Third-Order Adaptive Space-Time Edge-Based Solver for Two-Dimensional Unsteady Inviscid Flows

Hiroaki Nishikawa*
National Institute of Aerospace, Hampton, VA 23666, USA

Emmett Padway†
Department of Mechanical Engineering, University of Wyoming, Laramie, WY 82071, USA

A third-order space-time edge-based discretization is developed for solving two-dimensional unsteady inviscid-flow problems with adaptive tetrahedral grids. The resulting discrete problem is solver by a robust Jacobian-Free Newton-Krylov solver with the Gauss-Seidel relaxation scheme applied to a first-order Jacobian used as a variable preconditioner. Efficient unsteady computations are demonstrated with anisotropic grid adaptation, which dramatically reduces the grid size required to obtain a sufficiently accurate solution.

1. Introduction

Powerful computers with ever-increasing parallelism are promoting the use of high-fidelity Computational Fluid Dynamics (CFD) models in engineering analysis and design involving complex flow fields. However, time-dependent unstructured-grid simulations over complex geometries still require an enormous amount of CPU time, as long as months, even with thousands of cores [1]. For example, an aerodynamic simulation over a complete aircraft with a landing gear configuration on a 244-million-node grid (utilizing an implicit-time-stepping with 20 iterations per time step) took about four month with 3000 cores to simulate 0.3 real-time seconds (100,000 time steps) [1]. It is important to note that further increasing the number of cores does not necessarily speed-up the computation because of the latency barrier: the inter-processor communication begins to dominate the floating-point computation (which begins typically below $60,000$ nodes per core in unstructured-grid solvers). Note also that the individual CPU clock speed is limited for a physical reason (i.e., heating). Consequently, to achieve further reduction in computing time, we must exploit concurrency, i.e., develop algorithms that exploit the number of cores that still continues to increase [2]. For this reason, there is an increasing interest in space-time methods [3–6], where a two-dimensional (2D) unsteady problem, for example, is solved as a steady problem in a three-dimensional (3D) domain with the space-time coordinates (x, y, t), where (x, y) denotes the spatial coordinates and t the physical time. Compared with traditional sequential algorithms that march in time, space-time methods allow a significantly larger number of cores by solving an unsteady problem over the entire time in parallel. In the aircraft simulation example mentioned above, a simple tensor-product space-time grid (i.e., the one created by placing the 244-million-node grid at each time level and connecting them with edges along the time axis) will have $244\times10^6\times10^5 = 2.44\times10^{13}$ nodes, allowing $400,000,000$ cores with $60,000$ nodes per core, which is around the range of emerging exascale computing, whereas a traditional time-marching method will allow only up to 4066 cores. This example indicates the potential of space-time methods for future CFD simulations although it is still desirable to reduce the problem size for performing multiple simulations efficiently or running high-fidelity simulations with finer grids.

The most striking feature of space-time methods, radically different from the traditional time-marching methods, is that the time step corresponds merely the grid spacing along the time axis of an element in a space-time grid. That is, a large element corresponds to a large time step; and a small element corresponds

*Associate Research Fellow (hiro@nianet.org), 100 Exploration Way, Hampton, VA 23666 USA, Associate Fellow AIAA
†Graduate student, (epadway@uwyo.edu), Department of Mechanical Engineering, University of Wyoming, Member AIAA

to a small time step. Therefore, the required size of a space-time grid to capture salient flow features can be dramatically reduced especially on an unstructured grid, where small elements clustered only around regions for large solution variations (e.g., shock waves, vortex shedding, etc.) [7]. More specifically, anisotropic space-time grid adaptation (e.g., elongated element along the time axis where temporal variation is small), is expected to bring the problem size down well within the range of emerging exascale machines [6, 8]. Note that the problem size reduction is highly desirable since it will allow more simulations for a given CPU time; such will be required, for example, in a design optimization. Anisotropic grid adaptation involves local operations such as refinement/coarsening and node movement, and it has been known that such can be performed most efficiently on simplex (i.e., triangular and tetrahedral) grids [6, 8–13].

