# AIAA 2005–1222

## Unstructured Dynamic Meshes with Higher-order Time Integration Schemes for the Unsteady Navier-Stokes Equations

Zhi Yang
Dimitri J. Mavriplis

*Department of Mechanical Engineering,*

*University of Wyoming 1000 E. University Avenue Laramie, WY 82071*

**41th AIAA Aerospace Sciences Meeting and Exhibit**
**January 10–13, 2005/Reno, NV**

# Unstructured Dynamic Meshes with Higher-order Time Integration Schemes for the Unsteady Navier-Stokes Equations

Zhi Yang *
Dimitri J. Mavriplis †
*Department of Mechanical Engineering,*
*University of Wyoming 1000 E. University Avenue Laramie, WY 82071*

**Efficient techniques for computing time-dependent flows with dynamically deforming unstructured meshes are investigated. These include the formulation of robust mesh motion techniques, as well as the formulation of a third-order backwards difference time-integration scheme for the flow equations, which obeys a discrete geometric conservation law. Efficient multigrid solution techniques are devised for solving both the mesh motion equations, and the governing flow equations, using the same agglomerated coarse levels for both problems. Sample problems are used to demonstrate the accuracy and efficiency of these techniques in two and three dimensions.**

## Introduction

Unstructured mesh approaches have become well established for steady-state flow simulations due to the flexibility they afford for dealing with complex geometries. For unsteady flows with moving boundaries, such as fluid-structure problems (wing/tail buffet, flutter), implicit time-integration strategies are required for the efficient solution of the flow equations, while at the same time robust mesh deformation techniques are necessary for maintaining a suitable discretization of the evolving computational domain. In order to develop an efficient unsteady flow simulation capability, both mesh deformation and flow solution aspects must be considered, as well as the interaction between these two areas. In previous work,[8] we investigated the use of fast multigrid solvers for high-order implicit time-integration of unsteady flows on static meshes. In this work, we extend our investigation to the simultaneous development of robust and efficient unstructured mesh deformation techniques, as well as efficient higher-order time-integration strategies, and investigate the requirements for maintaining high time-accuracy in the presence of deforming meshes.

Several mesh-deformation strategies, such as the tension spring analogy,[3,25] the torsion spring analogy[5,20] and the linear elasticity analogy,[21] have been successfully demonstrated in the literature. However, there exist wide disparities in robustness and efficiency of these various methods. Our goal is to evaluate these various approaches and to develop efficient solution algorithms for the most robust mesh deformation strategies, suitable for long-time integration of large deformation, high-mesh resolution simulations on parallel computers.

For unsteady flow simulations, computational time remains an important issue. When small temporal errors are desired, higher-order time-integration (higher than second-order) has been shown to be more efficient than low-order time integration. Bijl, Carpenter and Vatsa[4] investigated and compared higher-order implicit Runge-Kutta schemes and Backward Differencing schemes on structured grids, while Jothiprasad, Mavriplis and Caughey[8] showed how a fourth order Runge-Kutta scheme (RK64) outperforms second-order Backward Differencing scheme (BDF2) on unstructured grids using a multigrid algorithm for solving the implicit system arising at each time step.

When dynamic meshes are used, the mesh velocities and other parameters related to geometry need to be considered carefully so that the errors introduced by the deformation of the mesh do not degrade the formal accuracy of the flow simulation. The discrete geometric conservation law (DGCL) provides a guideline on how to evaluate these quantities. First-order and second-order time-accurate and geometrically conservative schemes were presented and discussed in,[10,11] respectively. Guillard and Farhat have proved that to obtain at least first-order time accuracy, a DGCL condition must be satisfied.[6] For higher-order time integration schemes, a DGCL strategy which preserves the design order of the scheme must be explicitly constructed.

In addition to preserving time-accuracy, efficient solution strategies must be employed to avoid excessive computational times for long-time integration problems. Nonlinear and linear multigrid methods have been investigated previously for steady and unsteady flow simulations using unstructured meshes.[12,14,17] In this work we investigate the use of unstructured agglomeration multigrid for the time-integration of the unsteady flow equations, as well as for the solution of the equations governing the mesh

*Postdoctoral research associate, AIAA member; email: zyang@uwyo.edu
†Professor, AIAA Associate Fellow; email:mavriplis@uwyo.edu

deformation.

In the following sections, we first outline the governing equations and the base flow solver. We then discuss our choice and implementation of a high-order time integration scheme (BDF3), followed by a presentation of the geometric conservation law (GCL), which must be respected for this scheme. The various mesh deformation strategies which have been investigated are then described, followed by the multigrid and implicit line solution techniques used for the flow and mesh motion equations. Finally, a set of sample test cases is presented in order to illustrate the performance of these methods.

## Governing Equations in Arbitrary-Lagrangian-Eulerian (ALE) Form and Base Flow Solver

The Navier-Stokes equations in conservative form can be written as:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla.(\mathbf{F}(\mathbf{U}) + \mathbf{G}(\mathbf{U})) = 0 \tag{1}$$

where $\mathbf{U}$ represents the vector of conserved quantities (mass, momentum, and energy), $\mathbf{F}(\mathbf{U})$ represents the convective fluxes and $\mathbf{G}(\mathbf{U})$ represents the viscous fluxes. Integrating over a (moving) control volume $\Omega(t)$, we obtain:

$$\int_{\Omega(t)} \frac{\partial \mathbf{U}}{\partial t} dV + \int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U})\cdot\vec{\mathbf{n}})\mathbf{dS} + \int_{\partial\mathbf{\Omega(t)}} (\mathbf{G}(\mathbf{U})\cdot\vec{\mathbf{n}})\mathbf{dS} = \mathbf{0} \tag{2}$$

Using the differential identity

$$\frac{\partial}{\partial t}\int_{\Omega(t)} \mathbf{U} dV = \int_{\Omega(t)} \frac{\partial \mathbf{U}}{\partial t} dV + \int_{\partial\Omega(t)} (\dot{\mathbf{x}}\cdot\vec{\mathbf{n}})\mathbf{dS} \tag{3}$$

where $\dot{\mathbf{x}}$ and $\vec{\mathbf{n}}$ are the velocity and normal of the interface $\partial\Omega(t)$, respectively,
equation (2) becomes:

$$\begin{aligned} \frac{\partial}{\partial t}\int_{\Omega(t)} \mathbf{U} dV &+ \int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U})\cdot\vec{\mathbf{n}}\mathbf{dS} \\ &+ \int_{\partial\Omega(t)} \mathbf{G}(\mathbf{U})\cdot\vec{\mathbf{n}}\mathbf{dS} = \mathbf{0} \end{aligned} \tag{4}$$

Considering $\mathbf{U}$ as cell averaged quantities, these equations are discretized in space as:

$$\frac{\partial}{\partial t}(V\mathbf{U}) + \mathbf{R}(\mathbf{U}, \dot{\mathbf{x}}(\mathbf{t}), \vec{\mathbf{n}}(\mathbf{t})) + \mathbf{S}(\mathbf{U}, \vec{\mathbf{n}}(\mathbf{t})) = \mathbf{0} \tag{5}$$

where $R(\mathbf{U}, \dot{\mathbf{x}}, \vec{\mathbf{n}}) = \int_{\partial\mathbf{\Omega(t)}} (\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U})\cdot\vec{\mathbf{n}}\mathbf{dS}$ represents the discrete convective fluxes in ALE form, $S(\mathbf{U}, \vec{\mathbf{n}})$ represents the discrete viscous fluxes, and $V$ denotes the control volume. In the discrete form, $\dot{\mathbf{x}}(\mathbf{t})$ and $\vec{\mathbf{n}}(\mathbf{t})$ now represent the time varying velocities and surface normals of the control volume boundary faces.

The Navier-Stokes equations are discretized by a central difference finite-volume scheme with additional matrix-based artificial dissipation on hybrid meshes which may include triangular and quadrilateral elements in two dimensions, or tetrahedra, pyramids, prisms and hexahedra in three dimensions. Second-order accuracy is achieved using a two-pass construction of the artificial dissipation operator, which corresponds to an undivided biharmonic operator. A single unifying edge-based data-structure is used in the flow solver for all types of elements. The thin-layer form of the Navier-Stokes equations is employed and the viscous terms are discretized to second-order accuracy by finite-difference approximation for non-simplicial elements. For multigrid calculations, a first-order discretization is employed for the convective terms on the coarse grid level.[15,18]

## Higher-order Time Integration and the Discrete Geometric Conservation Law (GCL)

For unsteady flow simulations, a fully implicit time-integration strategy is most often adopted, using either multistep Backward Difference Formulas (BDF) or multistage Implicit Runge-Kutta (IRK) schemes. Although first-order (BDF1) and second-order (BDF2) backwards difference schemes are A-stable, higher-order multistep BDF schemes (beyond second-order) are not A-stable. However, the unstable region of the BDF3 scheme is very small and this scheme has most often been used successfully for unsteady flow simulations. Multistage IRK schemes of high-order which are A-stable and L-stable can easily be constructed. However, multiple nonlinear problems need to be solved at each time step using IRK schemes, which makes these more expensive alternatives to the BDF schemes.[4] In this paper, our discussion will be limited to higher-order BDF schemes (i.e. BDF2 and BDF3). The application of higher-order IRK schemes for dynamic unstructured mesh problems is currently under investigation.[19]

Equation (5) can be rewritten using the general formula for a k-step backward difference scheme as[4,8]

$$\alpha_1 (V\mathbf{U})^{n+1} + \sum_{i=0}^{1-k} \alpha_i (V\mathbf{U})^{n+i} = \Delta t \mathbf{R}((V\mathbf{U})^{n+1}, t^{n+1}) \tag{6}$$

