An Adaptive Explicit 3D Discontinuous Galerkin Solver for Unsteady Problems

Andrew C. Kirby * Dimitri J. Mavriplis †

Department of Mechanical Engineering, University of Wyoming, Laramie, WY 82071, USA

Andrew M. Wissink ‡

US Army Aviation Development Directorate - AFDD (AMRDEC), Moffett Field, CA 94035, USA

A block-structured Cartesian discontinuous Galerkin solver is developed for and implemented into the open source SAMRAI structured adaptive mesh refinement framework. The focus of the work is optimizing the computational efficiency of the block solver for the adaptive mesh refinement framework. A collocation scheme is implemented in a Cartesian setting for maximal efficiency. The discontinuous Galerkin solver is demonstrated in the SAMRAI framework and tested using an isolated vortex and comparing to an exact solution.

I. Introduction

The use of Cartesian grids in computational fluid dynamics (CFD) provides well known advantages in terms of computational efficiency and accuracy. Cartesian meshes represent the simplest grid structure which can be represented extremely compactly, thus minimizing solver memory footprints, optimizing parallel efficiency, and simplifying both adaptive mesh refinement implementations[1,2] and overset mesh search and interpolation tasks.[3,4] The principal drawback of Cartesian mesh approaches lies in the difficulties of dealing with non-simple geometries. Various approaches for dealing with complex geometries have been developed for use with Cartesian meshes including immersed boundary methods,[5,6,7] cut cell approaches,[8,9,10] and overlapping dual mesh paradigms where a body fitted mesh is used in near-body regions and a Cartesian mesh is used in off-body regions.[3,4,11]

The development of high-order accurate discretizations for computational aerodynamics has been pursued vigorously over the last decade and substantial advances have been demonstrated for continuous and discontinuous Galerkin (DG) methods.[12,13,14,15,16,17,18,19,20] However, the use of high-order methods for computational aerodynamics remains a research topic largely due to the poor robustness and large computational expense of these methods.[21] The approach taken in this work is to capitalize on the inherent advantages of Cartesian meshes to enable the development of an efficient and highly accurate discontinuous Galerkin solver specialized for Cartesian meshes. The basic approach consists of limiting the discretization to a collocation approach using a tensor-product basis formulation on hexahedral elements, which is well known to provide large gains in efficiency over the general element modal formulation.[22,23] Although numerous examples of high-order tensor-product implementations on structured hexahedral meshes have been described previously,[22,23,24] the current work seeks to obtain optimal solution accuracy and performance by further limiting the discretization to Cartesian meshes.

The use of a discontinuous Galerkin discretization offers advantages over traditional finite-difference or finite-volume discretizations in terms of accuracy per degree of freedom, which in turn enables the use of coarser grids, thus minimizing the overheads associated with managing the dynamic adaptive mesh refinement process. Additionally, the nearest neighbor stencil of the DG discretization simplifies the treatment of fringe data in transition regions between coarse and fine mesh blocks. Finally, using a variable order

*Doctoral Student, AIAA Student Member

†Professor, AIAA Associate Fellow

‡Aerospace Engineer, AIAA Member

implementation, we aim to enable combined h (mesh refinement) and p (order enrichment) adaptive methods, which have been shown to be optimal for error reduction.[25] Our approach for dealing with complex geometries is based on the dual mesh paradigm, as depicted in Figure 1, where an unstructured body fitted mesh is used in the near body regions and the Cartesian DG solver is used in off-body regions. Although this paper is restricted to the discussion of the off-body Cartesian mesh DG solver, previous work has shown this approach to be feasible for both low order and high-order discretizations.[26]

This paper begins by introducing the governing equations and their spatial discretization via the discontinuous Galerkin method. We build the block-solver framework and demonstrate validated results for three model problems: Ringleb flow, Taylor-Green vortex, and diagonally lid-driven cavity flow. Following the validation results, we present the block-solver in a self-contained parallel framework. The second half of the paper introduces the structured adaptive solver framework and presents results of two model problems: isentropic vortex convection and flow over a cube. The paper concludes with a summary of the advantages of moving to a SAMR setting and a discussion on future work.

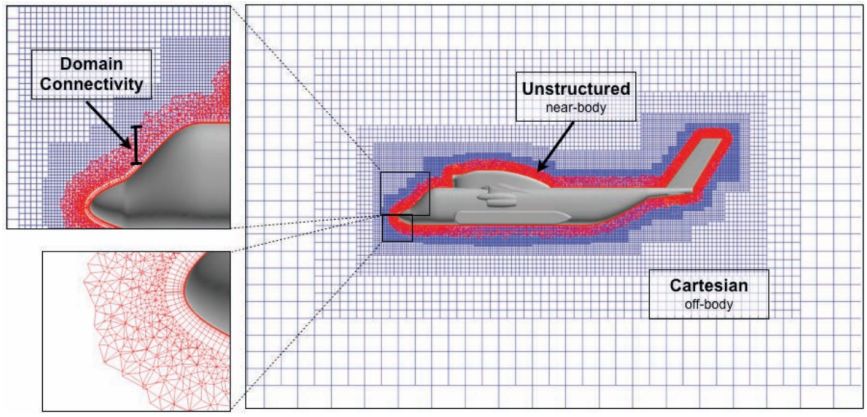


Figure 1. Dual-mesh, dual-solver paradigm in HELIOS.[3]

II. Governing Equations

The Navier-Stokes equations govern the dynamics of compressible fluids and can be written as:

$$\frac{\partial U_m}{\partial t} + \frac{\partial F_{mi}}{\partial x_i} = 0 \tag{1}$$

where they represent the conservation of mass, momentum, and energy. The solution vector U and flux F are defined as:

$$U = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{Bmatrix}, \quad F = \begin{Bmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + P - \tau_{11} & \rho uv - \tau_{12} & \rho uw - \tau_{13} \\ \rho uv - \tau_{21} & \rho v^2 + P - \tau_{22} & \rho vw - \tau_{23} \\ \rho uw - \tau_{31} & \rho vw - \tau_{32} & \rho w^2 + P - \tau_{33} \\ \rho uH - \tau_{1j}u_j + q_1 & \rho vH - \tau_{2j}u_j + q_2 & \rho wH - \tau_{3j}u_j + q_3 \end{Bmatrix} \tag{2}$$

where ρ is the density, u, v, w are the velocity components in each spatial coordinate direction, P is the pressure, E is total internal energy, $H = E + \frac{P}{\rho}$ is the total enthalpy, τ is the viscous stress tensor, and q is the heat flux. The viscosity is a function of the temperature given by the Sutherland's formula. These equations are closed using the ideal gas equation of state:

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

where $\gamma = 1.4$ is the ratio of specific heats. Hereinafter, Einstein notation is used where subscripts are not declared otherwise and the indices of m and n vary over the number of flow variables. Indices i and j have a range of 1 to 3.

III. Spatial Discretization

General finite-element methods scale per degree of freedom as $\mathcal{O}(N^d)$ where $N = p + 1$ is the order of accuracy, p is the polynomial degree, and d is the spatial dimension. Utilizing tensor-product basis functions and collocation of integration and interpolation points, the cost per degree of freedom is estimated as $\mathcal{O}(N)$ by reference.[23] This makes the cost comparable to compact finite difference methods. In this section, the DG finite-element method used to solve the Navier-Stokes equations is described.

The computational domain is partitioned into a block-structured Cartesian collection of hexahedra $\mathcal{T}_H$, of uniform element size H, into an ensemble of non-overlapping elements, such that $\Omega = \bigcup_{k \in \mathcal{T}_H} \Omega_k$, where Ω_k refers to the volume of an element k with $k \in \mathcal{T}_H$.

III.A. Discontinuous Galerkin Formulation

The discontinuous Galerkin discretization proceeds by formulating a weak statement of the governing equations. To obtain the weak formulation, multiply the equation by a set of test functions Ψ, with a maximum polynomial order of p. Note that the order of accuracy is $N = p+1$. Proceed by integrating over all elements k:

$$\int_{\Omega_k} \Psi \left(\frac{\partial U_m}{\partial t} + \frac{\partial F_{mi}}{\partial x_i} \right) \mathrm{d}\Omega_k = 0 \tag{3}$$

Integrating equation (3) by parts the total residual R_m is defined as:

$$R_m = \int_{\Omega_k} \Psi \frac{\partial U_m}{\partial t} \mathrm{d}\Omega \; - \int_{\Omega_k} \frac{\partial \Psi}{\partial x_i} F_{mi} \mathrm{d}\Omega_k + \int_{\Gamma} \Psi F^*_{mi} \mathrm{n}_i \mathrm{d}\Gamma = 0 \tag{4}$$

The residual now contains integrals over faces Γ and special treatment is needed for the fluxes in these terms. The advective fluxes F^* are calculated using Godunov's scheme or an approximate Riemann solver. Implemented schemes include Lax-Friedrichs[27] and Roe,[28] while the diffusive fluxes are handled using a symmetric interior penalty (SIP) method.[29,30] This work is focused on subsonic flows without the existence of shocks, thus the exact solution exists in C^∞. Therefore, we do not expect the order of convergence to be limited by solution irregularity.