Further improvements in efficiency are desirable and can be achieved in the discretization and the solver. For the discretization, we consider the third-order edge-based discretization, which is a highly economical discretization method comparable in cost to the second-order edge-based method [14–16]. Unlike other third-order methods for unstructured grids, this method does not require computations and storage of second derivatives [16] and achieves third-order accuracy on linear tetrahedral grids even over curved boundaries [17]. The resulting system of nonlinear residual equations can be efficiently solved by a Jacobian-Free Newton-Krylov solver, which in this work is constructed with the generalized conjugate residual (GCR) method with a variable-preconditioner [18]. For viscous problems, the hyperbolic Navier-Stokes method [15] is expected to bring further improvements in both efficiency and accuracy, especially in the gradient accuracy on adaptive grids; but it is not considered in this paper.

This paper presents preliminary results for two-dimensional unsteady inviscid flow problems towards the development of a robust and accurate hyperbolic viscous solver on adaptive grids. As such, issues with adaptive space-time methods are also discussed, which will be addressed in subsequent papers. In order to establish an accurate and reliable space-time solver, we will focus on 2D unsteady problems with 3D grids but develop methods extendable to 4D grids towards the ultimate goal of achieving space-time simulations for realistic 3D unsteady flow problems.

2. Governing Equations

Consider the unsteady Euler equations in two dimensions:

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}_x}{\partial x} + \frac{\partial \mathbf{f}_y}{\partial y} = 0, \tag{2.1}$$

where

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad \mathbf{f}_x = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{bmatrix}, \quad \mathbf{f}_y = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{bmatrix} \tag{2.2}$$

ρ is the density, u and v are the x- and y-velocity components, p is the pressure, E is the specific total energy, and $H = E + p/\rho$ is the specific total enthalpy. In this work, the system is considered as a steady system in the three-dimensional space with the coordinates x, y, and t. For convenience, we therefore write the system as

$$\frac{\partial \mathbf{f}_x}{\partial x} + \frac{\partial \mathbf{f}_y}{\partial y} + \frac{\partial \mathbf{g}}{\partial t} = 0, \tag{2.3}$$

where

$$\mathbf{g} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \tag{2.4}$$

and solve it over a three-dimensional domain.

3. Second-Order Space-Time Edge-Based Discretization

We consider a tetrahedral grid in the space-time domain and store the numerical solution at each node. The second-order edge-based discretization for the system (2.3) is given, at a node j, by

$$\sum_{k\in\{k_j\}} \mathbf{\Phi}_{jk} A_{jk} = 0, \tag{3.1}$$

where V_j is the measure of the dual control volume around the node j, $\{k_j\}$ is a set of neighbor nodes of j, A_{jk} is the magnitude of the directed area vector $\mathbf{n}_{jk} = (n_x, n_y, n_t)$, which is a sum of the directed-areas corresponding to the dual-triangular faces associated with all tetrahedral elements sharing the edge $[j,k]$ (see Figure **??**), and $\mathbf{\Phi}_{jk}$ is a numerical flux. The numerical flux is constructed as a sum of the 2D space flux and the time flux:

$$\mathbf{\Phi}_{jk} \;=\; \mathbf{\Phi}^{space}_{jk}|(n_x, n_y)| + \mathbf{\Phi}^{time}_{jk}|n_t|, \tag{3.2}$$

where the space flux $\mathbf{\Phi}^{space}_{jk}$ is computed by the 2D Roe flux with $\hat{\mathbf{n}} = (n_x, n_y)/|(n_x, n_y)|$,

$$\mathbf{\Phi}^{space}_{jk} = \frac{1}{2}\left[\mathbf{f}_L + \mathbf{f}_R\right] - \frac{1}{2}\left|\mathbf{A}_n\right|\left(\mathbf{u}_R - \mathbf{u}_L\right), \tag{3.3}$$

where $\mathbf{f}_R = \mathbf{f}_n(\mathbf{w}_R)$, $\mathbf{f}_L = \mathbf{f}_n(\mathbf{w}_L)$, $\mathbf{f}_n = [\mathbf{f}_x, \mathbf{f}_y]\cdot\hat{\mathbf{n}}$, $\mathbf{A}_n = \partial\mathbf{f}_n/\partial\mathbf{u}$ evalauted at the Roe averaged state, and the time flux is computed by the upwind flux with $\hat{n}_t = n_t/|n_t|$:

$$\mathbf{\Phi}^{time}_{jk} = \frac{1}{2}\left[\mathbf{g}_L + \mathbf{g}_R\right] - \frac{1}{2}\left(\mathbf{u}_R - \mathbf{u}_L\right), \tag{3.4}$$

where $\mathbf{g}_L = \mathbf{u}_L\hat{n}_t$ and $\mathbf{g}_R = \mathbf{u}_R\hat{n}_t$. The left and right states, $\mathbf{u}_L$ and $\mathbf{u}_R$, are computed directly from the primitive variables $\mathbf{w}_L$ and $\mathbf{w}_R$, respectively, linearly extrapolated by the kappa reconstruction scheme [19, 20]:

$$\mathbf{w}_L \;=\; \mathbf{w}_j + \left[\frac{1-\kappa}{2}(\partial_{jk}\mathbf{w}_j - \Delta\mathbf{w}_{jk}) + \frac{1+\kappa}{2}\Delta\mathbf{w}_{jk}\right], \tag{3.5}$$

$$\mathbf{w}_R \;=\; \mathbf{w}_k - \left[\frac{1-\kappa}{2}(\partial_{jk}\mathbf{w}_k - \Delta\mathbf{w}_{jk}) + \frac{1+\kappa}{2}\Delta\mathbf{w}_{jk}\right], \tag{3.6}$$

where κ is a real-valued parameter, and

$$\partial_{jk}\mathbf{w}_j = \nabla\mathbf{w}^{LSQ}_j\cdot(\mathbf{x}_k - \mathbf{x}_j), \quad \partial_{jk}\mathbf{w}_k = \nabla\mathbf{w}^{LSQ}_k\cdot(\mathbf{x}_k - \mathbf{x}_j), \quad \Delta\mathbf{w}_{jk} = \frac{1}{2}\left(\mathbf{w}_k - \mathbf{w}_j\right). \tag{3.7}$$

The nodal gradients $\nabla\mathbf{w}^{LSQ}_j$ and $\nabla\mathbf{w}^{LSQ}_k$ are computed by a weighted least-squares (LSQ) method with neighbor nodes. For the results shown in this abstract, we have used $\kappa = 0$.

The edge-based discretization as described above is designed to be exact for linear solution and fluxes, and therefore it is second-order accurate. Note that the numerical flux is computed only at the edge midpoint and this edge-based quadrature is exact for linear fluxes on arbitrary tetrahedral grids. It is in fact exact for quadratic fluxes as well. This is a special property of the edge-based quadrature, leading to a very economical third-order discretization as described in the next section.

4. Third-Order Space-Time Edge-Based Discretization

For the Euler equations, the edge-based method can be extended to third-order accuracy with very minor modifications: linear flux extrapolation,

$$\mathbf{f}_L \;=\; \mathbf{f}_n(\mathbf{w}_j) + \frac{1}{2}\left(\frac{\partial\mathbf{f}_n}{\partial\mathbf{w}}\right)_j \nabla\mathbf{w}^{LSQ}_j\cdot(\mathbf{x}_k - \mathbf{x}_j) \tag{4.1}$$

$$\mathbf{f}_R \;=\; \mathbf{f}_n(\mathbf{w}_k) - \frac{1}{2}\left(\frac{\partial\mathbf{f}_n}{\partial\mathbf{w}}\right)_k \nabla\mathbf{w}^{LSQ}_k\cdot(\mathbf{x}_k - \mathbf{x}_j) \tag{4.2}$$

and a quadratic LSQ method for computing the nodal gradients, $\mathbf{w}_j^{LSQ}$ and $\mathbf{w}_k^{LSQ}$. The quadratic LSQ method is implemented in two steps as described in Ref.[21]. If boundary conditions are imposed weakly, a special boundary flux quadrature formula will be required. But that is all required to achieve third-order accuracy. Note in particular that the numerical flux is still computed at the edge midpoint. This is one of the reasons that the third-order edge-based method is exceptionally more efficient than other third-order discretization methods.

