re^1$ is [8]

$$-\left(\frac{\ell}{12\phi\theta\Delta t} + \frac{\epsilon}{2\ell} + \frac{\mathrm{U}}{2}\right)_e Q_{j-1} + \left(\frac{6\ell}{12\phi\theta\Delta t} + \frac{\epsilon}{\ell}\right)_e Q_j - \left(\frac{\ell}{12\phi\theta\Delta t} + \frac{\epsilon}{2\ell} - \frac{\mathrm{U}}{2}\right)_e Q_{j+1} = b_j \tag{22}$$

For a nonuniform mesh, (22) is replaced as

$$\begin{aligned} &-\left(\frac{\ell_{\mathrm{L}}}{12\phi_{\mathrm{L}}\theta\Delta t} + \frac{\epsilon}{2\ell_{\mathrm{L}}} + \frac{\mathrm{U}}{2}\right) Q_{j-1} \\ &+\left(\frac{3\ell_{\mathrm{L}}}{12\phi_{\mathrm{L}}\theta\Delta t} + \frac{\epsilon}{2\ell_{\mathrm{L}}} + \frac{3\ell_{\mathrm{R}}}{12\phi_{\mathrm{R}}\theta\,\Delta t} + \frac{\epsilon}{2\ell_{\mathrm{R}}}\right) Q_j \\ &-\left(\frac{\ell_{\mathrm{R}}}{12\phi_{\mathrm{R}}\theta\,\Delta t} + \frac{\epsilon}{2\ell_{\mathrm{R}}} - \frac{\mathrm{U}}{2}\right) Q_{j+1} = b_j \end{aligned} \tag{23}$$

where ℓ_{R} and ℓ_{L} are the measures respectively right and left of finite element domains sharing node x_j, and similarly for ϕ_{L} and ϕ_{R}.

The Fourier modal analysis on (22)–(23) predicts algorithm phase accuracy, hence characterization of error-induced perturbation propagation throughout evolution of the fully discrete approximation. The amplification factor G for a single step of an ODE method [8] is defined as

$$G = \frac{Q_j^{t+\Delta t}}{Q_j^t} = \frac{G_{\mathrm{num}}}{G_{\mathrm{den}}} = \sum_{l=0}^{\infty} A_l m^l \tag{24}$$

where A_l is the coefficient set corresponding to lth power of m in the Laurent series expansion on m for the rational function G.

For the analytical solution to (1) in one dimension ($d = 1$), the corresponding amplification factor G_a is

$$\begin{aligned} G_a &= \frac{Q_j^{t+\Delta t}}{Q_j^t} = \mathrm{e}^{-\omega\,\Delta t(i\mathrm{U} + \omega\epsilon)} = \mathrm{e}^{-imC(1 - mC\xi)} \\ &= 1 - imC + \frac{(mC)^2(2i\xi - 1)}{2} - i\frac{(mC)^3(6i\xi - 1)}{3!} + \cdots = \sum_{l=0}^{\infty} A_l m^l \end{aligned} \tag{25}$$

where $i = \sqrt{-1}$, $\omega = 2\pi/\lambda$ is frequency with λ, the wavelength, $m_e = \omega\ell_e$, $\xi = \epsilon/(\mathrm{U}^2\,\Delta t)$ and the Courant number is $C_e = \mathrm{U}\,\Delta t/\ell_e$. In the last line of (25), the rational function G_a is expanded in a Laurent series on m up to and including the third order term.

The amplification factor G quantizes growth or decay of a solution perturbation. For $|G| < 1$ (i.e. $|G_{\mathrm{num}}| < |G_{\mathrm{den}}|$), a solution perturbation decays, hence the algorithm is stable. For $|G| = 1$ (i.e. $|G_{\mathrm{num}}| = |G_{\mathrm{den}}|$), the solution process is neutrally stable and solution perturbations are neither damped nor amplified as evolution proceeds. Finally, if $|G| > 1$ (i.e. $|G_{\mathrm{num}}| > |G_{\mathrm{den}}|$), a solution perturbation will amplify, hence eventually lead to divergence.

For algorithm (22), the numerator in (24) is

$$G_{\text{num}} = \frac{1}{12\phi}\left[6 - \left(e^{-im} + e^{im}\right)\right] - (1-\theta)\xi C^2\left[2 - \left(e^{-im} + e^{im}\right)\right] - (1-\theta)\frac{C}{2}\left(-e^{-im} + e^{im}\right) \tag{26}$$

while the denominator is

$$G_{\text{den}} = \frac{1}{12\phi}\left[6 - \left(e^{-im} + e^{im}\right)\right] + \theta\xi C^2\left[2 - \left(e^{-im} + e^{im}\right)\right] + \theta\frac{C}{2}\left(-e^{-im} + e^{im}\right) \tag{27}$$

Using the identities

$$\cos m = \frac{e^{-im} + e^{im}}{2}, \qquad i\,\sin m = \frac{-e^{-im} + e^{im}}{2} \tag{28}$$

the uniform mesh form for (24) is

$$G = \frac{\frac{1}{6\phi}[3 - \cos m] - (1-\theta)\xi C^2[2 - 2\cos m] - i(1-\theta)C\sin m}{\frac{1}{6\phi}[3 - \cos m] + \theta\xi C^2[2 - 2\cos m] + i\theta C\sin m} \tag{29}$$

Case I ($d = 1, \theta = 0$):

For the explicit Euler procedure, (29) becomes

$$G = \frac{\frac{1}{6\phi}[3 - \cos m] - \xi C^2[2 - 2\cos m] - iC\sin m}{\frac{1}{6\phi}[3 - \cos m]} \tag{30}$$

Using a symbolic manipulation program, e.g. Macsyma [9], via rational simplification, one may determine that

$$|G| = \sqrt{\frac{144C^4(1-\cos m)^2\phi^2\xi^2 - 24C^2(1-\cos m)(3-\cos m)\phi\xi + 36C^2\sin^2 m\phi^2 + (3-\cos m)^2}{(3-\cos m)^2}} \tag{31}$$

Hence, for (30), the condition for stability, $|G_{\text{den}}| \geqslant |G_{\text{num}}|$, leads to

$$2(3-\cos m)(1-\cos m)|\xi| \geqslant 3\phi\left[\sin^2 m + 4C^2\xi^2(1-\cos m)^2\right] \tag{32}$$

hence $1/\phi \geqslant 3\left[\sin^2 m + 4C^2\xi^2(1-\cos m)^2\right]/2(3-\cos m)(1-\cos m)|\xi|$ for stability. For the non-diffusive ($\xi = 0$) form of (1)

$$|G| = \frac{\sqrt{(3-\cos m)^2 + (6\phi C\sin m)^2}}{3-\cos m} > 1 \quad \forall m, \phi, C \tag{33}$$

and the algorithm is unconditionally unstable for any ϕ.