III.B. Reference Element Mapping

We wish to transform physical coordinates, $\vec{X} = (x, y, z)^T$, to reference coordinates, $\vec{\xi} = \left(\xi^1, \xi^2, \xi^3\right)^T$. The reference element E is the set of points spanned by $\vec{\xi}$ where $\xi \in [-1, 1]$, i.e. $E = [-1, 1]^3$. Since our domain discretization is Cartesian, the transformation between the physical coordinates and the reference coordinates is a pure dilation. Thus the integral of a general function G in physical space transformed to the reference space is given as

$$\int_{\Omega_k} G(\vec{X}) \mathrm{d}\Omega_k = \int_z \int_y \int_x G\left(x, y, z\right) \mathrm{d}x\mathrm{d}y\mathrm{d}z = \int_{\xi^3} \int_{\xi^2} \int_{\xi^1} G\left(\xi^1, \xi^2, \xi^3\right) \frac{dx}{d\xi^1} \frac{dy}{d\xi^2} \frac{dz}{d\xi^3} \mathrm{d}\xi^1 \mathrm{d}\xi^2 \mathrm{d}\xi^3 = \int_E G(\vec{\xi}) J \mathrm{d}\vec{\xi} \tag{5}$$

where

$$J = \frac{dx}{d\xi^1} \frac{dy}{d\xi^2} \frac{dz}{d\xi^3} \tag{6}$$

Due to the prescribed mesh framework, J is a precomputed constant.

III.C. Solution and Flux Approximation

The solution of each element is approximated by $U_m(\vec{\xi}) = a_{ms} \Psi_s(\vec{\xi})$ with degrees of freedom (DOF) a_{ms}. The index s runs over the total number of basis functions. We develop the basis functions to be a tensor-product of one-dimensional Lagrange interpolating polynomials of degree N in each spatial direction:

$$\Psi_s = \Psi_{ijk}(\vec{\xi}) = l_i(\xi^1) l_j(\xi^2) l_k(\xi^3), \tag{7}$$

The one-dimensional Lagrange polynomials are given as:

$$l_j(\xi) = \prod_{i=0, i \neq j}^{N} \frac{(\xi - \xi_i)}{(\xi_j - \xi_i)}, \quad j = 0, \cdots, N \tag{8}$$

possessing the Lagrange property:

$$l_j(\xi_i) = \delta_{ij}, \quad i,j = 0, \cdots, N \tag{9}$$

for a collection of points ξ_i in the one-dimensional reference space $[-1,1]$. By choosing the tensor-product of one-dimensional Lagrange polynomials as the basis, we develop a nodal finite-element method provided by the property in equation (9). The algorithm benefits computationally since the degrees of freedom a_{ms} are the solution values $\tilde{U}_{ms}$ at the collection of points. The points ξ_i are chosen to be the Gauss-Legendre quadrature points. By choosing the Gauss-Legendre quadrature points for the interpolation points, the DOF are equal to the solution values. Therefore there is no need to perform solution projections to the integration points, hence increasing performance. Additionally, choosing the Gauss-Legendre points yields more accuracy and efficiency in the discretization.[31] Gauss-Legendre quadrature is can integrate all polynomials up to degree $2N-1$ exactly in comparison to Gauss-Lobatto quadrature which can integrate polynomials up to $2N-3$ exactly.[32] We make note that by changing from a general basis to a tensor-product basis, we can rewrite the index s using three indices uvw that traverse each coordinate direction, i.e. $\tilde{U}_{ms} = \tilde{U}_{muvw}$. The flux vector F_{mi} has three disjoint flux components, namely F_{m1}, F_{m2}, and F_{m3}. When interpolating these functions, we evaluate these non-linear functions discretely with the Gauss-Legendre points ξ_i chosen above resulting in $\tilde{F}_{m1}\Psi_s(\vec{\xi}), \tilde{F}_{m2}\Psi_s(\vec{\xi})$, and $\tilde{F}_{m3}\Psi_s(\vec{\xi})$. However, there is a drawback to evaluating these non-linear fluxes at the interpolation points. Since we have non-linear fluxes, this can introduce truncation errors to the orthogonal polynomial infinite series representation and incur aliasing to the higher-order terms.[33] To alleviate the problem of polynomial aliasing, it has been suggested for the compressible Navier-Stokes equations to use $2N$ points for numerical integration.[34,35] By over-integrating, we trade the computational efficiency of collocation for numerical stability and accuracy.

III.D. Time Derivative Integral

This subsection describes the integration of the first term of the residual in equation (4). First noting that the mesh discretization is time-independent, we may factor the time derivative out of the integral. Selecting the basis function to be the same as the test function by the Galerkin approach,$\Psi = \Psi_{ijk}$:

$$\frac{\partial}{\partial t}\int_E U_m \Psi J \mathrm{d}\vec{\xi} = \frac{\partial}{\partial t}\int_E \left(\tilde{U}_{muvw}\Psi_{uvw}\right)\Psi_{ijk} J \mathrm{d}\vec{\xi} \tag{10}$$

Replacing the continuous integration with numerical quadrature, further reducing by collocation and finally using the Lagrange property in equation (9):

$$\frac{\partial}{\partial t}\int_E U_m \Psi J \mathrm{d}\vec{\xi} = J w_i w_j w_k \frac{\partial U_{muvw}}{\partial t} = J w_i w_j w_k \frac{\partial U_{ms}}{\partial t} \tag{11}$$

where w is the quadrature weight. We now define the mass matrix $\mathbb{M}$ and the inverse mass matrix $\mathbb{M}^{-1}$ as:

$$\mathbb{M} = M_{ijk} = J w_i w_j w_k \quad , \quad \mathbb{M}^{-1} = \left(J w_i w_j w_k\right)^{-1} \tag{12}$$

Equation (11) simplified becomes:

$$\mathbb{M}\frac{\partial U}{\partial t} \tag{13}$$

Noting that our Jacobian J is linear and the mesh contains only linear hexahedra, Gauss quadrature using the Gauss-Legendre points integrates the time derivative term exactly.

III.E. Volume Integral

This subsection describes the integration of the second term of the residual in equation (4). The volume integral is a disjoint sum of three flux integrals:

$$\int_{\Omega_k} \frac{\partial \Psi}{\partial \xi_i} F_{mi} \mathrm{d}\Omega_k = \int_{\Omega_k} \frac{\partial \Psi}{\partial \xi_1} F_{m1} \mathrm{d}\Omega_k + \int_{\Omega_k} \frac{\partial \Psi}{\partial \xi_2} F_{m2} \mathrm{d}\Omega_k + \int_{\Omega_k} \frac{\partial \Psi}{\partial \xi_3} F_{m3} \mathrm{d}\Omega_k \tag{14}$$

To do the general analysis of the fluxes, we only need to do the analysis on one of the fluxes in equation (14) as the rest are similar. Setting the test function equal to the basis function and replacing the integral

by Gauss quadrature:

$$\int_{\Omega_k} \frac{\partial \Psi}{\partial \xi_1} F_{m1} \mathrm{d}\Omega_k = \int_E \tilde{F}_{m1} \Psi_s(\vec{\xi}) \frac{\partial \Psi}{\partial \xi_1} \mathrm{d}\vec{\xi} \tag{15}$$

$$= \tilde{F}_{uwv1} l_u(\xi^1_\lambda) l_w(\xi^2_\mu) l_v(\xi^3_\nu) \frac{\partial \Psi_{ijk}(\vec{\xi})}{\partial \xi_1} w_\lambda w_\mu w_\nu \tag{16}$$

Noting that $l_u(\xi^1_\lambda) = \delta_{u\lambda}$ and taking the derivative of the basis function with respect to one coordinate removes the Lagrange property in that respective coordinate direction. Thus the surviving terms for the ξ^1-direction derivative are $\frac{dl_i(\xi)}{d\xi}\Big|_{\xi=\xi^1}$:

$$= \tilde{F}_{uwv1} l_u(\xi^1_\lambda) l_w(\xi^2_\mu) l_v(\xi^3_\nu) \frac{\partial \Psi_{ijk}(\vec{\xi})}{\partial \xi_1} w_\lambda w_\mu w_\nu \tag{17}$$

$$= \tilde{F}_{\lambda\mu\nu 1} \left.\frac{dl_i(\xi)}{d\xi}\right|_{\xi=\xi^1} \delta_{j\mu}\delta_{k\nu} w_\lambda w_\mu w_\nu \tag{18}$$

$$= w_j w_k \tilde{F}_{\lambda jk1} \left.\frac{dl_i(\xi)}{d\xi}\right|_{\xi=\xi^1} w_\lambda \tag{19}$$

Equation (19) can be viewed as a matrix-vector product but taking only a single component per flux. Finally we rewrite the volume integral as:

$$\int_{\Omega_k} \frac{\partial \Psi}{\partial \xi_i} F_{mi} \mathrm{d}\Omega_k = \left(w_j w_k \tilde{F}_{\lambda jk1} \left.\frac{dl_\lambda(\xi)}{d\xi}\right|_{\xi=\xi_i} w_\lambda \right) \tag{20}$$

$$+ \left(w_i w_k \tilde{F}_{i\mu k2} \left.\frac{dl_\mu(\xi)}{d\xi}\right|_{\xi=\xi_j} w_\mu \right) \tag{21}$$

$$+ \left(w_i w_j \tilde{F}_{ij\nu 3} \left.\frac{dl_\nu(\xi)}{d\xi}\right|_{\xi=\xi_k} w_\nu \right) \tag{22}$$

