AIAA Aviation
13-17 June 2016, Washington, D.C.
46th AIAA Fluid Dynamics Conference

Near-Body Grid Adaption for Overset Grids

Pieter G. Buning*
NASA Langley Research Center, Hampton, Virginia 23681

and

Thomas H. Pulliam†
NASA Ames Research Center, Moffett Field, California 94035

A solution adaption capability for curvilinear near-body grids has been implemented in the OVERFLOW overset grid computational fluid dynamics code. The approach follows closely that used for the Cartesian off-body grids, but inserts refined grids in the computational space of original near-body grids. Refined curvilinear grids are generated using parametric cubic interpolation, with one-sided biasing based on curvature and stretching ratio of the original grid. Sensor functions, grid marking, and solution interpolation tasks are implemented in the same fashion as for off-body grids. A goal-oriented procedure, based on largest error first, is included for controlling growth rate and maximum size of the adapted grid system. The adaption process is almost entirely parallelized using MPI, resulting in a capability suitable for viscous, moving body simulations. Two- and three-dimensional examples are presented.

I. Introduction

Grid adaption is used in computational fluid dynamics to improve the accuracy of flow simulations by reducing discretization error. Adaption can also be used to reduce the cost of a simulation by targeting grid refinement to areas of the domain where it improves the solution, as opposed to refining everywhere. Cost is measured in time to solution and computer resources required, and it is generally driven by the total size of the grid system.

This paper describes the development and implementation of a solution adaption capability for near-body grids used in the OVERFLOW computational fluid dynamics code.[1] OVERFLOW is a structured, overset grid framework where "near-body" grids are body-fitted curvilinear grids, and "off-body" grids are automatically generated Cartesian grids that fill the computational space between the near-body grids and the far-field boundary. (The use of off-body grids is optional, but they simplify grid generation for all but the simplest configurations.) An off-body solution adaption approach was presented in a previous paper,[2] and this work builds on that approach. Adaption for near-body grids is handled in a very similar manner, with grid refinement implemented in computational space rather than Cartesian space. This method has been developed with the specific intent to have the grid adaption as an integral part of the flow solver process and to ensure that it is efficient enough to use for time-dependent moving grid problems.

The adaption process described here is modeled after the off-body grid adaption developed by Meakin[3] and implemented in the OVERFLOW-D code.[4] This approach has been refined by Kamkar et al.[5] for rotorcraft applications in the Helios code, again for the off-body portion of the domain. Henshaw and Schwendeman[6-8] have developed an adaptive mesh refinement scheme for overset grids and applied it to high-speed inviscid flows. Most recently, Su[9] has demonstrated a grid adaption scheme for curvilinear multi-block structured grids that includes many of the features presented here, including parametric cubic interpolation to refine curvilinear grids and a sensor function based on second undivided differences of flow quantities.

New capabilities in this work include control of global size and growth of the near- and off-body grid system, an optional weighting of the sensor function by distance to the wall, use of blended central and one-sided differences with parametric cubic interpolation for grid refinement, and parallel performance of the grid adaption process suitable for viscous, moving body flow simulations using an overset structured grid approach.

* Aerospace Engineer, Computational AeroSciences Branch, MS 128, AIAA Associate Fellow.
† Aerospace Engineer, Fundamental Modeling and Simulation Branch, MS 258-2, AIAA Associate Fellow.

II. Approach

In this work, some basic ground rules for the adaption process are used, which match the off-body adaption scheme presented in Ref. 2. These include: allowing only isotropic grid refinement (using factors of two in each computational coordinate direction); arranging neighboring refinement regions so they differ by only one level of refinement; and providing communication between refinement regions by blanking out underlying coarser-level regions, leaving just enough overlap for interpolation of boundary flow conditions.

Several steps are involved in the adaption process. First, an error estimate or sensor function is computed as a field quantity. This function must be converted to a marker function that indicates what parts of the grid should be refined or coarsened. This marker function is further adjusted to satisfy a limit on global grid size. Next, the new grid system is created, with refined near-body grids preserving the smoothness and geometry features of the original grids. Finally, the flow solution is interpolated from the old grid system to the new system.

A. Sensor Function and Grid Marking

A sensor function is used in the grid adaption process to identify grid regions for coarsening or refinement. It is generally expected to represent a measure of the truncation error in the solution. One function that has proved useful for off-body adaption is the second undivided difference of the solution variables.[2] This measure was used for adaption by Meakin[3] in the original development of off-body Cartesian grids in OVERFLOW-D.[4] It has also been used by Warren et al.[10] and by Henshaw and Schwendeman[6-8] (in a form blended with the first undivided difference). In Ref. 9, Su uses second undivided differences of density, pressure, and temperature, normalized by a local average of the same flow quantity, as a sensor function. He states that this will identify both shock waves and vortices.

Here we use the primary flow variables (density, momentum, and stagnation energy per unit volume) in calculating the second undivided difference sensor function. We refer to these flow variables collectively as Q, and individual elements as q. The term "undivided" refers to these differences not being divided by the appropriate grid cell size term, i.e., Δx for the first-difference and Δx^2 for the second-difference. The second undivided difference is represented at each point as $S_i = (q_{i-1} - 2q_i + q_{i+1})/2$. It is equal to the difference between q_i and the linear interpolation (or average) of its neighbors q_{i+1} and q_{i-1}. In the context of computing S as a general function for adaption, we (a) normalize by a reference quantity q_{ref}, (b) square it to create a non-negative value, (c) take the maximum over all elements of Q, and (d) take the maximum over all coordinate directions:

$$S = \max_{i=j,k,l} \left\{ \max_{Q \text{ components}} \left[\left(\frac{q_{i-1} - 2q_i + q_{i+1}}{2q_{\text{ref}}} \right)^2 \right] \right\}$$

This function is nondimensional, independent of grid units, and becomes smaller as the grid is refined (where Q is smooth), all desirable properties for a sensor function for solution adaption. Further, it is simple to compute and the computation can be fully parallelized.

A smoothing step has been found to help the adaption process by removing local peaks or ridges in the sensor function, which may be caused by transients at inter-grid boundaries, for example. This smoothing step is implemented as consecutive sweeps in each of the coordinate directions, setting $S_i^* = \min[S_i, \max(S_{i-1}, S_{i+1})]$.

In some cases, flow features may persist far away from the geometry that created them. It is not often desirable to allow grid adaption to resolve these features to the far field, so some way to limit the extent of the adaption is needed. Two methods have been implemented. One is a simple geometric specification of a region, along with a maximum refinement level allowed. For off-body grids, the region is specified as an *(x,y,z)* box, with adaption control applied either inside or outside of the box. For near-body grids, the region is specified as a box in *(j,k,l)* computational space. This simple control is implemented at the stage when the refined grids are being generated. A second method for limiting the extent of the grid adaption is implemented as a distance weighting on the sensor function. This has the advantage of requiring much simpler user input that is not grid-specific. This weighting takes advantage of the wall distance function (usually calculated for the turbulence model). Two input parameters are specified: a characteristic length d_{char}, representing a one-grid refinement-level decay in the effective sensor function, and a distance offset from the wall d_{offset} to start the distance-based decay. This weighting factor is calculated as $w = (d^*/d_{\text{char}})/2^p$, where the modified distance $d^* = \max(d - d_{\text{offset}}, 0)$, and 2^p is the expected reduction in the sensor function with one level of grid refinement.

