43rd Aerospace Sciences Meeting and Exhibit, Jan 10-13, 2005, Reno, NV

Unstructured Algorithms for Inviscid and Viscous Flows Embedded in a Unified Solver Architecture: Flo3xx

Georg May*, Antony Jameson†

Stanford University, Stanford, CA, 94305, USA

This paper reports recent progress towards a new platform for computational aerodynamics on arbitrary meshes, tentatively designated Flo3xx. This tool is designed for maximum flixibility to serve as both an industrial strength flow solver on general grids, and as a framework for advanced research in the area of CFD and aerodynamic design. Such a flexible platform is crucial for transfering new research to industrial applications. We describe the capabilities and methods used in this tool, and show results from initial validation on inviscid and viscous testcases. Furthermore some open issues in CFD on unstructured grids will be addressed, such as viscous discretization and multigrid methods in an unstructured context.

I. Introduction

Numerical algorithms that use unstructured grids have become increasingly popular within the CFD community. For viscous flow calculations hybrid grids are predominantly used, with prisms or hexahedra to resolve shear layers, and tetrahedra for the rest of the computational domain. However, structured meshes are still being used as well, while some researchers prefer Cartesian meshes.

The variety of different meshes being used today and the immense effort required for the generation of high-quality meshes were a primary motivation for the development of our new hybrid solver architecture Flo3xx for three-dimensional viscous flow, which operates on arbitrary meshes, and different discretization techniques, such as cell-centered, or cell-vertex discretization. Furthermore, Flo3xx is capable of reproducing the exact residuals of structured flow solvers when operating on structured meshes. We will give a general overview of this new tool in the first section.

It is desirable to formulate all algorithms in a way that does not depend on the mesh topology. However, viscous discretization traditionally depends heavily on the type of mesh being used. Furthermore, the choice of control volumes is quite important in this context. Gradients may be needed at the element centroids or the nodes for the purpose of state reconstruction and/or turbulence source terms. They may also be needed at cell interfaces, when computing the the rate-of-strain tensor for viscous fluxes. These interfaces may be given by the faces of the primary elements or by polygonal constructs arising e.g. from the median dual. A construction of the gradient at cell cetroids or nodes and a subsequent suitable averaging for the computation of the rate-of-strain tensor has been used by many researchers, but it is not at all obvious how this can be done accurately and robustly in a completely mesh transparent way. We will present a new type of gas-kinetic viscous discretization, which is inspired by recent success of gas-kinetic schems, such as the Xu-Prendergast[1]

*PhD Candidate, Department of Aeronautics and Astronautics, Stanford University, AIAA Member.

†Professor, Department of Aeronautics and Astronautics, Stanford University, AIAA Member.

scheme, based on the BGK equation. Viscous fluxes are computed from a reconstructed distribution function in each cell, which potentially reduces the mesh dependence of the viscous discretization. Initial results have been obtained using this method with the new architecture, which will be presented in section III.

A strategy for the automatic generation of unstructured coarse meshes based on the edge-collapsing technique has been developed and successfully used for inviscid computations with purely tetrahedral meshes. For viscous flow, an extension for hybrid grids, involving prismatic boundary layers has been developed. This will be presented in section IV.

II. A New Flow Solver Architecture: Flo3xx

Flo3xx is a flow solver for the Euler and Navier-Stokes equations intended for use with any type of mesh and choice of control volumes. This new solver will be introduced in this section along with inviscid validation results. The subjects of viscous discretization and unstructured multigrid methods will be deferred to later sections.

A. Support for Arbitrary Meshes and Different Discretization Techniques

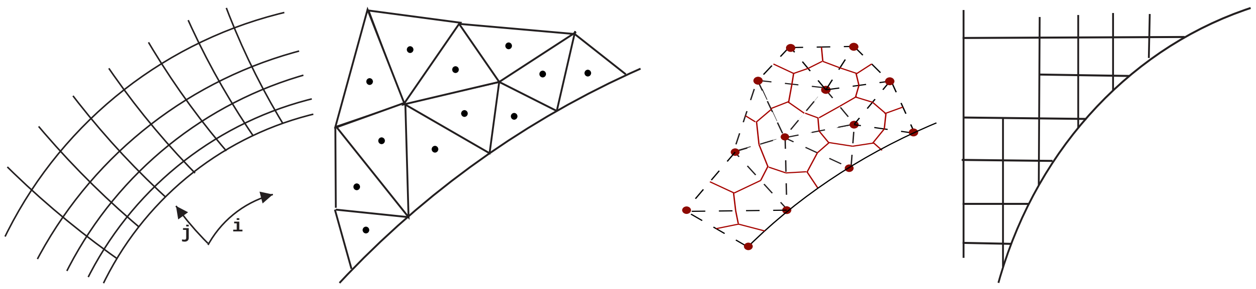


Figure 1. Examples of different mesh types and discretization techniques. From left to right: Structured, unstructured in cell-centered discretization (variables are stored at the cell-centers), unstructured in cell-vertex discretization using the median dual (variables are stored at the nodes), Cartesian.

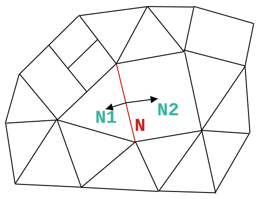


Figure 2. Two dimensional illustration of the face-based data structure.

Different types of meshes (structured, unstructured, Cartesian) as well as conceptually different methods of discretization are used in computational fluid dynamics today. Examples of discretization techniques are cell-centered discretization or cell-vertex discretization with one of the possible ways to define dual meshes from a given primary mesh topology. Figure 1 illustrates different mesh types and different discretization techniques. The newly developed program Flo3xx can operate on any of the above mesh types and discretization techniques. In fact, the flow solver module is at present completely oblivious to the underlying mesh topology so that it can run, in principle, on arbitrary meshes. A face-based data structure is used, which recognizes pointers from cell interfaces to adjacent nodes as the only connectivity information. This is illustrated in figure 2. Regardless of the mesh topology, there will always be a finite number of interfaces separating exactly two control volumes, including halo cells at boundaries. By looping over these interfaces all necessary flux computations can be carried out. The pseudo-code in figure A illustrates the basic algorithm used in the flow solver module. While this algorithm is simple, the connectivity information used is not normally the one stored in primary meshes. Computational complexity is shifted to the preprocessing stage, where the necessary data structure is generated along with metric data. Currently, preprocessing is implemented to provide metrics for the median dual mesh and the primary mesh for the four most common element types tetrahedra, prisms, pyramids, and hexahedra.

```
do i=ncell1,ncell2
   set residual(i) to zero
end do

do n=nface1,nface2
   N1 = ncf(1,n)
   N2 = ncf(2,n)
   flux(N) = f(solution(N1) , solution(N2))
   residual(N2) = residual(N2) + flux(N)
   residual(N1) = residual(N1) - flux(N)
end do

do i=ncell1,ncell2
   solution(i) = solution(i) - residual(i)
end do
```

Figure 3. Basic solution algorithm in Flo3xx

Preprocessing for hexahedra allows the treatment of structured meshes in an unstructured context. For structured meshes the data structure can also be augmented to recover the treatment that is normally used for this type of mesh. Figure 4 illustrates the usual stencil used for a second order flux computation on a structured mesh. The blue cell centers in figure 4 are pointees of the face between them in an unstructured context. By creating an additional level of pointers to the next-but-one neighbors (green cell centers), it is possible to reproduce the structured algorithms with the unstructured code. Note that there are thus two ways in which structured meshes can be treated in Flo3xx: One way is to look at them as unstructured hexahedral meshes and use a general-mesh treatment, involving gradient reconstruction (to be described below). Another way is to use the described extra data structure to reproduce the exact structured algorithms, which is faster compared to first method. It also serves to asses the minimum overhead that is incurred compared to a structured code due to the indirect addressing. Numerical experiments have shown this overhead to be around 25%.