For the BDF3 scheme, the coefficients are given as: $\alpha_1 = \frac{11}{6}$, $\alpha_0 = -3$, $\alpha_{-1} = \frac{3}{2}$, $\alpha_{-2} = -\frac{1}{3}$. By defining a nonlinear residual[8]

$$\begin{aligned} \Re^{n+1}(\mathbf{U}) \equiv \Re(\mathbf{U}^{n+1}) &\equiv \alpha_1 (V\mathbf{U})^{n+1} - \sum_{i=0}^{k-1} \alpha_i (V\mathbf{U})^{n-i} \\ &- \Delta t \mathbf{R}((V\mathbf{U})^{n+1}, t^{n+1}) \end{aligned} \tag{7}$$

the solution of equation (6) can be obtained by solving the non-linear problem $\Re^{n+1}(\mathbf{U}) = 0$ at each time step. Two different methods are used to solve the above equation: a

nonlinear multigrid full approximation storage (FAS) agglomeration method, and a linear agglomeration multigrid (LMG) method used as a solver at each stage within an approximate non-linear Newton iteration strategy. The details of these two methods can be found in reference.[8,17]

To apply the multistep BDF scheme in the presence of dynamic unstructured meshes, the so called geometric conservation law (GCL) should be satisfied to avoid degrading the formal accuracy of the scheme. The original statement of the geometric conservation law was introduced by Thomas and Lombard[24] for structured meshes. The discrete geometric conservation law requires that the state $\mathbf{U} = constant$ be an exact solution of equation (5). In this case, we have $S(\mathbf{U}, \vec{\mathbf{n}}) = \mathbf{0}$, since the viscous fluxes are based on gradients of $\mathbf{U}$. Additionally, we have:

$$\int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}) \cdot \vec{\mathbf{n}} d\mathbf{S} = \mathbf{R}(\mathbf{U}, \dot{\mathbf{x}}, \vec{\mathbf{n}}) = -\mathbf{U}\bar{\mathbf{R}}(\dot{\mathbf{x}}, \vec{\mathbf{n}}) \quad (8)$$

since the integral of the convective fluxes $\mathbf{F}(\mathbf{U})$ around a closed control volume must be zero for constant $\mathbf{U}$, for any spatially conservative scheme, with $\bar{\mathbf{R}}$ referring to the discretization of the second term in the above boundary integral. Thus, the GCL can be stated, in semi-discrete form as:

$$\frac{\partial V}{\partial t} - \bar{\mathbf{R}}(\dot{\mathbf{x}}(\mathbf{t}), \vec{\mathbf{n}}(\mathbf{t})) = \mathbf{0} \quad (9)$$

The Geometric Conservation Law must be satisfied in the discrete form, or so called Discrete Geometric Conservation Law. Lesoinne and Farhat[11] presented a first-order time-accurate backwards difference scheme (BDF1) which obeys the discrete geometric conservative law, while Koobus and Farhat[9] derived a second-order accurate backwards-difference scheme (BDF2) which obeys the discrete conservation law. Because higher-order time integration has been found to outperform low-order time integration for unsteady flow simulations,[4,8] a third-order time-accurate backwards-difference scheme (BDF3) which obeys the discrete geometric conservation law is derived according to the methods presented in[9,11] and presented below. The end result consists of a formula for computing the appropriate values of the grid point velocities and control volume face normals at each time step, which preserve both the formal accuracy of the BDF3 scheme, while respecting the GCL (i.e. admitting uniform flow as an exact solution). The final scheme is given as:

$$\sum_{i=1}^{-2} \alpha_i (V\mathbf{U})^{n+i} = \Delta t \mathbf{R}((V\mathbf{U})^{n+1}, t^{n+1}, \dot{x}(t), \vec{n}(t)) \quad (10)$$

where

$$\begin{aligned} \vec{n}(t) &= \tfrac{1}{2}\alpha_1(\vec{n}_1 + \vec{n}_2) + \tfrac{1}{2}(\alpha_1 + \alpha_0)(\vec{n}_3 + \vec{n}_4) \\ &- \tfrac{1}{2}\alpha_{-2}(\vec{n}_5 + \vec{n}_6) \\ \dot{x}(t) &= \tfrac{1}{2\Delta t}[\alpha_1(\vec{x}^{n+1} - \vec{x}^n) \cdot (\vec{n}_1 + \vec{n}_2) \\ &+ (\alpha_1 + \alpha_0)(\vec{x}^n - \vec{x}^{n-1}) \cdot (\vec{n}_3 + \vec{n}_4) \\ &- \alpha_{-2}(\vec{x}^{n-1} - \vec{x}^{n-2}) \cdot (\vec{n}_5 + \vec{n}_6)] \end{aligned}$$

The $\vec{n}_k$ represents the normal vector of a control volume boundary face evaluated at the different quadrature points located between the locations $n-2$ and $n+1$ in time, as detailed in the Appendix, while the $x^{n-2}$, $x^{n-1}$, $x^n$ and $x^{n+1}$ values refer to the grid point positions at the respective physical time steps. A full derivation of the GCL for BDF3 is given in the Appendix.

It should be noted that backward difference schemes beyond second-order temporal accuracy are not A-stable. However, for the third-order backward difference scheme, the unstable region is very small and we have not encountered stability issues in our test cases. Implicit Runge-Kutta schemes remain a viable alternative for achieving higher-order with complete A-stability.[4,8,19]

## Mesh Moving Strategies

**Tension spring analogy**

The tension spring analogy is perhaps the oldest and simplest strategy for unstructured mesh deformation (see for example[3]). In this approach, each edge of the mesh is represented by a spring whose stiffness is related to the length of the edge. The governing equations are closely related to a simple Laplace equation, as the displacements in each coordinate direction become decoupled and are governed by the equations:

$$(\Delta x_i)_m = \frac{\sum_j k_{ij}((x_j)_m - (x_i)_m)}{\sum_j k_{ij}}, \quad m = 1, 2, 3 \quad (11)$$

where $l_{ij} = \sum_{m=1}^{3}((x_i)_m - (x_j)_m)^2)^{\frac{1}{2}}$ and $k_{ij} = \frac{1}{l_{ij}^p}$. The parameter $p$ usually is set to 2.[25] Although these equations are relatively simple to solve, this approach tends to produce deformed meshes with collapsed or negative cell volumes especially for the high aspect-ratio meshes used in viscous flow problems. Since the Laplace equation obeys a maximum principle, it is easily seen that this approach is incapable of reproducing solid body rotation even in the presence of high spring constants, since for example, in the case of a pitching airfoil, this would require larger displacements away from the airfoil surface.

**Torsion spring analogy**

Murayama[20] improved the tension spring model by attaching torsion springs to each vertex. The stiffness of these torsion springs is related to the angle:

$$k_{ij}^{\tau} = \frac{1}{sin^2\theta_k} \quad (12)$$

As the angle $\theta_k \to 0$ or $\pi$, the stiffness $k_{ij}^{\tau} \to \infty$, thus the additional torsion springs can prevent vertices from crossing over edges or faces and avoid zero or negative cell areas or volumes. Compared to the tension spring method, the combination of tension spring and torsion springs is more robust and easily applied to three dimensional meshes.[20] Farhat has also presented a torsion spring method in reference.[5]

**Truss analogy**

In the truss analogy method,[5] each edge is represented by a bar or spring. Compared to the spring method presented above, the displacements of vertices in each direction are coupled in the truss method. Based on the bar equation,

$$\frac{d}{dx}\left(EA\frac{du}{dx}\right) = 0 \tag{13}$$

a global stiffness matrix can be assembled based on local elemental matrices:[22]

$$\begin{aligned}[K_{truss}] &= \tfrac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \\ [K_{global}] &= \left[R^T\right][K_{truss}][R]\end{aligned} \tag{14}$$

where $R$ is the transformation matrix and $EA$ in the stiffness matrix $K_{truss}$ is set to the reciprocal of the edge length.

**Linear elasticity analogy**

Various researchers have used the linear elasticity equations to simulate mesh deformation, due to the robustness of this approach.[2,7,21] The computational mesh is assumed to obey the linear elasticity equation, which can be written as:

$$\frac{\partial \sigma_{ij}}{\partial x_i} = -f_j, \qquad \sigma = D\epsilon, \qquad \epsilon = AU \tag{15}$$

where, in three dimensions, the stresses $\sigma_{ij}$, strains $\epsilon_{ij}$, and displacements $U_i$ are given as:

$\sigma = \{\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{23}, \sigma_{31}\}^T$,

$\epsilon = \{\epsilon_{11}, \epsilon_{22}, \epsilon_{33}, \epsilon_{12}, \epsilon_{23}, \epsilon_{31}\}^T$

$U = \left\{ \begin{array}{cc} u & v \quad w \end{array} \right\}^T$

and the remaining matrices are given as:

$$D = \alpha \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}-\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}-\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}-\nu \end{bmatrix}$$

$$A = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial z} \\ 0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial x} & \frac{\partial}{\partial z} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix}^T$$

where

$$\alpha = \frac{E}{(1+\nu)(1-2\nu)} \tag{16}$$

and where $E$ represents the modulus of elasticity, and $\nu$ is the Poisson ratio for a solid material.