Case II ($d = 1, \theta = 0.5$):

The *trapezoidal* implicit algorithm amplification factor is

$$G = \frac{\frac{1}{6\phi}[3 - \cos m] - \xi C^2[1 - \cos m] - \mathrm{i}\frac{C}{2}\sin m}{\frac{1}{6\phi}[3 - \cos m] + \xi C^2[2 - 2\cos m] + \mathrm{i}\frac{C}{2}\sin m} \tag{34}$$

Using Macsyma, the rational form of $|G|$ is

$$|G| = \sqrt{\frac{36C^4(1-\cos m)^2\phi^2\xi^2 - 12C^2(1-\cos m)(3-\cos m)\phi\xi + 36C^2\sin^2 m\phi^2 + (3-\cos m)^2}{36C^4(1-\cos m)^2\phi^2\xi^2 + 12C^2(1-\cos m)(3-\cos m)\phi\xi + 36C^2\sin^2 m\phi^2 + (3-\cos m)^2}} \tag{35}$$

For (34), the condition for stability, $|G_{\text{den}}| \geqslant |G_{\text{num}}|$, is

$$C^2\phi\xi(3-\cos m)(1-\cos m) \geqslant 0 \tag{36}$$

which accrues for all ϕ. For the non-diffusive ($\xi = 0$) form of (1)

$$|G| = 1 \quad \forall m, \phi, \Delta t \tag{37}$$

hence, the algorithm is unconditionally neutrally stable.

Case III ($d = 1, \theta = 1.0$):

For the backwards Euler time integration, (29) is

$$G = \frac{\frac{1}{6\phi}[3-\cos m]}{\frac{1}{6\phi}[3-\cos m] + \xi C^2[2-2\cos m] + iC\sin m} \tag{38}$$

Here again, using Macsyma, the rational form of $|G|$ is

$$|G| = \sqrt{\frac{(3-\cos m)^2}{144C^4(1-\cos m)^2\phi^2\xi^2 + 24C^2(1-\cos m)(3-\cos m)\phi\xi + 36C^2\sin^2 m\phi^2 + (3-\cos m)^2}} \tag{39}$$

The condition for stability is

$$2(3-\cos m)(1-\cos m)\xi + 3\phi\left[\sin^2 m + 4C^2\xi^2(1-\cos m)^2\right] \geqslant 0 \tag{40}$$

which accrues for all $\phi > 0$. For the non-diffusive form of (1)

$$|G| = \frac{3-\cos m}{\sqrt{(3-\cos m)^2 + (\phi C\sin m)^2}} < 1 \quad \forall m, \phi, \Delta t \tag{41}$$

and the algorithm is unconditionally stable for all ϕ.

An eigenvalue analysis of the system matrix stencil may determine monotonicity of a computational algorithm. For the algorithm algebraic system $[A]\{Q\} = \{b\}$, the solution vector $\{Q\}$ is non-oscillatory if and only if the eigenvalues of $[A]$ are devoid of an imaginary component. Furthermore, if the real part of these eigenvalues is non-negative, then the solution is also stable.

The $(h+1) \times (h+1)$ system matrix associated with (22) for a uniform h finite element discretization yields a tridiagonal matrix structure for which the eigenvalues are solvable in closed form as [7,10]

$$\lambda^j = \frac{6}{12\phi\theta\Delta t} + \frac{\epsilon}{\ell^2} - 2\sqrt{\left(\frac{1}{12\phi\theta\Delta t} + \frac{\epsilon}{2\ell^2} + \frac{U}{2\ell}\right)\left(\frac{1}{12\phi\theta\Delta t} + \frac{\epsilon}{2\ell^2} - \frac{U}{2\ell}\right)}\cos(j\pi\ell),$$

$$\text{where } 1 \leqslant j \leqslant h \tag{42}$$

Hence, λ^j can be represented as a Pick function of the form $\Phi(\zeta) = \alpha(\zeta) + i\beta(\zeta)$, and the ODE system solution will be monotone provided λ^j does not have an imaginary component. This can be assured by the particular choice for the renormalization parameter $\phi > 0$ such that

$$\left(\frac{1}{12\phi\theta\Delta t} + \frac{\epsilon}{2\ell^2} + \frac{U}{2\ell}\right)\left(\frac{1}{12\phi\theta\Delta t} + \frac{\epsilon}{2\ell^2} - \frac{U}{2\ell}\right) \geqslant 0 \tag{43}$$

Table 1
Coefficients A_l of m^l in (25) for different algorithms

Coefficients of m^l	$l=0$	$l=1$	$l=2$	$l=3$
Analytical G_a	1	$-iC$	$\frac{C^2}{2!}(2i\xi - 1)$	$\frac{C^3}{3!}(6\xi + i)$
GWS G^h	1	$-iC$	$-C^2(\xi + \theta)$	$iC^3\theta(2\xi + \theta)$
FV G^h	1	$-iC$	$-C^2(\xi + \theta)$	$iC^3\theta(2\xi + \theta) + \frac{iC}{6}$
MTA G^h	1	$-6iC\phi$	$-6C^2\phi(\xi + 6\phi\theta)$	$iC\phi\left(72C^2\phi\xi\theta + 216C^2\phi^2\theta^2 + \frac{5}{2}\right)$

i.e.

$$\left|\frac{1}{12\phi\theta\Delta t} + \frac{\epsilon}{2\ell^2}\right| \geqslant \left|\frac{U}{2\ell}\right| \tag{44}$$

The condition for ϕ to guarantee (44) is

$$\phi \leqslant \frac{\ell^2}{6\theta\Delta t(U\ell - \epsilon)} \quad \text{or,} \quad \frac{1}{\phi} \geqslant 6\theta(C - \xi C^2) \tag{45}$$

where θ is the ODE implicitness factor. Note that for a pure diffusion equation, $U = 0$, hence in (42) λ^j is always real and the algorithm solution is monotone for all $\phi\theta\xi C^2 \geqslant 0$.

The order-of-accuracy of MTA algorithm may be predicted by comparing the discrete and analytical amplification factor Laurent series. The corresponding Macsyma output for the GWS, diagonalized mass matrix (finite volume, FV) and MTA algorithms is documented in Appendices A and B. Table 1 summarizes the coefficients of m in the Laurent series expansion (25) for the GWS, FV and MTA algorithm solutions. For $l = 1$, it is confirmed the MTA algorithm is not time accurate for $\phi \neq 1/6$ as expected for a time acceleration procedure. For $\phi = 1/6$, the MTA algorithm is 2nd order accurate for $\theta = 0.5$ and for pure convective flow ($\xi = 0$), as are the FV and GWS algorithms, since G^h matches identically with G_a up to the m^2 terms in Laurent series expansion of m.

5. Discussion and results

5.1. 1-D unsteady heat conduction

Unsteady conduction in an axisymmetric cylinder is described by the solution to

$$\mathcal{L}(q) = \frac{\partial q}{\partial t} - \frac{1}{r}\frac{\partial}{\partial r}\left(r\epsilon\frac{\partial q}{\partial r}\right) = 0 \quad r_1 < r < r_2 \tag{46a}$$

$$\epsilon\nabla q \cdot \hat{\boldsymbol{n}} - \alpha q_n = 0 \qquad r = r_1 \tag{46b}$$

$$q(r_2) = q_b \qquad r = r_2 \tag{46c}$$

The analytical solution to (46a,c) is

$$q(r,t) = Ae^{-\lambda t + \left(-\epsilon \pm \sqrt{\epsilon^2 - 4r\epsilon\lambda}/2\epsilon\right)} \tag{47}$$

where A and λ are constants dependent on the data in (46a), i.e. $\epsilon, \alpha, q_n, r_1, r_2, t$. While (47) is smooth for all $t > t_0$, dependent on the time integration process, approximate solution monotonicity depends on the initial condition and the data.