The primary purpose of the sensor function is to mark grid points for coarsening or refinement. If the function value at a grid point exceeds a specified threshold for refinement, that point is marked for refinement; if the value is below a coarsening threshold, the point is marked for coarsening. However, it would also be useful to convert the sensor function into the expected number of refinement levels needed to bring the sensor function below the refinement

threshold, or the number of levels the grid could be coarsened to bring the sensor function up to the coarsen threshold. In order to do this, we need an estimate of the expected reduction in the sensor function with each level of refinement. Here we follow Nemec, Aftosmis, and Wintzer,[11] estimating the error reduction at 2^p, where p is the order of accuracy of the numerical scheme. The choice of p is not critical to the adaption process but allows the presentation of a diagnostic prediction of the effect of additional adaption steps. We hope to use a 5th-order scheme for convective flux terms, so set $p=5$ for now. (We do realize there is an inconsistency in using the second undivided difference *squared* as the sensor function, while expecting a drop of 2^p with refinement, rather than $(2^p)^2$. This inconsistency may be offset by the fact that the numerical solution procedure is formally 2nd-order accurate, not 5th-order.)

In order to examine the sensor function in terms of grid refinement or coarsening, we define threshold values S_{refine} and $S_{coarsen}$. The sensor function is then converted to an expected refinement level R as follows:

$$R = \begin{cases} \log_{2^p}\left(\dfrac{S}{S_{refine}}\right), S > S_{refine} \\ \log_{2^p}\left(\dfrac{S}{S_{coarsen}}\right), S < S_{coarsen} \\ 0, \quad S_{coarsen} \le S \le S_{refine} \end{cases}$$

The percentage of grid points with a value of R in each integer step is presented in a chart at each adaption step, showing the progression of points to lower values of the sensor function with successive adaption cycles and further convergence of the flow solution. We note that the ratio $S_{refine}/S_{coarsen}$ should be greater than 2^p, to avoid grid points that are refined on one adapt cycle being flagged for coarsening on the next cycle. A parameter *SIGERR* is adopted from the OVERFLOW-D code to conveniently specify both S_{refine} and $S_{coarsen}$ as $(1/8)^{SIGERR}$ and $(1/8)^{SIGERR+2}$, respectively. This provides a ratio $S_{refine}/S_{coarsen} = 2^6$, or slightly more than one expected level of refinement.

The grid refinement process uses the value of R at each point by agglomeration, taking the maximum value within a cube or "box" of points (e.g., 8x8x8 points). If any point is marked for refinement ($R_{max}>1$), the box is flagged for refinement. Alternatively, if all points are marked for coarsening ($R_{max}<-1$), the box is flagged for coarsening. If the box is not flagged for refinement or coarsening, it is left at the current refinement level. In the current implementation, grid regions can only coarsen or refine by one grid level at a time. These coarsening and refinement boxes are then used in the near-body refinement process to create a new adapted grid system.

B. Controlling Grid Size and Growth

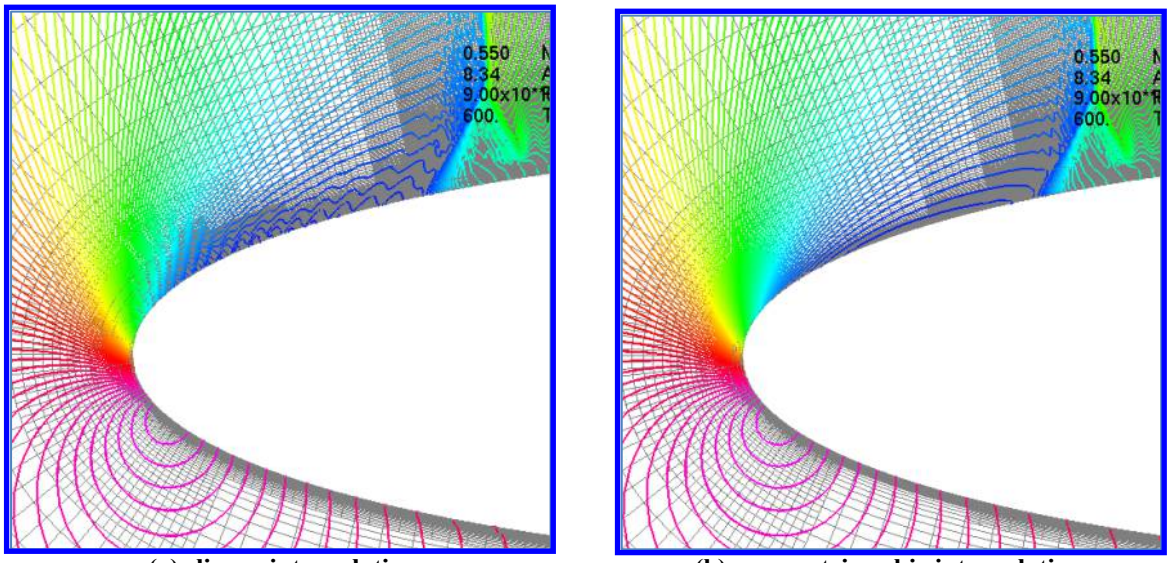
For this work, a goal-oriented procedure has been implemented to control the growth rate and maximum size of the combined near- and off-body grid system as it is adapted. This process is based on refining largest error cells first, following the concepts of Nemec, Aftosmis, and Wintzer,[11] and Nemec and Aftosmis.[12] While specific coarsening and refinement threshold values ($S_{coarsen}$, S_{refine}) for the sensor function are chosen by the user, the actual values used to trigger coarsening and refinement at each adaption step are adjusted automatically so the resulting grid size does not exceed a maximum size. This size is determined by the minimum of a percentage growth of the current size, and an absolute size. Limiting the growth rate to 20 or 30% at each adapt cycle results in a significantly more robust adaption process than refining all grid regions where the sensor function indicates refinement is needed. Further, this focuses refinement first on those regions showing the largest value of the sensor function. The ability to specify an absolute maximum grid size is useful so as to not exceed the available computing resources or job run time.

The specific mechanics of the adaption process are accomplished by calculating the new grid size based on input threshold values, then adjusting those values to meet the grid size limitations. A binary search method is chosen, since the relation between threshold values and grid size is not necessarily smooth. This matches the choice by Péron and Benoit.[13] Since the sensor function only has to be computed once and only the maximum value in each box of points needs to be examined, iterating to find acceptable thresholds takes little time compared to other adaption tasks such as interpolation of the flow solution onto the new grid system.