By introducing the shape functions $N$ and $U = NU^e$, taken as linear shape functions in our case, and applying a standard Galerkin method, we obtain

$$\int_\Omega (AN)^T D(AN)U^e dS = -\int_\Omega N^T f dS \tag{17}$$

which can be rewritten as

$$KU = F \tag{18}$$

where

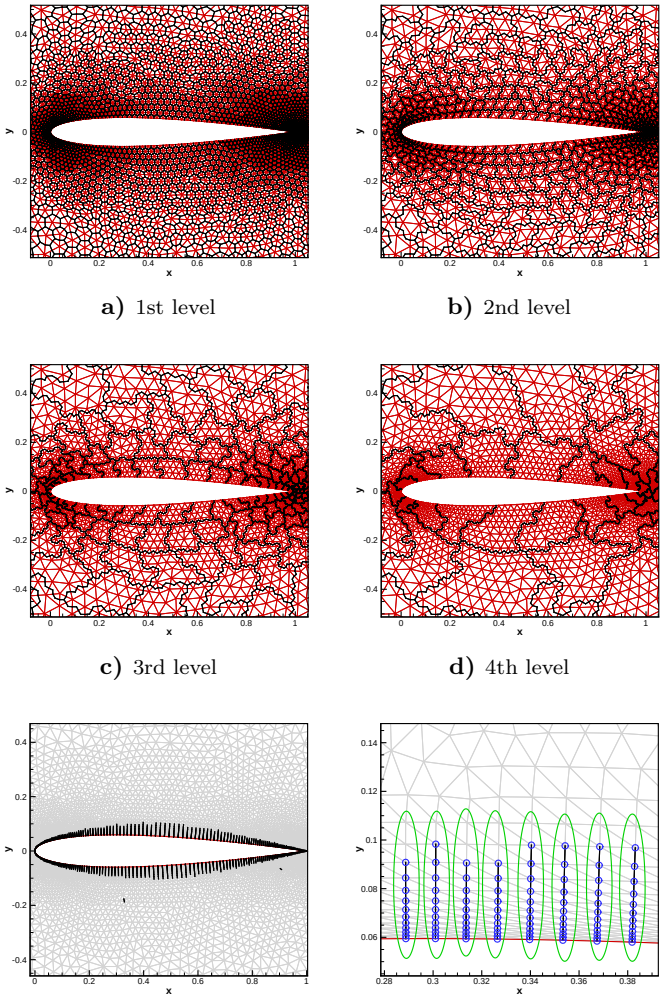
$K = \int_\Omega (B)^T D(B) dS, \quad F = -\int_\Omega N^T f dS, \quad B = AN$

In the mesh deformation case, the boundary displacements are given, so that the external forces $f_j$, of the force vector $F$ are not required. Rather, the homogeneous problem $KU = 0$ is solved, subject to Dirchlet conditions on the $U$ displacement vector. One advantage of the linear elasticity approach, is that regions of large E (modulus of elasticity) will be displaced as a solid body. Thus, an appropriate prescription of the distribution of E can be used to avoid severe mesh deformation in critical regions of the mesh. We employ a distribution of E which is inversely proportional to the cell volume[7] or to the distance from the deforming boundaries, thus relegating much of the mesh deformation to regions where the mesh is coarser and can sustain larger relative deformations. This turns out to be critical for avoiding invalid mesh cells in regions of high mesh stretching. Note that the approach adopted in reference[21] is based on the Navier equations, which govern the displacements in a continuous medium of constant modulus of elasticity, and therefore is not capable of simulating materials with variable E. On the other hand, in reference,[7] the inverse scaling of E with respect to the cell volume is achieved implicitly by omitting the Jacobian term in the integral equations. In our approach, we explicitly prescribe a distribution of E throughout the domain for increased control in difficult cases.

Because the discrete linear elasticity equations are assembled on an element basis, the edge-based data-structure of the flow solver is insufficient for assembling the linear elasticity equations in the presence of non-simplicial elements, and an element data-structure must be maintained as well.

## Acceleration Strategies

As mentioned previously, agglomeration multigrid methods are used to accelerate the solution of the non-linear problem arising at each time step of the unsteady flow equations. Agglomeration multigrid was originally developed as a steady-state solver for the Euler and Navier-Stokes equations on unstructured grids.[12,14,15,17] The idea of multigrid is to accelerate the solution on a fine grid by iteratively computing corrections to the fine grid problem on coarser grid levels where the cost of the iterations are lower, and the global error components are more easily reduced. Figures 1a-d show an example of the agglomeration multigrid procedure. Figure 1a shows the fine grid, while Figure 1b-d shows the 2nd, 3rd and 4th level coarse grid where each coarse cell is the combination of several fine cells. The agglomeration multigrid methods are also used to accelerate the solution of the mesh motion problems and share the same the coarse level meshes as the flow solver.



**a)** 1st level **b)** 2nd level

**c)** 3rd level **d)** 4th level

**e)** implicit lines **f)** lines(zoom in)

**Fig. 1 Illustration of agglomeration multigrid and implicit line construction.**

For high Reynolds number viscous flow, when highly stretched meshes are required to capture the thin boundary layer regions near the wall, the effectiveness of the multigrid approach degrades due to the anisotropic stiffness induced by the grid stretching. To relax this stiffness, an implicit line solution technique was introduced in.[13,16] The lines are constructed along the strong coupling direction in the mesh, as shown by the example depicted in Figure 1e-f for an unstructured mesh about a NACA0012 airfoil. In these regions, a block tridiagonal algorithm is used to solve all quantities along each line implicitly, thus replacing the simple explicit approach on all grid levels. A non-linear as well as a linear agglomeration multigrid algorithm based on line solvers has been used to solve the implicit time-integration problem for the flow equations. The various forms of the governing equations for mesh deformation presented above have also been solved using the line-based agglomeration multigrid algorithm, using the same line structures and coarse levels as for the flow solver.

For multigrid computations, the Gauss-Seidel iterative method provides excellent smoothing characteristics and is easily applied for two-dimensional problems on serial computers. However, for three dimensional problems, which must usually be executed on parallel computing hardware, the Gauss-Seidel method suffers from reduced parallel efficiency. The Chebyshev iterative method is an alternative smoother which delivers better smoothing rates than Jacobi methods, approaching the smoothing characteristics of Gauss-Seidel, without reduced parallel efficiency. Adams[1] investigated the performance of the Chebyshev method versus the Gauss-Seidel method for parallel multigrid smoothers, and demonstrated better overall efficiency using the Chebyshev smoother versus the Gauss-Seidel smoother for parallel computations. The Chebyshev iterative method belongs to the class of polynomial smoothers and can be written as

$$x^{n+1} = x^n + \sum_{0 \le k \le m} \alpha_k A^k (b - Ax^n) \qquad (19)$$

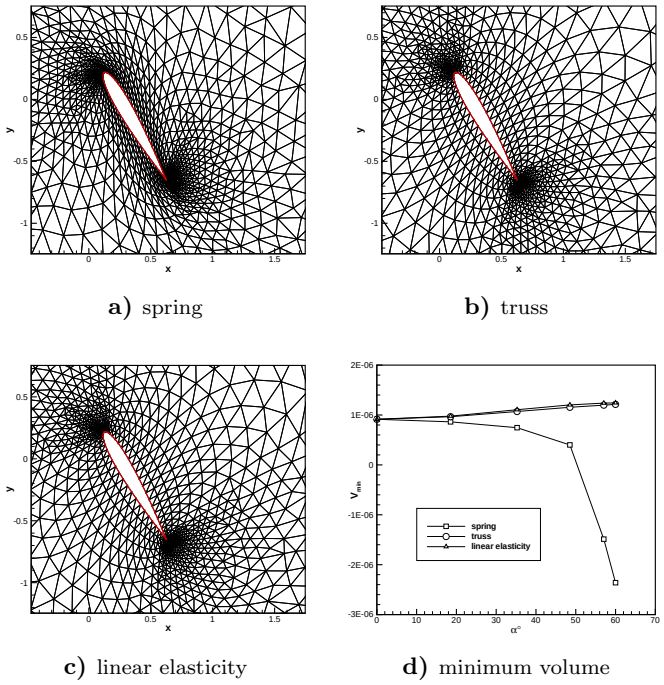
for the linear system $Ax = b$. When $m = 0$, the Chebyshev method reduces to the Jacobi iterative method. The coefficients of the Chebyshev method are computed by the Chebyshev polynomial and two other eigenvalues $\lambda_a$ and $\lambda_b$. The Chebyshev iteration is designed to minimize the errors for frequencies in the range between $\lambda_a$ and $\lambda_b$. When coupled with multigrid, $\lambda_a$ usually is set to the maximum eigenvalue and $\lambda_a$ is set to the value between the maximum and minimum eigenvalue in order to damp high frequency errors. One drawback of the Chebyshev method is that the maximum eigenvalue of the stiffness matrix needs to be calculated. However, in our case, the Chebyshev method is used in mesh motion calculations and the maximum eigenvalue can be computed and saved in the beginning of the simulation and used at each time step. The details of theory and implementation for the Chebyshev smoother can be found in references[1,23]

## Results and Discussion

### Mesh motion strategies

Several large deflection/deformation problems are presented to compare the quality of the meshes generated by the different mesh motion strategies. A NACA0012 airfoil is forced to oscillate around the quarter chord point with the angle of attack given as $\alpha = \alpha_{max} sin(ft)$, where $\alpha_{max}$ is set to 60°. Two meshes are considered for this configuration. The first mesh is an isotropic grid, suitable for inviscid flow simulations, which consists of 2139 nodes and is shown in Figure 2. The second one is a highly stretched mesh suitable for viscous flow calculations, which consists of 42960 nodes and is shown in Figure 3. The viscous mesh is highly stretched near the airfoil and the height of the first cell near the solid wall is $10^{-6}$ airfoil chords.

Figures 2a-c show the inviscid mesh configurations which result at the 60° pitching location for the spring analogy, truss analogy, and linear elasticity analogy mesh motion equations. The minimum cell area is plotted as a function of pitching angle in Figure 2d for the various




**a)** spring **b)** truss

**c)** linear elasticity **d)** minimum volume

**Fig. 2 Comparison of different mesh motion strategies for two-dimensional inviscid mesh case.**

mesh motion strategies. Because the spring analogy is incapable of reproducing solid body rotation displacements in regions near the airfoil, extreme mesh skewing results, leading to negative cell volumes prior to the airfoil reaching the 60° pitching angle. The other mesh motion methods are successful in avoiding negative cell areas up to the maximum pitching angle.