Cell-centered discretization using the primary mesh or cell-vertex discretization using the median dual mesh can be selected at run-time, the only difference being the preparation of the metric data. Usually, subtle differences between cell-centered and cell-vertex discretization exist even in a mesh transparent flow solver module, for example at the boundaries. In this implementation a unified treatment of the boundaries has been chosen, using halo-cells at boundaries for any kind of mesh, i.e. for cell-vertex schemes boundary conditions are not enforced at the nodes. During preprocessing boundary nodes become dual cell centers, which are no longer located on the boundary, but shifted to the dual cell centroids. The construction of halo cells proceeds in the same manner as for cell-centered discretization.

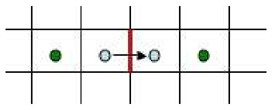


Figure 4. Stencil for 2^{nd} order flux computation on a structured mesh.

B. Numerical Methods

Several spatial discretization techniques are implemented in Flo3xx. Convective and diffusive terms are discretized using the H-CUSP or the E-CUSP scheme.[2] The JST scheme[3] is also available. These schemes are well known and well proven. A gas-kinetic scheme, based on the Xu-Prendergast BGK scheme, is also implemented, wich actually computes the full Navier-Stokes fluxes from locally reconstructed gas distribution functions. This is subject of another paper[4] and will not be discussed here. We will, however introduce a new viscous discretization technique based on gas-kinetic concepts in section III, along with conventional viscous discretization techniques.

A gradient-based linear reconstruction is used to construct left and right states for flux computations. Let index i denote a cell of the mesh, and index f a face of the cell. The reconstruction is written

$$w_f = w_i + \phi \nabla w_i (x_f - x_i) \ . \tag{1}$$

Here, w is a conserved flow variable. A limiter ϕ has to be used to prevent oscillations in nonsmooth regions. Currently Venkatakrishnan's limiter is used

$$\phi = \min_{f \in \mathcal{L}} \left\{ \frac{\alpha^2 + 2\alpha\Delta w + \epsilon}{\alpha^2 + \alpha\Delta w + 2\Delta w^2 + \epsilon} \right\} \ , \tag{2}$$

where $\mathcal{L}$ denotes the set of faces separating cell i from its neighbors. $\Delta w = \nabla w_i(x_f - x_i)$ and

$$\alpha = \begin{cases} \beta\left(w_{max} - w_i\right) & \Delta w \geq 0 \\ \beta\left(w_{min} - w_i\right) & \Delta w < 0 \end{cases} , \tag{3}$$

where w_{max} and w_{min} are local extrema of w over all cells adjacent to faces in $\mathcal{L}$. Strict positivity is not enforced at smooth extrema, where Δw is small. However, the value of the limiter approaches zero for large Δw, which ensures positivity at discontinuities.

The constant β in equation 2 can be used to tune the limiter. Normally, $\beta = 1$. The constant ϵ ensures that the limiter becomes inactive whenever both α and Δw are small, i.e. in smooth regions.

C. Time Stepping

Explicit Runge-Kutta time stepping with convergence acceleration techniques such as local time stepping, enthalpy damping, residual smoothing, and multigrid provides a good compromise between computational cost and complexity on one hand, and convergence rate on the other hand. This has been the default implementation in Flo3xx.

Recently the nonlinear symmetric Gauss-Seidel method, proposed and validated on structured meshes by Jameson and Caughey,[5] has proven to be an attractive alternative, offering superb convergence characteristics at a computational cost comparable to the Runge-Kutta method. For Flo3xx a formulation for unstructured grids has been implemented. To outline the method consider a flux-split semi-discretization of the one dimensional conservation law

$$\frac{dw}{dt} + D_x^- f^+(w) + D_x^+ f^-(w) = 0 ,$$

where the splitting is such that the jacobians $A^\pm = \frac{\partial f^\pm}{\partial w}$ have positive and negative eigenvalues, respectively. $D_x^\pm$ are forward and backward difference operators. The linearized implicit scheme becomes

$$\left\{I + \lambda\left(\delta_x^+ A^- + \delta_x^- A^+\right)\right\}\delta w + \Delta t R = 0 , \tag{4}$$

where $\lambda = \frac{\Delta t}{\Delta x}$ and R is the residual. Upon rearranging,

$$\delta w_i + \lambda\left(A_{i+1}^- \delta w_{i+1} - A_i^- \delta w_i + A_i^+ \delta w_i - A_{i-1}^+ \delta w_{i-1}\right) + \Delta t R_i = 0 , \tag{5}$$

we obtain the symmetric Gauss-Seidel scheme as

$$\delta w_i^{(1)} + \lambda\left(A_i^+ - A_i^-\right)\delta w_i^{(1)} - \lambda A_{i-1}^+ \delta w_{i-1}^{(1)} + \Delta t R_i = 0 ,$$

$$\delta w_i^{(2)} + \lambda\left(A_i^+ - A_i^-\right)\delta w_i^{(2)} + \lambda A_{i+1}^- \delta w_{i+1}^{(2)} - \lambda A_{i-1}^+ \delta w_{i-1}^{(1)} + \Delta t R_i^{(1)} = 0 .$$

By noting that the terms $\Delta t R_i - \frac{\Delta t}{\Delta x} A_{i-1}^+ \delta w_{i-1}^{(1)}$ are a linearization of $\Delta t R_i$ evaluated with $w_{i-1}^{(1)} = w_{i-1} + \delta w_{i-1}^{(1)}$, the scheme can be recast as

$$\left\{I + \frac{\Delta t}{\Delta x}\left(A_i^+ - A_i^-\right)\right\}\delta w_i^{(1)} + \Delta t R_i^{(01)} = 0 , \tag{6}$$

$$\left\{I + \frac{\Delta t}{\Delta x}\left(A_i^+ - A_i^-\right)\right\}\delta w_i^{(2)} + \Delta t R_i^{(12)} = 0 , \tag{7}$$

where

$$R_i^{(01)} = \frac{1}{\Delta x}\left({f_{i+1}^-}^{(0)} - {f_i^-}^{(0)} + {f_i^+}^{(0)} - {f_{i-1}^+}^{(1)}\right) ,$$
$$R_i^{(12)} = \frac{1}{\Delta x}\left({f_{i+1}^-}^{(2)} - {f_i^-}^{(1)} + {f_i^+}^{(1)} - {f_{i-1}^+}^{(1)}\right) ,$$

and

$$w_i^{(1)} = w_i^{(0)} + \delta w_i^{(1)} \ , \qquad f_i^{\pm(1)} = f^{\pm}(w_i^{(1)}) \ ,$$
$$w_i^{n+1} = w_i^{(2)} = w_i^{(0)} + \delta w_i^{(2)} \ , \qquad f_i^{\pm(2)} = f^{\pm}(w_i^{(2)}) \ .$$