C. Grid Generation and Connectivity

Refined regions of near-body grids are generated using parametric cubic interpolation. In one dimension, the interpolation function f is expressed in parametric space ξ as

$$f(\xi) = f(0)[2\xi^3 - 3\xi^2 + 1] - f(1)[2\xi^3 - 3\xi^2] + f_\xi(0)[\xi^3 - 2\xi^2 + \xi] + f_\xi(1)[\xi^3 - \xi^2]$$



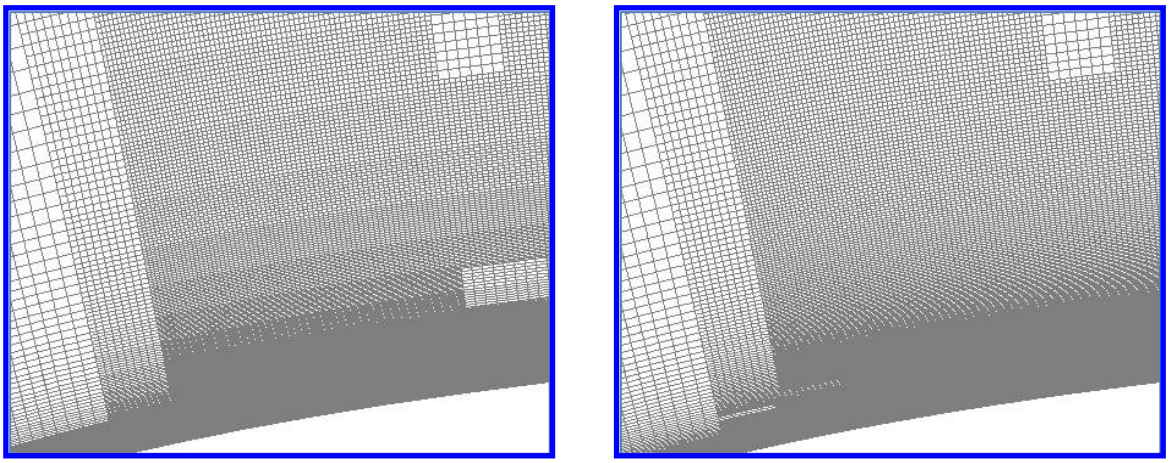
(a) linear interpolation **(b) parametric cubic interpolation**

Figure 1. Pressure contours about the nose of an airfoil when using linear or parametric cubic interpolation for grid refinement.

Here $\xi \in [0,1]$ represents the interpolation interval, one cell for the purpose of grid generation. To evaluate the function, values of f and its derivative in parametric space f_ξ are needed at the corners of the cell $\xi = 0$ and $\xi = 1$. Parametric cubic interpolation in three dimensions is similar, but first derivatives and all combinations of cross-derivatives in the (ξ, η, ζ) coordinate directions are needed at the cell corners.

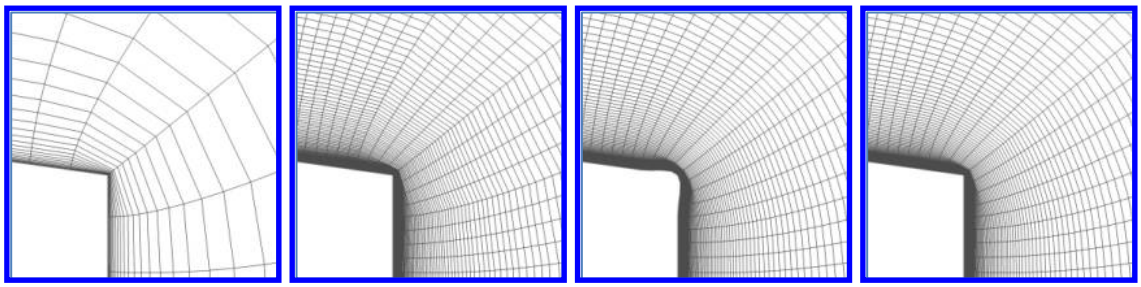
This approach preserves smooth geometry, avoiding faceting of the surface that would occur with linear interpolation. Figure 1 compares a transonic airfoil flow field using linear and parametric cubic interpolation. Pressure contours are smooth when using cubic interpolation but display oscillations resulting from the faceted surface representation when using linear interpolation. Parametric cubic interpolation also preserves grid stretching, which is critical for highly stretched viscous grids. A comparison of grid stretching using linear and parametric cubic interpolation is shown in Fig. 2. In the stretched direction, refined regions using linear interpolation show sudden jumps in spacing corresponding to grid cells of the original grid. This is because linear interpolation yields constant spacing for refinement within each original grid cell. Since the original grid is stretched, the refined grid changes spacing at each original cell boundary. With cubic interpolation, the grid spacing changes smoothly in the refined regions.

A disadvantage of using parametric cubic interpolation is that sharp corners in the original grid become rounded as the grid is refined. To avoid this problem, derivative information at cell corners are determined by blending central



(a) linear interpolation **(b) parametric cubic interpolation**

Figure 2. Stretched grid refinement regions about the nose of an airfoil when using linear or parametric cubic interpolation.



(a) original grid **(b) linear interpolation** **(c) parametric cubic interpolation** **(d) parametric cubic interpolation with blended differences**

Figure 3. Original and refined grids around a sharp corner.

and one-sided differences based on local grid line curvature and stretching ratios. If a derivative is computed using central differencing, the value will be the same for neighboring cells, and the parametric space will have C1 continuity. If one-sided differences are used, the parametric space will be continuous in value, but not in slope. While this scheme does not preserve all geometry features, it can produce smooth refined grids that preserve significant corners. This is illustrated by a close-up of the trailing edge of a blunt airfoil in Fig. 3. Linear interpolation preserves the sharp corner, while the refined grid produced with cubic interpolation rounds the geometry very close to the corner. Using the blended central and one-sided differencing allows the cubic interpolation scheme to maintain smoothness in most of the grid, while preserving the sharp corner, and transitioning smoothly between the two. For the purpose of grid refinement, the problem of detecting features such as corners exists for the entire volume grid, not just the surface. If the surface grid line were to preserve the sharp corner but the next line off the surface was smooth, the resulting grid would have a sudden jump in spacing and would not be suitable for use.

In this work, two blending functions are used: one to handle turning of grid lines (B_θ) and one for changes in grid spacing (B_{SR}). Both blending functions range from 1 (slopes determined by central differencing) to 0 (slopes calculated with one-sided differences). The angle blending function is calculated as

$$B_\theta = \begin{cases} 2cos^2\theta - 1, & |\theta| \le 45\ deg. \\ 0, & otherwise \end{cases}$$

For grid lines in a coordinate direction, the cosine of the turning angle θ at point i is easily calculated as the dot product of the normalized direction vector between points i and i-1, and the vector between i+1 and i. With this formulation, anywhere a grid line turns by more than 45 deg. the slope discontinuity will be fully preserved. As the turning angle approaches zero, the slope will be determined by central differencing, resulting in the same slope in the refined grid on both sides of the original grid point.

A lack of smoothness in grid spacing must also be addressed. For this, we define a blending function using the grid stretching ratio. The stretching ratio SR is calculated as the distance between points i+1 and i, divided by the distance between i and i-1. If the stretching ratio is less than one, we take the inverse, so that the function is independent of direction along the grid line and is always greater than or equal to one. We use a simple blending function

$$B_{SR} = \begin{cases} 0, & SR \ge 5 \\ 1, & SR \le 3 \\ linear, in\ between \end{cases}$$

Both blending functions are computed in each coordinate direction, at each original grid point. The product $B = B_\theta B_{SR}$ is used to blend central and one-sided differences to represent a derivative in that coordinate direction, with 2nd-order differences used where possible:

$$f_\xi(0) = B_i \frac{(f_{i+1} - f_{i-1})}{2} + (1 - B_i) \frac{(-3f_i + 4f_{i+1} - f_{i+2})}{2}$$

This formulation can be further improved to reduce the one-sided difference from 2nd-order to 1st-order based on the blending function at i+1.

$$f_\xi(0) = (f_{i+1} - f_i) - B_i \frac{(f_{i+1} - 2f_i + f_{i-1})}{2} - (1 - B_i)B_{i+1} \frac{(f_{i+2} - 2f_{i+1} + f_i)}{2}$$

These blending functions are not perfect. Two issues have been observed, one where the blending function changes too rapidly in highly skewed regions of the original grid. This may result in negative volumes in the refined grid. The other issue is that even with slight curvature there is a small amount of one-sided differencing used for local slopes. This discourages the use of very coarse original grids, as curved geometry will not be fully preserved with refinement.