Implicit quadratic gradient method: In this method, the gradients are determined as a solution to a global system of equations. It has a great potential for further reducing the third-order error on relatively coarse grids. It will be reported in the final paper.

5. Boundary conditions and flux quadrature

Boundary conditions are implemented weakly through the right state sent to the numerical flux. For the edge-based discretization, a special flux quadrature formula is requried to preserve the design accuracy. For second- and third-order accuracy, we employed the linearly-exact formula [22] and the quadratically-exact formula [17], respectively. Further details on the boundary conditions will be given in the final paper.

6. Nonlinear Solver

The global system of residual equations

$$\mathbf{Res}(\mathbf{U}) = 0, \tag{6.1}$$

is solved iteratively,

$$\mathbf{U}^{m+1} = \mathbf{U}^m + \Delta\mathbf{U}, \tag{6.2}$$

where m is the iteration counter and the correction $\Delta\mathbf{U}$ is computed by solving a linearized system. The Jacobian-Free Newton-Krylov solver is a robust and efficient method, where the exactly linearized system is solved without forming the exact Jacobian. It is constructed here based on the algorithm presented in Ref.[18] with the generalized conjugate residual (GCR) method: with $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_0 = -\mathbf{Res}(\mathbf{U}^m)$, $\mathbf{p}_0 = \tilde{\mathbf{A}}^{-1}\mathbf{r}_0$, perform for $i = 0, 1, 2, \cdots, i_{max}$

$$\alpha_i = \frac{(\mathbf{A}\mathbf{p}_i)^T\mathbf{r}_i}{(\mathbf{A}\mathbf{p}_i)^T(\mathbf{A}\mathbf{p}_i)}, \quad \mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i\mathbf{p}_i, \quad \mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i\mathbf{A}\mathbf{p}_i, , \tag{6.3}$$

where i_{max} is a user-defined integer, and, if not converged yet, compute $\mathbf{p}_{i+1} = \tilde{\mathbf{A}}^{-1}\mathbf{r}_{i+1}$, perform the orthogonalization for $k = 0, 1, 2, \cdots, i$:

$$\beta_i = \frac{(\mathbf{A}\mathbf{p}_i)^T(\mathbf{A}\mathbf{p}_k)}{(\mathbf{A}\mathbf{p}_k)^T(\mathbf{A}\mathbf{p}_k)}, \quad \mathbf{p}_{i+1} = \mathbf{p}_{i+1} + \beta_k\mathbf{p}_k, \quad \mathbf{A}\mathbf{p}_{i+1} = \mathbf{A}\mathbf{p}_{i+1} - \beta_k\mathbf{A}\mathbf{p}_k, \tag{6.4}$$

and go back to the step (6.3). At convergence, we obtain the correction as $\Delta\mathbf{U} = \mathbf{x}_k$. The symbol $\tilde{\mathbf{A}}^{-1}$ denotes a variable-preconditioner based on the multi-color Gauss-Seidel relaxation, which has a good smoothing property and can be made partition-independent, applied to approximately invert the lower-order compact Jacobian $\tilde{\mathbf{A}} = \frac{\partial\overline{\mathbf{Res}}}{\partial\mathbf{U}}$, where $\overline{\mathbf{Res}}$ is the residual with zero LSQ gradients, with the tolerance of 50% residual reduction or for a specified maximum number 15 of relaxations. Furthermore, $\mathbf{A} = \partial\mathbf{Res}/\partial\mathbf{U}$ is the true Jacobian, but the computation and storage are avoided by computing the matrix-vector product, e.g., $\mathbf{A}\mathbf{p}_k$, by the Fréchet derivative approximation:

$$\mathbf{A}\mathbf{p}_k = \frac{\mathbf{Res}(\mathbf{U}^m + \epsilon\mathbf{p}_k) - \mathbf{Res}(\mathbf{U}^m)}{\epsilon}, \tag{6.5}$$

