A Finite-Element Approach for Modeling Inviscid and Viscous Compressible Flows using Prismatic Grids

S. A. Pandya *
NASA Ames Research Center
Moffett Field, CA 94035

M. Hafez †
University of California, Davis
Davis, CA 95616

Abstract

The Galerkin finite-element method is used to solve the Euler and Navier–Stokes equations on prismatic meshes. It is shown that the prismatic grid is advantageous for correctly and efficiently capturing the boundary layers in high Reynolds number flows. It can be captured accurately because of the ability to cluster grid points normal to the body. The efficiency derives from the implicit treatment of the normal direction. To treat the normal direction implicitly, a semi–implicit Runge–Kutta time stepping scheme is developed. The semi-implicit algorithm is validated on simple geometries for inviscid and viscous flows and its convergence history is compared to that of the explicit Runge-Kutta scheme. The semi–implicit scheme is shown to be a factor of 3 to 4 faster in terms of CPU time to convergence.

Introduction

Many methods have been attempted for both the generation of prismatic grids and solution of flows using prismatic grids. Prismatic meshes have been generated by optimization methods [1], by solution of hyperbolic PDEs [2], and by algebraic methods [3]. Unstructured tetrahedral meshes that have been cut from prisms have also been created using an algebraic approach developed by Lohner [4] and later modified by Marcum [5] as well as another approach developed by Pirzadeh [6]. Prismatic meshes have also been shown to be an efficient way to obtain inviscid [2, 7] and viscous [8, 9] solutions for compressible flows.

In the past, however, explicit finite-volume methods were used to solve the Euler and Navier–Stokes equations on prismatic grids with the exception of the work described in Ref. [7] where a semi–implicit line Jacobi method was used to solve the Euler equations. On unstructured tetrahedral grids, the linelet method of solving the problem semi–implicitly has also been demonstrated [10, 11].

In the present paper, an improved method for treating the structured direction in a prismatic grid in an implicit manner is proposed(Figure 1). The method is based on Runge–Kutta time stepping, which is similar to, but not the same as that used in Ref. [12] for structured grids. The new time stepping method has better stability characteristics and as a result has better convergence properties [13]. The time stepping method was implemented in the Galerkin finite-element framework which was used for discretizing the Euler and Navier–Stokes equations.

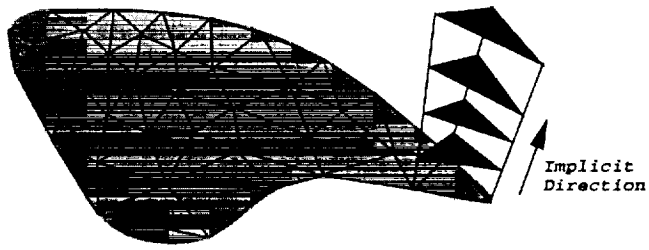


Fig. 1 Column of prisms on a triangulated surface

The basic idea behind the semi–implicit method is as follows. If a fully implicit scheme is used, a matrix with a large bandwidth would have to be solved. However, due to the structure in the body–normal direction of a prismatic grid, such a matrix always has a block–tridiagonal subset. If all the terms which are not contributing to this subset are dropped, then the problem reduces to the solution of a 5×5 block tridiagonal matrix for three dimensional problems. This means that the standard Thomas algorithm [14] can be used. The use of this semi–implicit concept is embedded in a diagonally dominant manner in the Runge-Kutta multi stage scheme.

The implicit treatment of the body-normal direction can be thought of as a line-Jacobi algorithm. However, unlike the work in Ref. [12], all terms contributing to the main diagonal from all directions are used in the solution of the block tridiagonal matrix system. Accounting for these terms is similar to a block-Jacobi method in the body-tangent directions. The block-

*Aerospace Engineer, NASA Ames Research Center, Member AIAA

†Professor, Aeronautical Engineering, AIAA Fellow

Jacobi method has better stability and convergence properties than explicit methods [15] which improves the stability and convergence properties of the present scheme.

The present paper uses a Galerkin finite-element method for spatial discretization. The unsteady terms of the equations are discretized and lumped to the diagonal to drive the solution to the steady state. It is in this artificial time stepping that both a standard explicit Runge-Kutta and the semi-implicit Runge-Kutta schemes are implemented.

We first present the governing equations and discuss the Galerkin finite-element discretization of the spatial terms. We then discuss the temporal discretization and present the time stepping scheme. Comparisons between the explicit and semi-implicit schemes are shown for several inviscid and viscous test cases. These comparisons show that the semi-implicit algorithm is approximately 3 to 4 times more efficient than its explicit counterpart for the solution of both Euler and Navier-Stokes equations. Several solutions are also presented as validation of the present algorithm.

Governing equations

The Navier-Stokes equations for describing viscous fluid flow in three dimensions can be written in integral form as

$$\iiint (\partial_t \vec{Q}) d\mathcal{V} + \iiint \nabla \cdot \mathcal{F} d\mathcal{V} = 0 \qquad (1)$$

where Q is the state vector of density, momentum, and internal energy per unit mass.

$$\vec{Q} = (\rho, \rho u, \rho v, \rho w, e) \qquad (2)$$

$$\mathcal{F} = (F - F_v)\hat{i} + (G - G_v)\hat{j} + (H - H_v)\hat{k} \qquad (3)$$

F, G, and H are the convective flux terms and F_v, G_v, and H_v are the viscous flux terms in x, y and z directions respectively. The pressure term in the flux functions is defined with respect to $\vec{Q}$ as follows.

$$p = (\gamma - 1)(e - \frac{1}{2}\rho(u^2 + v^2 + w^2))$$

For inviscid flow simulations, $F_v = G_v = H_v = 0$. The viscous fluxes are defined in terms of the viscous stress tensor [16].