Overset grid connectivity is established for refined grid regions by first allowing holes to be cut by neighboring geometry in the same way as they are cut in the corresponding original grid. For example, if a wing grid is cut by the fuselage, then all refined grid regions of that wing grid may also be cut by the fuselage. A refinement region (or the original near-body grid) may also have holes created due to finer refinement regions. Boundary conditions for a refined grid region may come from (1) physical boundary conditions (such as a solid wall) inherited from the original grid, (2) interpolation of flow variables from original or refined regions of the same grid, or (3) interpolated data from a different grid. All but the last are set up automatically as a result of the grid refinement process, leading to an efficient determination of grid connectivity.

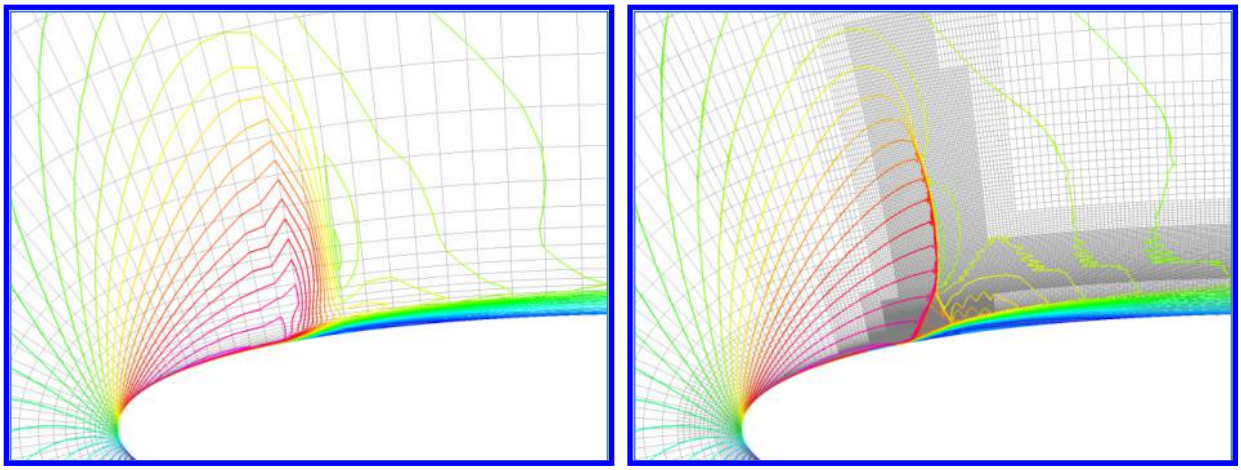
D. Grid and Solution Interpolation

In contrast to the creation of refinement regions for off-body grids (which are simply equispaced Cartesian grids), the grids for near-body refinement regions are generated through interpolation of the associated original grid. We create regions of any level of refinement (2x, 4x, etc.) directly from the corresponding subset of the original grid, rather than from the next-coarser refinement level. This has the advantage of requiring the least amount of information for the interpolation process. This is an advantage for parallel processing, when processes creating refinement regions for a given near-body grid must receive appropriate subsets of the original grid for interpolation. Alternatively, if the interpolation was done from the next-coarser refinement level rather than the original level, transfer of a finer grid than the original grid would be required. In addition, the next-coarser level might not be available until partway through the new grid generation process, impeding parallelism due to dependency. The current approach provides some level of parallelism by using non-blocking sends and blocking receives to exchange pieces of the grid needed to generate refinement regions. Grids for the new regions are then generated independently by each process as needed.

Once a new grid system has been generated, the flow solution has to be interpolated from the old grids to the new ones. This process is done (for each new refinement region) by interpolating flow quantities from any overlapping old refinement regions associated with the same original near-body grid, or the original grid itself. Old regions are checked coarse-to-fine, so that the final interpolated solution represents the finest available solution in each part of the grid. Interpolation weightings are easy to determine, since both old and new regions are referenced to the computational space of the original near-body grid. The current process uses trilinear interpolation (in computational space). This procedure is parallelized by structuring the interpolation as an outer loop through old refinement regions and an inner loop through new refinement regions. The MPI process that owns the old grid extracts appropriate subsets and sends them to processes that own overlapping new grids, using non-blocking sends. The receiving processes use blocking receives, and interpolate the grid subset immediately. If the owner of the old grid also owns an overlapping new grid, interpolation is done in-place. In this way, much of the processing of each old grid is performed in parallel, while only sending the required subsets of the grid to those processes that need them. This interpolation process is done for each near-body grid used in the simulation.

III. Examples

Five examples are presented here, representing steady and unsteady flows. These cases illustrate the capability and characteristics of the near-body solution adaption scheme, including improved accuracy at reduced cost (compared to global refinement), and parallel performance suitable for moving body simulations. These examples demonstrate the ability to refine flow features to allow investigation of physics, track and resolve features that impinge on other bodies, and reduce computational cost by resolving flow without global refinement of the grid.



(a) original grid (b) adapted grid

Figure 4. Comparison of Mach contours from the original grid and the adapted grid using four levels of adaption.

A. Transonic Airfoil

The first example is of transonic flow about a NACA 0012 airfoil. At Mach 0.55, 8.34 deg. angle-of-attack, and a Reynolds number of 9 million per chord, the flow accelerates around the leading edge and forms a shock wave at about 10% chord. This flow condition corresponds to Case A2 of the 1987 Viscous Transonic Airfoil Workshop, results of which were reported in Ref. 14. We use an O-grid with 253 points around the airfoil and 73 points in the viscous direction normal to the surface as the original near-body grid. In this case we use no off-body grids, and the near-body grid extends to the far field. Using an HLLE++ upwind scheme with 3rd-order MUSCL,[15] the flow solution on the original grid captures the shock in two grid cells. When four levels of grid adaption are used, the structure of the shock/boundary layer interaction is resolved more clearly, showing a lambda shock and a larger flow separation than shown in the original grid (Fig. 4). Weak shock reflections behind the lambda shock are evidenced by oscillations in the Mach contours. Finer contour levels would reveal alternating slightly supersonic and subsonic regions. (In contrast, oscillations behind the main shock are an artifact of the upwind scheme, where the shock wave stair-steps through the grid. The adaption scheme has refined the grid to preserve these features as they are convected downstream.) The improved flow feature resolution is reflected in the surface pressure coefficient as shown in Fig. 5, a result of the lambda shock and more pronounced separation bubble.