For the example problem, the initial condition is $q(r,0) = q_0(r) = 0$, $1 < r < 2$ in (46a) and the flux (Robin) and Dirichlet boundary conditions are $\alpha = 1$, $q_n = 1$ and $q_b = 0$ in (46b,c). Summaries of early-time solution non-monotonicity are presented in Fig. 2. For the initial timestep of 0.0001, the temporal ($\theta = 0.5$) solution evolution for both $k = 1$ and $k = 2$ Lagrange GWS (no ϕ) method is distinctly non-monotone on this meshing, (Fig. 2a and b), although nodal accuracy for surface temperature (at $r = r_1 = 1$) is acceptable for engineering purposes. In comparison, for the initial timestep of 0.0001, the FV (also no ϕ) solution on the same meshing is truly monotone and acceptably time-accurate, (Fig. 2c).

Table 2
Effect of ϕ on a uniform 9-node mesh

Algorithm	k	ϕ	tstep	$Q_{\min} \times 10^2$	$Q_{\max} \times 10^2$	Evolution	CPU time (s)
GWS	1		1820	0.0	58.0 (steady)	Non-monotone	200
GWS	2		1820	0.0	58.1 (steady)	Non-monotone	215
FV	1		1845	0.0	58.1 (steady)	Monotone	210
FV	1		453	0.0	58.1 (steady)	Non-monotone	52
MTA	1	$\frac{1}{6}$	990	0.0	58.1 (steady)	Monotone	110
MTA	1	5	82	0.0	58.1 (steady)	Monotone	10

The companion $\phi = 1/6$ MTA solution procedure, (Fig. 2e), generates a monotone and second-order time-accurate solution on this meshing and initial timestep.

For efficiency, over-relaxing the temporal ($\theta = 0.5$) solution process via selecting $\phi \geqslant 1/6$ can facilitate evolutionary monotone attainment of the steady state using a much smaller number of integration timesteps, Fig. 3. As presented in Table 2, for GWS $k = 1$, the trapezoidal ($\theta = 0.5$ in (8)) rule time integration scheme solution becomes steady, i.e. insensitive to time up to 4th decimal, after 1820 timesteps, with the initial timestep of 0.0001, yielding the data graphed in Fig. 2a and b. In distinction, for the MTA procedure, (Fig. 2f), with the same initial Δt and θ, only 82 timesteps are required with $\phi = 5$ to monotonely reach a time station with solution independent of timestep. For a large initial timestep ($\geqslant 0.01$), the FV algorithm solution evolution is oscillatory and, for an initial timestep of 0.05, solution approaches steady state after 453 timesteps, (Fig. 2d). Both GWS and FV algorithms take progressively increasing numbers of steps as the initial timestep increases. For example, for an initial timestep of 0.1, the GWS algorithm takes 2472 timesteps to reach steady state. The CPU time documented in Table 2 are for Sun Sparc station 20 with Solaris 2 operating system. Note in Fig. 3 that as the time relaxation parameter increases the CPU time requirement and the total number iterations for steady state decreases sharply. However, after reaching an optimum value ($\phi \approx 8$) the CPU time and total iteration count slowly increase with ϕ.

5.2. *Quasi-1D compressible shocked-flow verification*

The quasi-one dimensional inviscid (Euler) form of (1) is

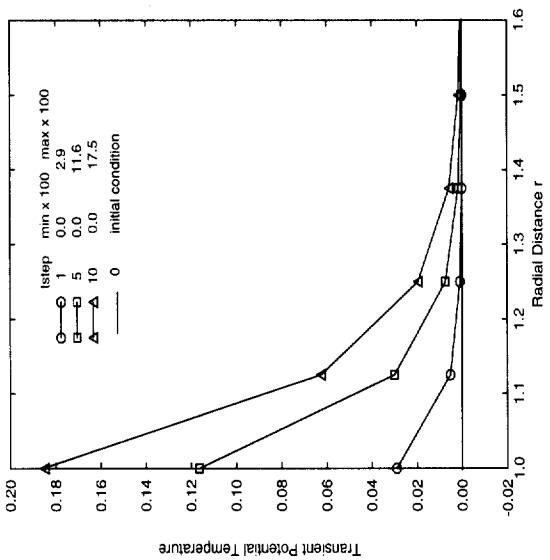
$$\mathcal{L}(q) = \frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} - s = 0 \tag{48a}$$

$$q = \begin{pmatrix} \rho \\ m \\ E \end{pmatrix}, \qquad f = \begin{pmatrix} m \\ \dfrac{m^2}{\rho} + p \\ \dfrac{(E+p)m}{\rho} \end{pmatrix} \tag{48b}$$

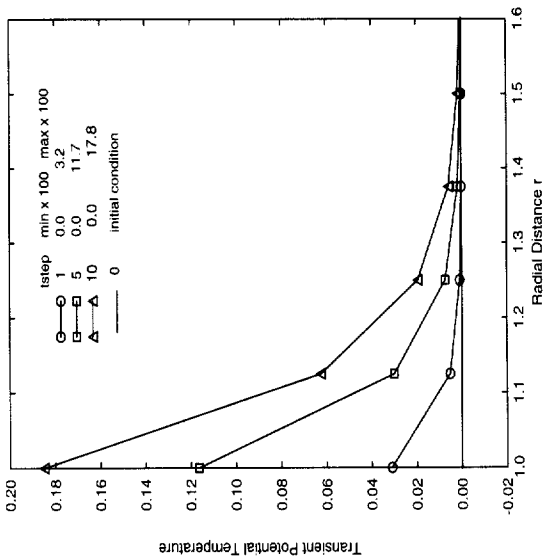
$$s = \begin{pmatrix} -m\dfrac{\mathrm{d}\,\ln A(x)}{\mathrm{d}x} \\ -\dfrac{m^2}{\rho}\dfrac{\mathrm{d}\,\ln A(x)}{\mathrm{d}x} \\ -\dfrac{(E+p)m}{\rho}\dfrac{\mathrm{d}\,\ln A(x)}{\mathrm{d}x} \end{pmatrix} \tag{48c}$$

$$p = (\gamma - 1)\left[E - \frac{m^2}{2\rho}\right] \tag{48d}$$

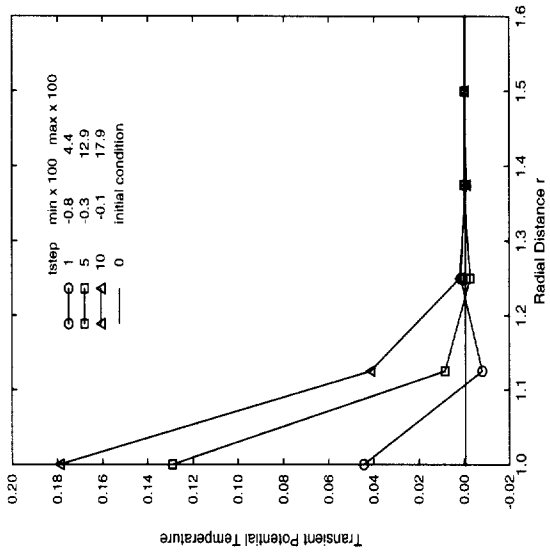
where ρ is density, m is momentum, E is volume specific total energy, p is pressure, γ is the ratio of specific heats and $A(x)$ is the nozzle cross-sectional area distribution. For a perfect gas, (48a–c) is closed