In the limit $\Delta t \to \infty$ the sweeps 6 and 7 reduce to

$$|A|\delta w^{(1)} + \sigma R^{(01)} = 0 \tag{8}$$
$$|A|\delta w^{(2)} + \sigma R^{(12)} = 0 \ , \tag{9}$$

where $|A|$ is the absolute Jacobian matrix,

$$|A| = A^+ - A^- \ ,$$

and σ is a relaxation factor ~ 1.

The extension to higher dimensions and a finite-volume formulation can be accomplished by replacing $|A|$ with the sum of absolute directed Jacobians. The generic formulation 8,9 remains valid. In practice the nonlinear formulation reduces computational complexity, since a simple cell-based algorithm can be used, in which the fluxes at all faces surrounding the cell are computed, and the solution in the cell is updated before moving to the next cell. This means that fluxes are computed twice at each face during both the forward and backward sweep. Thus the baseline scheme is roughly equivalent in complexity to a four stage Runge-Kutta scheme.

For an unstructured implementation two major differences compared to the structured version need to be addressed. Firstly, there is no obvious ordering in which the updates are to be performed. At the very least, a reasonably balanced stencil should be generated by the ordering, so that flux computations always use a combination of updated and non-updated cells. For external aerodynamics it has been found that performing the sweeps in incresing and decreasing x_n, where x_n is the free stream direction, yields good results. Thus the nodes are ordered in increasing x_n. Should a boundary of the domain be aligned with a direction normal to x_n an arbitrary ordering in that direction could result. This can be prevented by rotating the ordering direction very slightly about an axis normal to x_n.

The second important difference is that, unlike structured meshes, where the reconstruction can be performed on the fly using a simple stencil of direct neighbors and next-but-one neighbors, for unstructured meshes the reconstruction is precomputed using gradients and limiters. Since the solution variables are continuously updated during the loop over the cells, corrections to the reconstruction must be made. In the current implementation the reconstruction for the SGS scheme uses a least-squares method to compute gradients. For a cell, i say, the gradient is obtained by nminimizing the functional

$$\frac{1}{2}\sum_{l\in\mathcal{L}} \left\{\nabla w_i \cdot \Delta x_l - \Delta w_l\right\}^2 \tag{10}$$

where l is the index of a face separating cell i from a neighboring cell j. All such faces form the set $\mathcal{L}$. We denote by $\Delta x_l = x_i - x_j$ the distance between the cells i and j, and Δw_l is defined in a similar fashion. This leads to

$$A\vec{\nabla} w_i = \sum_{l\in\mathcal{L}} \Delta w_l \Delta \vec{x}_l \tag{11}$$

where

$$A = \sum_{l\in\mathcal{L}} \begin{pmatrix} \Delta x_l \Delta x_l & \Delta x_l \Delta y_l & \Delta x_l \Delta z_l \\ \Delta x_l \Delta y_l & \Delta y_l \Delta y_l & \Delta y_l \Delta z_l \\ \Delta x_l \Delta z_l & \Delta y_l \Delta z_l & \Delta z_l \Delta z_l \end{pmatrix} \ . \tag{12}$$

Suppose the value of w in node i has changed according to $\widetilde{w} = w + dw$, and we wish to update the gradient in node i and the surrounding stencil. The matrix A in equation 12 depends only on metric terms, and its inversion can be precomputed and stored for each node. Let the new value of the gradient in i be denoted by $\widetilde{w}$. By inspection of equation 11 we see that

$$\vec{\nabla}\widetilde{w} = \vec{\nabla}w + A^{-1}\vec{r} \; , \tag{13}$$

where

$$\vec{r} = dw_i \sum_j (\vec{x}_i - \vec{x}_j)$$

for cell i and

$$\vec{r} = dw_i(\vec{x}_i - \vec{x}_j) \tag{14}$$

for any node neighboring cell j. The limiter can be updated in a similar fashion.

D. Inviscid Validation

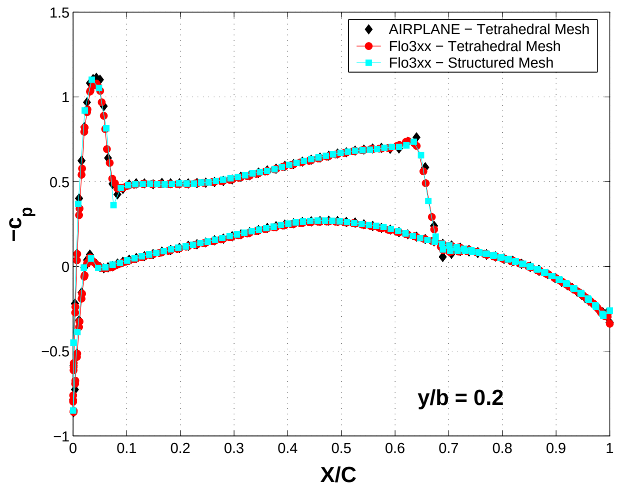


Figure 5. Pressure distribution in the wing section $y/b = 0.2$ for the Onera M6 wing at Mach number $M = 0.84$ and $\alpha = 3.06^o$ angle of attack. Results are shown for a structured mesh with 1,1 million nodes and for a tetrahedral mesh with 316,000 nodes.

Figure 5 provides initial validation of the unified-mesh algorithm. Both a structured mesh with approximately 1.1 million nodes and an unstructured tetrahedral mesh with 316,000 nodes have been used to calculate the flow for the Onera M6 testcase at Mach number $M = 0.84$ and $\alpha = 3.06^o$ angle of attack. The c_p distribution at the wing section $y/b = 0.2$ is compared to results obtained with the solver AIRPLANE on the same unstructured tetrahedral mesh. All computations have been carried out using cell-vertex discretization on a three-level multigrid sequence.

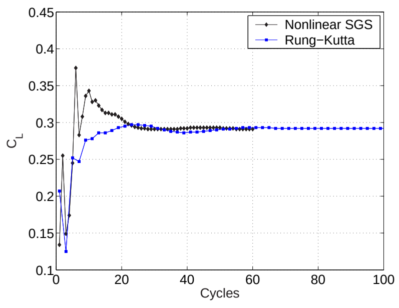
Convergence histories for the lift coefficient for both Runge-Kutta time stepping and the nonlinear SGS scheme are shown for several meshes in figure 6. It can be observed that both methods converge well. The SGS scheme, however, reaches the steady state faster than the Runge-Kutta scheme. For both schemes the convergence is nearly mesh independent. For the two coarser meshes in 6 a three mesh multigrid sequence has been used, whereas for the finest mesh a four mesh sequence was employed. All multigrid computations were carried out using a w-cycle.