III.F. Surface Integral

The surface integral is drastically simplified in the block-structured Cartesian framework. This occurs since the outward-facing unit normal for the ξ^1-direction is $n_1 = (\pm 1, 0, 0)^T$ with analogous expression for the other reference coordinate directions. We are able to eliminate the other components of the numerical flux F^*. For example, the surface integral at $\xi^1 = 1$,

$$\int_\Gamma \Psi F^*_{mi} \mathrm{n}_i \mathrm{d}\Gamma = \int_{-1}^{1}\int_{-1}^{1} \Psi_{\hat{s}1}(\xi^1 = 1) \xi F^*_{jk1} \frac{dy}{\xi^2}\frac{dz}{\xi^3} \mathrm{d}\xi^2 \mathrm{d}\xi^3 \tag{23}$$

Evaluating the Lagrange polynomials and numerically integrating, we get:

$$F^*_{jk1} l_i(1) w_j w_k \frac{dy}{\xi^2}\frac{dz}{\xi^3} \tag{24}$$

The other fluxes are analogous to equation (24) while paying particular attention to the outward-facing unit normal. In order to evaluate the surface integrals, we need to project the solution to the faces. To project to the surface points, we do a simple one-dimensional scalar product of the solution coefficients and the basis functions evaluated at the Gauss-Legendre points. For example, to project to the $\xi^1 = 1$ surface, $U_{jk} = U_{ijk} l_i(\xi = 1)$. Fig.2 shows the locations of Gauss-Legendre quadrature points in addition to boundary flux point locations. There is an imagined one-dimensional line of quadrature points for the desired surface point.

Additional benefits are provided in the Cartesian framework by recalling that the outward-facing unit normals are single component only varying in sign, thus eliminating the other directional components of the flux. Specialized flux routines are implemented in each coordinate direction to optimize the total number of operations.

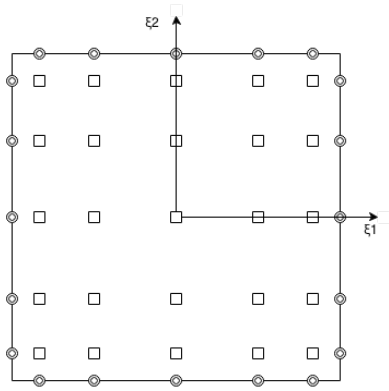


Figure 2. Gauss-Legendre quadrature points $\square$ for $N = 5$ and boundary quadrature points.

IV. Temporal Discretization

From the previous section on numerical discretization, we can combine the volume integral and the surface integral into a spatial residual, $R_X(U)$. To evolve the equations temporally, we use an explicit time-stepping scheme. Via the method of lines, the semi-discrete equation of equation (3) is given as:

$$\mathbb{M}\frac{\partial U}{\partial t} + R_X(U) = 0 \tag{25}$$

where $\mathbb{M}$ is the mass matrix defined previously. Equation (25) is a system of coupled ODEs which is solved numerically using an explicit time-stepping scheme. Current explicit time-schemes that are implemented are Forward-Euler and the classical Runge-Kutta four-stage method (RK4).[36,37] In this formulation, at each stage of the explicit scheme, the mass matrix must be inverted. For the collocation DG formulation, we demonstrated the mass matrix is block-diagonal allowing for easy inversion. We have precomputed the mass matrix and embedded the components amongst the basis functions while making all possible cancellations. Additionally, explicit time-marching methods have a maximum stable time-step restricted by the CFL condition. We compute the time-step under the assumption that the CFL number is constant and is equal to one. With ν as the kinematic viscosity and constants $C_1,\ C_2,\ C_3$ which are found heuristically, we compute the time-step as:[38]

$$dt = \min\left(C_1\frac{h}{(|U|+c)}, C_2\frac{h^2}{\nu}, C_3\frac{h}{\sigma\nu}\right) \tag{26}$$

$$h = \frac{\min(dx, dy, dz)}{2(p+1)^{1.8}} \tag{27}$$

$$\sigma = \frac{(p+1)^2}{\min(dx, dy, dz)} \tag{28}$$

V. Block Solver Results

In this section, we validate the accuracy of the developed DG solver and examine computational performance. Three model problems are used to demonstrate these properties. A mesh resolution study is performed using the Ringleb flow problem. The second problem is the Taylor-Green vortex.[39,40] The third problem is a three-dimensional, diagonally lid-driven cavity flow. This problem is relatively new in terms of three-dimensional code validation and was introduced in references.[41,42,43,44,45]

V.A. Mesh Resolution Study: Ringleb Flow

We use an exact solution to verify the accuracy of the finite-element formulation. Ringleb flow is an exact solution to the steady state Euler equations and is solved analytically using the hodograph method.[46] Although it is only a two-dimensional solution, the three-dimensional equations can be verified by setting

the momentum in the third dimension to zero. Characteristic boundary conditions are used in the x and y directions and an inviscid wall boundary condition is used in the z-direction. The domain is a $[1, 1]$ square discretized with a hexahedral mesh. Various meshes ranging from 1 x 1 x 1 to 40 x 40 x 1 are used for different solution orders. The flow is initialized using the exact solution and then the residual is driven to machine precision.

The errors in a finite-element formulation should decrease asymptotically following the power law Ch^{p+1}, where C is a constant, h is the mesh size, and p is the polynomial degree. By comparing the analytic solution to the numerical solution the error can be measured. The error is measured using an L_∞ norm for $p = 1, 2, 4, 6, 9$ at all mesh resolutions. The L_∞ error versus mesh size h is plotted in Fig.3. We see from the plot that we get the desired design accuracy noting that at $p = 9$, the L_∞ error is near machine precision thus giving a slight degradation in the slope of the error.

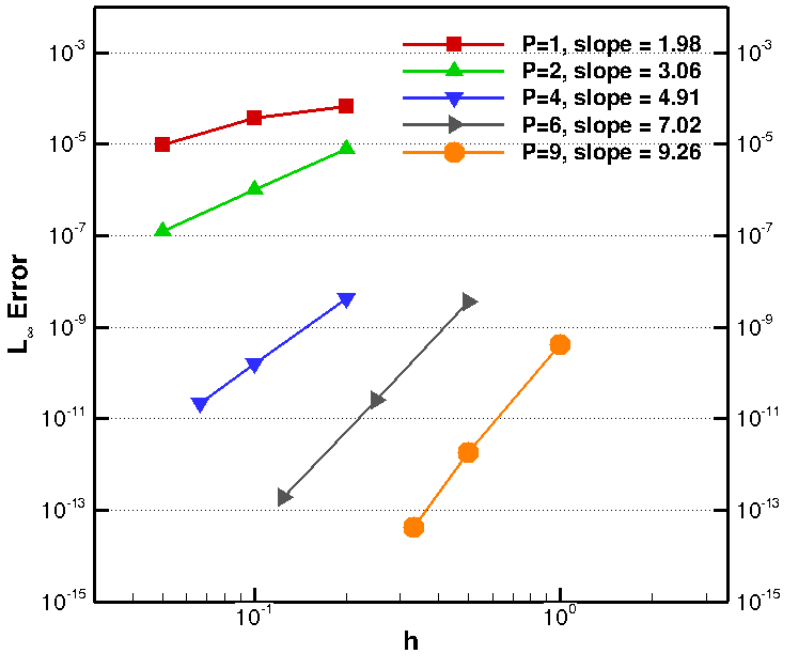


Figure 3. Ringleb flow: L_∞ error vs. mesh size.

V.B. Taylor-Green Vortex

The Taylor-Green vortex flow is simulated using the compressible Navier-Stokes equations at $M_0 = 0.1$. The flow is solved on an isotropic domain which spans $[-\pi L, \pi L]$ in each coordinate direction where L is the characteristic length. The initial conditions are given by

$$\begin{aligned} u &= V_0 \sin(x/L)\cos(y/L)\cos(z/L) \\ v &= -V_0 \cos(x/L)\sin(y/L)\cos(z/L) \\ w &= 0 \\ p &= \rho_0 V_0^2 \left[\frac{1}{\gamma M_0^2} + \frac{1}{16}\left(\cos(2x) + \sin(2y)\right)\left(\cos(2z) + 2\right)\right] \end{aligned}$$

where u, v and w are the components of the velocity in the x-, y- and z-directions, p is the pressure and ρ is the density. The flow is initialized to be isothermal $\left(\frac{p}{\rho} = \frac{p_0}{\rho_0} = RT_0\right)$. The initial flow is laminar and subsequently transitions to turbulence, with the creation of small scales, followed by a decay phase similar to decaying homogeneous turbulence. The domain contains no boundary conditions as the problem is fully

periodic. Fig.4 shows the temporal evolution of the kinetic energy dissipation rate $-\frac{dE_k}{dt}$ for the Taylor-Green vortex problem computed at $M_0 = 0.1$, $Pr = 0.71$, $Re = \frac{\rho_0 V_0 L}{\mu} = 1600$, and 256 degrees of freedom in each coordinate direction with comparison to an incompressible spectral code with 512 degrees of freedom in each coordinate direction.[47] The temporal evolution of the kinetic energy integrated over the domain Ω is calculated as:

$$E_k = \frac{1}{\rho_0 \Omega} \int_\Omega \rho \frac{\vec{v} \cdot \vec{v}}{2} d\Omega \tag{29}$$

As the order of accuracy increases, the discontinuous Galerkin solver dissipation solution tends towards the incompressible spectral code. The DG scheme becomes less dissipative with higher-order basis functions resulting in the solver's ability to more accurately match the spectral solver results. Reference[22] demonstrates similar dissipation curves with an unstructured version of the current approach.