Spatial discretization

The steady version of the Navier-Stokes equations can be written as

$$\nabla \cdot \mathcal{F} dV = 0 \qquad (4)$$

where $\mathcal{F}$ is the flux function. By the weighted residual method, we multiply by a test function ϕ and integrate by parts. The resulting integral equation can be expressed as a sum of the integrals over each prism.

$$\sum_{e=1}^{n_e} \left(\iint_{\partial\Omega_e} \phi \mathcal{F} \cdot \hat{n} dS - \iiint_{\Omega_e} \nabla\phi \cdot \mathcal{F} d\mathcal{V} \right) = 0 \qquad (5)$$

where e is the prism counter and n_e is the number of prisms. Ω_e is the prism itself and $\partial\Omega_e$ refers to the faces of the prism.

The surface integral in equation 5 cancels at each internal face of the grid. At the boundaries of the domain, the surface integral does not vanish. Therefore, this surface integral is evaluated at the boundary face and its contributions are accumulated in the terms for the vertices that make up the face.

A standard Galerkin finite-element scheme with bilinear shape functions is used to approximate the volume integral terms. The flux function is evaluated with the shape functions N_m as follows.

$$\mathcal{F} = \sum_{m=1}^{6} \mathcal{F}_m N_m \qquad (6)$$

where $\mathcal{F}_m$ is the flux at the vertex m and N_m is the associated shape function.

Another choice for approximating a flux function for non-linear flux terms exists. For example if the flux function $\mathcal{F} = u^2$, we could approximate it as

$$u^2 = \left(\sum_{m=1}^{6} u_m N_m \right) \left(\sum_{m=1}^{6} u_m N_m \right) \qquad (7)$$

This second choice has been shown to not conserve mass [17].

Substituting the test function and equation 6 into equation 5, we get

$$\iiint_{\Omega_e} \sum_{m=1}^{6} \nabla N_m \cdot \sum_{n=1}^{6} N_n \mathcal{F}_n dx dy dz = 0 \qquad (8)$$

Due to the compact support property of the shape function, the test function is only non-zero at the vertex m. So equation 8 yields one equation for every vertex in the mesh.

The integral in equation 8 is evaluated on each prism and the appropriate contributions are distributed to the vertices of that prism. In order to accurately approximate this integral, a six point quadrature rule is used on the right triangular prism(Figure 2) to evaluate the integral. The coordinates (ξ,η,ζ) are the local directions on the right triangular prism. The values of the weights and the locations of the quadrature points are tabulated in [13] along with expressions for the coordinate transformation.

Applying this coordinate transformation to equation 8, we get the following integral on a right triangular prism.

$$\iiint [J^{-1}] \hat{\nabla} N_m \cdot N_n \mathcal{F}_n |J| d\xi d\eta d\zeta = 0 \qquad (9)$$

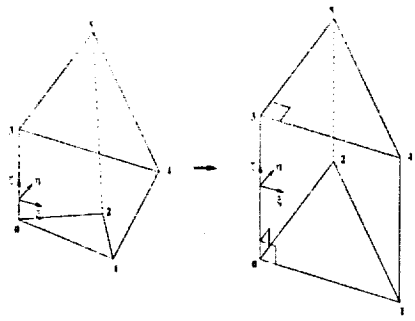


Fig. 2 Mapping a pentahedron to a right triangular prism

where $\hat{\nabla}N = [N_\xi, N_\eta, N_\zeta]^T$ and J is the coordinate transformation Jacobian.

Solving equation 9 at each vertex corresponds to the solution of a system of nonlinear equations. A common method of solving a nonlinear system is by linearizing the equations. We linearize equation 9 by Newton's method using $\mathcal{F}^{k+1} = \mathcal{F}^k + \mathcal{A}^k \delta Q$ where k can be thought of as an iteration index.

After linearization, equation 9 becomes

$$\iiint \left[[J^{-1}]\hat{\nabla}N_m \cdot N_n \mathcal{A}_n |J| d\xi d\eta d\zeta\right] \delta Q_n = \quad (10)$$
$$- \iiint [J^{-1}]\hat{\nabla}N_m \cdot N_n \mathcal{F}_n |J| d\xi d\eta d\zeta$$

The resulting system of equations can be written as

$$A\delta Q = \mathcal{R} \quad (11)$$

where A is the coefficient matrix which gets contributions from the coefficient to δQ on the left hand side of equation 11. We solve for the unknowns δQ. $\mathcal{R}$ is the residual.