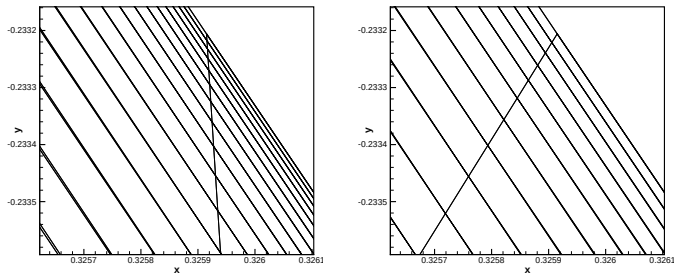
Figure 3a-d show the equivalent results for the anisotropic viscous mesh at a pitching angle of 60° for the spring analogy, truss analogy and linear elasticity analogy mesh motion equations. Two linear elasticity mesh motion cases are considered: one where the modulus of elasticity E is prescribed as a constant value throughout the domain, and another where E is prescribed as inversely proportional to the cell area. Negative area mesh cells (interior to the airfoil surface) are observed for the truss and constant E linear elasticity methods in Figures 3 b and c. The variable modulus of elasticity (E) linear elasticity approach is is shown to be the most robust method, since only this approach is capable of producing a valid mesh at the maximum pitching angle. Figure 3e-f shows a closeup of the mesh in the the mid-chord region near the airfoil wall, illustrating the relative mesh deformation which occurs near the airfoil surface for the spring analogy approach, as compared to the mesh produced by the linear elasticity approach, which is relatively undeformed in this region, since it tends to be displaced as a solid body translation and rotation under the variable E linear elasticity approach.

Figure 4a illustrates a three-dimensional unstructured mesh about a wing-body configuration in which a large spanwise deflection has been prescribed at the wing surface, thus inducing a deformation in the mesh, which was



**a)** spring **b)** truss

**c)** linear elasticity(constant E) **d)** linear elasticity



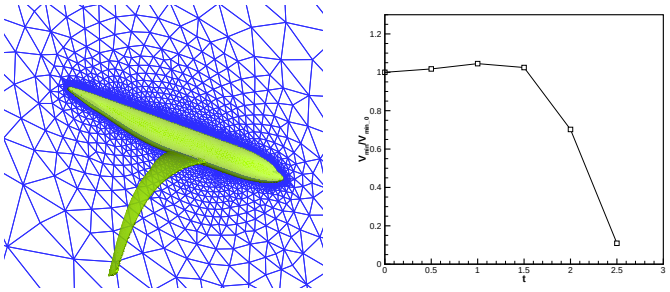
**e)** spring(zoom in) **f)** linear elasticity(zoom in)

**Fig. 3 Comparison of different mesh motion strategies for two-dimensional viscous mesh case.**

generated with the wing in the original horizontal position. This mesh is highly stretched near the aircraft surface, with a normal spacing of approximately $10^{-6}$ chords on the wing surface, and contains a total of 473,025 vertices. The linear elasticity method (with E prescribed as inversely proportional to the cell volume) was the only method capable of producing a valid mesh for this case. Figure 4b shows the minimum cell volume in the mesh vs. the time t. The time t=2.5 corresponds the wing position shown in Figure 4a.

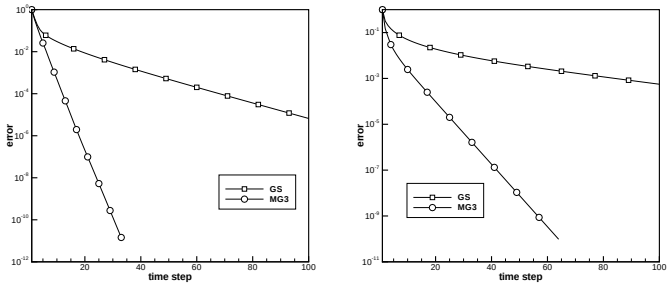
### Convergence of the mesh motion equations

The convergence of the mesh motion equations for the inviscid two-dimensional pitching airfoil problem is depicted in Figures 5a-b, for the spring analogy equations, and the linear elasticity equations (with variable E), respectively. A block Gauss-Seidel smoothing approach is used either on the fine grid as a solver, or as a smoother on each grid level of the multigrid sequence. The residuals are reduced by 10 orders of magnitude in these examples in order to examine the effectiveness of the solution strategies, although such stringent convergence criteria would nor-



**a)** **b)**

**Fig. 4 Illustration of three-dimensional viscous mesh deformation problem and minimum cell size as a function of spanwise deflection.**

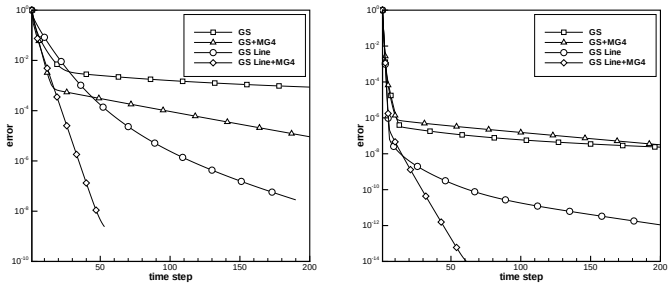



**a)** spring **b)** linear elasticity

**Fig. 5 Convergence history for mesh motion strategies for two-dimensional inviscid grid problem.**

mally not be required in the context of a dynamic mesh flow simulation. A three-level multigrid scheme using 6 Gauss-Seidel smoothing passes on each level achieves over 10 orders of residual reduction for the spring analogy equations for this case. This rapid multigrid convergence is expected, since the spring analogy equations correspond to scaled Poisson equations, which are easily solved by standard multigrid methods. The multigrid scheme achieves close to an order of magnitude speedup over the single grid scheme in this case. The convergence of the linear elasticity equations without multigrid is substantially slower than that of the spring analogy equations, even for this relatively simple test case. However, the multigrid algorithm applied to these equations achieves 10 orders of residual reduction in 65 multigrid cycles, which is only a factor of 2 slower than that achieved for the spring analogy equations.

The convergence rates for the spring analogy and linear elasticity (with variable E) analogy equations for the viscous airfoil grid are depicted in Figures 6a-b. In this case, a four level multigrid scheme is employed, with 6 point or line Gauss-Seidel smoothing passes on each grid level. For both mesh motion analogies, the single grid point-Gauss-Seidel and even the multigrid point-Gauss-Seidel solvers converge relatively slowly, due to the stiffness associated with the highly stretched cells in the boundary layer and wake regions for this viscous mesh. In fact, the single grid line-Gauss-Seidel approach converges more effectively than the point-Gauss-Seidel approach with multigrid. However, the line-Gauss-Seidel multigrid scheme provides the fastest



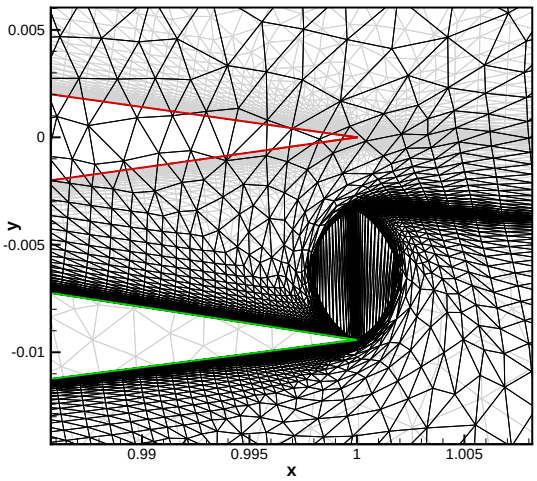

**a)** spring **b)** linear elasticity

**Fig. 6 Convergence history for mesh motion strategies for two-dimensional viscous grid problem.**

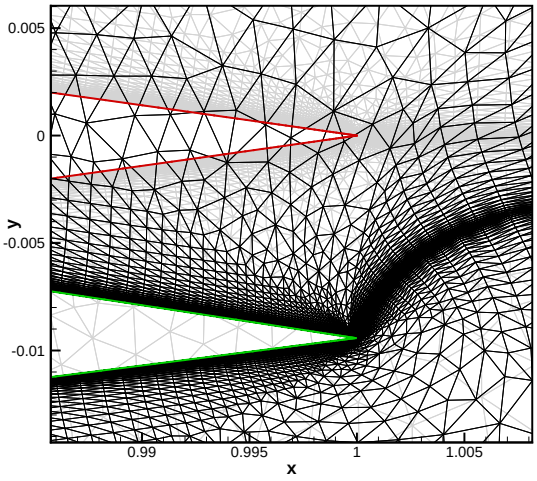
convergence, reducing the residuals by over 9 orders of magnitude in both cases in 50 cycles. The convergence rates achieved by this approach for the viscous anisotropic mesh are close to those achieved on the inviscid isotropic mesh of the previous example.

The effectiveness of the line solver for the mesh motion equations is demonstrated in Figures 7a-c, where the airfoil trailing edge undergoes a vertical displacement from $y = 0$ to $y = -0.01$. After ten point-Gauss-Seidel multigrid iterations, the highly stretched wake cells near the trailing edge are severely distorted, as their computed displacements remain far from convergence. However, after ten single grid line-Gauss-Seidel passes, the mesh structure in this region is much better behaved, although still not fully converged. The effect of the line solver is to rapidly propagate surface displacements through these boundary layer and wake cells. In fact, boundary layer cells in regions of low surface curvature are often displaced approximately as a solid body, due to the instantaneous propagation of the surface displacements along the lines by the line solver, often producing a valid mesh in these regions in a single line solver pass. In the current example, after ten multigrid cycles using the line-Gauss-Seidel smoother, the grid point displacements are nearly fully converged, as shown in Figure 7c.