(a) GWS solution, no ϕ, $k = 1$



(c) FV solution, no ϕ, $k = 1$



(b) GWS solution, no ϕ, $k = 2$

(e) MTA solution, $\phi = 1/6$

(d) FV solution, no ϕ, $k = 1$ (to steady state)

(f) MTA solution, $\phi = 5$ (to steady state)

Fig. 2. Solution monotonicity and effect of ϕ for the transient evolution.

by the polytropic equation of state (48d) and for velocity definition $u \equiv m/\rho$, the volume specific total energy E is

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho u^2 \tag{49}$$

The de Laval nozzle geometry, Fig. 4, and test problem specification of [11] is used for the verification test where

$$A(x) = \begin{cases} 1.75 - 0.75\cos[2(x - 0.5)\pi] , & 0.0 \leqslant x \leqslant 0.5 \\ 1.25 - 0.25\cos[2(x - 0.5)\pi] , & 0.5 \leqslant x \leqslant 1.0 \end{cases} \tag{50}$$

The non-dimensional initial condition for q is derived from the sonic, shock-free isentropic solution for (50). The boundary conditions are $\rho(0,t) = 1.0 = \rho_{\text{in}}$, $E(0,t) = 1.0 = E_{\text{in}}$, and $p(1,t) = p_{\text{out}} = 0.909$. The corresponding isentropic inlet Mach number is $\text{Ma}_{\text{in}} = 0.2395$.

The verification steady state condition corresponds to an impulsive change from isentropic flow by decreasing the exit pressure p_{out} from 0.909 to 0.84. The resultant expansion wave propagates upstream to the throat, hence triggers formation of a (normal) shock which moves downstream to $x = 0.65$ with the analytical shock Mach number, $\text{Ma}_s = 1.40$.

The nozzle domain is uniformly discretized into 100 elements. The linear Lagrange basis ($k = 1$) finite element form of the Taylor weak statement (TWS) algorithm, cf. [1], is employed, with $\beta = 0.2\{1,1,1\}$ the dissipation level for all tests. Table 3 documents the TWS, FV and MTA steady state solution performance, where CPU time is for a Sun Sparc station 10. The TWS solution took 68 constant timesteps of $\Delta t = 0.02$ to converge to (10^{-4}) at the first iteration, which is accepted as steady state. The FV solution similarly converged in 69 constant timesteps ($\Delta t = 0.02$). Both solutions capture a shock, smeared over three computational cells centered at $x = 0.65$, with upstream (shock) Mach number $\text{Ma}_s = 1.377$. A larger initial timestep may be used to achieve faster convergence in TWS and FV algorithms. For $\Delta t = 0.1$, TWS and FV algorithms take 30 steps (66 iterations) to reach steady state. However, convergence becomes increasingly difficult for $\Delta t > 0.1$, and for $\Delta t > 0.5$ both the TWS and FV Newton solution processes diverge.

In comparison, the MTA algorithm with $\phi = 6$ produces a steady state solution nominally identical to the TWS and FV solutions in 10 steps (25 iterations). The MTA solution evolution remains essentially non-oscillatory (ENO), and Fig. 5 plots the associated initial condition and steady state solution Mach number, density and pressure distributions along the de Laval nozzle axis.

5.3. 2D compressible flow through a converging duct

The two-dimensional inviscid (Euler) form of (48a)–(49) is

$$\mathcal{L}(q) = \frac{\partial q}{\partial t} + \frac{\partial f_j}{\partial x_j} = 0, \quad 1 \leqslant j \leqslant 2 \tag{51a}$$

Table 3
Effect of ϕ on a uniform 101-node mesh compressible flow solution

Algorithm	k	ϕ	Iterations	Shock location x	Evolution	CPU time (s)
TWS	1		162	0.65	Monotone	78
TWS	1		65	0.65	Non-monotone	32
FV	1		164	0.65	Monotone	79
FV	1		66	0.65	Non-monotone	32
MTA	1	$\frac{1}{6}$	160	0.65	Monotone	76
MTA	1	6	25	0.65	Monotone	12

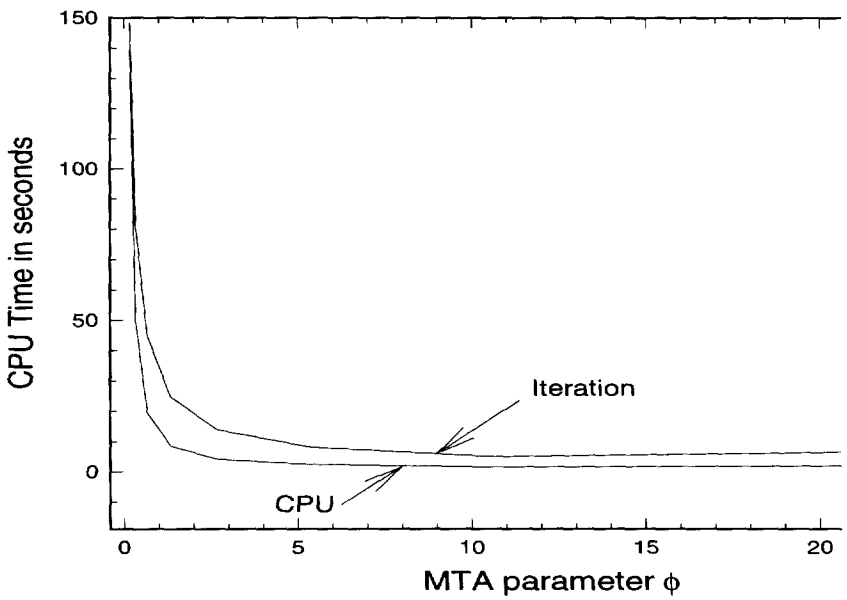


Fig. 3. CPU time and required iterations for steady state vs. ϕ.