where

$$\epsilon = \max(1, |\mathbf{U}^m|)\sqrt{10^{-16}}. \tag{6.6}$$

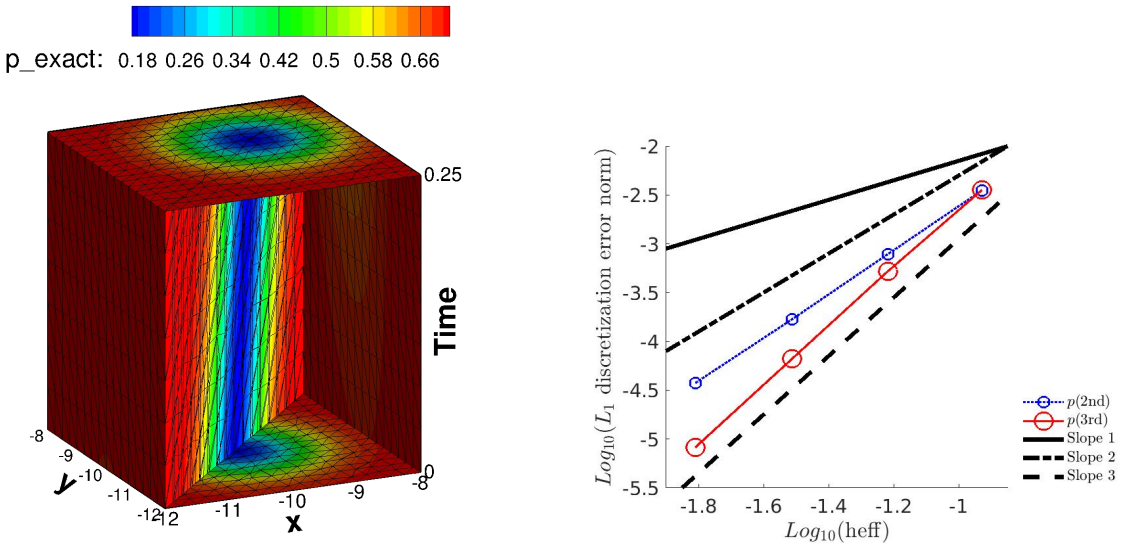
For all computations in this study, we set $i_{max} = 10$.

7. Results

At the time of submitting the abstract, results for the inviscid vortex test case are available as shown below. More results will be shown in the final paper.

7.1. Accuracy verification

We first verify third-order accuracy on space-time tetrahedral grids with the exact inviscid vortex solution as described in Ref.[23]. The domain is taken to be $(x, y, t) \in [-12, 08] \times [-12, -8] \times [0, 0.25]$. The inviscid vortex initially placed at $(x, y) = (-10, -10)$ will move to $(x, y) = (-9.5, -9.5)$ at $t = 0.25$, which is shown in the space-time domain in Figure 1(a) with the exact solution. For accuracy verification purpose, we fix the initial solution at $t = 0$, apply the weak Dirichlet condition on all side planes, and no values are specified at the final time plane $t = 0.25$. To determine the error convergence, we solved the problem on four levels of irregular tetrahedral grids with 2312, 17424, 135200, and 1065024 nodes. Results obtained with the second- and third-order methods are shown in Figure 1(b), where the L_1 error convergence is shown for the pressure, and second- and third-order accuracy are verified.



(a) Exact solution on the coarsest grid.

(b) Error convergence.

Figure 1: The coarsest grid and the error convergence result with the inviscid vortex solution.