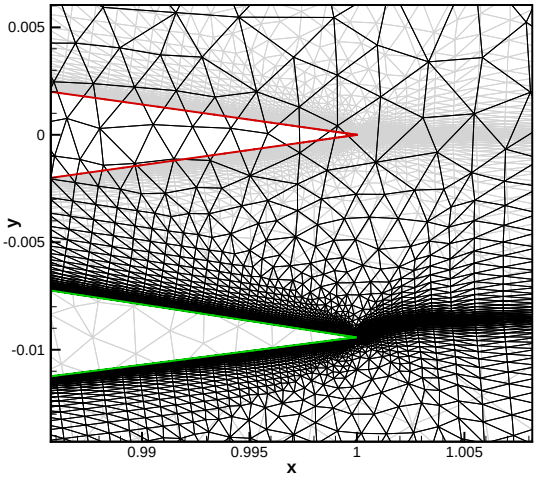
Figure 8 depicts the convergence achieved for the linear elasticity equations for the same case, using point and line-Jacobi smoothers in the place of Gauss-Seidel smoothers. The convergence of the Jacobi smoothers is substantially slower than that achieved with the Gauss-Seidel smoothers for these equations, more so than the typical factor of 2 observed for Poisson type equations. This is due to the fact that the discrete linear elasticity equations are not necessarily diagonally dominant, and Gauss-Seidel provides much better smoothing characteristics than Jacobi or under-relaxed Jacobi methods. The convergence of the Chebyshev smoother, optimized for high-frequency damping, is shown to provide convergence of the multigrid algorithm which is only slightly slower than the Gauss-Seidel approach. While the line-Gauss-Seidel driven multigrid scheme achieves a residual reduction of 14 orders of magnitude in 60 iterations, the line-Chebyshev driven multigrid scheme achieves the same residual reduction in just 80 iterations. The advantage of the Chebyshev smoother is that




**a)** point solver + MG4

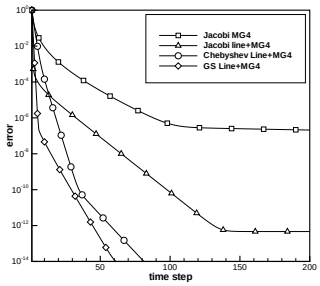



**b)** line solver + MG1

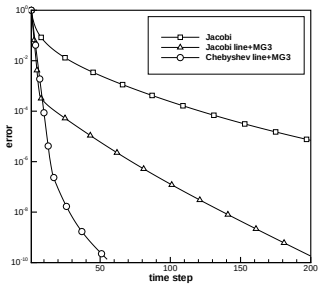



**c)** line solver + MG4

**Fig. 7 Comparison of a deforming mesh configuration after 10 multigrid cycles using point-smoother, after 10 single grid cycles using line smoother, and after 10 multigrid cycles using line smoother.**




**a)** linear elasticity, viscous grid




**b)** linear elasticity, 3D grid

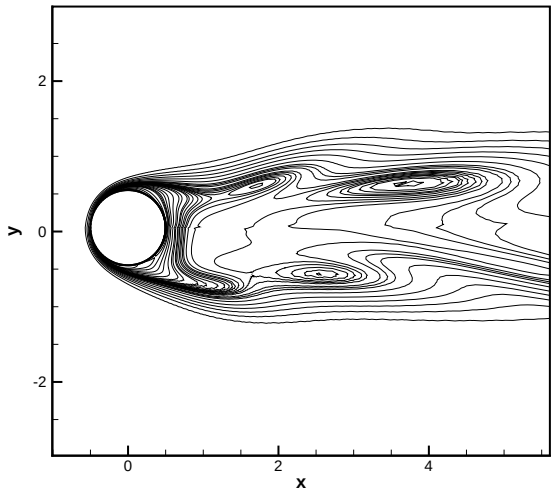
**Fig. 8 Convergence history for Jacobi, Gauss-Seidel and Chebyshev smoothers for linear elasticity mesh motion analogy for two-dimensional and three-dimensional test case.**

it is more fully parallelizable than a Gauss-Seidel strategy.

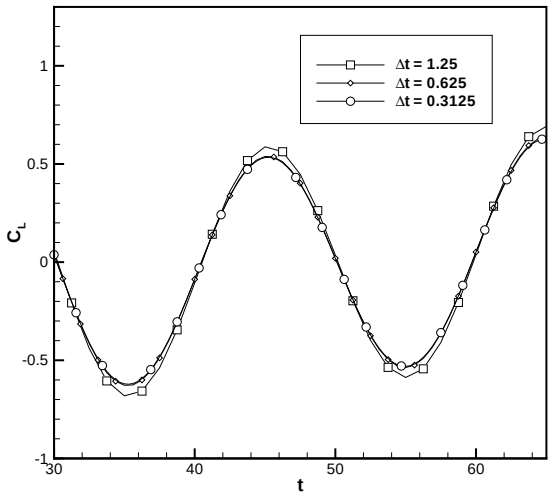
For there reasons, only the Jacobi and Chebyshev smoothers are implemented in the three-dimensional case. Figure 8b compares the convergence of the three-dimensional deforming viscous mesh case, using line-Jacobi, line-Jacobi driven multigrid, and line-Chebyshev driven multigrid, using a three-level multigrid scheme. The convergence rate of the line-Jacobi multigrid scheme is approximately twice as fast as the single grid line-Jacobi approach, while the line-Chebyshev multigrid scheme is four times faster than the line-Jacobi multigrid scheme, illustrating the superior high-frequency damping properties of the Chebyshev smoother. At this rate, the solution of the mesh motion equations for linear elasticity represents a small fraction of the overall solution time within an unsteady flow solution problem.

### Time-Accuracy Validation for Dynamic Unstructured Mesh Problem

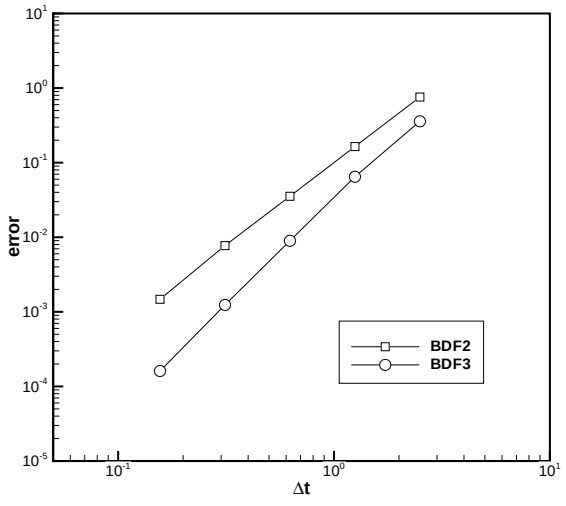
The temporal accuracy of the GCL compliant 2nd-order (BDF2) and third-order (BDF3) backwards-difference time-integration schemes is examined for a two-dimensional dynamic mesh viscous flow problem. The test case consists of an oscillating cylinder of unit diameter, which undergoes a vertical displacement given by $y = Asin(ft)$, where $A = 0.1$ and $f = 0.1\pi$. The two-dimensional unstructured mesh consists of 19012 nodes and 37632 triangles. The mesh on the cylinder surface is displaced according to the prescribed cylinder motion, while the outer boundary of the domain is held fixed. The mesh deformation is modeled using the spring analogy approach and the Gauss-Seidel multigrid method is used to solve the mesh motion equations at each time step. The freestream Mach number of the flow is $M = 0.2$, and the Reynolds number is $Re = 185$. Figure 9a shows the entropy contours for this case at the time step corresponding to $t = 65$. Figure 9b shows the comparison in the lift coefficients as a function of physical time, for the third-order BDF3 scheme using different time steps, showing good agreement for the cases computed using the two finest time steps. In Figure 9c, the temporal error as a function of the physical time step is plotted in log-log format. The tem-




**a)** entropy contours




**b)**




**c)**

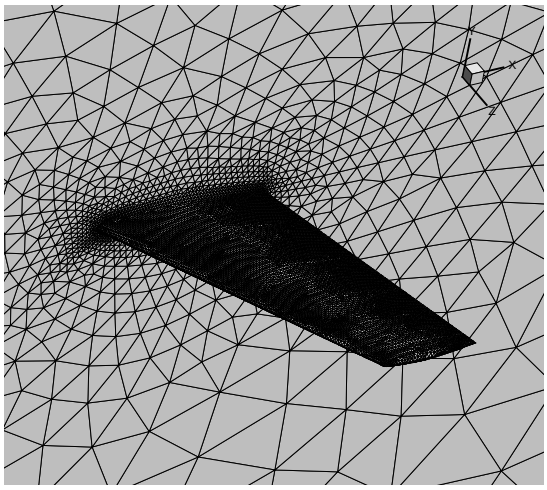
**Fig. 9 Flow solution and measured temporal error as a function of time step size for BDF2 and BDF3 schemes for two-dimensional oscillating cylinder.**

poral error is defined as the difference in the lift coefficient at $t = 65$ between the considered solution and a reference solution computed with a smaller time step of 0.078 using the BDF3 scheme. The slope of the BDF2 scheme in the figure is 2.0, indicating that design accuracy is achieved, while the slope of the BDF3 error curve is 2.85, which is close to the design accuracy of 3. To achieve an error of $5 \times 10^{-3}$, the time steps required for BDF2 and BDF3 are 0.258 and 0.512, respectively. Considering that the CPU time for BDF2 and BDF3 are nearly identical, the BDF3 scheme is seen to be approximately twice as efficient for this case, with larger benefits occurring for lower temporal error tolerances.