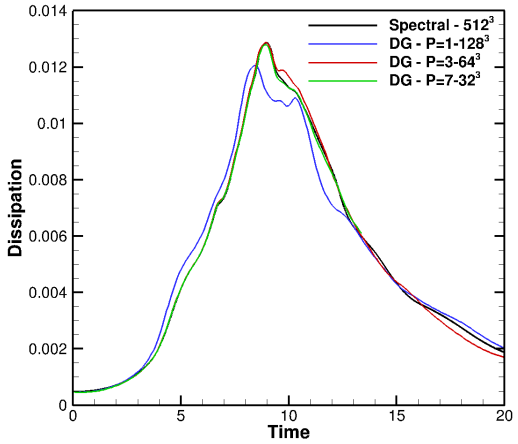


Figure 4. Taylor-Green vortex problem at $M = 0.1$, $Re = 1600$ computed using 256^3 degrees of freedom.

For further solver validation, a comparison to a three-dimensional mixed-element DG solver with shock capturing[18] was conducted using the Taylor-Green vortex problem. Results are demonstrated in Fig.5 for $p = 4$ and a 64^3 mesh. Kinetic energy dissipation, enstrophy, and the pressure dilation contribution to kinetic energy dissipation are plotted in Fig.5, respectively. Enstrophy is computed as:

$$\epsilon = \frac{1}{\rho_0 \Omega} \int_\Omega \rho \frac{\vec{\omega} \cdot \vec{\omega}}{2} d\Omega \tag{30}$$

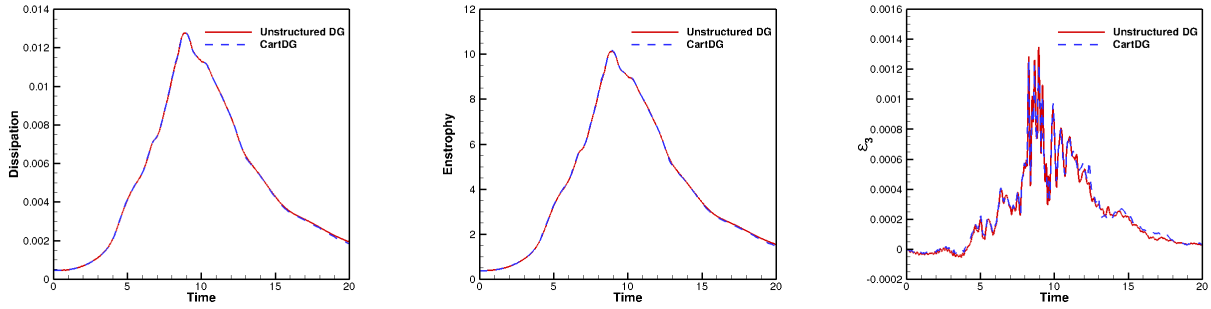
The pressure dilation contribution term ϵ_3 is computed as:

$$\epsilon_3 = -\frac{1}{\rho_0 \Omega} \int_\Omega p \nabla \cdot \vec{v} \; d\Omega \tag{31}$$

As seen from Fig.5, the DG solver developed in this paper gives nearly identical solution curves for kinetic energy dissipation and enstrophy as the mixed-element, shock-capturing, unstructured discontinuous Galerkin solver of Reference.[18] The pressure dilation term ϵ_3 curve for CartDG is slightly different than the unstructured code. Overall, the trends in the ϵ_3 plots are very similar and can be attributed to both codes solving the compressible Navier-Stokes equations and using a symmetric interior penalty method for the viscous flux calculations.

V.C. Diagonally Lid-Driven Cavity

Recent efforts for an extension to the standard two-dimensional lid-driven cavity benchmark[48] have been proposed by Povitsky[49,41,42] and Feldman et. al.[43] The extension is a steady flow driven cavity with the lid moving at 45° to the cube wall. The $x-$ and $z-$ velocities have equal magnitudes. For this study, we use a Reynolds number of 1000. Fig.6 depicts the lid-driven flow at a diagonal. The physical domain is $[0, 1]^3$.



(a) kinetic energy dissipation (b) enstrophy (c) ϵ_3

Figure 5. Taylor-Green Vortex comparison between an unstructured DG solver and CartDG: $p = 4, 64^3$ mesh at $M = 0.1$, $Re = 1600$.

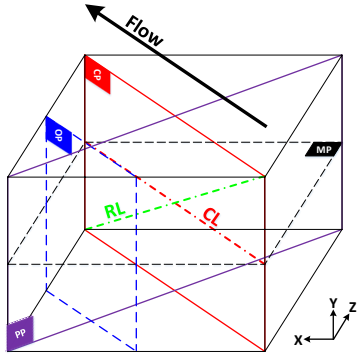


Figure 6. Cubic lid-driven cavity with lid moving at 45° to the x-axis, $Re = 1000$.

We present results for this test case in Table 1 using Feldman et. al.[43] as a comparison reference. We note that the solver used in Feldman et. al. is a second-order conservative finite volume scheme with a full pressure-velocity coupling that solves the incompressible Navier-Stokes equations.[48,45] Their calculations were performed on 152^3 and 200^3 grids, though we use the latter results for comparison. Our computation was performed at $p = 3$ using a grid of 32^3 mesh. The total degrees of freedom in the case of the present calculation is just over two million where as the comparative results used eight million DOF. We consider the flow to be converged when the L_2 norm of the spatial residual is less than $1E - 12$. The results in Table 1 demonstrate that the V_x and V_z velocities towards the boundaries of the computational domain are similar in magnitude to the comparison reference. In the middle of the domain near $(0.5, 0.5, 0.5)$, the V_x and V_z velocities vary the most compared to the rest of the domain. The V_y velocities of the DG solver and the incompressible finite volume solver agree more on the lower half of the domain in comparison to the upper half of the domain. Fig.7 shows the streamlines in the center plane (cp). In Fig.8, visualization of the streamlines of the converged flow observed in the center plane direction and parallel plane (pp) direction are presented. In Fig.9, visualization of the contour of velocity magnitude is presented.

V.D. Computational Performance and Parallel Scalability

In this section we examine the computational performance and parallel scalability of the discontinuous Galerkin solver. The numerical discretization allowed us to make several cost-saving simplifications. Designing a nodal collocation method by choosing the interpolation points to be the same as the integration points saves two projection calculations. Additionally, the tensor-product basis converts costly projections and integrations to be one-dimensional products therefore boosting the performance significantly. Combining these properties with the block-structured Cartesian framework makes the DG implementation competitive with finite-difference discretizations.

$Re = 1000$

y	$V_x, V_z \cdot 10^3$		$V_y \cdot 10^3$	
	Reference[43]	**Present**	**Reference[43]**	**Present**
1.0	707.1	707.1	0.0	0.0
0.9766	417.8	418.2	5.357	9.735
0.9688	341.3	339.5	8.774	9.804
0.9609	277.2	276.7	12.47	9.622
0.9531	226.6	227.9	16.03	13.90
0.8516	76.82	76.06	30.13	29.99
0.7344	62.56	61.74	22.28	21.95
0.6172	41.77	40.68	5.439	4.832
0.5	-1.649	-4.218	-34.41	-36.76
0.4531	-31.93	-35.25	-65.23	-68.61
0.2813	-131.0	-130.7	-160.5	-160.1
0.1719	-134.7	-133.0	-138.0	-136.2
0.1016	-143.0	-141.8	-86.68	-85.15
0.0703	-158.8	-157.6	-52.73	-51.60
0.0625	-161.9	-160.2	-43.93	-42.98
0.0547	-162.2	-160.4	-35.40	-34.60
0.0	0.0	0.0	0.0	0.0

Table 1. Velocity components along the vertical center line $(0.5, y, 0.5)$ for diagonally lid-driven cavity flow: comparison between the reference[43](200^3 grid) and the present ($p = 3, 32^3$ grid) solution.

The CPU time for a single residual evaluation normalized by the number of degrees of freedom for a viscous flow is shown in Fig.10 for two different CPU architectures. The CPU times are obtained by performing a simulation of the Taylor-Green vortex problem. The first results in Fig.10(a) are computed in parallel on the NCAR-Wyoming Supercomputer (NWSC) Yellowstone[50] using 1,024 cores on Intel Xeon E5-2670 processors with a clock speed of 2.6Ghz and 2GB per core memory. For reference, the TAU benchmark for this architecture runs in 8.4s.[50] The second timing results presented in Fig.10(b) are computed in parallel using eight cores on an Intel Core i7-5960X Haswell-E processor with a clock speed of 3GHz and 4GB per core memory. The results are obtained using a GNU compiled version of the code. Using the GNU compiler, the TAU benchmark for this architecture is 5.25s. Hindenlang et. al. report a computational time per DOF for a single residual evaluation is $2.0E-06$ seconds for their DG spectral-element solver at sixth-order compared to their in-house compact finite-difference solver at $4.0E-06$ seconds.[23] Reference[22] report their similar unstructured DG solver averaging $7.5E-07$ seconds per DOF for a single residual evaluation which is comparable to the cost for a residual evaluation using the OVERFLOW finite-difference code for a Navier-Stokes simulation. Reference[22] also report they were able to obtain a computationally cost independent of order of accuracy up to 16th order through optimized computational kernels and carefully alignment of data and unit-strided memory operations. Current implementations of CartDG achieve a similar computational cost to Reference[22] at lower orders of accuracy although further optimizations are needed in order to achieve the computational cost independence of order of accuracy achieved by Reference.[22]

Aside from the encouraging computational efficiency of the DG method, its main advantages are based on its parallel scalability. Distributed memory parallelism is realized through the MPI application program interface (API). The DG algorithm is inherently parallel, since all elements communicate only with their direct neighbors. Only surface data is needed by neighboring elements thus minimizing the data transfer. This is an advantage over finite-difference methods, which couple distant neighboring elements and thus grow the fringe area.