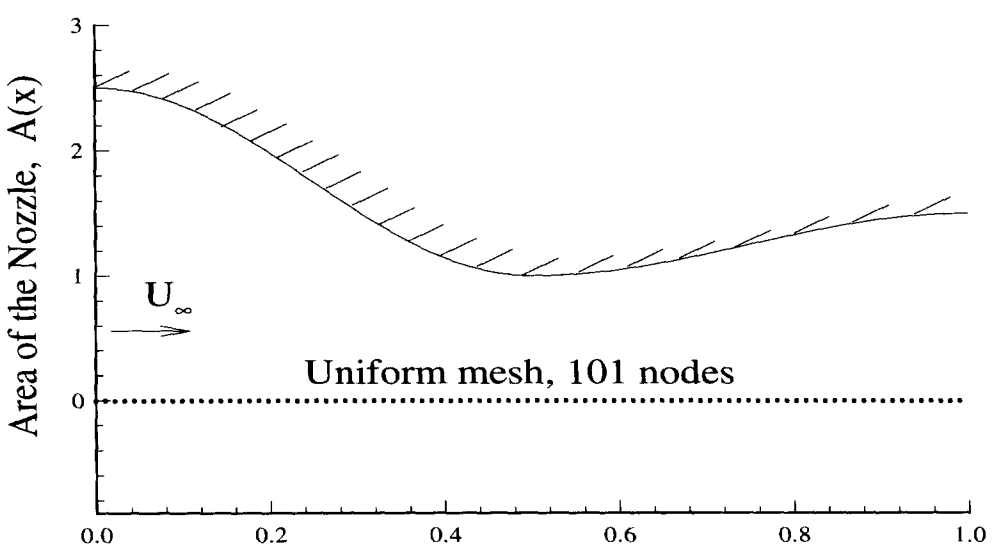
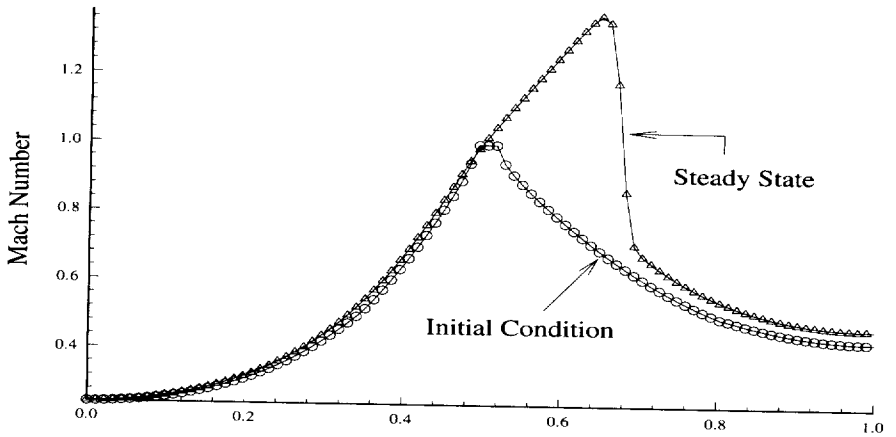
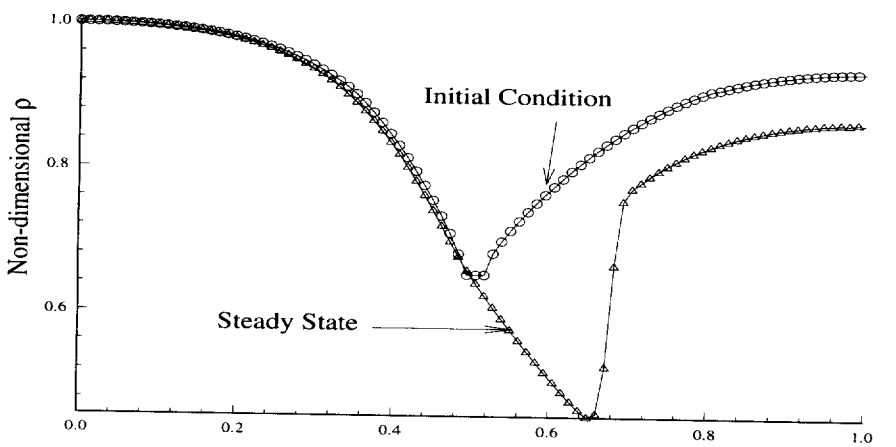


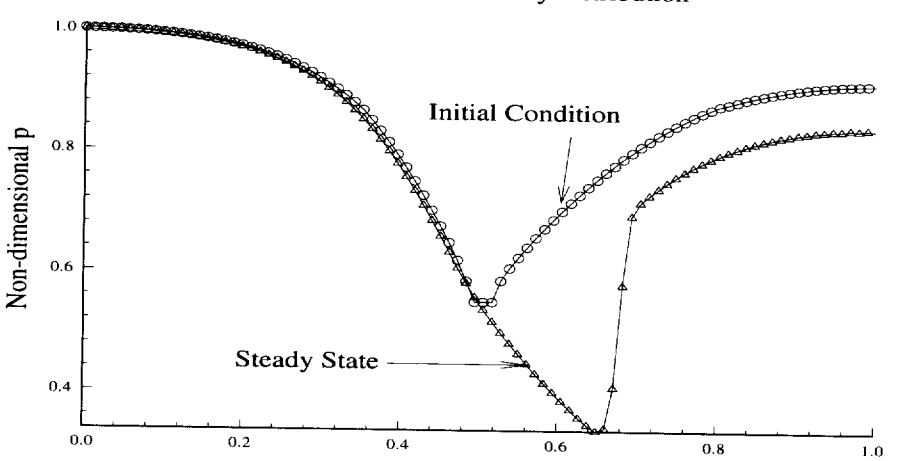
Fig. 4. de Laval nozzle geometry.



a) Solution Mach number



b) Solution density distribution



c) Solution pressure distribution

Fig. 5. Steady state solution for compressible flow through de Laval nozzle.

Table 4
Effect of ϕ on a 2-D compressible flow solution

Algorithm	k	ϕ	Iterations	Normalized CPU time
TWS	1		638	100
MTA	1	$\frac{1}{6}$	620	98.7
MTA	1	$\frac{1}{3\sqrt{2}}$	284	49.9
MTA	1	$\frac{1}{3}$	189	33.2
MTA	1	$\frac{2}{3}$	172	30.4
MTA	1	$\frac{4}{3}$	181	31.7

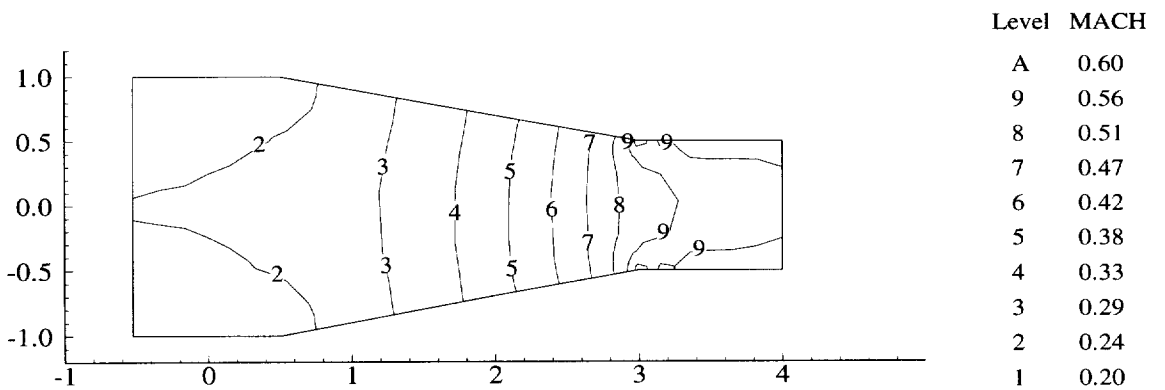


Fig. 6. Steady state solution Mach number isolines for flow through a converging duct.