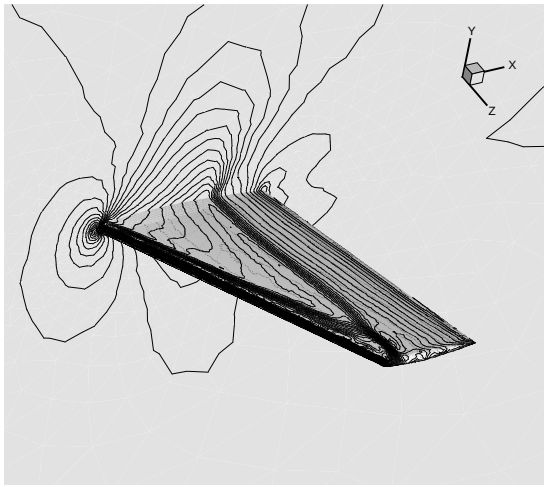
### Three-Dimensional Inviscid Unsteady Flow Simulation

A three-dimensional unsteady flow simulation with a dynamically deforming mesh is computed for an ONERA M6 wing undergoing a forced twisting motion. The three-dimensional unstructured mesh for the M6 wing consists of 53961 vertices and 287962 tetrahedrons, and is depicted in Figure 10a. A four level agglomeration multigrid algorithm is used, where the same agglomerated levels are shared for the flow and dynamic mesh motion equation solvers. The freestream Mach number is $M = 0.84$, and the initial incidence is $\alpha = 3.06^o$. The computation is divided to two parts. First, the steady-state flow over the stationary wing is computed. This flow-field, which is illustrated in Figure 10b, is then used to initialize the twisting wing calculation, during which the wing is forced to twist around the quarter chord line, with the angle of attack given as $\alpha = \alpha_m sin(ft)$, where $\alpha_m$ is set to 2.51° at the wing tip, and $\alpha_m$ varies linearly from this value to zero at the wing root, and the reduced frequency is 0.1628.

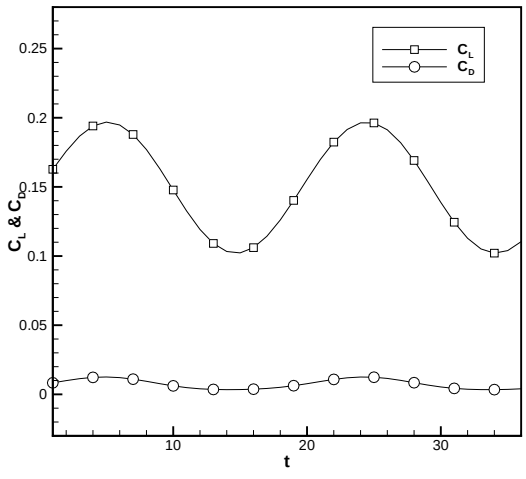
The computed unsteady lift and drag coefficients for the twisting wing are shown in Figure 10c. These calculations are performed using the second-order backwards difference scheme (BDF2), with a time step of $\Delta t = 2.0$. At each physical time-step, the non-linear residuals for the flow equations are converged using either the linear multigrid scheme or the non-linear multigrid scheme, and the spring analogy mesh-motion equations are converged using a Jacobi-driven multigrid scheme with 3 Jacobi smoothing passes on each level. Figures 11a-b and 12a-b examine the convergence efficiency of the linear multigrid scheme as a function of the number of smoothing passes and the number of linear multigrid cycles per non-linear update. In Figure 11a-b, using two linear multigrid cycles per non-linear update, the optimal convergence of the non-linear flow solution in terms of cpu-time is obtained with 3 smoothing passes on each grid level (even though larger numbers of smoothing passes produce faster convergence rates on a multigrid cycle basis). Note that the system diverges for insufficient numbers of smoothing passes. In Figure 12a-b the non-linear convergence per multigrid cycle is seen to asymptote to a lower bound as the number of linear multigrid cycles is increased, due to the fact that the



a)

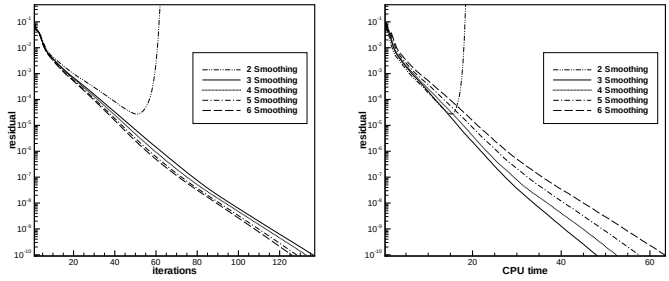


b)



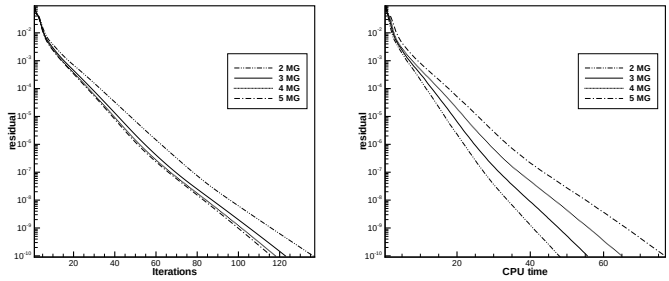

c)

**Fig. 10 Unstructured mesh and computed density contours at initial position, and time-varying lift and drag coefficients for twisting ONERA M6 wing inviscid test case.**



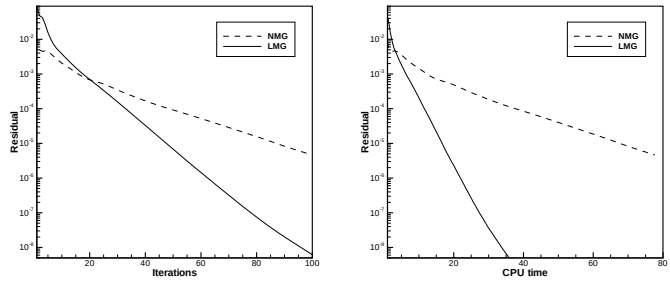

a) b)

**Fig. 11 Effect of the number of smoothing iterations on overall convergence of linear multigrid scheme for twisting ONERA M6 wing case.**




a) b)

**Fig. 12 Effect of number of linear multigrid cycles (per non-linear update) on overall convergence for twisting ONERA M6 wing case.**




a) b)

**Fig. 13 Comparison of non-linear multigrid and linear multigrid convergence in terms of non-linear iterations and cpu time for twisting ONERA M6 wing case.**

linear multigrid solver operates on an approximate (first-order accurate) Jacobian of the non-linear flow equations (i.e. linear multigrid is used to drive an approximate Newton scheme of the non-linear flow equations). In terms of cpu-time, the optimal convergence efficiency is obtained with 2 linear multigrid cycles at each non-linear update. Using the optimal parameters for the linear multigrid approach, the convergence efficiency of the non-linear flow residual at a given time step is compared for the optimized linear multigrid method, and the non-linear (FAS) multigrid method, which uses the same coarse levels, and a three-stage multi-stage Jacobi-preconditioned smoother on each grid level. In Figure 13a, the linear multigrid scheme is seen to be approximately twice as fast as the non-linear multigrid scheme, based on the number of non-

linear residual updates, which is expected, since the linear multigrid scheme performs twice as many multigrid cycles as the non-linear multigrid scheme per non-linear update, and the two schemes can be expected to converge at the same asymptotic rates per multigrid cycle.[17] However, in terms of cpu time, the linear multigrid scheme is substantially faster than the non-linear scheme, due to the lower cost of the linear multigrid iterations. Table 1 shows the required cpu time for a non-linear flow-solution time step solution, using the non-linear (FAS) multigrid approach, the optimized linear multigrid approach, and the cpu time required to converge the mesh motion equations to the equivalent level of accuracy as the flow equations (eight orders of residual reduction). The table shows that the linear multigrid approach is more than four times as efficient as the non-linear multigrid solver for the flow equations, while the mesh motion equations require of the order of 10% or less of the time required to solve the unsteady flow solution problem at a given time step. In practice, less stringent convergence tolerances may be employed for unsteady dynamic mesh flow simulations, but the relative performances of these solvers should remain approximately the same.

| NMG | LMG | Mesh MG |
|---|---|---|
| 159s | 34s | 3.48s |

**Table 1 A comparison of the cpu time required for an 8 order of magnitude reduction in the unsteady flow residual at a given time step, using the non-linear multigrid (NMG) scheme, and the linear multigrid scheme (LMG), and the cpu time required for an equivalent convergence level of the mesh motion equations using multigrid.**

## Conclusions

Techniques for the efficient solution of unsteady flow dynamic mesh motion problems have been developed. A third-order backwards difference time-integration scheme which respects the discrete GCL has been developed and validated on a two-dimensional viscous flow problem. Various mesh-motion strategies have been investigated, and the linear elasticity method, using a variable prescription of the modulus of elasticity has been found to be the most robust method. Agglomeration multigrid methods are developed for solving the flow and mesh motion equations, using the same coarse agglomerated levels for both sets of equations. A line-implicit driven multigrid approach results in an effective solver for the mesh-motion equations on highly-stretched viscous meshes, with line-Gauss-Seidel or line-Chebyshev smoothers proving particularly effective for the linear elasticity mesh motion equations. For the flow equations, a linear multigrid approach operating on an approximate Newton method for the full non-linear problem outperforms the use of a non-linear (FAS) multigrid solver applied directly to the flow equations at each time step. The combined solution of the mesh motion and flow solution equations is demonstrated on a three-dimensional inviscid flow problem, where it is shown that the mesh motion solution requires a small fraction of the overall cpu time at each time step. Future work is underway to derive and implement higher-order implicit Runge-Kutta time-integration strategies which respect the GCL. These techniques will also be applied to large three-dimensional viscous turbulent flow simulations, for the purposes of simulating aeroelastic effects.