III. Viscous Discretization

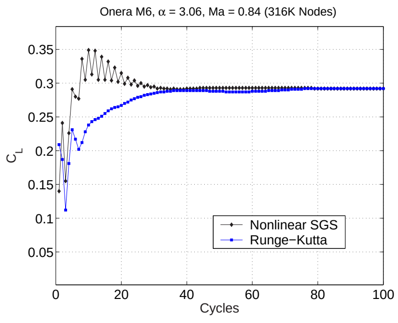
Since gradients are needed for state reconstruction they are initially computed for the nodes of the primary mesh or the centroids of the primary elements, depending on the discretization type chosen. For viscous dicretization the gradients must be evaluated at the interfaces between edjacent cells. This can be done by a suitable average, which typically depends on the mesh structure and choice of control volume. Since the goal for the computational platform Flo3xx is complete mesh transparency, this dependence on the mesh topology is unsatisfactory.

In the current implementation gradients are constructed using either the unweighted least-squares method described in section II or a gauss-type reconstruction, i.e.

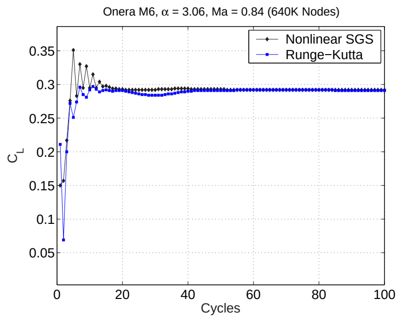
$$\nabla w_i = \frac{1}{V_i}\int_A \bar{\phi}\, dA \; , \tag{15}$$



(a) Onera M6 Testcase on 94k node tetrahedral mesh



(b) Onera M6 testcase on 316k node tetrahedral mesh



(c) Onera M6 testcase on 640k node tetrahedral mesh

Figure 6. Comparison of convergence history of the lift coefficient for the nonlinear SGS scheme and Runge-Kutta time-stepping.

where V_i is the cell volume, and the overbar denotes a suitable face average.

For conventional viscous discretization an initial value for the gradient at the cell interface may be obtained by averaging the values of adjacent cells, ϕ_i and ϕ_j, to give

$$\overline{(\nabla\phi)_{ij}} = \frac{1}{2}\left[(\nabla\phi)_i + (\nabla\phi)_j\right]. \tag{16}$$

Subsequently a correction in the direction $\vec{s}_{ij}$ is applied to avoid odd-even decoupling, where

$$\vec{s}_{ij} = \frac{\vec{r}_j - \vec{r}_i}{|\vec{r}_j - \vec{r}_i|} .$$

This leads to the following gradient on the interface

$$(\nabla\phi)_{ij} = \overline{(\nabla\phi)_{ij}} - \left[\overline{(\nabla\phi)_{ij}} \cdot \vec{s}_{ij} - \frac{\phi_j - \phi_i}{|\vec{r}_j - \vec{r}_i|}\right] \cdot \vec{s}_{ij} . \tag{17}$$

This modification results in a stable discretization of the viscous terms and does not affect the accuracy as long as the interface integration point is located halfway between the two cell-centers. This, however, is only the case for the median dual control volume, rendering this type of viscous discretization highly mesh dependent

Using concepts of kinetic gas theory, viscous discretization can be accomplished in a different fashion. Many researchers have used the Boltzmann equation or its BGK simplification,[6] i.e. the BGK equation, as a basis for the construction of numerical fluxes.[7,1] Firstly we note that the Navier-Stokes equations can be obtained by Chapman-Enskog expansion of both the full Boltzmann equation as well as the BGK equation, the difference being only in the values of the transport coefficients.

Suppose that f is a distribution function describing the fluid state $\Phi = (\rho, \rho u, \rho v, \rho w, \rho e)^T$, i.e.

$$\Phi(x,y,z,t) = \int \boldsymbol{\phi}\, f(x,y,z,u,v,w,t,\xi)\, du\, dv\, dw\, d\xi \ , \tag{18}$$

where

$$\boldsymbol{\phi}(x,y,z,t) = \left(1, u, v, w, \frac{1}{2}(u^2+v^2+w^2+\xi^2)\right)^T ,$$

and u, v, w are the phase-space velocities. The variable ξ stands for internal degrees of freedom.

Consider a finite-volume discretization of the Navier-Stokes equations, for a cell i, say,

$$W_i^{n+1} = W_i^n - \frac{1}{V_i} \int_0^{\Delta t} \int \vec{F} \cdot d\vec{A} dt \ , \tag{19}$$

where W_i is a volume average. By employing a gas-kinetic description of the flow field, see equation 18, one obtains the Navier-Stokes flux at the face f as

$$F = \begin{pmatrix} F_\rho \\ F_u \\ F_v \\ F_w \\ F_E \end{pmatrix} = \int u \phi f(\vec{x}_f, \vec{u}, t) \, d\Xi \ , \tag{20}$$

where $d\Xi = du \, dv \, dw \, d\xi$. It is assumed that the direction x, and the corresponding phase-space velocity u, are normal to the face (one typically works in rotated coordinates for these computations). For a gas-kinetic scheme, it remains to find a valid distribution function. For Navier-Stokes fluxes, this distribution function must be consistent with the Chapman-Enskog expansion up to the Navier-Stokes order, i.e. with the first-order Chapman Enskog expansion. This means that the distribution function may be constructed from local Maxwellians and their first derivatives.

Consider now the BGK equation:

$$\frac{\partial f}{\partial t} + \vec{u} \cdot \vec{\nabla} f = -\frac{f - g}{\tau} \ . \tag{21}$$

Prendergast and Xu have proposed a construction of the local distribution function for the Euler and Navier Stokes equations based on a local analytic solution of the BGK equation, using local Maxwellians and their derivatives.[8] This method has since been refined by many researchers, most notably Xu.[1] More recently Ohwada has elucidated the relationship between the Navier-Stokes equations and gas-kinetic schemes in general, and the Xu-Prendergast BGK scheme in particular, establishing that the resulting fluxes are consistent with the Chapman-Enskog expansion of the BGK equation up to the Navier-Stokes level.[9] We have recently extended the method to three-dimensional viscous multigrid computations on general unstructured meshes using the framework described herein, including multigrid. While this work is subject of another publication,[4] we will very briefly outline the method insofar as it is relavant for the gas-kinetic viscous discretization presented here.