In addition to the finite-element method's straight-forward parallelization, the block-structured Cartesian grid framework allows for nearly optimal load balancing by requiring all domain decomposition blocks to be identical. This method also takes advantage of the Cartesian virtual topology framework in MPI simplifying

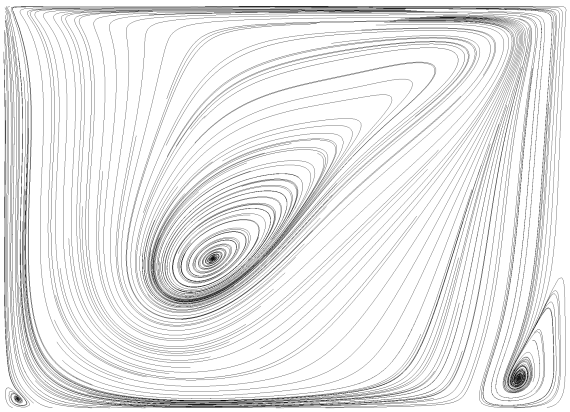
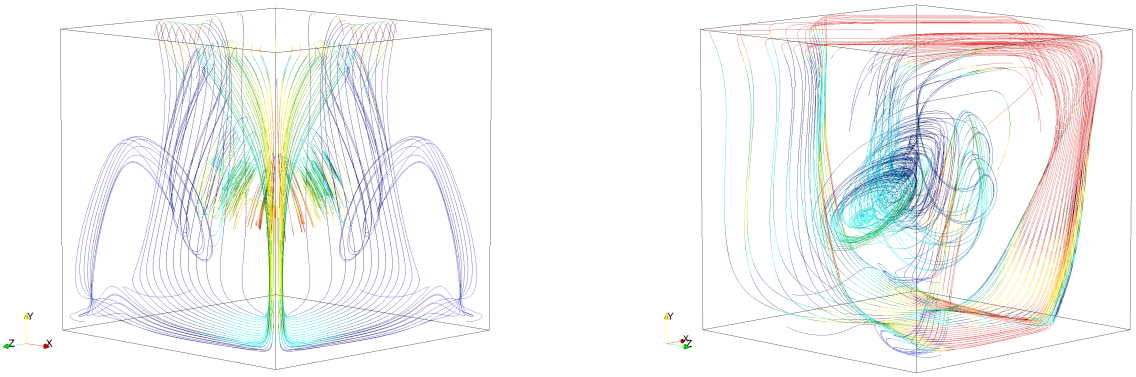


Figure 7. Center plane (cp) streamlines in the direction of the flow



(a) flow towards observer

(b) flow left to right

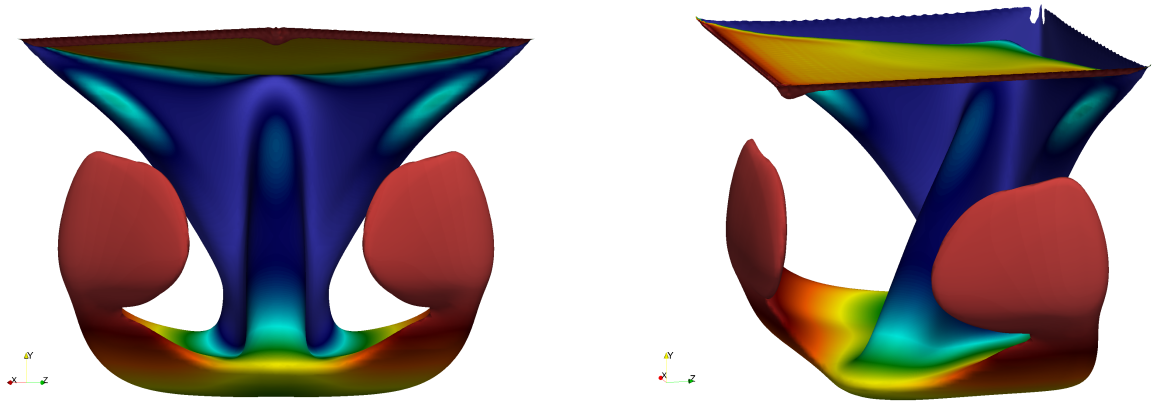
Figure 8. Cubic lid-driven cavity flow: streamlines colored by velocity magnitude.

the implementation. The DG algorithm can be split into the two calculations: the volume integral and the surface integral. The volume integral calculation only depends on element local DOF whereas the surface integral calculation requires neighboring element data. This fact can help to hide communication latency by exploiting local element operations and will further reduce the negative influence of data transfer on efficiency. It is therefore possible to send surface data while simultaneously performing volume data operations.

Parallel scalability results were obtained on Yellowstone.[50] Yellowstone is composed of 72,576 processor cores. The cores are Intel Xeon E5-2670 processors with a clock speed of 2.6Ghz and 2GB per core memory with Advanced Vector Extensions (AVX). The strong scalability results are obtained by performing a simulation of the Taylor-Green vortex problem using a 128 x 128 x 128 mesh. This mesh equates to approximately 260 million DOF, 1 billion DOF, and 2 billion DOF for $p = 4$, 7, and 9, respectively. Strong scaling results up to 32,768 processors for polynomial degrees of 4, 7, and 9 are shown in Fig.11. The scaling improves as the order of accuracy increases. At 32,768 processors, the problem contained 8,000, 32,768, and 64,000 degrees of freedom on each processor for $p = 4, 7$, and 9, respectively.

VI. Structured Adaptive Solver

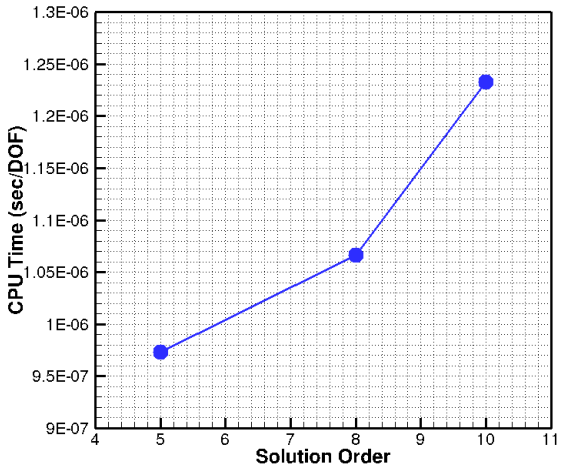
The overall goal of this work is to capture shed wakes from wind turbines and rotary-wing vehicles in a multi-solver paradigm. In able to achieve high-order accuracy without using excessive resources, we have implemented the block-solver CartDG previously discussed into a structured adaptive mesh framework (SAMR). By working in an adaptive mesh refinement framework, we can focus resolution to areas of increased



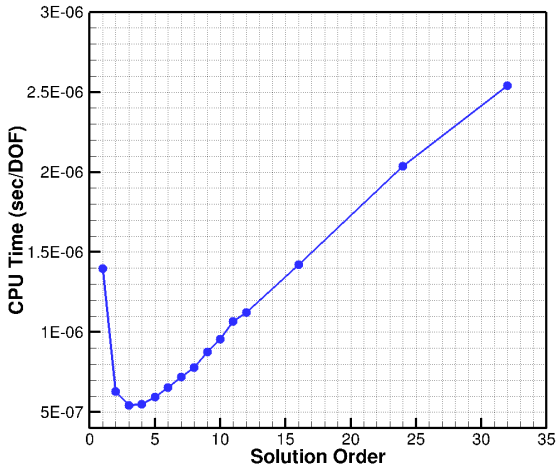
(a) flow away from observer

(b) flow left to right

Figure 9. Cubic lid-driven cavity flow: velocity magnitude contour of 0.005 colored by y-velocity.



(a) CartDG performed on 1,024 cores on the NCAR-Wyoming Supercomputer:[50] TAU Benchmark 8.4s with Intel compiler



(b) CartDG performed on 8 cores on Intel i7-5960X: TAU Benchmark 5.25s with GNU compiler

Figure 10. CPU time for a single residual evaluation per degree of freedom.