Temporal discretization

To implement a time-stepping scheme for the prismatic grid in the Galerkin finite-element framework, the unsteady term of the Navier-Stokes equations

$$\iiint (\partial_t \vec{Q}) d\mathcal{V} \quad (12)$$

is multiplied by the shape function and subsequently the time derivative is approximated as

$$\iiint \phi(\partial_t \vec{Q}) d\mathcal{V} = \iiint \phi \frac{\delta q}{\Delta t} d\mathcal{V} \quad (13)$$

where $\delta q = Q^{n+1} - Q^n$ and n is the time step counter. Using a bilinear shape function and equation 6 with $\mathcal{F} = \delta q$, we approximate equation 13 as

$$\left(\iiint \sum_{m=1}^{6} N_m \sum_{n=1}^{6} N_n d\mathcal{V}\right) \frac{\delta q_j}{\Delta t} \quad (14)$$

This integral results in a left hand side which once again has a sparse coefficient matrix. We lump all terms from this integral to the diagonal. This step is not necessary for semi-implicit or implicit schemes. The following time term results at every vertex m

$$\left(\iiint N_m \sum_{n=1}^{6} N_n d\mathcal{V}\right) \frac{\delta q_m}{\Delta t} \quad (15)$$

Defining the integral term in equation 15 as ΔV, explicit and semi-implicit Runge-Kutta schemes are developed below. A p-step explicit time stepping scheme for taking Q from time n to time $n+1$ can be written as

$$Q^{(0)} = Q^n \quad (16)$$
$$Q^{(k)} = Q^{(k-1)} \quad (17)$$
$$+ \frac{1}{p+1-k} \frac{\Delta t}{\Delta V} \mathcal{R}(Q^{(k-1)})$$
$$- (Q^{(k-1)} - Q^n)$$
$$Q^{n+1} = Q^{(k=p)} \quad (18)$$

where k refers to step number and ranges from 1 to p. The variable $\mathcal{R}$ is the residual from the inviscid and viscous spatial terms of the equation.

A implicit version of the same method is developed by taking the residual at the new step and linearizing with respect to the previous step. The kth step of the linearized implicit Runge–Kutta procedure can be written as

$$\left[(p+1-k)I - \frac{\Delta t}{\Delta V} \frac{\partial \mathcal{R}(Q^{(k-1)})}{\partial Q}\right] \delta Q^{(k)} = \quad (19)$$
$$\frac{\Delta t}{\Delta V} \mathcal{R}(Q^{(k-1)}) - (p+1-k) \quad (Q^{(k-1)} - Q^n)$$

where $\delta Q^{(k)} = Q^{(k)} - Q^{(k-1)}$, and the bracketed part on the left hand side would form the left hand side matrix.

For the semi-implicit scheme, we only retain the components of the flux Jacobian matrix which contribute to the tridiagonal part of the matrix. This results in a left hand side matrix with a 5 × 5 block tridiagonal structure. The scheme can be thought of as the line-Jacobi scheme for a single Runge-Kutta step with contributions to the diagonal from all neighbor points.

$$\left[(p+1-k)I - \frac{\Delta t}{\Delta V} TRID\left(\frac{\partial \mathcal{R}(Q^{(k-1)})}{\partial Q}\right)\right] \delta Q^{(k)} =$$
$$\frac{\Delta t}{\Delta V} \mathcal{R}(Q^{(k-1)}) - (p+1-k)(Q^{(k-1)} - Q^n) \quad (20)$$

where the $TRID$ symbolizes that only the contributions to the tridiagonal portion of the matrix are retained.

Time step computation

To compute a conservative estimate of the time step, the monotonicity analysis done by Barth [18] is used.

Fig. 5 Pressure on the ONERA M6 wing at $M_\infty = 0.84$, $\alpha = 3.06$, inviscid flow

case was 9.1688×10^{-4}seconds/iteration/vertex.

Inviscid flow over ONERA M6 wing

The inviscid flow over a ONERA M6 wing is presented to show the use of the prismatic grid method on a wing geometry at a transonic Mach number. Figure 5 shows a solid surface plot of the pressure on the upper surface. A lambda shock structure is easily seen in this view. Figure 6 shows the C_p comparison to the experimental data [21] and the convergence behavior of the explicit Runge–Kutta method. Due to the high Re number of the experiment which was 11×10^6, we are able to compare the inviscid solution to the experiment. We find that the pressure at the surface was predicted by the present method with reasonable accuracy. The location of the shock was not captured precisely. The present solution agrees with the trend documented in inviscid flow literature such as Ref. [18].

Flat plate boundary layer

To validate the viscous capability of the code, the flat plate boundary layer flow is computed and compared to the Blasius velocity profile. The grid for this test case is set up as four rows of triangles along the plate surface and then prism columns emanating into the direction away from the plate. The grid is highly stretched with cells near the surface having aspect ratios of the order of 750:1.

A velocity profile is compared to the Blasius solution [22] of the boundary layer equations in Figure 7. The numerical solution compares to the exact solution very well.

Laminar viscous flow over ONERA M6 wing

Finally, the viscous flow over a ONERA M6 wing is computed. Since a turbulence model is not implemented, we compute the flow around a ONERA M6 wing at a Re number of 50000. The test case is performed at a Mach number of 0.6 and zero angle of attack. The grid is composed of 171135 vertices and 325248 prisms. A c-type grid topology was used to cluster points in the wake region. Because no experimental observations exist at this Re number, the pressure coefficient is compared to a numerical solution from the OVERFLOW code [23] in Figure 8 at several stations along the span. The two numerical solutions agree very well. OVERFLOW is a structured grid compressible Navier–Stokes solver.