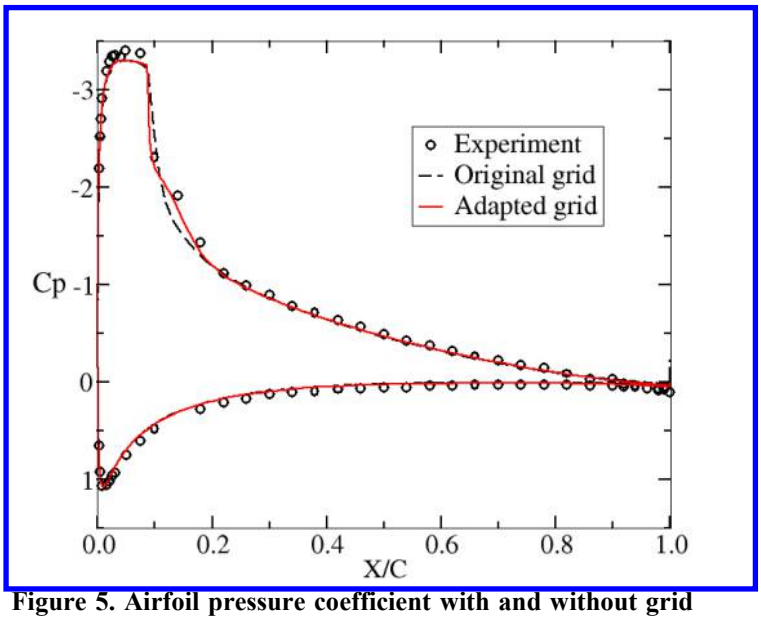
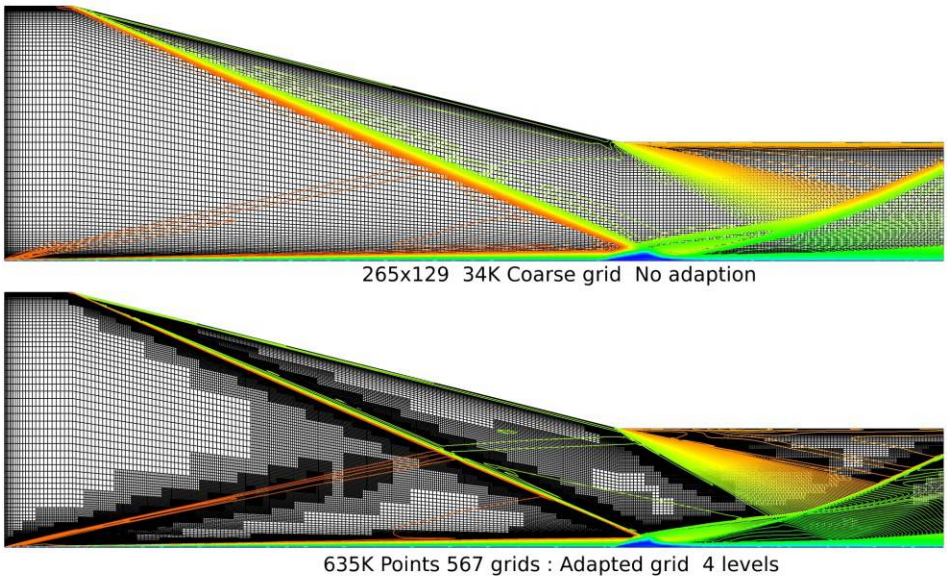


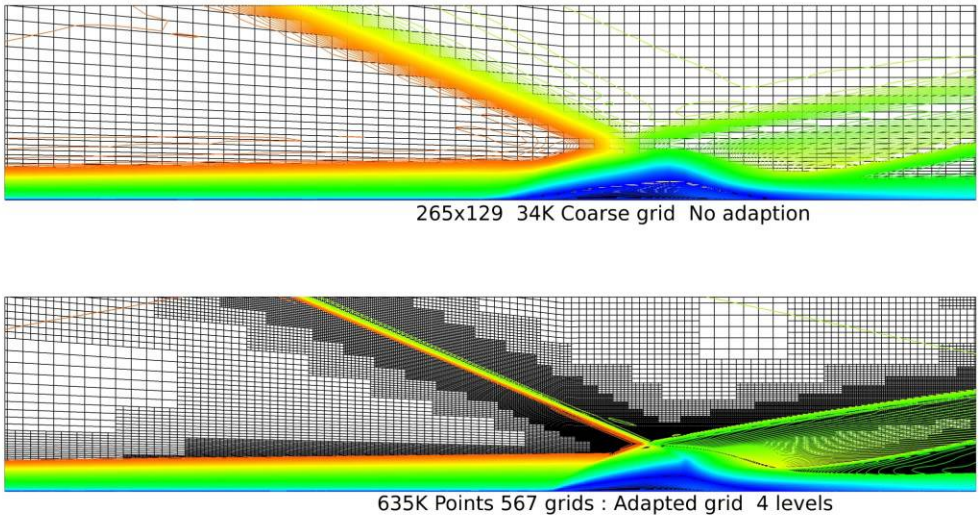
Figure 5. Airfoil pressure coefficient with and without grid adaption, compared to experimental data (from Ref. 14).

B. Shock Wave/Turbulent Boundary Layer Interaction

The second example is a two-dimensional Mach 5 shock wave/turbulent boundary layer interaction (SWTBLI) test case,[16] with a 14 deg. ramp on the upper surface creating a shock that interacts with the boundary layer on the lower surface. This flow field is designed to be a test for turbulence models. The use of grid adaption allows more confidence that the computed solution obtained is not affected by poor grid resolution. A coarser version of the grid used by Lillard et al.[17] is used here as the original near-body grid. (Again, no off-body grids are used.) Second-order central differencing with matrix dissipation[18] is used for the inviscid fluxes, along with the Menter SST turbulence



(a) Overall comparison of supersonic inlet grid and Mach contours.



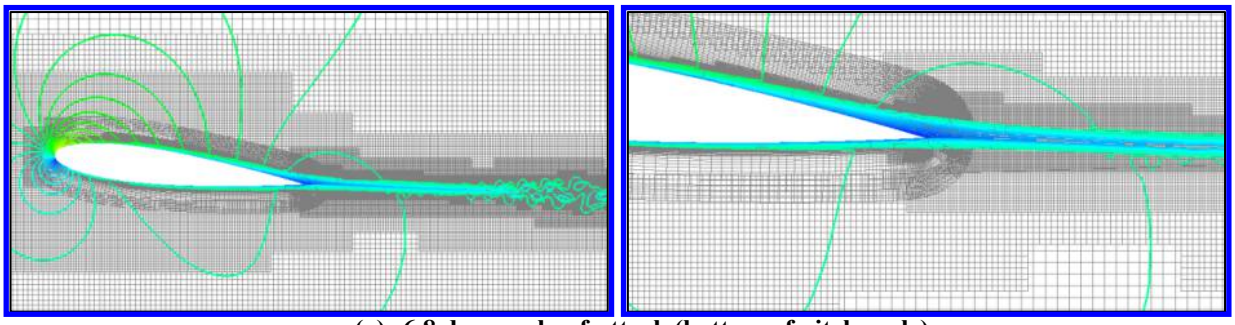
(b) Close-up of shock/boundary layer interaction.

Figure 6. Comparison of supersonic inlet grid and flow field with and without grid adaption.