## Acknowledgments

The first author would like to thank Dr. M. Adams for help on the Chebyshev method. This work was supported in part by grant NNL04AA63G from the NASA Langley Research center, and by contract number Z205800 from the Ball Aerospace Co.

## References

[1]M. F. Adams, M. Brezina, J. J. Hu, and R. S. Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. *Journal of Computational Physics*, 188(2):593–610, 2003.

[2]T. Baker and P. A. Cavallo. Dynamic adaptation for deforming tetrahedral meshes. AIAA 1999-3253, 1999.

[3]J. T. Batina. Unsteady euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal*, 28(8):1381–1288, 1990.

[4]H. Bijl, M. H. Carpenter, and V. N. Vatsa. Time integration schemes for the unsteady navier-stokes equations. AIAA 2001-2612, 2001.

[5]C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional spring for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering*, 163:231–245, 1998.

[6]H. Guillard and C. Farhat. On the significance of the geometric conservation law for flow computations on moving meshes. *Computer Methods in Applied Mechanics and Engineering*, 190:1467–1482, 2000.

[7]A. A. Johnson and T. E. Tezduyar. Simulation of multiple spheres falling in a liquid-filled tube. *Computer Methods in Applied Mechanics and Engineering*, 134:351–373, 1996.

[8]G. Jothiprasad, D. J. Mavriplis, and D. A. Caughey. Higher-order time integration schemes for the unsteady navier stokes equations on unstructured meshes. *Journal of Computational Physics*, 191:542–566, 2003.

[9]B. Koobus and C. Farhat. On the implicit time integration of semi-discrete viscous fluxes on unstructured dynamic meshes. *International Journal for Numerical Methods in Fluids*, 29:975–996, 1999.

[10]B. Koobus and C. Farhat. Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes. *Computer Methods in Applied Mechanics and Engineering*, 170:103–129, 1999.

[11]M. Lesoinne and C. Farhat. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering*, 134:71–90, 1996.

[12]D. J. Mavriplis. Multigrid techniques for unstructured meshes. In *VKI Lecture Series*, number 1995-02 in VKI-LS. 1995.

[13]D. J. Mavriplis. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. Technical Report 98-6, ICASE, 1998.

[14]D. J. Mavriplis. On convergence acceleration techniques for unstructured meshes. AIAA 1998-2966, 1998.

[15]D. J. Mavriplis. Three-dimensional high-lift analysis using a parallel unstructured multigrid solver. Technical Report 98-20, ICASE, 1998.

[16]D. J. Mavriplis. Directional agglomeration multigrid techniques for high-reynolds number viscous flow. *AIAA Journal*, 27(10):1222–1230, 1999.

[17]D. J. Mavriplis. An assessment of linear versus non-linear multigrid methods for unstructured mesh solvers. *Journal of Computational Physics*, 175:302–325, 2002.

[18]D. J. Mavriplis and V. Venkatakrishnan. A unified multigrid solver for the navier-stokes equations on mixed element meshes. *International Journal of Computational Fluid Dynamics*, (8):247–263, 1997.

[19]D. J. Mavriplis and Z. Yang. Achieving higher-order time accuracy for dynamic unstructured mesh fluid flow simulations: Role of the GCL. submit to AIAA, 2005.

[20]M. Murayama, K. Nakahashi, and K. Matsushima. Unstructured dynamic mesh for large movement and deformation. AIAA 2002-0122, 2002.

[21]E. J. Nielsen and W. K. Anderson. Recent improvements in aerodynamic design optimization on unstructured meshes. AIAA 2001-0596, 2001.

[22]J. N. Reddy. *An Introduction To The Finite Element Method*. McGraw-Hill, 1993.

[23]Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, 1996.

[24]P. D. Thomas and C. K. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17(10):1030–1037, 1979.

[25]V. Venkatakrishnan and D. J. Mavriplis. Implicit method for the computation of unsteady flows on unstructured grids. *Journal of Computational Physics*, 127:380–397, 1996.

## Appendix

Apply a third order backward difference scheme

$$\sum_{k=1}^{-2} \alpha_{n+k} V^{n+k} u^{n+k} = \Delta t^n \sum R(u^{n+1}, n, \dot{x}) \qquad (20)$$

The above third order scheme satisfies the GCL if

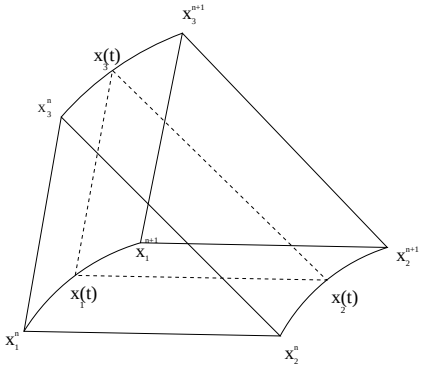
$$\begin{aligned} \sum_{k=1}^{-2} \alpha_{n+k} V^{n+k} u_\infty &= \Delta t^n \sum R(u_\infty, n, \dot{x}) \\ &= u_\infty \Delta t^n \sum G(n, \dot{x}) \\ &= u_\infty \int_{t^n}^{t^{n+1}} \int_\Delta \dot{x} \cdot \hat{n} ds dt \end{aligned} \qquad (21)$$

For the right hand side, let

$$\begin{aligned} I_\Delta^n &= \int_{t^n}^{t^{n+1}} \int_\Delta \dot{x} \cdot \hat{n} ds dt \\ x(t) &= \eta_1 x_1(t) + \eta_2 x_2(t) + (1-\eta_1-\eta_2) x_3(t) \\ \dot{x}(t) &= \eta_1 \dot{x}_1(t) + \eta_2 \dot{x}_2(t) + (1-\eta_1-\eta_2) \dot{x}_3(t) \end{aligned} \qquad (22)$$

where

$$\begin{aligned} &\eta_1 \in [0,1], \quad \eta_2 \in [0, 1-\eta_1], \quad t \in [t^n, t^{n+1}] \\ &x_i(t) = \xi_{n+1}(t) x_i^{n+1} + \xi_n(t) x_i^n + \xi_{n-1}(t) x_i^{n-1} \\ &\qquad -(1 - \xi_{n-1}(t) - \xi_n(t) - \xi_{n+1}(t)) x_i^{n-2}, i = 1,2,3 \\ &\xi_{n+1}(t^{n+1}) = 1, \quad \xi_{n+1}(t^n) = 0 \\ &\xi_n(t^{n+1}) = 0, \quad \xi_n(t^n) = 1 \\ &\xi_{n-1}(t^{n+1}) = 0, \quad \xi_{n-1}(t^n) = 0 \end{aligned}$$




Then

$$\begin{aligned} I_\Delta^n &= \int_{t^n}^{t^{n+1}} \int_0^1 \int_0^{1-\eta_1} (\eta_1 \dot{x}_1 + \eta_2 \dot{x}_2 + (1-\eta_1-\eta_2)\dot{x}_3) \\ &\quad \cdot (\Delta x_{13} \times \Delta x_{23}) d\eta_2 d\eta_1 dt \\ &= \int_{t^n}^{t^{n+1}} \frac{1}{6} (\dot{x}_1 + \dot{x}_2 + \dot{x}_3) \cdot (\Delta x_{13} \times \Delta x_{23}) dt \\ &= \frac{1}{6} \int_{t^n}^{t^{n+1}} f \cdot g dt \end{aligned} \qquad (23)$$

where

$$\begin{aligned} &\Delta x_{13} = x_3 - x_1, \quad \Delta x_{23} = x_3 - x_2 \\ f &= \dot{x}_1 + \dot{x}_2 + \dot{x}_3 \\ &= \dot{\xi}_{n+1}(x_1^{n+1} + x_2^{n+1} + x_3^{n+1}) \\ &+ \dot{\xi}_n(x_1^n + x_2^n + x_3^n) \\ &+ \dot{\xi}_{n-1}(x_1^{n-1} + x_2^{n-1} + x_3^{n-1}) \\ &- (\dot{\xi}_{n+1} + \dot{\xi}_n + \dot{\xi}_{n-1})(x_1^{n-2} + x_2^{n-2} + x_3^{n-2}) \\ g = &[\xi_{n+1}\Delta x_{13}^{n+1} + \xi_n \Delta x_{13}^n + \xi_{n-1}\Delta x_{13}^{n-1} + \\ &(1 - \xi_{n-1} - \xi_n - \xi_{n+1})\Delta x_{13}^{n-2}] \times \\ &[\xi_{n+1}\Delta x_{23}^{n+1} + \xi_n \Delta x_{23}^n + \xi_{n-1}\Delta x_{23}^{n-1} + \\ &(1 - \xi_{n-1} - \xi_n - \xi_{n+1})\Delta x_{23}^{n-2}] \end{aligned}$$

Using a six-point integration rule to approximate the $I_\Delta^n$

$$\begin{aligned} I_\Delta^n &\approx \widetilde{I_\Delta^n} = \frac{\Delta t^n}{6} \sum_{k=1}^{6} \omega_k f^k \cdot g^k \\ f^k &= \dot{\xi}_{n+1}^{pk}(x_1^{n+1} + x_2^{n+1} + x_3^{n+1}) + \\ &\quad \dot{\xi}_n^{pk}(x_1^n + x_2^n + x_3^n) + \\ &\quad \dot{\xi}_{n-1}^{pk}(x_1^{n-1} + x_2^{n-1} + x_3^{n-1}) + \\ &\quad (\dot{\xi}_{n+1}^{pk} + \dot{\xi}_n^{pk} + \dot{\xi}_{n-1}^{pk})(x_1^{n-2} + x_2^{n-2} + x_2^{n-2}) \\ g^k &= [\xi_{n+1}^{pk}\Delta x_{13}^{n+1} + \xi_n^{pk}\Delta x_{13}^n + \xi_{n-1}^{pk}\Delta x_{13}^{n-1} + \\ &\quad (1 - \xi_{n-1}^{pk} - \xi_n^{pk} - \xi_{n+1}^{pk})\Delta x_{13}^{n-2}] \times \\ &\quad [\xi_{n+1}^{pk}\Delta x_{23}^{n+1} + \xi_n^{pk}\Delta x_{23}^n + \xi_{n-1}^{pk}\Delta x_{23}^{n-1} + \\ &\quad (1 - \xi_{n-1}^{pk} - \xi_n^{pk} - \xi_{n+1}^{pk})\Delta x_{23}^{n-2}] \\ \xi_{(\bullet)}^{pk} &= \xi_{(\bullet)}(t^{pk}) \\ \dot{\xi}_{(\bullet)}^{pk} &= \dot{\xi}_{(\bullet)}(t^{pk}) \end{aligned}$$