Convergence studies

Parabolic bump

The semi–implicit method was first tested by solving the Euler equations on a 10% parabolic bump. The comparisons were performed on an SGI Indigo2 with a R10000 processor on a stretched grid containing 4805 vertices and 7200 prisms. The 2-, 3-, and 4–step semi–implicit Runge–Kutta methods are compared to the 3- and 4–step explicit Runge-Kutta method. In these multi step methods, variations are possible where the left hand side matrix is not recalculated at all the steps [24]. For the 4–step Runge–Kutta method, we compare the scheme where the matrix is recomputed only at the first and third steps to assess the effect of not updating the left hand side.

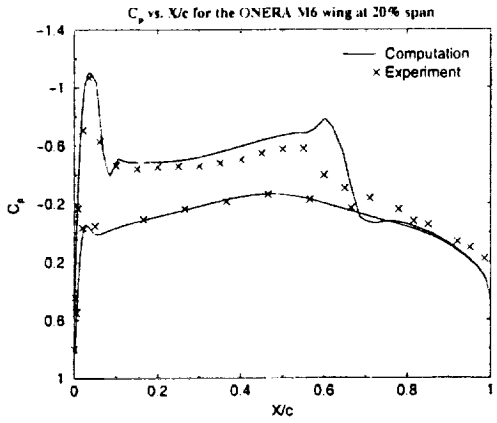
The convergence history for explicit and semi–implicit methods is presented in Figure 9 and is a clear indication that the four step semi–implicit Runge-Kutta method is the fastest method. A closer look at the numbers in Table 1 unveils several details. The Mem column reveals that the semi–implicit methods take up more than twice the memory of the explicit method. From the normalized CPU time consumed per iteration in column $\overline{CPU}$ we discover that the semi–implicit method can be more than twice as expensive as the explicit methods. However, the $\overline{Time}$ column reveals that the overall efficiency(defined as time to convergence) of the semi–implicit schemes is up to 4 times better than the explicit methods. Finally, not recomputing the matrix at every step slows convergence.

The number of iterations and total CPU time required to converge the solution to a correction of 10^{-14} is reported in Table 1.

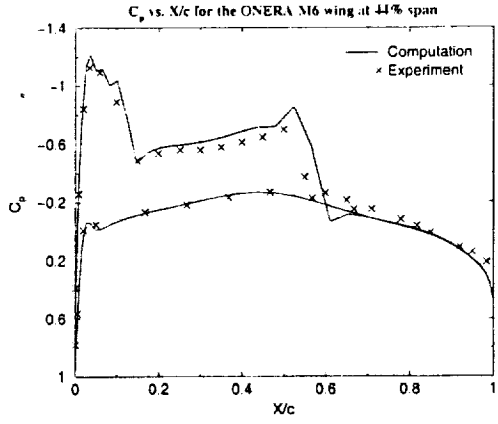
Sphere

To further verify these results, we perform another comparison of the 4 step explicit method and the 4 step semi–implicit method. We use the inviscid flow around a sphere for this comparison. The prismatic grid used for this comparison is larger than the bump case with 18000 grid points to make sure the parabolic bump results were not due to the small problem size.

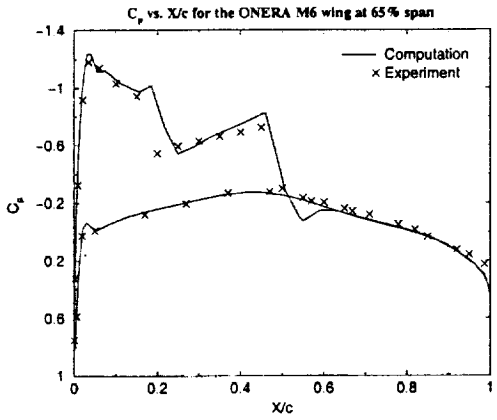
The convergence history from the sphere test case is shown in Figure 10 and the CPU times are compared in Table 2. Once again, the semi–implicit method per-