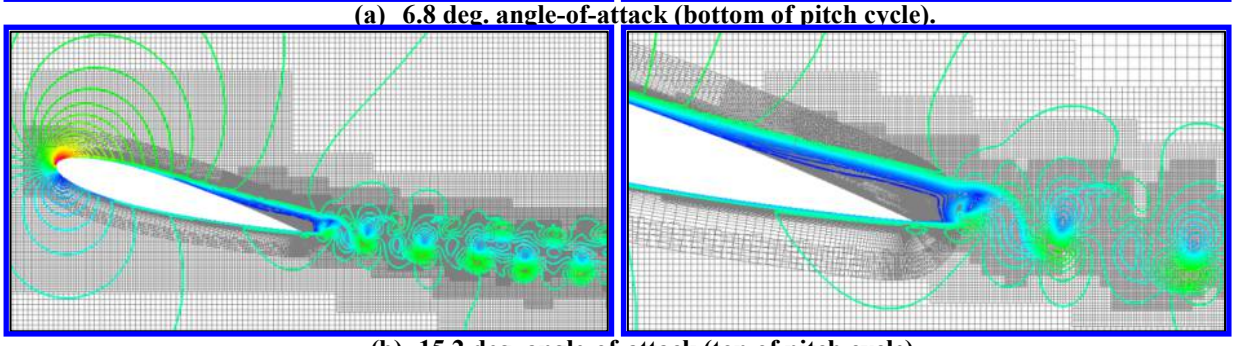
model.[19] Images showing the overall grid and Mach contours, and close-ups of the shock/boundary layer interaction, are shown in Fig. 6. The solution with 4 levels of grid adaption clearly shows the details of the interaction, including the separation bubble, shock intersections, expansion fan, and recompression shock on the back side of the separation.

C. Pitching Airfoil

This example is a two-dimensional simulation of a pitching NACA 0015 airfoil, with experimental data from Piziali.[20] This case has a free-stream Mach number of 0.29, Reynolds number of 1.95 million per chord, and an angle-of-attack range of 6.8 to 15.2 deg. The reduced frequency of the oscillation is 0.2. The purpose of this test case is to demonstrate the ability of the grid adaption to follow flow features that are moving, using a coupled near-body/off-body grid system. Here the airfoil is resolved by an O-grid that extends 0.1 chords away from the surface. The near-body grid adaption process has not been programmed to handle periodic grid boundaries, so the O-grid is split into two overlapping grids. The initial off-body grid resolution near the airfoil is 0.01 chords. We use a 5th-order WENO[15]



(a) 6.8 deg. angle-of-attack (bottom of pitch cycle).



(b) 15.2 deg. angle-of-attack (top of pitch cycle).

Figure 7. Mach contours for pitching airfoil, showing grid adaption of boundary layer and wake (close-up of trailing edge region on left).

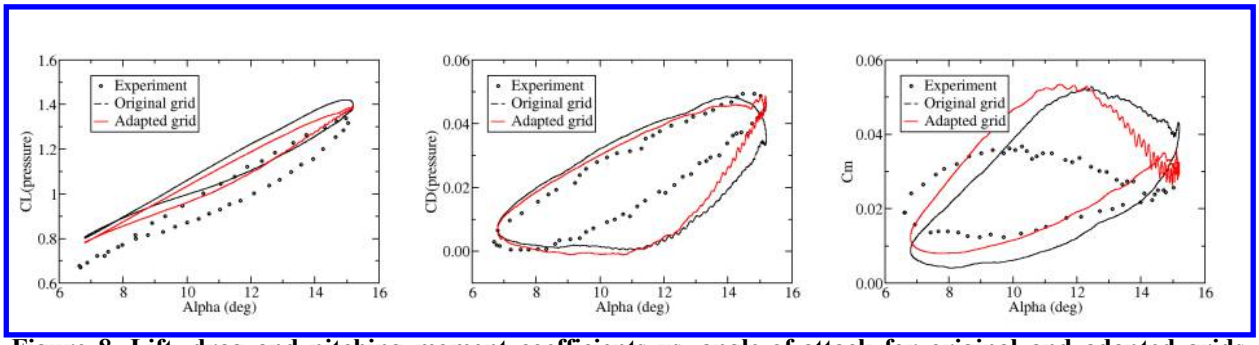


Figure 8. Lift, drag and pitching moment coefficients vs. angle-of-attack for original and adapted grids, compared to experimental data from Ref. 20.

scheme with HLLE++ upwind differencing for the inviscid fluxes. The Spalart-Allmaras turbulence model[21] is used, with the SARC rotational and curvature correction.[22] Further, delayed detached eddy simulation[23] (DDES) is used to reduce eddy viscosity in separated regions. While the use of DDES in two-dimensional simulations is not physically appropriate (and we are not endorsing it here), it enhances the richness of flow features in this calculation, providing a more interesting test for grid adaption.

The flow simulation is first converged with the airfoil fixed at the lowest angle-of-attack, and grid adaption is started with two levels of refinement on both the near- and off-body grids. Airfoil pitching is started, with 5760 time-steps per cycle and 100 subiterations per time-step, using dual time-stepping for 2nd-order accuracy in time. The grid is adapted every 10 time-steps. The adapted grid system and Mach contours are shown in Fig. 7, at the bottom and top of the pitching cycle. We see that the near-body refinement regions have followed the thickening of the boundary layer on the upper surface, and refined the trailing edge wake region. The off-body grids have continued this refinement in the wake.

The two-dimensional grid system averaged 670 thousand grid points. This simulation was performed using 16 MPI processes, with grids being redistributed for load-balancing at every adaption step. The adaption process averaged 0.24% of the total simulation time per pitch cycle. We consider this an acceptable cost for the advantage of having solution adaption as part of the simulation process. For comparison, lift, drag, and pitching moment coefficients vs.

angle-of-attack are plotted in Fig. 8 for the original and adapted grid systems, along with experimental data from Piziali.[17] Grid refinement in the separated flow segment of the pitch cycle accentuates the trend toward negative pitch damping at the top of the pitch cycle. The effect of reduced eddy viscosity due to DDES is indicated by the large oscillations in pitching moment, a result of shed vortices being stronger in two dimensions than in a real three-dimensional flow.

D. Blade/Vortex Interaction

This case simulates the wake and tip vortex of one wing impinging on a trailing wing. It is meant to mimic blade/vortex interaction in a helicopter rotor flow field. A wind tunnel test of this configuration was performed by Wittmer et al.[24] Both leading and trailing wings use the NACA 0012 airfoil with the same chord, and have a flat wingtip. Both wings are mounted on the tunnel wall at 5 deg. angle-of-attack, with the trailing wing 14 chords directly behind the leading wing. The leading and trailing wings have spans of 4.33 and 4.94 chords, respectively. The intent of the experiment was to measure the effect of the leading wing tip vortex on the pressure distribution of the trailing wing.

Here we test the combination of near-body and off-body grid adaption on a steady three-dimensional flow. The near-body grids extend 0.1 chords from each wing. Each wing is gridded with a (split) O-grid covering the main part of the span, and a tip cap grid coving the flat tip and overlapping the main wing grids. The tunnel wall is modeled as an inviscid wall. A 5th-order WENO scheme with Roe upwind differencing is used for inviscid fluxes, to help preserve the wakes and tip vortices. Two levels of refinement are used, with adaption performed every 20 or 50 steps, for a total of 21 adaption cycles. Figure 9 shows a detail of surface pressure on the tip of the leading wing, with grid refinement on a slice of the tip grid. The grid adaption in the wing grids helps generate a more concentrated tip vortex from the leading wing by reducing numerical dissipation due to the grid. Figure 10 shows a slice of the flow field ahead of the trailing wing showing the incoming wake, and a slice just downstream of the trailing wing, showing wakes from both wings. The leading wing wake has been sliced by the trailing wing, with the majority of the wake passing below the wing and the tip vortex grazing the top of the wing. Near-body and off-body grid adaption has allowed this grid system to propagate the vortex to the trailing wing and beyond.