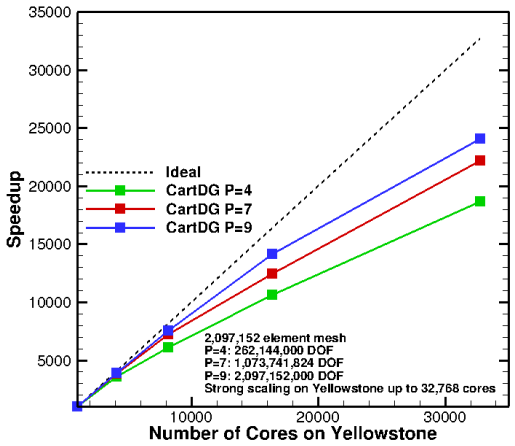


Figure 11. Strong scaling for polynomial degrees 4, 7, and 9 on the NCAR-Wyoming Supercomputer.[50]

activity, particularly wakes with high vorticity. Particularly working with a SAMR framework, we can adapt the grid dynamically in a time-dependent manner to resolve unsteady effects with relative ease. In a parallel compute setting, every time we adapt the mesh, the grid needs to be re-partitioned for load balancing and parallel communications must be redirected. AMR is simpler in a Cartesian setting giving it an advantage over unstructured methods in terms of efficiency. The SAMR framework has significant advantages in frequent, dynamic adaption because of the concise block-structure descriptions allowing for quick re-gridding procedures.[51] This section discusses the details of the structured adaptive mesh framework and the interpolation operators needed for transferring data between levels. Previous work in SAMR discontinuous Galerkin methods using an octree structure have done by Kopera and Giraldo.[52] Our SAMR approach is patch-based.

VI.A. Structured Adaptive Mesh Refinement Framework

The adaptive solver is built using the open source SAMRAI package from Lawrence Livermore National Lab. SAMRAI is an objected-oriented C++ library that provides flexible adaptive meshing and data management for parallel SAMR applications.[53] The code that couples SAMRAI and CartDG is referred to as SAMCart-DG. It is comprised of a C++ driver with the block-solver CartDG written in Fortran90. SAMRAI uses a block-structured grid hierarchy of nested refinement levels. Each level is composed of a union of logically-rectangular grid regions. On each level the grid spacing is fixed and the ratio of refinement between two consecutive levels is generally two, although the environment allows for other refinement ratios pending the construction of appropriate refinement and coarsening operators. The Cartesian grid levels are constructed from coarsest to finest by first tagging cells for refinement, clustering them into blocks, and refining these blocks into a new level. SAMRAI performs all grid generation, load balancing, and parallel communications between blocks. Each level is partitioned into a set of disjoint patches.

The governing equations for each patch on all levels are solved. Each level performs an explicit Runge-Kutta (RK) stage with a uniform time step. The time step is governed by the finest mesh spacing. At the beginning of a RK stage, data on the fine patch boundaries are updated by copying data from a neighboring patch if the resolution is the same or data is interpolated from a coarser level. Then the RK stage is performed independently on each patch. Next, data from the finer levels are injected into the coarser levels where finer levels overlap coarser levels via a coarsening operator. All data communications between levels and inter-level patches are done in parallel using MPI which is managed by SAMRAI. After a complete time step and regridding is requested, cells are tagged for refinement using a feature detection criteria implemented by the user. Tagged cells are covered by a disjoint union of boxes which form a new refined mesh level.[54] Our current tagging algorithm is based on vorticity magnitude.

VI.B. Refinement Operator

In a hierarchical, patch-based SAMR scheme, data from coarser levels need to be interpolated to finer levels. Two occurrences require interpolated data from coarse levels to fine levels: coarse-fine boundaries and coarse cells are that overlapped from newly formed fined cells. In the case of coarse-fine boundaries, ghost cells need to be provided on the fine level to do the flux calculation. Alternatively when a coarse level is refined, newly refined cells need to have their data interpolated from the underlying coarse cells. To interpolate data from a coarse cell to a fine cell, an interpolation mechanism called a refinement operator is needed.

Interior quadrature points (as in Fig.2) are located in each shaded cell in Fig.12. Since each of the refined cells is properly nested in a respective coarse cell, our refine operator is a Galerkin projection to the respective mapped subregion of the coarse element. For example in one-dimension, we let the native coarse coordinates be represented by $\xi_{\text{coarse}} = [-1, 1]$ as seen in Fig.13. Each of the native fine cells that cover the coarse cell have local reference coordinates $\xi_{\text{fine}} = [-1, 1]$. We can map the local reference fine cell coordinates to the coarse cell coordinates $\xi_1 = [-1, 0]$ and $\xi_2 = [0, 1]$ for the overlapping left and right fine cells, respectively. That is, we can observe the local reference coordinates for the fine cells to be a subset of the coarse reference coordinates. To get the left fine solution q^L in Fig.13 from the coarse solution coefficients U_j, a coordinate transformation $\xi_{\text{fine}} = [-1, 1] : \rightarrow \xi_1 = [-1, 0]$ is used then giving the projection P_1:

$$q^L\left(\xi_{fine}\right) = \sum_{j=0}^{N} \Psi_j \left(\frac{1}{2}\xi_{\text{fine}} - \frac{1}{2}\right) U_j \tag{32}$$

where Ψ_j are the coarse cell basis functions. Similarly, to get the right fine solution q^R, a coordinate transformation $\xi_{\text{fine}} = [-1, 1] : \rightarrow \xi_2 = [0, 1]$ is used then giving the projection P_2:

$$q^R\left(\xi_{fine}\right) = \sum_{j=0}^{N} \Psi_j \left(\frac{1}{2}\xi_{\text{fine}} + \frac{1}{2}\right) U_j \tag{33}$$

In three dimensions, we transform each coordinate to the respective coarse subset coordinates, ξ_1 or ξ_2, then perform the Galerkin projection using the basis evaluated at the transformed coordinates.

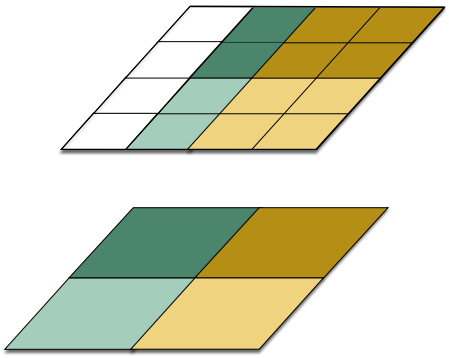


Figure 12. Coarse-fine boundary interface: fine green cells are ghost cells that are interpolated from coarse green cells.

VI.C. Coarsening Operator

Conversely to the refine operator, a interpolation mechanism is needed to transfer data on fine levels to coarse levels. This mechanism is called a coarsening operator. Coarse cells that have fine cells overlaying them are called non-valid cells in contrast to exposed cells which are called valid cells. The coarsening operator is performed for all non-valid cells after every Runge-Kutta stage to provide more accurate solution data from overlapping fine cells. In order to maintain discrete conservation of the conserved variables, we employ a mass matrix Galerkin projection approach for the coarsening operator. We need to gather two fine element solutions and form a coarse element solution as shown in Fig.14 for the one-dimensional example. To solve

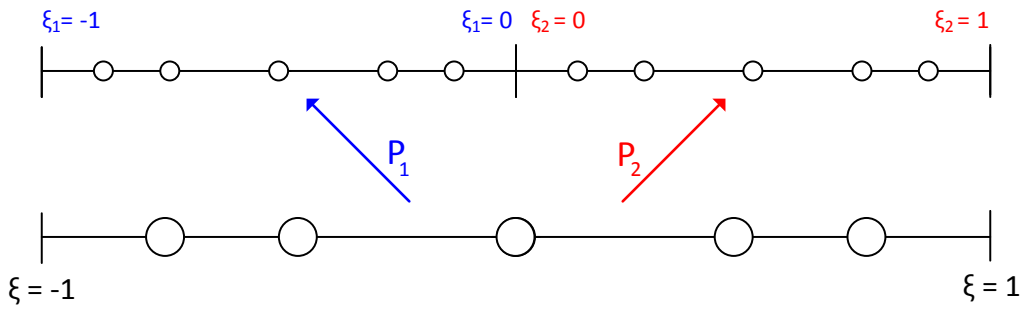


Figure 13. One-dimensional refine operator via Galerkin projection.

for the coarse solution U_j, the mass matrix Galerkin projection starts as:

$$\sum_{j=1}^{N} \Psi_j(\xi) U_j = \vec{q}(\xi) \tag{34}$$

where $\vec{q}$ is the composite vector of fine solution coefficients in the two overlapping fine level cells. Next, we multiply both sides of equation (34) with test functions and integrate over the standard element noting that the right hand side can be split across the fine elements:

$$\sum_{j=1}^{N} \int_{-1}^{1} \Psi_i(\xi) \Psi_j(\xi) U_j \, \mathrm{d}\xi = \int_{-1}^{0} \Psi_i(\xi) q_1 \, \mathrm{d}\xi + \int_{0}^{1} \Psi_i(\xi) q_2 \, \mathrm{d}\xi \quad \text{for} \quad i = 1, \cdots, N \tag{35}$$

where q_1 is the left fine solution and q_2 is the right fine solution. The elemental mass matrix is:

$$M_{ij} = \int_{-1}^{1} \Psi_i \Psi_j \mathrm{d}\xi \tag{36}$$

Lastly, we note that using nodal basis functions, $q_1 = \Psi^T(2\xi + 1)$ and $q_2 = \Psi^T(2\xi - 1)$ and invert the mass matrix to form a matrix G:

$$G = M^{-1} \cdot \left[\int_{-1}^{0} \Psi(\xi) \Psi^T(2\xi + 1) \, \mathrm{d}\xi, \int_{0}^{1} \Psi(\xi) \Psi^T(2\xi - 1) \, \mathrm{d}\xi \right] \tag{37}$$

We then obtain the coarse solution U from the fine solutions q_1 and q_2:

$$U = G \cdot [q_1; q_2] \tag{38}$$

Thus the resulting coarse solution is obtained by a matrix-vector product of the matrix G and the composite vector consisting of the respective fine element solutions that overlap the coarse solution.