For the left hand side, let

$$\begin{aligned} J_\Delta^n &= \alpha_{n+1} V^{n+1} + \alpha_n V^n + \alpha_{n-1} V^{n-1} + \alpha_{n-2} V^{n-2} \\ &= \alpha_{n+1}(V^{n+1} - V^n) + (\alpha_{n+1} + \alpha_n)(V^n - V^{n-1}) - \end{aligned}$$

$$
\begin{aligned}
& \alpha_{n-2}(V^{n-1} - V^{n-2}) \qquad \text{where } \alpha_{n+1} = \tfrac{11}{6},\ \alpha_n = \tfrac{-18}{6},\ \alpha_{n-1} = \tfrac{9}{6},\ \alpha_{n-2} = \tfrac{-2}{6} \\
= \ & \alpha_{n+1} \int_{t^n}^{t^{n+1}} \int_\Delta \dot{x} \cdot \hat{x} ds dt + \\
& (\alpha_{n+1} + \alpha_n) \int_{t^{n-1}}^{t^n} \int_\Delta \dot{x} \cdot \hat{n} ds dt - \\
& \alpha_{n-2} \int_{t^{n-2}}^{t^{n-1}} \int_\Delta \dot{x} \cdot \hat{n} ds dt
\end{aligned}
$$

According to,[11] $\int_{t^n}^{t^{n+1}} \int_\Delta \dot{x} \cdot \hat{n} ds dt$, $\int_{t^{n-1}}^{t^n} \int_\Delta \dot{x} \cdot \hat{n} ds dt$, $\int_{t^{n-2}}^{t^{n-1}} \int_\Delta \dot{x} \cdot \hat{n} ds dt$ can be evaluated exactly, so

$$
\begin{aligned}
J_\Delta^n = \ & \tfrac{\alpha_{n+1}}{18}(\Delta x_1^n + \Delta x_2^n + \Delta x_3^n) \cdot \\
& (\Delta x_{13}^{n+1} \times \Delta x_{23}^{n+1} + \tfrac{1}{2}\Delta x_{13}^n \times \Delta x_{23}^{n+1} + \\
& \Delta x_{13}^n \times \Delta x_{23}^n + \tfrac{1}{2}\Delta x_{13}^{n+1} \times \Delta x_{23}^n) + \\
& \tfrac{\alpha_{n+1}+\alpha_n}{18}(\Delta x_1^{n-1} + \Delta x_2^{n-1} + \Delta x_3^{n-1}) \cdot \\
& (\Delta x_{13}^n \times \Delta x_{23}^n + \tfrac{1}{2}\Delta x_{13}^{n-1} \times \Delta x_{23}^n + \\
& \Delta x_{13}^{n-1} \times \Delta x_{23}^{n-1} + \tfrac{1}{2}\Delta x_{13}^n \times \Delta x_{23}^{n-1}) + \\
& \tfrac{\alpha_{n-2}}{18}(\Delta x_1^{n-2} + \Delta x_2^{n-2} + \Delta x_3^{n-2}) \cdot \\
& (\Delta x_{13}^{n-1} \times \Delta x_{23}^{n-1} + \tfrac{1}{2}\Delta x_{13}^{n-2} \times \Delta x_{23}^{n-1} + \\
& \Delta x_{13}^{n-2} \times \Delta x_{23}^{n-2} + \tfrac{1}{2}\Delta x_{13}^{n-1} \times \Delta x_{23}^{n-2})
\end{aligned}
$$

where $\Delta x_i^m = x_i^{m+1} - x_i^m$, $i = 1, 2, 3$ and $m = n, n-1, n-2$

To satisfy the GCL, set

$$
\widetilde{I}_\Delta^n = J_\Delta^n \tag{24}
$$

The relations below satisfy the DGCL above

$$
\begin{aligned}
& \delta_1 = \tfrac{1}{2}(1 + \tfrac{1}{\sqrt{3}}) \quad \delta_1 = \tfrac{1}{2}(1 + \tfrac{1}{\sqrt{3}}) \\
& \xi_{n+1}^{p1} = \delta_1 \quad \xi_{n+1}^{p2} = \delta_2 \quad \xi_{n+1}^{p3} = 0 \\
& \xi_{n+1}^{p4} = 0 \quad \xi_{n+1}^{p5} = 0 \quad \xi_{n+1}^{p6} = 0 \\
& \xi_n^{p1} = \delta_2 \quad \xi_n^{p2} = \delta_1 \quad \xi_n^{p3} = \delta_1 \\
& \xi_n^{p4} = \delta_2 \quad \xi_n^{p5} = 0 \quad \xi_n^{p6} = 0 \\
& \xi_{n-1}^{p1} = 0 \quad \xi_{n-1}^{p2} = 0 \quad \xi_{n-1}^{p3} = \delta_2 \\
& \xi_{n-1}^{p4} = \delta_1 \quad \xi_{n-1}^{p5} = \delta_1 \quad \xi_{n-1}^{p6} = \delta_2 \\
& \dot{\xi}_{n+1}^{p1} = \tfrac{1}{\Delta t} \quad \dot{\xi}_{n+1}^{p2} = \tfrac{1}{\Delta t} \quad \dot{\xi}_{n+1}^{p3} = 0 \\
& \dot{\xi}_{n+1}^{p4} = 0 \quad \dot{\xi}_{n+1}^{p5} = 0 \quad \dot{\xi}_{n+1}^{p6} = 0 \\
& \dot{\xi}_n^{p1} = -\tfrac{1}{\Delta t} \quad \dot{\xi}_n^{p2} = -\tfrac{1}{\Delta t} \quad \dot{\xi}_n^{p3} = \tfrac{1}{\Delta t} \\
& \dot{\xi}_n^{p4} = \tfrac{1}{\Delta t} \quad \dot{\xi}_n^{p5} = 0 \quad \dot{\xi}_n^{p6} = 0 \\
& \dot{\xi}_{n-1}^{p1} = 0 \quad \dot{\xi}_{n-1}^{p2} = 0 \quad \dot{\xi}_{n-1}^{p3} = -\tfrac{1}{\Delta t} \\
& \dot{\xi}_{n-1}^{p4} = -\tfrac{1}{\Delta t} \quad \dot{\xi}_{n-1}^{p5} = \tfrac{1}{\Delta t} \quad \dot{\xi}_{n-1}^{p6} = \tfrac{1}{\Delta t} \\
& \omega_1 = \omega_2 = \tfrac{1}{2}\alpha_{n+1} \\
& \omega_3 = \omega_4 = \tfrac{1}{2}(\alpha_{n+1} + \alpha_n) \\
& \omega_5 = \omega_6 = -\tfrac{1}{2}\alpha_{n-2}
\end{aligned}
$$

Then

$$
\begin{aligned}
& x_j^{p1} = \delta_1 x_j^{n+1} + \delta_2 x_j^n \qquad \dot{x}_j^{p1} = \dot{x}_j^{p2} = \tfrac{x_j^{n+1} - x_j^n}{\Delta t} \\
& x_j^{p2} = \delta_2 x_j^{n+1} + \delta_1 x_j^n \\
& x_j^{p3} = \delta_1 x_j^n + \delta_2 x_j^{n-1} \qquad \dot{x}_j^{p3} = \dot{x}_j^{p4} = \tfrac{x_j^n - x_j^{n-1}}{\Delta t} \\
& x_j^{p4} = \delta_2 x_j^n + \delta_1 x_j^{n-1} \\
& x_j^{p5} = \delta_1 x_j^{n-1} + \delta_2 x_j^{n-2} \quad \dot{x}_j^{p5} = \dot{x}_j^{p6} = \tfrac{x_j^{n-1} - x_j^{n-2}}{\Delta t} \\
& x_j^{p6} = \delta_2 x_j^{n-1} + \delta_1 x_j^{n-2}
\end{aligned}
$$

where $j = 1, 2, 3$

$$
\begin{aligned}
\vec{n} = \ & \tfrac{1}{2}\alpha_{n+1}(\vec{n}_1 + \vec{n}_2) - \tfrac{1}{2}\alpha_{n-2}(\vec{n}_5 + \vec{n}_6) \\
& \tfrac{1}{2}(\alpha_{n+1} + \alpha_n)(\vec{n}_3 + \vec{n}_4) \\
\dot{x} = \ & \tfrac{1}{2\Delta t}\alpha_{n+1}(x^{n+1} - x^n) \cdot (\vec{n}_1 + \vec{n}_2) + \\
& \tfrac{1}{2\Delta t}(\alpha_{n+1} + \alpha_n)(x^n - x^{n-1}) \cdot (\vec{n}_3 + \vec{n}_4) + \\
& \tfrac{1}{2\Delta t}\alpha_{n-2}(x^{n-1} - x^{n-2}) \cdot (\vec{n}_5 + \vec{n}_6)
\end{aligned}
$$