To evaluate the computational requirements of the adaption process, we look at the percent of total run time spent on the grid adaption. For this example, 6 grid adaptions were performed during a run of 300 steps. The adaption process accounted for 1% of the total run time. The grid system averaged 115 million points, which was split among 20 MPI processes.

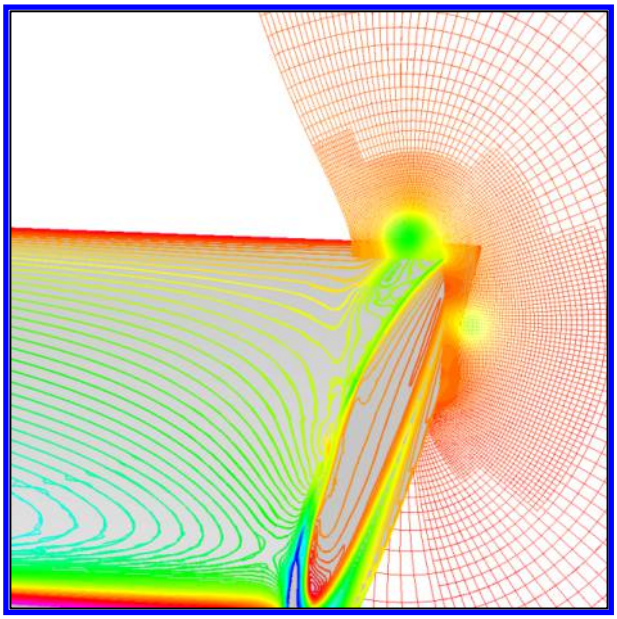


Figure 9. Surface pressure and slice of adapted tip grid on leading wing.

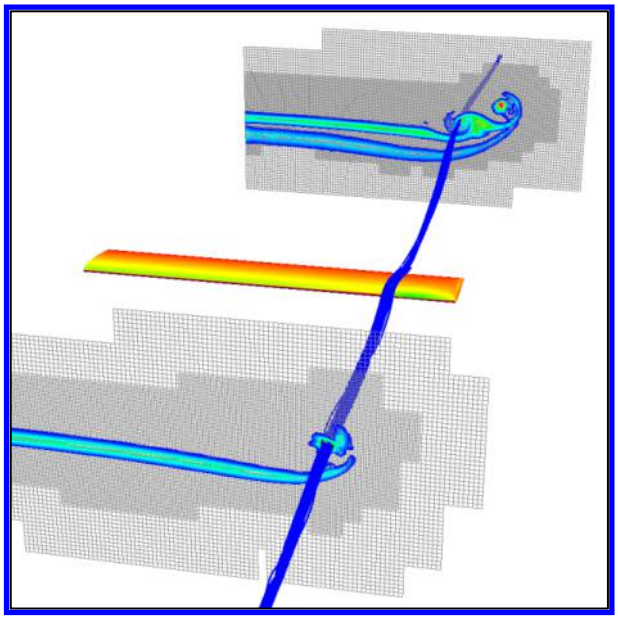


Figure 10. Trailing wing surface pressure and entropy contours on slices of adapted off-body grids. Particle traces indicate tip vortex location.

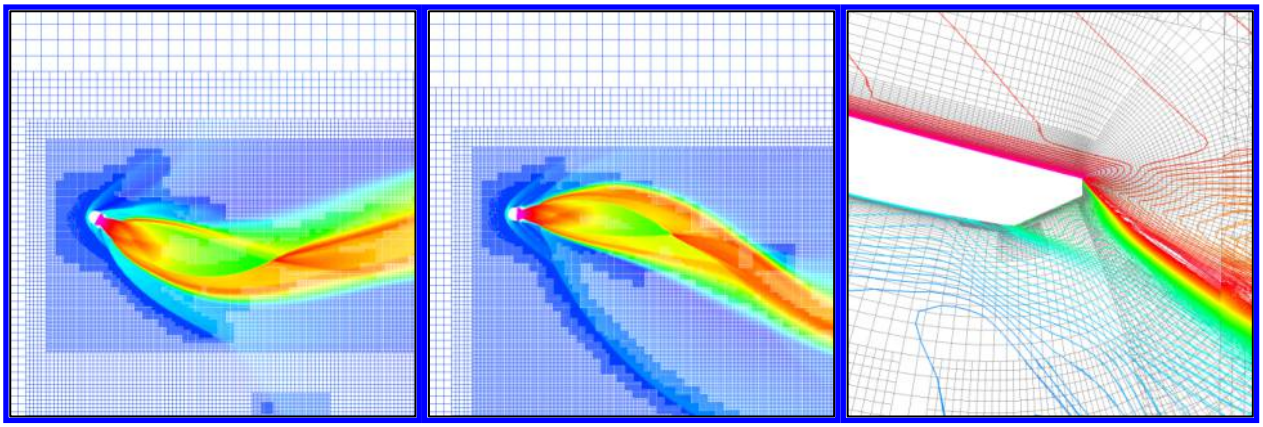


Figure 11. Pitching rocket plume in Mach 2 flow, at two different angles. Adapted grid is colored by logarithm of temperature. At right is a close-up of the nozzle lip region.

E. Rocket Plume in Cross-Flow

A final test case evaluates computational resources for grid adaption in an unsteady flow. Here a generic rocket engine is throttled up as it pitches from 0 to 30 deg. cross-flow angle in a Mach 2 free-stream flow. The Reynolds number is 4.7 million based on the nozzle diameter. One level of grid refinement is used, with adaption every 10 time-steps and 20 subiterations per time-step. This case serves as a testbed for a simulation of Space Launch System stage separation, which uses booster separation motors firing in cross-flow to push the solid rocket motors away from the core vehicle.

Figure 11 shows contours of the log of temperature on the symmetry plane at two different pitch angles. The off-body grid refinement has followed the plume structure, showing that the adaption frequency is sufficient to keep up with the changing flow field. A detail of the flow near the nozzle illustrates the effect of near-body adaption in resolving the shear layer as it leaves the nozzle lip, and the plume structure near the nozzle. This simulation averages 60 million grid points in 3000 grids, and uses 200 MPI processes. Grid adaption takes 2.8% of the total run time. This performance level is quite adequate for unsteady flow simulations on grids of the order of 100 million grid points.

IV. Problems and Future Work

The example cases shown in the previous section demonstrate significant success in implementation of a grid adaption scheme for near-body curvilinear grids in an overset grid approach. However, certain aspects of the process have proved troublesome and need additional work. The blended central/one-sided differencing for parametric cubic refinement causes negative cell volumes for some grids, specifically in highly skewed original grids. Further refinement of the blending formulas may alleviate this problem.

The current adaption process only allows isotropic refinement in computational space. While this is sufficient to provide grid refinement and is less costly than global refinement, grid sizes can still be quite large. Anisotropic refinement could allow spanwise refinement without increasing viscous clustering, or shear layer refinement without adding points in the chordwise direction. However, the increased coding complexity is daunting. A related problem is the reduced viscous wall spacing resulting from adaption at the wall. Three levels of adaption can result in an initial spacing an order of magnitude smaller than required, and may result in slow convergence of the flow solver and turbulence model. In Ref. 9, Su notes that slow convergence or divergence of the turbulence model can be caused by subtleties of wall distance calculation in refined regions.