Equation 21 can be solved analytically to give

$$f(\vec{x}, \vec{u}, t, \xi) = \frac{1}{\tau} \int_0^t g(\vec{x}', \vec{u}, t', \xi) e^{-\frac{-(t-t')}{\tau}} dt' + \ e^{-\frac{t}{\tau}} f_0(\vec{x} - \vec{u}t, \vec{u}, \xi) \ ,$$

where $\vec{x}' = \vec{x} - \vec{u}(t - t')$. It can be found by Chapman-Enskog expansion that in order to recover the Navier-Stokes equations the collision time τ is related to the dynamical viscosity by

$$\mu = \tau p \ . \tag{22}$$

By setting the collision time appropriately, viscous fluxes can thus be computed directly as moments of the distribution function, provided it is based on the Chapman-Enskog expansion of the current flow solution. Restricting ourselves to one-dimension for simplicity,the intitial distribution f_0 in the Xu-Prendergast scheme is obtained as

$$f_0 = \begin{cases} g^l + x g^l_x - \tau(g^l_t + u g^l_x) \ , & x \le 0 \ , \\ g^r + x g^r_x - \tau(g^r_t + u g^r_x) \ , & x \ge 0 \ . \end{cases} \tag{23}$$

The derivatives of the Maxwellian can be computed using the most current local reconstruction. Similarly, the time-dependent equilibrium distribution g in equation 22 has the general form

$$g = g^0 + (1 - H(x))g^0_{x^-}x + H(x)g^0_{x^+}x + g^0_t t \ , \tag{24}$$

where $H(x)$ is the Heaviside function

$$H(x) = \begin{cases} 1 \ , & x \geq 0 \\ 0 \ , & x < 0 \end{cases} \ , \tag{25}$$

and g^0 is the Maxwellian distribution obtained with macroscopic variables averaged on the face using a conservation constraint (see[4] for details).

The equilibrium portion of the initial distribution is given by the local Maxwellian and its spatial expansion,

$$f_{00} = g + g_x x \ . \tag{26}$$

The portion

$$f_{01} = \tau(g_x u + g_t) \tag{27}$$

is formally identical to the first term of the Chapman-Enskog expansion of the distribution function, which gives the rate-of strain tensor and the heat flux vector. Hence we use the current Chapmann-Enskog expansion to the Navier-Stokes level as input data, which, along with the requirement that the reconstructed flux be consistent with the solution of the exact gas-kinetic equation corresponding to the first-order Chapman-Enskog expansion, guarantees that the Navier Stokes fluxes will be recovered.

In the Xu-Prendergast scheme the derivatives are replaced by finite differences obtained by limited reconstruction, and the expansion is carried out normal to the face at which the flux is computed. Both these restrictions are problematic. Limited gradients are necessary for a stable discretization of the convective terms, but the viscous rate-of-strain tensor should be evaluated with the exact gradients. Furthermore, if the reconstruction is restricted to the direction normal to the face, the complete rate-of-strain tensor is not computed. Including the tangential derivatives in the expansion for the distribution function is expensive. Since it is strictly needed for the viscous term only, we propose to use only the part in the distribution function corresponding to the nonequilibrium terms, and combine it with a conventional Riemann solver, such as the CUSP scheme for the convective terms.

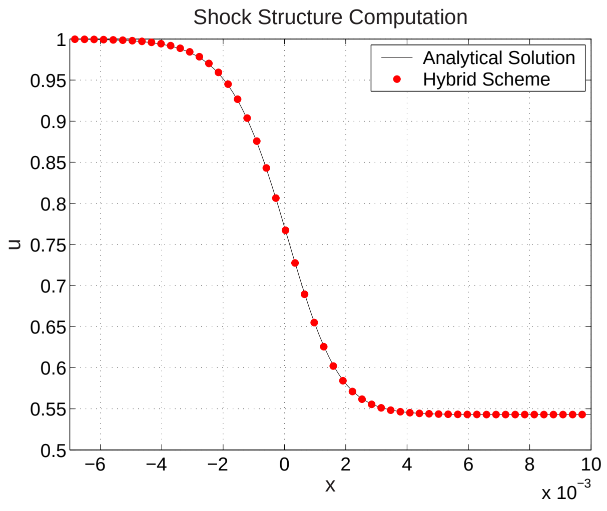
The viscous fluxes can then be disrcetized as

$$F_v = \frac{\mu}{p} \int u\boldsymbol{\phi} \left\{ H(u)(g^l_t + ug^l_x) + (1 - H(u))(g^r_t + ug^r_x) \right\} d\Xi \ . \tag{28}$$

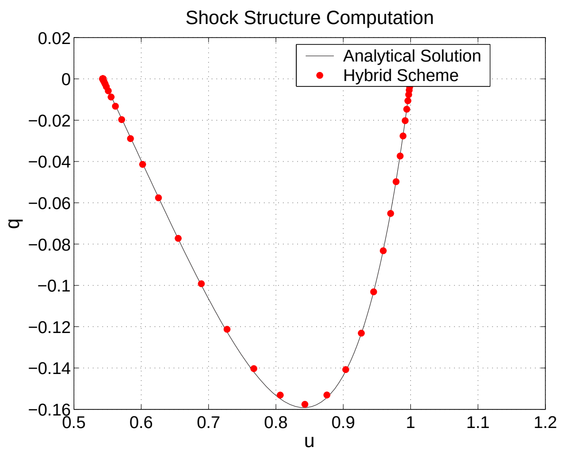
The multidimensional extension is straight forward. Within the BGK framework this type of discretization has been succesfully used on a variety of different meshes in exactly the same way. Since each Maxwellian is inegrated over only half the u-velocity space, this gives rise to a nonlinear average of the two states adjacent to the cell, when written in the macroscopic flow variables.

For initial validation we provide the solution of a resolved shock structure obtained with the new scheme, see Figure 7. It should be pointed out at this point that, since the Chapmann-Enskog expansion of the BGK equation gives the transport coefficients as a sole function of the collision time τ, only one parameter can be set automatically to the right value. This will usually be the dynamic viscosity. For accurate heat flux computations, the reconstruction involving the temperature gradient has to be scaled appropriately, so as to correspond to the correct Prandtl number (see[4] for details).

For furter validation consider a zero-pressure-gradient boundary layer. Figures 8 shows the solution computed for comparison with Flo3xx and a JST scheme with the conventional viscous discretization using central averaging of gradients and directional correction. The result computed with the new gas-kinetic discretization is shown in figure 9. This type of viscous discretization is certainly mesh independent. It has been well proven in gas-kinetic schemes as part of the overall kinetic construction of gas-kinetic fluxes, and its validity is out of question. Further analysis, however, is required, with primarily issues related to robustness and stability in mind. This will be considered for a future publication.