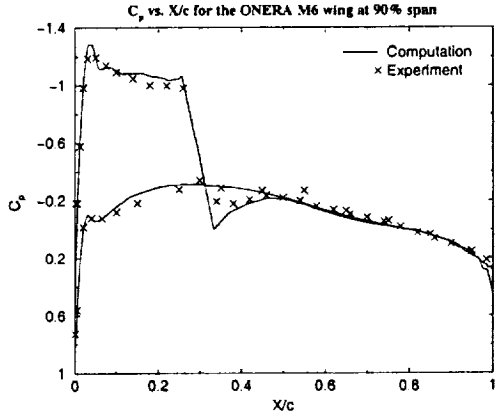
a) 20% span station



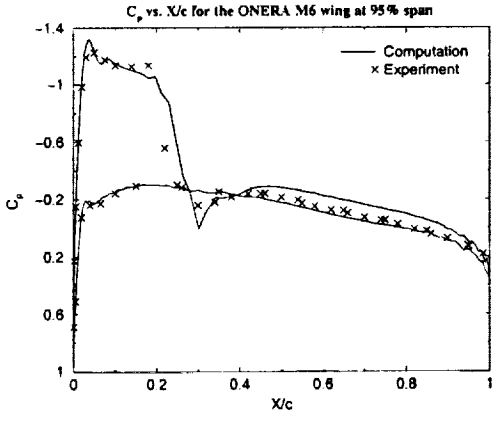
b) 44% span station



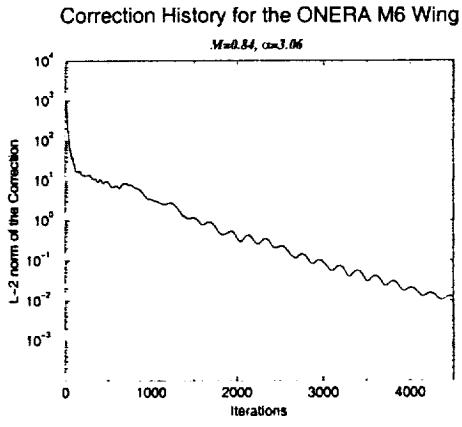
c) 65% span station



d) 90% span station



e) 95% span station



f) Convergence History

Fig. 6 C_p **at various span stations with comparison to experiment of Ref. [21] and convergence history on a ONERA M6 wing at** $M_\infty = 0.84$, $\alpha = 3.06$, **inviscid flow**

R-K Steps	Method	CPU [s/iter]	$\overline{CPU}$	Mem [Kb]	CFL	Iterations	Time [s]	$\overline{Time}$
3	Explicit	1.145	0.79	2843	2.5	20000	22900	1.27
4	Explicit	1.445	1	2843	4	12500	18062	1
2	Semi-implicit	1.702	1.17	7081	2.5	11000	18722	1.04
3	Semi-implicit	2.429	1.68	7081	6	2850	6922	0.38
4	Semi-implicit(1,3)	2.492	1.72	7081	9	2200	5482	0.3
4	Semi-implicit(Full)	3.156	2.18	7081	9	1500	4734	0.26

Table 1 Comparison of explicit Runge–Kutta schemes to the semi–implicit schemes for inviscid flow over a parabolic bump

R-K Steps	Method	CPU [s/iter]	$\overline{CPU}$	Mem [Kb]	CFL	Iterations	Time [s]	$\overline{Time}$
4	Explicit	9.766	1	6857	5	15000	146488	1
4	Semi-implicit(Full)	22.377	2.3	18917	8	2500	55943	0.38

Table 2 Comparison of the explicit scheme to the semi–implicit schemes for inviscid flow over a sphere

AR	R-K Steps	Method	CPU [s/iter]	$\overline{CPU}$	Mem [Kb]	CFL	Iterations	Time [s]	$\overline{Time}$
5	4	Explicit	8.84	1	6888	5	13000	114920	1
5	4	Semi-implicit	18.38	2.1	20108	20	2500	45950	0.4
10	4	Explicit	17.82	1	13549	4.5	25000	445500	1
10	4	Semi-implicit	40.97	2.3	39601	30	3000	122910	0.27

Table 3 Comparison of the explicit scheme to the semi–implicit scheme for viscous flow over a flat plate at $M_\infty = 0.5$, and $Re = 100000$

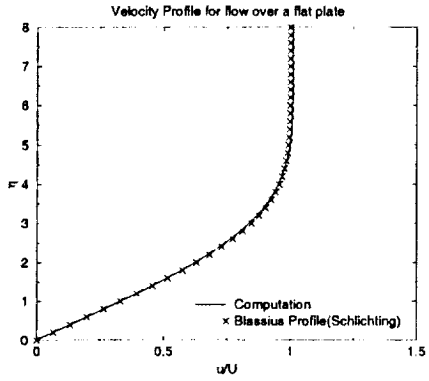


Fig. 7 Velocity profile for flow over a flat plate at $M_\infty = 0.5$, $Re = 100,000$

forms better. The semi–implicit method is an order of magnitude faster by number of iterations and about 3 times more efficient in terms of CPU time.