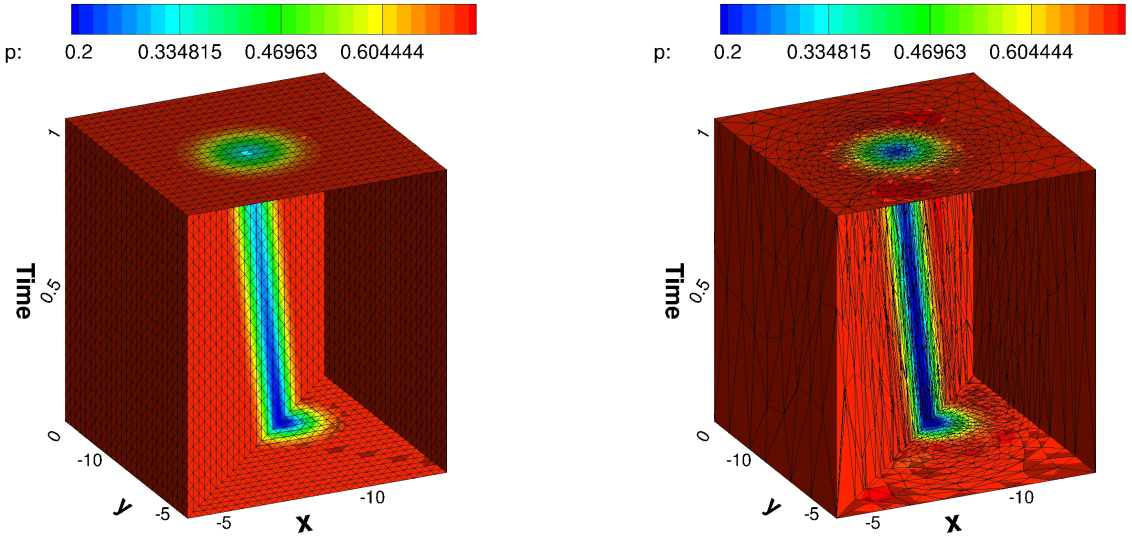
7.2. Inviscid vortex

To demonstrate the efficiency of adaptive space-time computations, we take the vortex problem and apply anisotropic grid adaptation. In this case, we extend the domain to $(x, y, t) \in [-14, -6] \times [-14, -6] \times [0, 1.0]$, i.e., extend the final time to $t = 1.0$. We begin the adaptive computation with the initial grid as shown in Figure 2(a), where the pressure contours obtained with the third-order edge-based method are shown. As can be seen, the vortex is rapidly dissipated towards the final time (i.e., the top boundary). This initial grid has 15,000 nodes and 79,488 tetrahedra. We then applied the anisotropic grid adaptation once and recomputed the solution on the adapted grid. See Figure 2(b). The grid adaptation was performed by using the open-source tool *refine* [11] developed by Mike Park at NASA Langley Research Center and available at `https://github.com/nasa/refine`. The adaptive grid has only 4,309 nodes and 21,128 tetrahedra. The L_1 errors in the pressure are 7.0e-03 on the initial grid and 4.3e-03 on the adaptive grid. Observe that the vortex is much better resolved on the adaptive grid. The results clearly indicate that the adaptive-grid method is significantly more efficient, achieving a lower error on a quarter-size grid.

If compared with a conventional time-stepping scheme, the adaptive space-time method is even more efficient as it naturally incorporates adaptive time-step controls in a very local manner. Comparison with a

conventional method will be shown in the final paper.

This result is expected to be further improved with an implicit quadratic gradient method (instead of the quadratic LSQ method), which is currently being implemented. Updated results and iterative convergence histories and CPU time comparison will be reported in the final paper.



(a) Initial grid.

(b) Adaptive grid.

Figure 2: Pressure contours on the initial grid and the adaptive grid for the inviscid vortex problem.

8. Final Paper

In the final paper, we plan to present the following items:

1. Improved results with an implicit quadratic gradient method.
2. Further details of the algorithms, including boundary conditions.
3. Additional results, e.g., a shock diffraction problem.
4. Comparison with a conventional time-stepping method.

References

[1]Eric J. Nielsen and Boris Diskin. High-performance aerodynamic computations for aerospace applications. *Parallel Computing*, 64:20–32, 2017.

[2]R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, J. B. Schroder, and S. Vandewalle. Multigrid methods with space-time concurrency. *Computing and Visualization in Science*, 18:123–143, 2017.

[3]C. M. Klaij, J. J. W. van der Vegt, and H. van der Ven. Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations. *J. Comput. Phys.*, 217:589–611, 2006.

[4]M. Behr. Simplex space-time meshes in finite element simulations. *Int. J. Numer. Meth. Fluids*, 57:1421–1434, 2008.