VII. Structured Adaptive Solver Results

Two model problems are used to demonstrate the SAMR framework of SAMCart-DG. The first case is the inviscid convection of a two-dimensional isentropic vortex. This problem demonstrates the ability of the SAMR framework to track features in a flow while maintaining the same computational error as a completely refined, fixed Cartesian grid while utilizing a fraction of the total number of elements. The second case is an illustrative example of flow over a cube. This problem highlights the SAMR solver's ability to maintain flow characteristics by resolving unsteady wake regions while maintaining low element counts in comparison to a fully refined fixed grid.

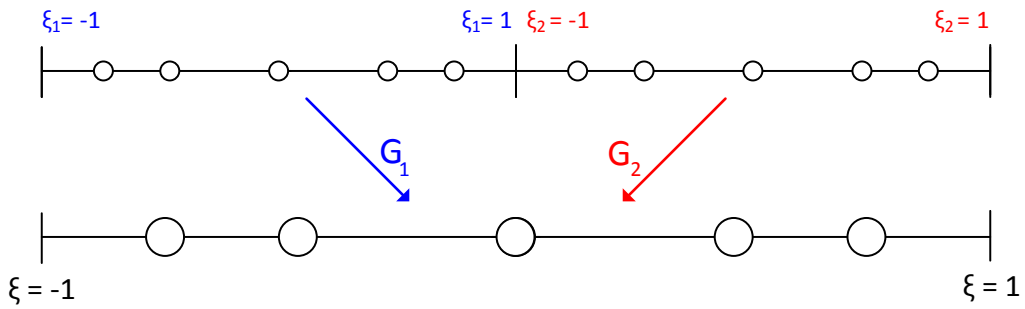


Figure 14. One-dimensional coarsen operator via mass matrix Galerkin projection.

VII.A. Convection of an Isentropic Vortex

The inviscid isentropic vortex is initialized with a perturbation of the uniform flow described as:

$$\delta u = -U_0 \beta \frac{y - y_c}{R} e^{0.5\left(1-r^2\right)} \tag{39}$$

$$\delta v = V_0 \beta \frac{x - x_c}{R} e^{0.5\left(1-r^2\right)} \tag{40}$$

$$T = -\frac{\left(\gamma - 1\right)\beta^2}{16\gamma\pi^2} e^{2\left(1-r^2\right)} \tag{41}$$

$$\rho = \left(1 + T\right)^{\frac{1}{1-\gamma}} \tag{42}$$

$$p = p_0 + \rho^\gamma - 1 \tag{43}$$

$$\tag{44}$$

where U_0, V_0 is the convecting speed of the vortex in the $x-$ and $y-$directions, $\beta = 1$ is the vortex strength, $R = 2\pi$ is the characteristic radius, x_c, y_c are the vortex center coordinates, and p_0 is the free-stream pressure. The exact solution of this flow is a non-dissipating convection of the vortex at the speed U_0, V_0. The following results use a free-stream Mach number of 0.5 with the vortex traveling in the $x-$direction. The fixed grid uses a 256 x 128 x 2 mesh. The SAMR meshes are set to have the same maximum resolution as the fixed mesh. Elements are selected for refinement when the magnitude of vorticity at any quadrature point located in the element is larger than $10^{-7}s^{-1}$. Fig.15 shows the fixed grid and the final adapted grids using five levels of refinement for $p = 2$ and $p = 4$ with the refinement ratio set to $(2, 2, 1)$ meaning when an element is selected for refinement, the x- and y-directions are refined by a factor two whereas the z-direction has a fixed resolution. To demonstrate the SAMR's ability to track the vortex while maintaining accuracy, we have simulated three solution orders: third-, fourth-, and fifth-order, using a maximum number of refinement levels of two, three, and five. In Fig.16, we show trends for solution time versus RMS error and solution time versus relative percent error. The RMS error is computed using the analytic solution at the final solution time of the simulation and is calculated as:

$$\mathrm{RMS}\left(U\right) = \sqrt{\frac{\int_\Omega \left(U - U_{analytic}\right)^2 \mathrm{d}\Omega}{|\Omega|}} \tag{45}$$

where Ω is the valid region domain and $|\Omega|$ is the valid region area. The relative percent error is computed using the RMS error of the solution obtained on the fully refined fixed grid, U_{fixed} as the reference solution which is computed as:

$$\text{Relative Percent Error}\left(U\right) = \frac{\left|\mathrm{RMS}\left(U\right) - \mathrm{RMS}\left(U_{\mathrm{fixed}}\right)\right|}{\mathrm{RMS}\left(U_{\mathrm{fixed}}\right)} \cdot 100\% \tag{46}$$

It is demonstrated that the SAMR can maintain the error relative to that of the fixed grid to within 0.0004%, 0.025%, and 0.08% relative error for $p = 2, 3$, and 4, respectively. We note that the fixed grid RMS error drops by a factor of 10 when increasing the order of accuracy by one, making the relative error more sensitive for higher order of accuracies. We demonstrate the cumulative space-time element counts of each of the isentropic vortex simulations in Fig.17. We sum all elements on all levels at each iteration. For higher orders of accuracy, a smaller time step is required thus increasing the total number of time steps and total number of space-time elements for the entire simulation. For fifth-order using five levels of refinement, we observe a 15x element reduction over the fixed grid for this two-dimensional problem. The SAMR significantly reduces the total number of space-time elements used during a simulation and maintains virtually the same error as the fixed mesh solution compared to the analytic solution. Although CPU time does not exactly correlate to the number of cells, efficiency improvements of the AMR operations are currently under development and we expect the CPU times to have a similar reduction as the reduction of space-time elements with minimal overhead in future work.

VII.B. Flow over a Cube

To demonstrate the three-dimensional feature tracking ability of SAMCart-DG, we model a flow over a cube by placing a 2 x 2 viscous wall boundary condition centered on the left face of a rectangular domain given by non-dimensional coordinates $(0, -10, -10)$ x $(40, 10, 10)$. Using a fourth-order simulation, cells are refined when the magnitude of vorticity at any quadrature point in a cell is greater than $0.2s^{-1}$. In Fig.18, we illustrate the adaptive meshing capability to track vorticies. In Fig.19, we demonstrate a time sequence of vorticity contours and their respective grids. The maximum refinement resolution of this simulation is 320 x 160 x 160. We allow a total of five levels during the simulation and start the simulation with a base grid of 20 x 10 x 10. In comparison to a fixed refined grid, the total space-time element count is approximately ten times less using the SAMR framework. We note that the vorticies expand to fill a majority of the computational domain in this example.

VIII. Concluding Remarks and Future Work

We have developed a collocated, discontinuous Galerkin solver in a block-structured Cartesian framework utilizing a tensor-product basis formulation. Additionally, we have successfully implemented this high-order, Cartesian mesh-based solver into a structured adaptive mesh refinement framework using the open source framework SAMRAI. We have demonstrated the competitive accuracy and efficiency attributes of our Cartesian DG implementation, and have shown that the structured adaptive mesh refinement (SAMR) approach is capable of maintaining the same error levels as a fixed fine grid while significantly reducing the total element count for time-dependent simulations. We also have demonstrated the SAMR framework in a three-dimensional model problem which was able to track features in the flow while reducing the space-time element counts by an order of magnitude with a majority of the computational domain having significant flow features.

At present, the efficiency of certain aspects of the implementation remain non-optimal such as the high-order coarsen and refine operations, and in certain cases the adaptive meshing operations can constitute a significant fraction of the overall solution time. Through further optimization we expect the CPU times for the adaptive mesh solver to scale proportionally with the total number of space-time elements in the simulation.

To further extend the adaptive capability of the solver, future work will extend the SAMR capability to include p-enrichment in the SAMRAI framework, in order to enable combined h-p refinement. To apply this solver to real engineering problems, following work will adopt the multi-solver, multi-mesh paradigm using an overset methodology following previous work with low-order solvers.

IX. Acknowledgments

This work was partially funded under ONR Grant N00014-14-1-0045 and by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under Award # DE-SC0012671. Computer time was provided by the University of Wyoming Advanced Computing Research Center (ARCC) and the NCAR-Wyoming Supercomputer Alliance.

References

[1]Hornung, R. D., Wissink, A. M., and Kohn, S. R., "Managing complex data and geometry in parallel structured AMR applications," *Engineering with Computers*, Vol. 22, No. 3-4, 2006, pp. 181–195.

[2]Adams, M., Colella, P., Graves, D. T., Johnson, J., Keen, N., Ligocki, T. J., Martin, D. F., McCorquodale, P., Schwartz, D. M. P., Sternberg, T., and Straalen, B. V., "Chombo Software Package for AMR Applications Design Document," 2014, Lawrence Berkeley National Laboratory Technical Report LBNL-6616E.

[3]Wissink, A., Kamkar, S., Pulliam, T., Sitaraman, J., and Sankaran, V., "Cartesian Adaptive Mesh Refinement for Rotorcraft Wake Resolution," AIAA Paper 2010-4554, 28th AIAA Applied Aerodynamics Conference, Chicago, IL, June 2010.