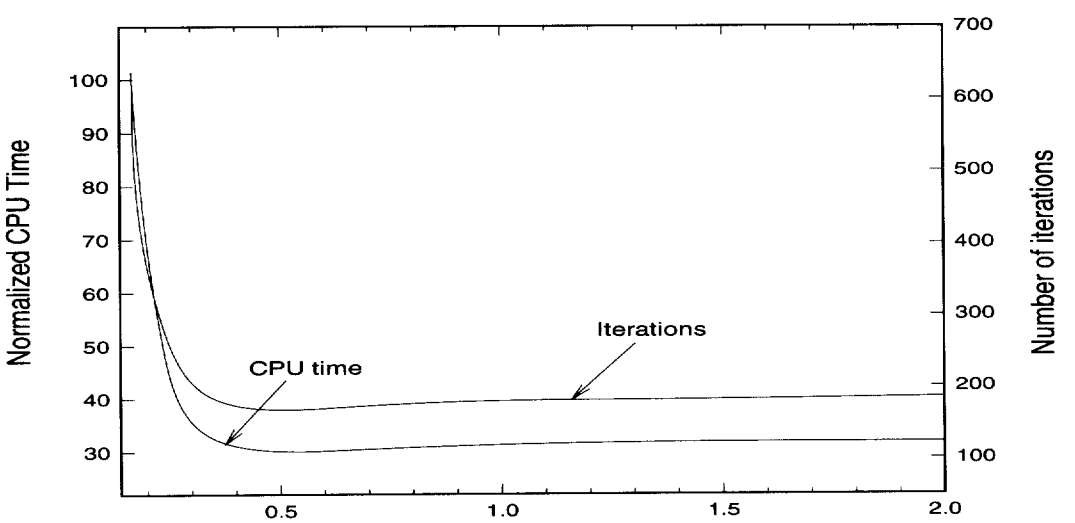


Fig. 7. CPU time and iteration count for steady state vs. ϕ in 2-D.

$$q = \begin{pmatrix} \rho \\ m_i \\ E \end{pmatrix}, \qquad f_j = \begin{pmatrix} m_j \\ \dfrac{m_i m_j}{\rho} + p\delta_{ij} \\ \dfrac{(E+p)m_j}{\rho} \end{pmatrix} \tag{51b}$$

with $s = 0$, velocity definition $u_j = m_j/\rho$, and pressure (48d). The non-dimensional initial conditions for q is homogeneous zero, except for inflow, whereat the boundary conditions are $\rho(0,t) = 1.0 = \rho_{\text{in}}$, $E(0,t) = 1.0 = E_{\text{in}}$, and at outflow where $p(1,t) = p_{\text{out}} = 0.84$. The corresponding isentropic flow inlet Mach number is $\text{Ma}_{\text{in}} = 0.2395$.

Table 4 compares the TWS, with $\beta_q = 0.1\{1,1,1\}$, and MTA steady state Euler solutions for flow in a converging duct. The TWS solution took 400 variable timesteps requiring 638 iterations to converge to 10^{-3} at the first iteration (defined as steady state). The CPU time comparisons are normalized by the solution time taken for this TWS execution. In comparison, the MTA algorithm, with $\phi \geqslant 1/6$, produces a steady state solution in a lesser number of iterations and the solution evolution process is ENO. Fig. 6 graphs the Mach number isolines for the TWS and MTA solutions, which are essentially identical at steady state. The trend of iterations and normalized CPU times for Sun Sparc station 10 are plotted in Fig. 7, and an optimum $\phi \approx 2/3$ exists.

6. Conclusions

A weak statement monotone time acceleration technique based on local static condensation of a Lagrange $k > 1$ mass matrix has been developed for generating faster convergence to steady state for CFD algorithm constructions employing an unsteady formulation. The presented MTA theory analysis is complete for $d = 1$, has been verified extensible to $d = 2$ and is very easy to incorporate in any dimension. In 1-D, the suitable choice of the time acceleration parameter, ϕ rapidly accelerates the Newton unsteady solution process to steady state in less than 5% (for heat conduction) to 15% (for compressible flow) time to that of the standard Galerkin (and Taylor) WS. However, in higher dimensions, the MTA solution process becomes sensitive to large ϕ, hence an optimal value is computationally verified to exist. The associated theoretical analysis should be developed.

Acknowledgments

This work has been partially supported by grant MSS-9015912 from the National Science Foundation. The authors acknowledge the support provided by the UT CFD Laboratory and Computational Mechanics Corporation. Mr. Joseph Orzechowski has provided thoughtful discussions and significant programming help in this reported work.

Appendix A. Amplification factors for the MTA algorithm

Nomenclatures for the Macsyma output:
theta = θ, c = $C = \mathrm{U}\Delta t/\ell$, xi = $\xi = \epsilon/\mathrm{U}^2\Delta t$ and g = G. t0 = $|G|$ for $\theta = 0$. t05 = $|G|$ for $\theta = 0.5$. t10 = $|G|$ for $\theta = 1$.