[5]M. J. Gander. 50 years of time parallel time integration. In *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113. Springer, 2015.

[6]Philip Caplan, Robert Haimes, David Darmofal, and Marshall Galbraith. Extension of local cavity operators to 3d+t space-time mesh adaptation. In *AIAA Scitech 2019 Forum*, AIAA Paper 2019-1992, San Diego, CA, 2019.

[7]T. J. R. Hughes and G. M. Hulbert. Space-time finite element methods for elastodynamics: formulations and error estimates. *Computer Methods in Applied Mechanics and Engineering*, 66:339–363, 1988.

[8]S. Jayasinghe, D. L. Darmofal, N. K. Burgess, M. C. Galbraith, and S. R. Allmaras. A space-time adaptive method for reservoir flows: formulation and one-dimensional application. *Comput. Geosci.*, 2:107–123, 2018.

[9]E. M. Lee-Rausch, C. L. Rumsey, and M. A. Park. Grid-adapted FUN3D computations for the second high lift prediction workshop. *J. Aircraft*, 52(4):1098–1111, 2014.

[10]X. Li, M. S. Shephard, and M. W. Beall. 3D anisotropic mesh adaptation by mesh modification. *Computer Methods in Applied Mechanics and Engineering*, 194(48):4915–4950, 2005.

[11]Michael A. Park and David L. Darmofal. Parallel anisotropic tetrahedral adaptation. In *Proc. of 46th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2008-917, Reno, Nevada, 2008.

[12]F. Alauzet, A. Dervieux, and A. Loseille. Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. *J. Comput. Phys.*, 229(8):2866–2897, 2010.

[13]T. Michal and J. Krakos. Anisotropic mesh adaptation through edge primitive operations. In *Proc. of 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, AIAA Paper 2012-159, Nashville, Tennessee, 2012.

[14]A. Katz and V. Sankaran. An efficient correction method to obtain a formally third-order accurate flow solver for node-centered unstructured grids. *J. Sci. Comput.*, 51:375–393, 2012.

[15]Yi Liu and Hiroaki Nishikawa. Third-order inviscid and second-order hyperbolic Navier-Stokes solvers for three-dimensional inviscid and viscous flows. In *46th AIAA Fluid Dynamics Conference*, AIAA Paper 2016-3969, Washington, D.C., 2016.

[16]Hiroaki Nishikawa and Yi Liu. Accuracy-preserving source term quadrature for third-order edge-based discretization. *J. Comput. Phys.*, 344:595–622, 2017.

[17]Hiroaki Nishikawa. Accuracy-preserving boundary flux quadrature for finite-volume discretization on unstructured grids. *J. Comput. Phys.*, 281:518–555, 2015.

[18]Mohagna J. Pandya, Boris Diskin, James L. Thomas, and Neal T. Frink. Improved convergence and robustness of USM3D solutions on mixed element grids. *AIAA J.*, 54(9):2589–2610, September 2016.

[19]B. van Leer. Upwind-difference methods for aerodynamic problems governed by the Euler equations. In *Fluid Mechanics, Lectures in Applied Mathematics*, volume 22, pages 327–335, 1985.

[20]Clarence O. E. Burg. Higher order variable extrapolation for unstructured finite volume RANS flow solvers. AIAA Paper 2005-4999, 2005.

[21]Hiroaki Nishikawa. First, second, and third order finite-volume schemes for advection-diffusion. *J. Comput. Phys.*, 273:287–309, 2014.

[22]H. Nishikawa. Beyond interface gradient: A general principle for constructing diffusion schemes. In *Proc. of 40th AIAA Fluid Dynamics Conference and Exhibit*, AIAA Paper 2010-5093, Chicago, 2010.

[23]Katate Masatsuka. I do like CFD, VOL.1, Second Edition, version 2.6. `http://www.cfdbooks.com`, 2018.

Acknowledgments

This work has been funded by the U.S. Army Research Office under the contract/grant number W911NF-19-1-0429 with Dr. Matthew Munson as the program manager. The authors would like to thank Mike Park (NASA Langley Research Center) for helping us perform the grid adaptation with refine.