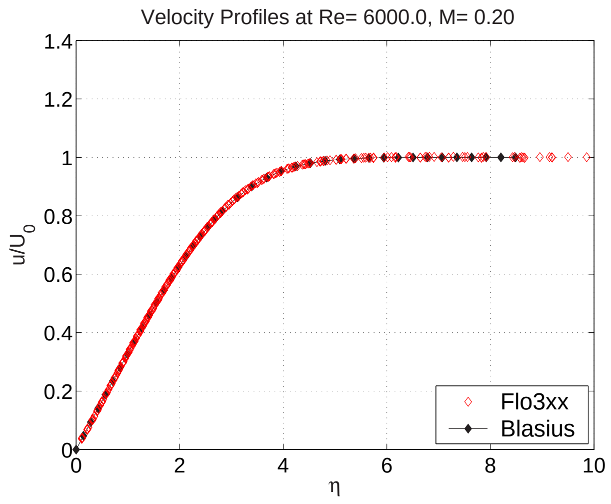


(a) Velocity distribution across a normal shock.

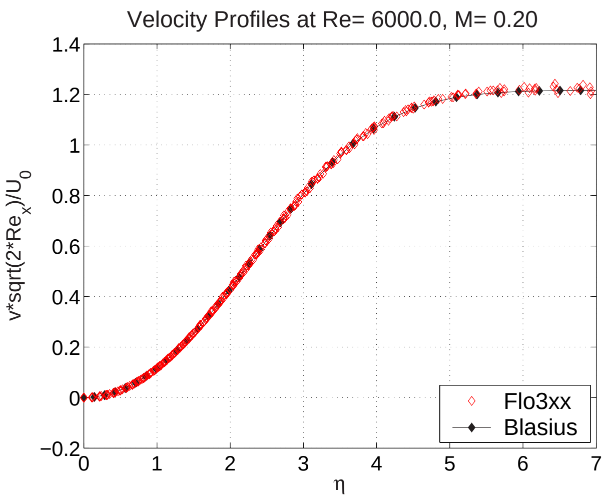


(b) Heat flux across a normal shock.

Figure 7. Resolved shock structure for upstream Mach number $M = 1.6$ and a Prandtl number of $Pr = 0.72$

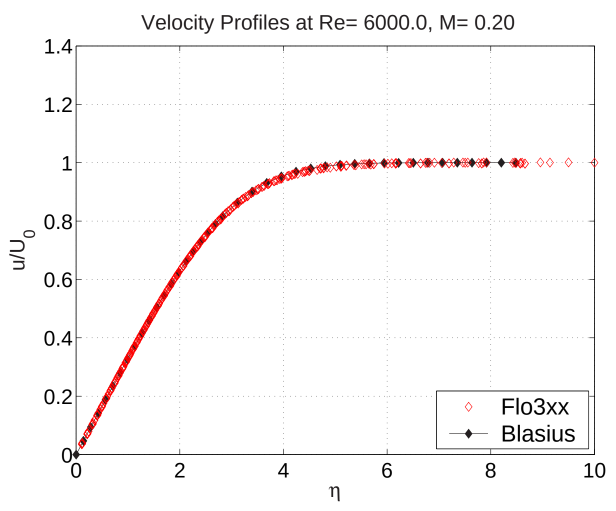


(a) x-Velocity.

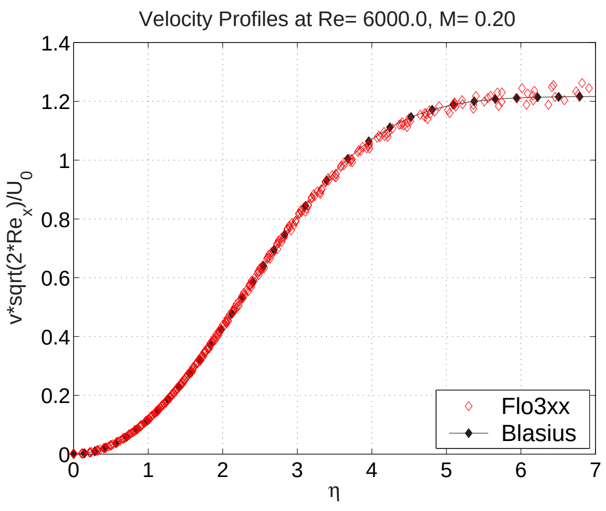


(b) y-velocity.

Figure 8. Laminar Boundary Layer at $Re = 6000$ and $M = 0.2$ using the JST scheme and direction-corrected central discretization of the viscous terms.



(a) x-Velocity.



(b) y-velocity.

Figure 9. Laminar Boundary Layer at $Re = 6000$ and $M = 0.2$ using the CUSP-gas-kinetic hybrid scheme.

IV. Multigrid schemes for unstructured grids

A general framework for the development of full-approximation multigrid methods for nonlinear equations can be outlined as follows. Consider

$$\mathcal{L}(u) = \mathcal{F}.$$

A numerical estimate of the solution, v_h say, will not satisfy this equation exactly. Assume a correction δv_h can be found such that

$$\mathcal{L}_h(v_h + \delta v_h) = \mathcal{F}_h \ .$$

Then, after linearization,

$$\mathcal{A}_h \delta v_h + R_h = 0 \ ,$$

where $R_h = L_h(v_h) - F_h$ is the residual and A_h is the Jacobian of the nonlinear operator L_h. On the coarse grid the above equation can be replaced by

$$\mathcal{A}_{2h} \delta v_{2h} + I_{2h}^{h} R_h = 0 \ , \tag{29}$$

where I_{2h}^{h} represents the aggregation or restriction operator. To avoid using the Jacobian explicitly add

$$\mathcal{L}_{2h}(v_{2h}) - \mathcal{F}_{2h} - R_{2h} = 0$$

to equation 29 to get

$$\mathcal{L}_{2h}(v_{2h} + \delta v_{2h}) - \mathcal{F}_{2h} + I_{2h}^{h} R_h - R_{2h} = 0 \ .$$

This is formally identical to the original equation, except for an additional source term involving the residuals on the coarse and fine mesh. The solution can thus be advanced on the coarse mesh using the same routines as on the fine mesh. Let $v_{2h} + \delta v_{2h} = v_{2h}^{+}$. After v_{2h}^{+} has been computed on the coarse grid the corrected solution on the fine grid can be written

$$v_h^{+} = v_h + I_h^{2h}(v_{2h}^{+} - v_{2h}) \ ,$$

where I_h^{2h} is an interpolation operator.

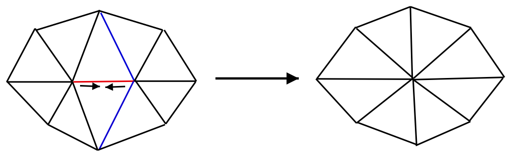


Figure 10. 2D Illustration of edge collapsing

We will consider the case of purely tetrahedral meshes first. It is desirable to generate only the finest mesh of a multigrid sequence manually, which is then successively coarsened by an automatic method. One such procedure is given by the method of repeated edge collapsing, which is used extensively in image rendering, and has been applied to CFD by Crumpton et. al.[10,11] The basic idea behind edge collapsing is to coarsen a grid by deleting two nodes connected by an edge and replace them with one node at the midpoint of the edge, or a point optimized according to some figure of merit. A two-dimensional illustration of the concept is shown in fig. 10. The concept carries over to three-dimensional meshes in a straightforward way, as long as the mesh is composed only of tetrahedra.