Flat plate boundary layer

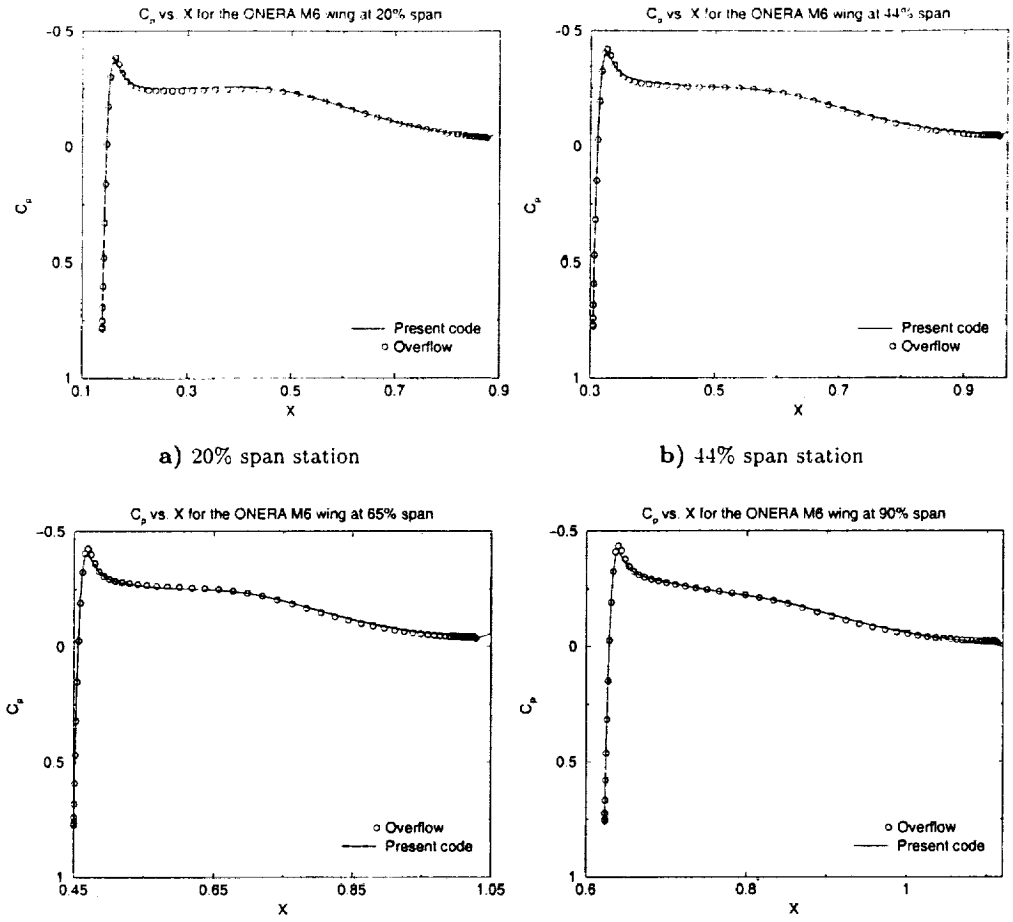
The advantage of the semi–implicit method is also demonstrated for viscous flows. The simulation of the boundary layer flow over a flat plate is chosen as the first test case. The comparison is done with two separate uniform grids of cell aspect ratio 5 and 10 respectively to emphasize the effect of the aspect ratio on the rate of convergence obtained by the semi–implicit scheme. The cell aspect ratio here is computed based on the cell width divided by the cell height. The width is computed using the fact that the flat plate triangulation was made from a uniform structured grid.

The aspect ratio 5 grid has a total of 15655 vertices and 24000 prisms, whereas the aspect ratio 10 grid is simply twice as dense in the direction normal to the plate. The convergence histories are shown in Figure 11 and the statistics are tabulated in Table 3. The CFL numbers are based on the explicit time step in order to establish a trend with respect to the aspect ratio of the grid.

A comparison of the statistics for the aspect ratio 5 results to the aspect ratio 10 results shows that though the explicit method takes almost twice as long on the denser grid, the semi–implicit method takes approximately the same number of iterations. The principle reason for this can be attributed to the fact that the same time step for both aspect ratio grids can be used for the semi–implicit method, while a smaller time step is required for the explicit method due to stability restrictions.

ONERA M6 wing

Finally, the ONERA M6 wing test case is used to verify that the convergence rate improvement is also realized on a mesh with varying aspect ratios in different parts of the grid. The viscous solution over the ONERA M6 wing is performed at $M_\infty = 0.6$ at zero



a) 20% span station

b) 44% span station

c) 65% span station

d) 90% span station

Fig. 8 Pressure coefficient on an ONERA M6 wing at $M_\infty = 0.6$, $Re = 50000$

R-K Steps	Method	CPU [s/iter]	$\overline{CPU}$	Mem [Kb]	CFL	Iterations	Time [s]	$\overline{Time}$
4	Explicit	68	1	61776	0.75	9000	612000	1
4	Semi-implicit	99	1.46	172317	3.0	1800	178200	0.3

Table 4 Comparison of the explicit scheme to the semi-implicit scheme for viscous flow over an ONERA M6 wing at $M_\infty = 0.6$, and $Re = 50000$

angle of attack and a $Re = 50000$. The grid for the test case consists of 171135 vertices, and 325248 prisms. Although the cell aspect ratio varies in different parts of the mesh, near the surface of the wing they ranged between 100 and 1000.

The comparison between the residual histories of the four step explicit and semi-implicit Runge-Kutta methods is shown in Figure 12, and the associated statistics are compiled in Table 4. The CPU times reported are for a single CPU on an Origin2000. Once again, the semi-implicit method is seen to have better convergence properties and is the more efficient method.

Concluding remarks

The use of prismatic grids allows for fast convergence and for accurate solution of the flow field. The finite element flux formulation is validated for inviscid and viscous flows with a variety of aspect ratio grids. The inviscid flow over a sphere is presented and is in favorable agreement with the known potential solution. The inviscid flow over a ONERA M6 wing is computed at transonic Mach numbers. It is found that the pressure at the surface of the wing is in reasonable agreement with the experimental results [21] except for the location of the shock.