Condition for $G \leqslant 1$ iff *cond* $\geqslant 0$.

```
(c3) theta:0$
(c4) n1:(3-cos(m))/(6*phi)$
(c5) n2:2*xi*c^2*(1-cos(m))$
(c6) n3:c*sin(m)$
(c7) d1:n1$
(c8) gr:(n1-n2)/d1$
(c9) gi:n3/d1$
(c10) g:gr-%i*gi$
(c11) absg:abs(g)$
(c12) t0:ratsimp(absg)$
(c13) t0sq:t0*t0$
(c14) t0num:num(t0sq)$
(c15) t0denom:denom(t0sq)$
(c16) cond:t0denom-t0num$
(c17) t0cond:ratsimp(cond)$
(c18) theta:0.5$
(c19) n1:(3-cos(m))/(6*phi)$
(c20) n2:xi*c^2*(1-cos(m))$
(c21) n3:c*sin(m)$
(c22) d1:n1$
(c23) d2:n2$
(c24) d3:n3$
(c25) g:(n1-n2-%i*n3)/(d1+d2+%i*d3)$
(c26) absg:abs(g)$
(c27) t05:ratsimp(absg)$
(c28) t05sq:t05*t05$
(c29) t05num:num(t05sq)$
(c30) t05denom:denom(t05sq)$
(c31) cond:t05denom-t05num$
(c32) t05cond:ratsimp(cond)$
(c33) theta:1.0$
(c34) n1:(3-cos(m))/(6*phi)$
(c35) d1:n1$
(c36) d2:2*xi*c^2*(1-cos(m))$
(c37) d3:c*sin(m)$
(c38) g:n1/(d1+d2+%i*d3)$
(c39) absg:abs(g)$
(c40) t10:ratsimp(absg)$
(c41) t10sq:t10*t10$
(c42) t10num:num(t10sq)$
(c43) t10denom:denom(t10sq)$
(c44) cond:t10denom-t10num$
(c45) t10cond:ratsimp(cond)$
(c46) t0;
                  4     2           4                 4      2    2
```

```
(d46) sqrt((144 c  cos (m) - 288 c  cos(m) + 144 c ) phi  xi
          2    2          2             2               2    2       2
 + (- 24 c  cos (m) + 96 c  cos(m) - 72 c ) phi xi + 36 c  sin (m) phi
      2                           2
 + cos (m) - 6 cos(m) + 9)/sqrt(cos (m) - 6 cos(m) + 9)
(c47) t05;
                  4    2          4              4     2   2
(d47) sqrt(((36 c  cos (m) - 72 c  cos(m) + 36 c ) phi  xi
          2    2          2             2               2    2       2
 + (- 12 c  cos (m) + 48 c  cos(m) - 36 c ) phi xi + 36 c  sin (m) phi
      2                           4    2          4              4     2   2
 + cos (m) - 6 cos(m) + 9)/((36 c  cos (m) - 72 c  cos(m) + 36 c ) phi  xi
        2    2          2              2               2    2       2      2
 + (12 c  cos (m) - 48 c  cos(m) + 36 c ) phi xi + 36 c  sin (m) phi  + cos (m)
 - 6 cos(m) + 9))
(c48) t10;
              2                                  4    2           4
(d48) sqrt(cos (m) - 6 cos(m) + 9)/sqrt((144 c  cos (m) - 288 c  cos(m)
        4     2   2         2    2          2              2
 + 144 c ) phi  xi  + (24 c  cos (m) - 96 c  cos(m) + 72 c ) phi xi
       2    2       2      2
 + 36 c  sin (m) phi  + cos (m) - 6 cos(m) + 9)
(c49) t0cond;
             4    2           4               4     2   2
(d49) (- 144 c  cos (m) + 288 c  cos(m) - 144 c ) phi  xi
        2    2          2              2               2    2       2
  + (24 c  cos (m) - 96 c  cos(m) + 72 c ) phi xi - 36 c  sin (m) phi
(c50) t05cond;
                     2    2          2              2
(d50)           (24 c  cos (m) - 96 c  cos(m) + 72 c ) phi xi
(c51) t10cond;
            4    2           4               4     2   2
(d51) (144 c  cos (m) - 288 c  cos(m) + 144 c ) phi  xi
        2    2          2              2               2    2       2
  + (24 c  cos (m) - 96 c  cos(m) + 72 c ) phi xi + 36 c  sin (m) phi
(c52) xi:0$
(c53) t0m:ev(t0,eval);
                       2    2       2      2
              sqrt(36 c  sin (m) phi  + cos (m) - 6 cos(m) + 9)
(d53)         ------------------------------------------------
                                 2
                        sqrt(cos (m) - 6 cos(m) + 9)
(c54) cond:ev(t0cond,eval);
                                   2    2       2
(d54)                       - 36 c  sin (m) phi
(c55) t05m:ev(t05,eval);
(d55)                                1
(c56) cond:ev(t05cond,eval);
(d56)                                0
(c57) t10m:ev(t10,eval);
                                 2
                        sqrt(cos (m) - 6 cos(m) + 9)
(d57)         ------------------------------------------------
```

```
                        2    2       2      2
                sqrt(36 c  sin (m) phi  + cos (m) - 6 cos(m) + 9)
(c58) cond:ev(t10cond,eval);
                                     2    2       2
(d58)                          36 c  sin (m) phi
(c59) xi:1$
(c60) t0m:ev(t0,eval);
                2    2       2         4    2           4               4     2
(d60) sqrt(36 c  sin (m) phi  + (144 c  cos (m) - 288 c  cos(m) + 144 c ) phi
          2    2          2              2           2
 + (- 24 c  cos (m) + 96 c  cos(m) - 72 c ) phi + cos (m) - 6 cos(m) + 9)
                                    2
                          /sqrt(cos (m) - 6 cos(m) + 9)
(c61) cond:ev(t0cond,eval);
           2    2       2           4    2           4               4     2
(d61) - 36 c  sin (m) phi  + (- 144 c  cos (m) + 288 c  cos(m) - 144 c ) phi
                       2    2          2              2
                + (24 c  cos (m) - 96 c  cos(m) + 72 c ) phi
(c62) t05m:ev(t05,eval);
                 2    2       2        4    2          4               4     2
(d62) sqrt((36 c  sin (m) phi  + (36 c  cos (m) - 72 c  cos(m) + 36 c ) phi
          2    2          2              2           2
 + (- 12 c  cos (m) + 48 c  cos(m) - 36 c ) phi + cos (m) - 6 cos(m) + 9)
     2    2       2        4    2          4              4     2
/(36 c  sin (m) phi  + (36 c  cos (m) - 72 c  cos(m) + 36 c ) phi
        2    2          2              2           2
 + (12 c  cos (m) - 48 c  cos(m) + 36 c ) phi + cos (m) - 6 cos(m) + 9))
(c63) cond:ev(t05cond,eval);
                       2    2          2              2
(d63)               (24 c  cos (m) - 96 c  cos(m) + 72 c ) phi
(c64) t10m:ev(t10,eval);
              2                                2    2       2
(d64) sqrt(cos (m) - 6 cos(m) + 9)/sqrt(36 c  sin (m) phi
         4    2           4               4     2
 + (144 c  cos (m) - 288 c  cos(m) + 144 c ) phi
        2    2          2              2           2
 + (24 c  cos (m) - 96 c  cos(m) + 72 c ) phi + cos (m) - 6 cos(m) + 9)
(c65) cond:ev(t10cond,eval);
          2    2       2         4    2           4               4     2
(d65) 36 c  sin (m) phi  + (144 c  cos (m) - 288 c  cos(m) + 144 c ) phi
                       2    2          2              2
                + (24 c  cos (m) - 96 c  cos(m) + 72 c ) phi
(c66) quit();
```

Appendix B. Time accuracy for GWS, FV and MTA algorithms

Algorithm time accuracy is estimated by comparing the m^l coefficient terms (A_l in (24)) with the analytical solution amplification factor (25). The mass matrices used for GWS, FV and MTA algorithms are

$$\text{GWS } [M_1]_e = \frac{\ell_e}{3}\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}_{(2,2)}$$

$$\text{FV } [M_1]_e = \frac{\ell_e}{2}\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}_{(2,2)}$$