The edge collapsing cycle is summarized in fig. 12. To ensure minimal distortion of the mesh, the shortest edge meeting at a given node is collapsed. Since each edge deletion affects the mesh only locally, on-the-fly checks of the quality of the generated new cells can be easily implemented before actually carrying out a node deletion. Figures of merit that can be used to monitor the mesh quality include aspect ratio, element edge-ratios, minimum element volume, and similar standard mesh quality measures.

Surface preservation is another important issue in mesh manipulation. Blindly collapsing edges on surfaces will ultimately violate the surface, and can in fact severely distort the shape of a meshed object. We generate a linear representation of the original surface on which all new nodes, which are generated when collapsing edges on the boundary, are projected. An efficient binary search tree is used to find the closest point in the original surface representation.

Projecting new nodes onto the original surface is not quite sufficient for a smooth representation of the entire surface. Even with two vertices lying on the original surface, the edge which connects them can (and will) penetrate the surface. On highly curved surfaces, such as wing leading edges this can lead to a significant distortion. The current implementation of edge collapsing uses the above-mentioned distance search to determine the surface penetration, and rejects a node deletion if it exceeds a specified tolerance.

Mesh	n_V	$\Lambda < 10$	$10 < \Lambda < 100$	$\Lambda > 100$
Fine	94493	97.98 %	2.01%	.01%
Medium	17294	87.96 %	12.02%	.02%
Coarse	3524	78.07 %	20.86%	.07%

Table 1. Representative Distribution of aspect ratio Λ among tetrahedra

In a certain sense all surface protection must be heuristic, because any new mesh resulting from a movement of vertices on the boundary will violate the original surface to some extent. By using a higher-order surface representaion this violation can be made arbitrarily small, but the computational cost and complexity becomes considerable, which is why in this work a linear surface representation with heuristic algorithms for boundary protection has been chosen. An acceptable quality of the new mesh can be ensured, as is demonstrated in table 1. The edge collapsing procedure is carried out in a repeated loop over all nodes until the desired coarsening ratio is reached. After each loop the existing data structure can be used to copy a new list of cells and nodes from the fine mesh. Figure 11 shows a representative example of coarsened meshes for the Onera M6 wing.

The coarsening ratios achieved by this procedure depend heavily on the quality restrictions applied during the coarsening. For most cases, coarsening ratios of approximately five have been found a good compromise between mesh quality and convergence. For high-quality fine meshes, coarsening ratios of 8 have also been used with very good results.

Given the initial fine mesh, repeated edge collapsing is a completely automatic procedure, and can be controlled to achieve a desired level of coarseness with respect the the overall mesh size, while ensuring acceptable mesh quality in measurable quantities.

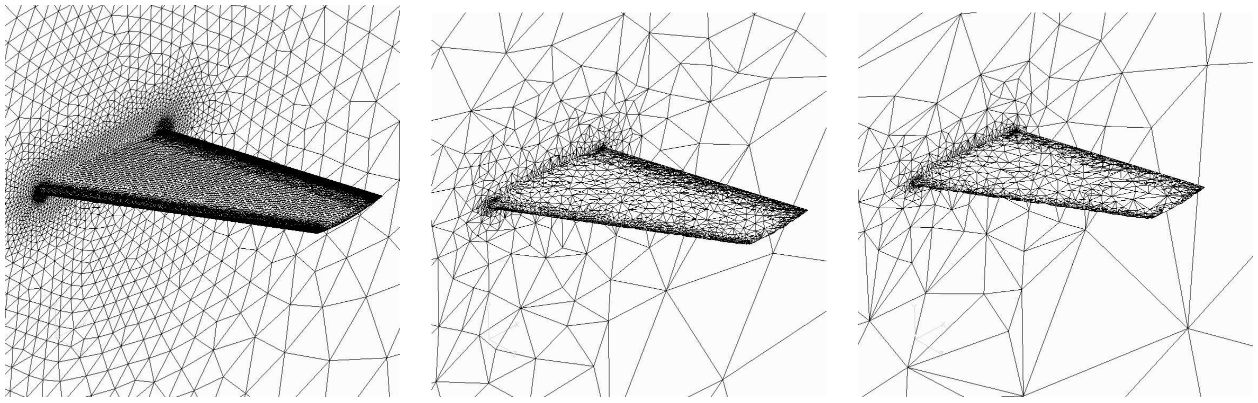
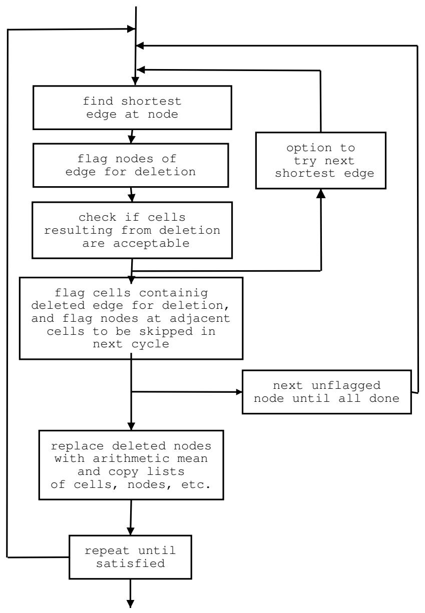

(a) Baseline (fine) mesh (b) Medium mesh (c) Coarse mesh

Figure 11. Examples of coarse meshes generated from baseline fine mesh by repeated edge-collapsing.



find shortest edge at node
flag nodes of edge for deletion
check if cells resulting from deletion are acceptable
option to try next shortest edge
flag cells containig deleted edge for deletion, and flag nodes at adjacent cells to be skipped in next cycle
next unflagged node until all done
replace deleted nodes with arithmetic mean and copy lists of cells, nodes, etc.
repeat until satisfied

Figure 12. The edge collapsing cycle

Edge collapsing has been used to generate most of the multigrid sequences used in this work. The convergence rates shown earlier should suffice to suggest that the method is viable. In this section we compare the edge-collapsing technique to the commonly used method of agglomeration, which fuses stencils of cells to form coarse volumes. Agglomeration is perhaps the most accepted mesh coarsening algorithm for unstructuered meshes at the moment. The flow solver Flo3xx has been used to compute the inviscid Onera M6 test case for both agglomerated and edge-collapsed meshes. Due to the general data structure implemented in the code, both methods can be used, although mesh coarsening by agglomeration is not currently implemented. A four mesh sequence with identical coarsening ratios of 5 to 6 between the meshes has been used for this comparison. The fine mesh consists of 316,275 nodes and 1,940,182 tetrahedra. The convergence history is shown in figure 13. It can be seen that both methods have very similar convergence properties.