For the validation of the prismatic grid method for viscous flows, the case of the flat plate boundary layer is compared to the analytical solution of Blasius [22]. It is found that the computed velocity profile is in

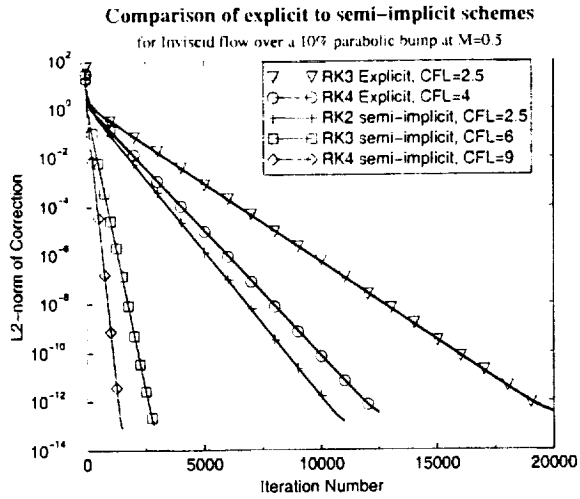


Fig. 9 Convergence history comparison for inviscid flow over a 10% parabolic bump at $M_\infty = 0.5$

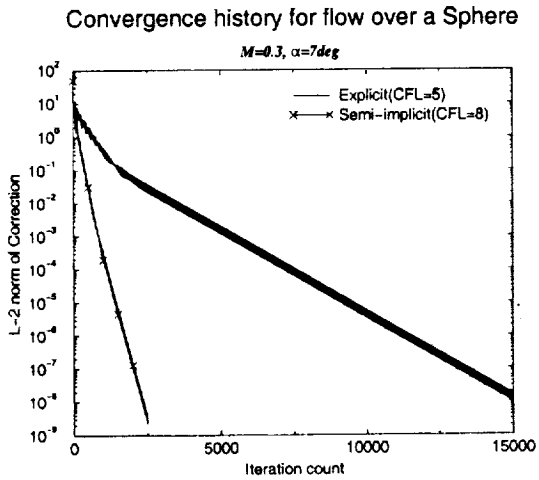
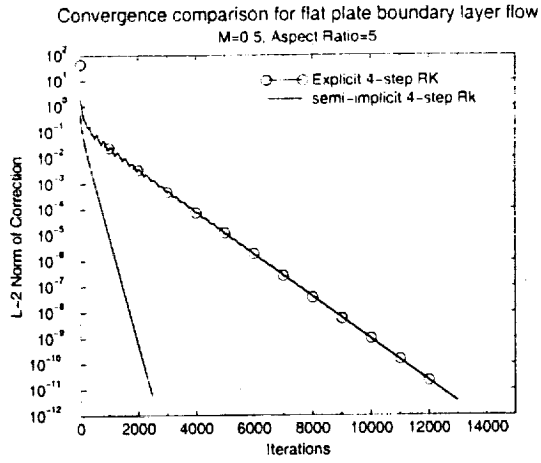
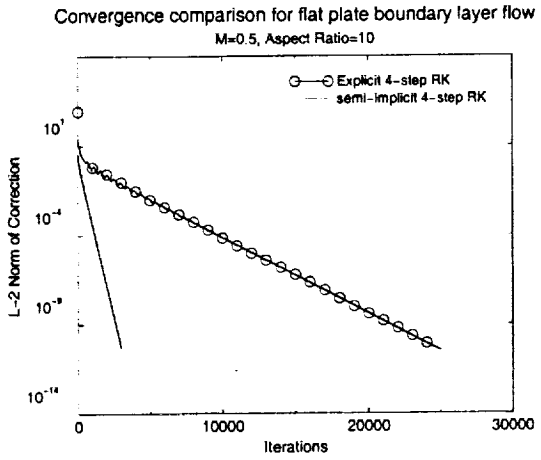


Fig. 10 Convergence history comparison for inviscid flow over a sphere at $M_\infty = 0.3$



a) $AR = 5$ grid



b) $AR = 10$ grid

Fig. 11 Convergence history comparison for viscous flow over a flat plate at $M_\infty = 0.5$, and $Re = 100000$

respectable agreement with the analytic solution. Finally, viscous flow simulation over the ONERA M6 wing is presented and compared to the numerical solution from the OVERFLOW code. The pressure distributions at the surface were found to be in good agreement.

Comparisons are made with test cases to analyze the convergence benefits of the semi-implicit method over the explicit methods. The semi-implicit method is 3 to 4 times faster than the explicit method and it becomes more useful with higher cell aspect ratios. This was shown by performing numerical experiments where the convergence rates for several test cases of varying sizes were used to compare the semi-implicit method to explicit methods. For the cases where the aspect ratio of the prism cells was carefully controlled, we see convergence acceleration consistent with that predicted by theory. Tables 2 and 4 reveal that for a mesh where the grid aspect ratio is not carefully controlled, the semi-implicit method is still faster than a traditional Runge-Kutta method by a factor of 3.

Further work in assessing the usefulness of the semi-implicit schemes is still needed. The relationship of the semi-implicit methods to the line-Jacobi preconditioning methods of Allmaras [25] should be investigated. The possibility of implementing agglomeration multi-grid in the unstructured directions is also attractive. Another attractive method worth analyzing is the LU-SGS implementation by Nakamura [26] on unstructured grids and the subsequent application on prisms by Sharov [27]. A combination of this method with line relaxation should be developed.

References

[1]Nakahashi, K. "Optimum Spacing Control of the Marching Grid Generation". AIAA Paper 91-0103, 1991.

[2]Melton, J. E., S. A. Pandya, and J. L. Steger. "3D Euler Flow Solutions using Unstructured Cartesian and Prismatic Grids". AIAA Paper 93-0331, 1993.

[3]Kallinderis, Y. and S. Ward. "Prismatic Grid generation with an efficient algebraic method for aircraft configurations".