Finally, feature-based adaption, or adapting the grid to specific flow features or characteristic properties of the flow variables, is generally accepted to be less efficient than output-based adaption such as the adjoint method for error estimation, which can guide grid adaptation specifically to improve a functional output such as lift or drag (cf. Ref. 25). The incorporation of some form of output-based error estimate as the sensor function to drive the present adaption scheme could improve its usefulness for engineering applications.

V. Summary

We have presented a scheme for solution adaption of the curvilinear near-body grids in a structured, overset grid flow solver. The grid adaption and solution interpolation are an integral part of the flow solver, providing a high level of efficiency. Parametric cubic interpolation is used to create refined regions of the original near-body grids in computational space, preserving both smooth geometry and smooth grid stretching. A blending of central and one-sided differencing is used to determine spatial derivatives at cell corners, allowing preservation of sharp corners and other grid discontinuities.

The scheme is implemented in a parallel fashion using MPI; in particular, the interpolation of the grid and flow solution onto the new grid system. Load-balancing of the flow solution process is maintained over adaption cycles. Further, a procedure for controlling growth rate and maximum size of the adapted grid system has been implemented, which benefits the robustness of the adaption process and control of computational resources. Efficient adaptation for both steady and unsteady flows has been demonstrated.

This scheme, coupled with off-body adaption already developed, provides an engineering capability to improve the accuracy of flow simulations while reducing the computational cost compared to global refinement of the grid. By providing an automated adaption process, the scheme also reduces the burden on the user to refine grids by hand.

Acknowledgments

This work is sponsored by the NASA Revolutionary Vertical Lift Technology and Transformational Tools and Technologies Projects, and the NASA Space Launch System Program. Computations were performed on the NAS Pleiades supercomputer at NASA Ames Research Center.

References

[1]Nichols, R. H., Tramel, R. W., and Buning, P. G., "Solver and Turbulence Model Upgrades to OVERFLOW 2 for Unsteady and High-Speed Applications," AIAA Paper 2006-2824, June 2006.

[2]Buning, P. G., and Pulliam, T. H., "Cartesian Off-Body Grid Adaption for Viscous Time-Accurate Flow Simulation," AIAA Paper 2011-3693, June 2011.

[3]Meakin, R. L., "An Efficient Means of Adaptive Refinement Within Systems of Overset Grids," AIAA Paper 95-1722, June 1995.

[4]Chan, W. M., Meakin, R. L., and Potsdam, M. A., "CHSSI Software for Geometrically Complex Unsteady Aerodynamic Applications," AIAA Paper 2001-0593, Jan. 2001.

[5]Kamkar, S. J., Wissink, A. M., Sankaran, V., and Jameson, A., "Combined Feature-Driven Richardson-Based Adaptive Mesh Refinement for Unsteady Vortical Flows," *AIAA Journal*, Vol. 50, No. 12, 2012, pp. 2834–2847.

[6]Henshaw, W. D., and Schwendeman, D. W., "An Adaptive Numerical Scheme for High-Speed Reactive Flow on Overlapping Grids," *Journal of Computational Physics*, Vol. 191, 2003, pp. 420–447.

[7]Henshaw, W. D., and Schwendeman, D. W., "Moving Overlapping Grids with Adaptive Mesh Refinement for High-Speed Reactive and Non-Reactive Flow," *Journal of Computational Physics*, Vol. 216, 2006, pp. 744–779.

[8]Henshaw, W. D., and Schwendeman, D. W., "Parallel Computation of Three-Dimensional Flows Using Overlapping Grids with Adaptive Mesh Refinement," *Journal of Computational Physics*, Vol. 227, 2008, pp. 7469–7502.

[9]Su, X., "Accurate and Robust Adaptive Mesh Refinement for Aerodynamic Simulation with Multi-Block Structured Curvilinear Mesh," *Int. J. Numer. Meth. Fluids*, Vol. 77, 2015, pp. 747–766.

[10]Warren, G. P., Anderson, W. K., Thomas, J. L., and Krist, S. L., "Grid Convergence for Adaptive Methods," AIAA Paper 91-1592, June 1991.

[11]Nemec, M, Aftosmis, M. J., and Wintzer, M., "Adjoint-Based Adaptive Mesh Refinement for Complex Geometries," AIAA Paper 2008-0725, Jan. 2008.

[12]Nemec, M., and Aftosmis, M. J., "Toward Automatic Verification of Goal-Oriented Simulations," NASA/TM-2014-218386, Aug. 2014.

[13]Péron, S., and Benoit, C., "Automatic Off-Body Overset Adaptive Cartesian Mesh Method Based on an Octree Approach," *Journal of Computational Physics*, Vol. 232, 2013, pp. 153–173.

[14]Holst, T. L., "Viscous Transonic Airfoil Workshop Compendium of Results," AIAA Paper 87-1460, June 1987.

[15]Tramel, R. W., Nichols, R. H., and Buning, P. G., "Addition of Improved Shock-Capturing Schemes to OVERFLOW 2.1," AIAA Paper 2009-3988, June 2009.

[16]Schülein, E., Krogmann, P., and Stanewsky, E., "Documentation of Two-Dimensional Impinging Shock/Turbulent Boundary Layer Interaction Flow," Report 1B 223-96 A 49, DLR, German Aerospace Center, Göttingen, Germany, Oct. 1996.

[17]Lillard, R. P., Olsen, M. E., Oliver, A. B., Blaisdell, G. A., and Lyrintzis, A. S., "The lagRST Model: a Turbulence Model for Non-Equilibrium Flows," AIAA Paper 2012-0444, Jan. 2012.

[18]Swanson, R. C., and Turkel, E., "On Central-Difference and Upwind Schemes," *Journal of Computational Physics*, Vol. 101, 1992, pp. 292–306.

[19]Menter, F. R., "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications," *AIAA Journal*, Vol. 32, No. 8, 1994, pp. 1598–1605.
[20]Piziali, R. A., "2-D and 3-D Oscillating Wing Aerodynamics for a Range of Angles of Attack Including Stall," NASA TM-4632, Sept. 1994.
[21]Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *La Recherche Aérospatiale*, No. 1, 1994, pp. 5–21.
[22]Spalart, P. R., and Shur, M. L., "On the Sensitization of Turbulence Models to Rotation and Curvature," *Aerospace Science and Technology*, Vol. 1, No. 5, 1997, pp. 297–302.
[23]Spalart, P. R., Deck, S., Shur, M. L., Squires, K. D., Strelets, M. K., and Travin, A. K., "A New Version of Detached-Eddy Simulation, Resistant to Ambiguous Grid Densities," *Theor. Comp. Fluid Dyn.*, Vol. 20, 2006, pp. 181–195.
[24]Wittmer, K. S., Devenport, W. J., Rife, M. C., and Glegg, S. A. L., "Perpendicular Blade Vortex Interaction," AIAA Paper 94-0526, Jan. 1994.
[25]Park, M. A., "Adjoint-Based, Three-Dimensional Error Prediction and Grid Adaptation," *AIAA Journal*, Vol. 42, No. 9, 2004, pp. 1854–1862.