Edge collapsing has the advantage that very basic primary mesh data structure is the only input and output. Thus the mesh coarsening can be carried out before entering the Preprossesing stage, where the metrics for either cell-centered or cell-vertex formulation can be prepared by a simple loop over all meshes in precisely the same way for all meshes. Subsequently transfer coefficients for the transfer of the solution between meshes can be computed using efficient search algorithms such as binary trees. A disadvantage is certainly the nontrivial parallelization of the process. So far our implementation is sequential, which becomes a restrictive factor for fine meshes.

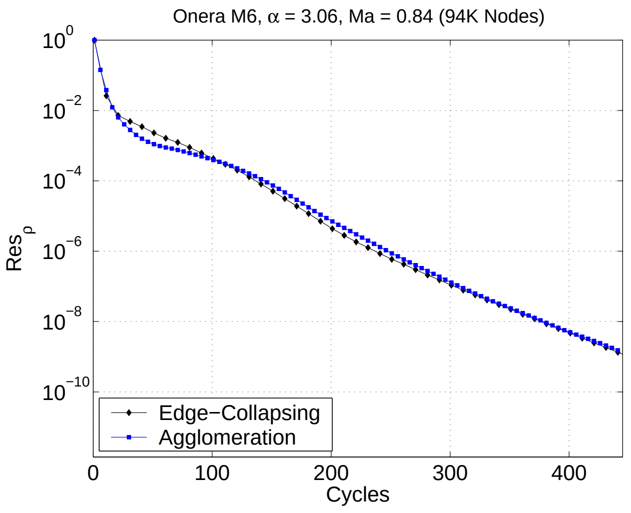
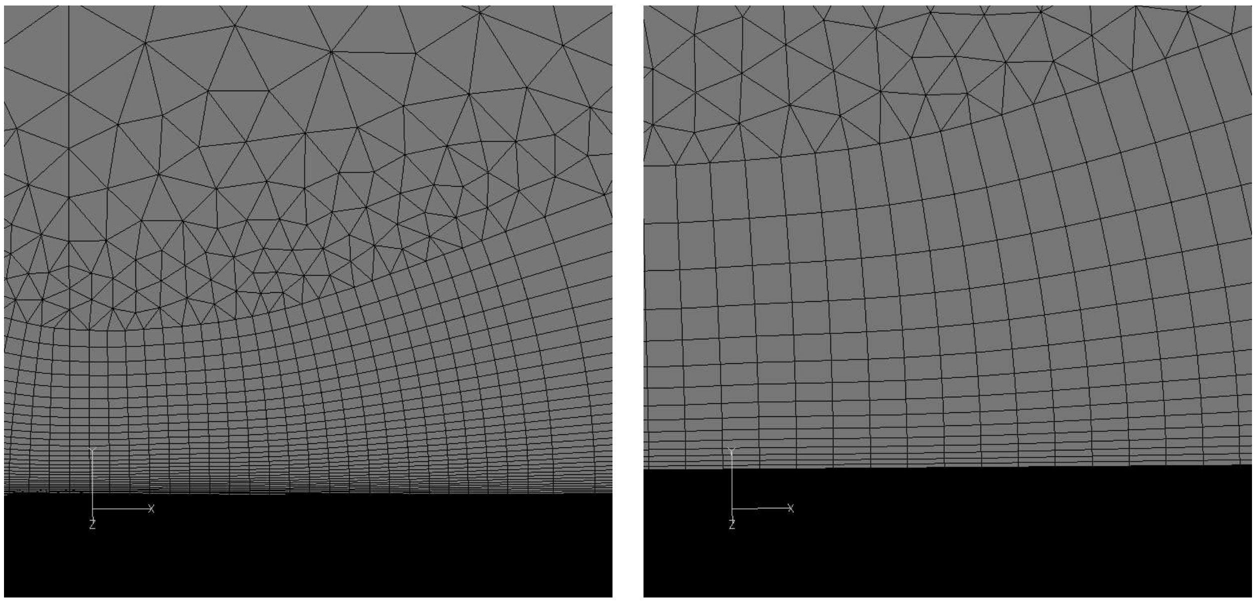


Figure 13. Convergence history for edge collapsing and agglomeration

An extension to hybrid meshes with prismatic elements has recently been implemented. The basic idea behind the algorigthm is that layers of prismatic elements are coarsened using a structured coarsening in normal direction, i.e. every other prism is deleted. The interface between prisms and hexahedra is treated as a boundary on the tetrahedral side. Figure 14 shows an example of a coarsened hybrid mesh using the semi-structured coarsening.



(a) Medium mesh

(b) Coarse mesh

Figure 14. Examples of coarse meshes generated from baseline hybrid fine mesh by semi-structured coarsening.

So far only constant layers of prisms are supported, which is a severe restriction. This will be addressed in future work. Furthermore, an option to coarsen the layers of triangles connecting the prisms should be available. One option is to carry out edge-collapsing on the tetrahedral side, treat the prism interface as a boundary, while allowing edge deletions on the interface, which are subsequently propagated through the layers of prisms.

References

[1]Xu, K., "A Gas-Kinetic BGK Scheme for the Navier-Stokes Equations and its Connection with Artificial Dissipation and Godunov Method," *J. Comp. Phys.*, Vol. 171, No. 48, 2001, pp. 289–335.

[2]Jameson, A., "Analysis and design of numerical schemes for gas dynamcics 2: Artificial diffusion and discrete shock structure," *Int. J. Comp. Fluid. Dyn.*, Vol. 5, 1995, pp. 1–38.

[3]A. Jameson, W. Schmidt, E. T., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," Aiaa paper 1981-1259, 1981.

[4]May, G., Srinivasan, B., and Jameson, A., "Three-Dimensional Flow Calculations on Arbitrary Meshes Using a Gas-Kinetic BGK Finite-Volume Method," Aiaa paper 2005-1397, 2005.

[5]Jameson, A. and Caughey, D. A., "How Many Steps are Required to Solve the Euler Equations of Steady Compressible Flow: In Search of a Fast Solution Algorithm," Aiaa paper 2001-2673, 2001.

[6]Bhatnagar, P., Gross, E., and Krook, M., "A Model for Collision processes in Gases I: Small Amplitude Processes in Charged and Neutral One-Component Systems," *Phys. Rev.*, Vol. 94, 1954, pp. 511.

[7]Chou, S. Y. and Baganoff, D., "Kinetic Flux-Vector Splitting for the Navier-Stokes Equations," *J. Comp. Pys.*, Vol. 130, 1996, pp. 217–230.

[8]Prendergast, K. and Xu, K., "Numerical Hydrodynamics from Gas-Kinetic Theory," *J. Comp. Phys.*, Vol. 109, 1993, pp. 53–66.

[9]Ohwada, T. and Kobayashi, S., "Management of Discontinuous Reconstruction in Kinetic Schemes," *J. Comp. Phys.*, Vol. 197, 2004, pp. 116–138.

[10]Crumpton, P. I., "Implicit time accurate solutions on unstructured dynamic grids," Aiaa paper 95–1671, 1995.

[11]Crumpton, P. I., "Design optimisation for complex geometries," *15th ICNMFD Conference, Monterey, USA*, 1996.