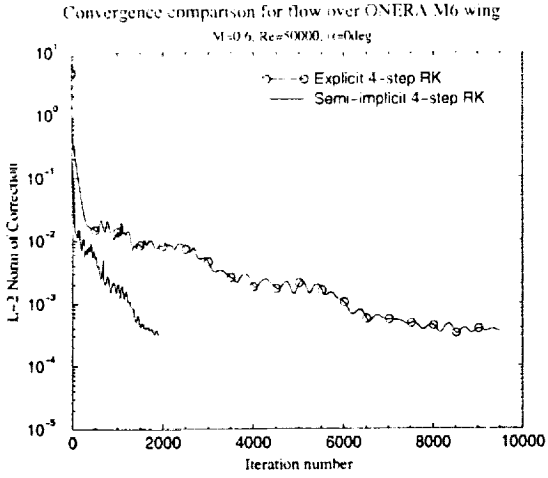


Fig. 12 Convergence history comparison for viscous flow over a ONERA M6 wing at $M_\infty = 0.6$, and $Re = 50000$

AIAA Paper 92-2721, 1992.

[4]Lohner, Rainald. "Matching Semi-Structured and Unstructured Grids for Navier-Stokes Calculations". AIAA Paper 93-3348-CP, 1993.

[5]Marcum, David L. "Generation of Unstructured Grids for Viscous Flow Applications". AIAA Paper 95-0212, 1995.

[6]Pirzadeh, Shahyar. Three-dimensional unstructured viscous grids by the advancing-layers method. *AIAA Journal*, 34(1):43-49, January 1996.

[7]Pandya, S. A. and M. M. Hafez. "A semi-Implicit Finite Volume Scheme for Solution of Euler Equations on 3-D Prismatic Grids". AIAA Paper 93-3431, 1993.

[8]Nakahashi, Kazuhiro. "Adaptive-Prismatic-Grid Method for External Viscous Flow Computations". AIAA Paper 93-3314-CP, 1993.

[9]Parthsarathy, V., Y. Kallinderis, and K. Nakajima. "Hybrid Adaptation Method and Directional Viscous Multigrid with Prismatic-Tetrahedral Meshes". AIAA Paper 95-0670, 1995.

[10]Martin, Dorothee and Rainald Lohner. An implicit linelet-based solver for incompressible flows. AIAA Paper 92-0668, January 1992.

[11]Hassan, O., E. J. Probert, K. Morgan, and J. Peraire. "Line Relaxation Methods for the Solution of 2D and 3D Compressible Flows". AIAA Paper 93-3366, 1993.

[12]Loyd, B., K. Lee, and E. Murman. Semi-implicit navier-stokes solver(sinss) calculations of separated flows around blunt delta wings. AIAA Paper 90-0590, January 1990.

[13]Pandya, S. A. *A Finite Element Approach for Modelling of Inviscid and Viscous Compressible Flows using Prismatic Grids*. PhD thesis, University of California, Davis, October 1998.

[14]Thomas, L. H. "Elliptic Problems in Linear Difference Equations over a Network". Technical report, Columbia University, New York, 1949. Watson Sci. Comput. Lab. Rept.

[15]Pierce, Niles A. and Michael B. Giles. Preconditioning on stretched meshes. Technical Report 95/10, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, England, OX1 3QD, June 1995.

[16]Hirsch, C. *Numerical Computation of Internal and External Flows*. Wiley, 1990.

[17]Chattot, J. J., J. Guiu-Roux, and J. Lamine. Numerical solution of a first-order conservation equation by a least square method. *International Journal for Numerical Methods in Fluids*, 2:209-219, 1982.

[18]Barth, T. J. "Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations". AGARD-R-787, 1993.

[19]Winterstein, R. L. and M. M. Hafez. "Euler Solutions for Blunt Bodies using Triangular Meshes: Artificial Viscosity Forms, and Numerical Boundary Conditions". AIAA Paper 93-3333-CP, 1993.

[20]White, Frank M. *Fluid Mechanics*. McGraw-Hill Book Company, 1986.

[21]Schmitt V. and Charpin F. "Pressure Distributions on the ONERA-M6 Wing at Transonic Mach Numbers". In *AGARD Report*, number 138 in AR, pages B1-1 - B1-44, 1979.

[22]Schlichting, Hermann. *Boundary-Layer Theory*. McGraw-Hill, seventh edition, 1979.

[23]Buning, Pieter. G et al. Overflow user's manual. Version 1.8, NASA Ames Research Center, February 1998.

[24]Chakravarthy, S. R. Relaxation methods for unfactored implicit upwind schemes. AIAA Paper 84-0165, January 1984.

[25]Allmaras, Steven R. Analysis of semi-implicit preconditioners for multigrid solution of the 2-d compressible navier-stokes equations. AIAA Paper 95-1651, 1995.

[26]Men'shov, I. and Y. Nakamura. Implementation of the lu-sgs method for an arbitrary finite volume discretization. In *9th Japanese Symposium on CFD*, pages 123,124, Chuo University, Tokyo, Japan, 1995.

[27]Sharov, D. and K. Nakahashi. Reordering of hybrid unstructured grids for lower-upper symmetric gauss-siedel computation. *AIAA Journal*, 36(3):484-486, March 1998.