$$\text{MTA } [M_2]_e^R = \frac{\ell_e}{12\phi}\begin{bmatrix}3 & -1\\ -1 & 3\end{bmatrix}_{(2,2)}$$

```
(c2) n1:(4+2*cos(m))/6$
(c3) n2:2*xi*c^2*(1-t)*(1-cos(m))$
(c4) n3:(1-t)*c*sin(m)$
(c5) d1:n1$
(c6) d2:2*xi*c^2*t*(1-cos(m))$
(c7) d3:t*c*sin(m)$
(c8) gwsg:(n1-n2-%i*n3)/(d1+d2+%i*d3)$
(c9) gwst:taylor(gwsg,m,0,6)$
(c10) n1:(3-cos(m))/(12*phi)$
(c11) n2:2*xi*c^2*(1-t)*(1-cos(m))$
(c12) n3:(1-t)*c*sin(m)$
(c13) d1:n1$
(c14) d2:2*xi*c^2*t*(1-cos(m))$
(c15) d3:t*c*sin(m)$
(c16) mtag:(n1-n2-%i*n3)/(d1+d2+%i*d3)$
(c17) mtat:taylor(mtag,m,0,6)$
(c18) anlg:%e^-(%i*m*c*(1-m*c*xi))$
(c19) angt:taylor(anlg,m,0,6)$
(c20) term1g:coeff(gwst,m,1);
(d20)/r/                                  - %i c
(c21) term1m:coeff(mtat,m,1);
(d21)/r/                                - 6 %i c phi
(c22) term1a:coeff(angt,m,1);
(d22)/r/                                  - %i c
(c23) solve([term1m-term1a],[phi]);
                                             1
(d23)                                   [phi = -]
                                             6
(c24) term2g:coeff(gwst,m,2);
                                          2       2
(d24)/r/                                - c  xi - c  t
(c25) term2m:coeff(mtat,m,2);
                                    2               2    2
(d25)/r/                    - 6 c  phi xi - 36 c  phi  t
(c26) term2a:coeff(angt,m,2);
                                          2       2
                                   2 %i c  xi - c
(d26)/r/                           ---------------
                                         2
(c28) term3g:coeff(gwst,m,3);
                                     3            3  2
(d28)/r/                      2 %i c  t xi + %i c  t
(c29) term3m:coeff(mtat,m,3);
                   3    2                3    3  2
          144 %i c  phi  t xi + 432 %i c  phi  t  + 5 %i c phi
```

```
(d29)/r/   ------------------------------------------------
                                    2
(c30) term3a:coeff(angt,m,3);
                                   3          3
                                6 c  xi + %i c
(d30)/r/                        ---------------
                                       6

(c31) xi:0$
(c32) t:0.0$
(c33) gwst:ev(gwsts,eval)$
(c34) mtat:ev(mtats,eval)$
(c35) term1g:coeff(gwst,m,1);
(d35)/r/                             - %i c
(c36) term1m:coeff(mtat,m,1);
(d36)/r/                          - 6 %i c phi
(c37) term1a:coeff(angt,m,1);
(d37)/r/                             - %i c
(c38) solve([term1m-term1a],[phi]);
                                          1
(d38)                                [phi = -]
                                          6
(c39) term2g:coeff(gwst,m,2);
(d39)/r/                               0
(c40) term2m:coeff(mtat,m,2);
(d40)/r/                               0
(c41) term2a:coeff(angt,m,2);
                                         2
                                        c
(d41)/r/                              - --
                                        2
(c42) term3g:coeff(gwst,m,3);
(d42)/r/                               0
(c43) term3m:coeff(mtat,m,3);
                                   5 %i c phi
(d43)/r/                           ----------
                                       2
(c44) term3a:coeff(angt,m,3);

                                         3
                                      %i c
(d44)/r/                              -----
                                        6

(c45) t:0.5$
(c46) gwst:ev(gwsts,eval)$
(c47) mtat:ev(mtats,eval)$
(c48) term1g:coeff(gwst,m,1);
(d48)/r/                             - %i c
(c49) term1m:coeff(mtat,m,1);
(d49)/r/                          - 6 %i c phi
(c50) term1a:coeff(angt,m,1);
(d50)/r/                             - %i c
```

```
(c51) solve([term1m-term1a],[phi]);
                                        1
(d51)                             [phi = -]
                                        6
(c52) term2g:coeff(gwst,m,2);
                                       2
                                      c
(d52)/r/                            - --
                                      2
(c53) term2m:coeff(mtat,m,2);
                                        2    2
(d53)/r/                          - 18 c  phi
(c54) term2a:coeff(angt,m,2);
                                       2
                                      c
(d54)/r/                            - --
                                      2
(c55) term3g:coeff(gwst,m,3);
                                        3
                                    %i c
(d55)/r/                            -----
                                      4
(c56) term3m:coeff(mtat,m,3);
                                  3    3
(d56)/r/                (108 %i c  phi  + 5 %i c phi)/2
(c57) term3a:coeff(angt,m,3);
                                        3
                                    %i c
(d57)/r/                            -----
                                      6
(c58) t:1.0$
(c59) gwst:ev(gwsts,eval)$
(c60) mtat:ev(mtats,eval)$
(c60) term1g:coeff(gwst,m,1);
(d60)/r/                            - %i c
(c60) term1m:coeff(mtat,m,1);
(d60)/r/                          - 6 %i c phi
(c61) term1a:coeff(angt,m,1);
(d61)/r/                            - %i c
(c62) solve([term1m-term1a],[phi]);
                                        1
(d62)                             [phi = -]
                                        6
(c63) term2g:coeff(gwst,m,2);
                                       2
(d63)/r/                            - c
(c64) term2m:coeff(mtat,m,2);
                                        2    2
(d64)/r/                          - 36 c  phi
(c65) term2a:coeff(angt,m,2);
                                       2
                                      c
(d65)/r/                            - --
```

```
                                        2
(c66) term3g:coeff(gwst,m,3);
                                         3
(d66)/r/                                %i c
(c67) term3m:coeff(mtat,m,3);
                                   3    3
(d67)/r/                  (432 %i c  phi  + 5 %i c phi)/2
(c68) term3a:coeff(angt,m,3);
                                         3
                                      %i c
(d68)/r/                              -----
                                        6
(c2) n1:1$
(c3) n2:2*xi*c^2*(1-t)*(1-cos(m))$
(c4) n3:(1-t)*c*sin(m)$
(c5) d1:n1$
(c6) d2:2*xi*c^2*t*(1-cos(m))$
(c7) d3:t*c*sin(m)$
(c8) fvg:(n1-n2-%i*n3)/(d1+d2+%i*d3)$
(c9) fvts:taylor(fvg,m,0,6)$
(c10) term1fv:coeff(fvts,m,1);
(d10)/r/                              - %i c
(c11) term2fv:coeff(fvts,m,2);
                                     2       2
(d11)/r/                          - c  xi - c  t
(c12) term3fv:coeff(fvts,m,3);
                              3              3  2
                     12 %i c  t xi + 6 %i c  t  + %i c
(d12)/r/             ---------------------------------
                                        6
(c13) xi:0$
(c14) t:0$
(c15) term1f:ev(term1fv,eval);
(d15)/r/                              - %i c
(c16) term2f:ev(term2fv,eval);
(d16)/r/                                0
(c17) term3f:ev(term3fv,eval);
                                      %i c
(d17)/r/                              ----
                                        6
(c18) t:0.5$
(c19) term1f:ev(term1fv,eval);
(d19)/r/                              - %i c
(c20) term2f:ev(term2fv,eval);
                                        2
                                       c
(d20)/r/                             - --
                                       2
(c21) term3f:ev(term3fv,eval);
                                     3
                                3 %i c  + 2 %i c
(d21)/r/                        ----------------
                                       12
```

```
(c22) t:1.0$
(c23) term1f:ev(term1fv,eval);
(d23)/r/                                - %i c
(c24) term2f:ev(term2fv,eval);
                                           2
(d24)/r/                                 - c
(c25) term3f:ev(term3fv,eval);
                                           3
                                  6 %i c  + %i c
(d25)/r/                          --------------
                                         6
(c26) quit();
```

References

[1] A.J. Baker and J.W. Kim, Int. J. Numer. Methods Fluids 7 (1987) 489–520.

[2] C. Hirsch, Numerical Computation of Internal and External Flows 2, Computational Methods for Inviscid and Viscous Flows (John Wiley & Sons Ltd., West Sussex, UK, 1992).

[3] E.L. Wachspress, Iterative Solution of Elliptic Systems (Prentice Hall, Englewood Cliffs, NJ, 1966).

[4] J. Chung and G.L. Cole, NASA Technical Memorandum 107194, ICOMP-96-05 (1996)

[5] A.J. Baker, Finite Element Computational Fluid Dynamics (Hemisphere Publ. Co., New York, 1983).

[6] O.C. Zienkiewicz and A.W. Craig, The Mathematical Basis of Finite Element Methods, D.F. Griffiths, ed. (Clarendon Press, Oxford, 1984).

[7] S. Roy, On Improved Methods for Monotone CFD Solution Accuracy, Ph.D. Dissertation, The Univ. of Tennessee, Knoxville, 1994.

[8] S. Roy and A.J. Baker, A weak statement perturbation CFD algorithm with high-order phase accuracy for hyperbolic problems, Comput. Methods Appl. Mech. Engrg. 131 (1996) 209–232.

[9] Macsyma™ Reference Manual, 1988, Version 13, Symbolics, Inc., Cambridge, MA.

[10] H.O. Kreiss and J. Lorenz, Initial-Boundary Value Problems and the Navier–Stokes Equations (Academic Press, New York, 1989).

[11] M.S. Liou and B. van Lerr, Choice of implicit and explicit operators for the upwind differencing method, Tech. Paper AIAA 88-0624, 26th Aerospace Meeting, Reno, Nevada, 1988.

[12] F.M. White, Fluid Mechanics (McGraw Hill, New York, 1979).