[4]Wissink, A., Jayaraman, B., Datta, A., Sitaraman, J., Potsdam, M., Kamkar, S., Mavriplis, D., Yang, Z., Jain, R., Lim, J., et al., "Capability enhancements in Version 3 of the Helios high-fidelity rotorcraft simulation code," AIAA Paper 2012-713, 50th AIAA Aerospace Sciences Meeting, Nashville, TN, January 2012.

[5]Fadlun, E., Verzicco, R., Orlandi, P., and Mohd-Yusof, J., "Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations," *Journal of Computational Physics*, Vol. 161, No. 1, 2000, pp. 35–60.

[6]Peskin, C. S., "The immersed boundary method," *Acta numerica*, Vol. 11, 2002, pp. 479–517.

[7]Mittal, R. and Iaccarino, G., "Immersed boundary methods," *Annu. Rev. Fluid Mech.*, Vol. 37, 2005, pp. 239–261.

[8]Aftosmis, M. J., "Solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries," *VKI Lecture Series*, Vol. 2, 1997.

[9]Ingram, D. M., Causon, D. M., and Mingham, C. G., "Developments in Cartesian cut cell methods," *Mathematics and Computers in Simulation*, Vol. 61, No. 3, 2003, pp. 561–572.

[10]Marshall, D. D. and Ruffin, S. M., "A new inviscid wall boundary condition treatment for embedded boundary Cartesian grid schemes," AIAA Paper 2004-583, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2004.

[11]Sitaraman, J., Mavriplis, D., and Duque, E. P., "Wind Farm Simulations Using a Full Rotor Model for Wind Turbines," AIAA Paper 2014-1086, 32nd ASME Wind Energy Symposium, National Harbor, MD, January 2014.

[12]Cockburn, B., Karniadakis, G. E., and Shu, C.-W., *The development of discontinuous Galerkin methods*, Springer, 2000.

[13]Luo, H., Baum, J. D., and Löhner, R., "A discontinuous Galerkin method based on a Taylor basis for the compressible flows on arbitrary grids," *Journal of Computational Physics*, Vol. 227, No. 20, 2008, pp. 8875–8893.

[14]Ceze, M. and Fidkowski, K. J., "Drag prediction using adaptive discontinuous finite elements," *Journal of Aircraft*, Vol. 51, No. 4, 2014, pp. 1284–1294.

[15]Darmofal, D. L., Allmaras, S. R., Yano, M., and Kudo, J., "An adaptive, higher-order discontinuous Galerkin finite element method for aerodynamics," AIAA Paper 2013-2871, 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 2013.

[16]Haga, T., Gao, H., and Wang, Z., "A high-order unifying discontinuous formulation for the Navier-Stokes equations on 3D mixed grids," *Mathematical Modelling of Natural Phenomena*, Vol. 6, No. 03, 2011, pp. 28–56.

[17]Hartmann, R., "Higher-order and adaptive discontinuous Galerkin methods with shock-capturing applied to transonic turbulent delta wing flow," *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, 2013, pp. 883–894.

[18]Brazell, M. J. and Mavriplis, D. J., "3D Mixed Element Discontinuous Galerkin with Shock Capturing," AIAA Paper 2013-3064, 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA., June 2013.

[19]Wang, L., Anderson, W. K., Erwin, J. T., and Kapadia, S., "Discontinuous Galerkin and Petrov Galerkin methods for compressible viscous flows," *Computers & Fluids*, Vol. 100, 2014, pp. 13–29.

[20]Glasby, R. S., Burgess, N., Anderson, K., Wang, L., Allmaras, S., and Mavriplis, D., "Comparison of SU/PG and DG finite-element techniques for the compressible Navier-Stokes equations on anisotropic unstructured meshes," AIAA Paper 2013-691, 51st AIAA Aerospace Sciences Meeting, Grapevine, TX, January 2013.

[21]Huynh, H. and Kroll, N., "Third International Workshop on High-Order CFD Methods," `https://www.grc.nasa.gov/hiocfd/`.

[22]Diosady, L. T. and Murman, S. M., "Design of a Variational Multiscale Method for Turbulent Compressible Flows," AIAA Paper 2013-2870, 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 2014.

[23]Hindenlang, F., Gassner, G. J., Altmann, C., Beck, A., Staudenmaier, M., and Munz, C.-D., "Explicit discontinuous Galerkin methods for unsteady problems," *Computers & Fluids*, Vol. 61, 2012, pp. 86–93.

[24]El Khoury, G. K., Schlatter, P., Noorani, A., Fischer, P. F., Brethouwer, G., and Johansson, A. V., "Direct numerical simulation of turbulent pipe flow at moderately high Reynolds numbers," *Flow, turbulence and combustion*, Vol. 91, No. 3, 2013, pp. 475–495.

[25]Solin, P., Segeth, K., and Dolezel, I., *Higher-order finite element methods*, CRC Press, 2004.

[26]Brazell, M. J., Mavriplis, D. J., and Sitaraman, J., "An Overset Mesh Approach for 3D Mixed Element High Order Discretizations," AIAA Paper 2015-1739, 53rd AIAA Aerospace Sciences Meeting, Kissimmee, FL, January 2014.

[27]Harten, A., Lax, P. D., and Leer, B. v., "On upstream differencing and Godunov-type schemes for hyperbolic conservation laws," *SIAM review*, Vol. 25, No. 1, 1983, pp. 35–61.

[28]Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of computational physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[29]Hartmann, R. and Houston, P., "An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier-Stokes equations," *J. Comput. Phys.*, Vol. 227, No. 22, 2008/11/20, pp. 9670 – 85.

[30]Shahbazi, K., Mavriplis, D., and Burgess, N., "Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations," *J. Comput. Phys.*, Vol. 228, No. 21, 2009/11/20, pp. 7917 – 40.

[31]Kopriva, D. A. and Gassner, G., "On the quadrature and weak form choices in collocation type discontinuous Galerkin spectral element methods," *Journal of Scientific Computing*, Vol. 44, No. 2, 2010, pp. 136–155.

[32]Kahaner, D., Moler, C., and Nash, S., "Numerical methods and software," *Englewood Cliffs: Prentice Hall, 1989*, Vol. 1, 1989.

[33]Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A., *Spectral methods: fundamentals in single domains*, Springer, 2006.

[34]Kirby, R. M. and Karniadakis, G. E., "De-aliasing on non-uniform grids: algorithms and applications," *Journal of Computational Physics*, Vol. 191, No. 1, 2003, pp. 249–264.

[35]Gassner, G. J. and Beck, A. D., "On the accuracy of high-order discretizations for underresolved turbulence simulations," *Theoretical and Computational Fluid Dynamics*, Vol. 27, No. 3-4, 2013, pp. 221–237.

[36]Butcher, J. C., *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*, Wiley-Interscience, 1987.

[37]Jameson, A., Schmidt, W., Turkel, E., et al., "Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes," AIAA Paper 1981-1259, 14th Fluid and Plasma Dynamics Conference, Palo Alto, CA, June 1981.

[38]Guarini, S., *Direct Numerical Simulation of Supersonic Turbulent Boundary Layers*, Ph.D. thesis, Stanford University, June 1998.

[39]Taylor, G. and Green, A., "Large Ones," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, Vol. 158, No. 895, 1937, pp. 499–521.

[40]Brachet, M., "Direct simulation of three-dimensional turbulence in the TaylorGreen vortex," *Fluid Dynamics Research*, Vol. 8, No. 1, 1991, pp. 1–8.

[41]Povitsky, A., "High-incidence 3-D lid-driven cavity flow," AIAA Paper 2001-2847, 15th AIAA Computational Fluid Dynamics Conference, Anaheim, CA, June 2001.

[42]Povitsky, A., "Three-dimensional flow in cavity at yaw," *Nonlinear Analysis: Theory, Methods & Applications*, Vol. 63, No. 5, 2005, pp. e1573–e1584.

[43]Feldman, Y. and Gelfgat, A. Y., "From multi-to single-grid CFD on massively parallel computers: Numerical experiments on lid-driven flow in a cube using pressure–velocity coupled formulation," *Computers & Fluids*, Vol. 46, No. 1, 2011, pp. 218–223.

[44]d'Humières, D., "Multiple–relaxation–time lattice Boltzmann models in three dimensions," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, Vol. 360, No. 1792, 2002, pp. 437–451.

[45]Gelfgat, A. and Feldman, Y., "Reply to a letter of A. Povitsky regarding benchmark problem of 3D flow in a cubic cavity driven by a diagonally moving lid," *Computers & Fluids*, Vol. 92, 2014, pp. 224.

[46]Ringleb, F., "Exakte Loesungen der Differentialgleichungen einer adiabatischen Gasstroemung," *A. Angew. Math. Mech.*, Vol. 20, No. 4, 1940, pp. 185–198.

[47]van Ress, W., Leonard, A., Pullin, D., and Koumoutsakos, P., "A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high Reynolds number," *J. Comput. Phys.*, Vol. 230, 2011, pp. 2794–2805.

[48]Albensoeder, S. and Kuhlmann, H. C., "Accurate three-dimensional lid-driven cavity flow," *Journal of Computational Physics*, Vol. 206, No. 2, 2005, pp. 536–558.

[49]Povitsky, A., "Three-dimensional flow in cavity at Yaw," Tech. rep., DTIC Document, 2001.

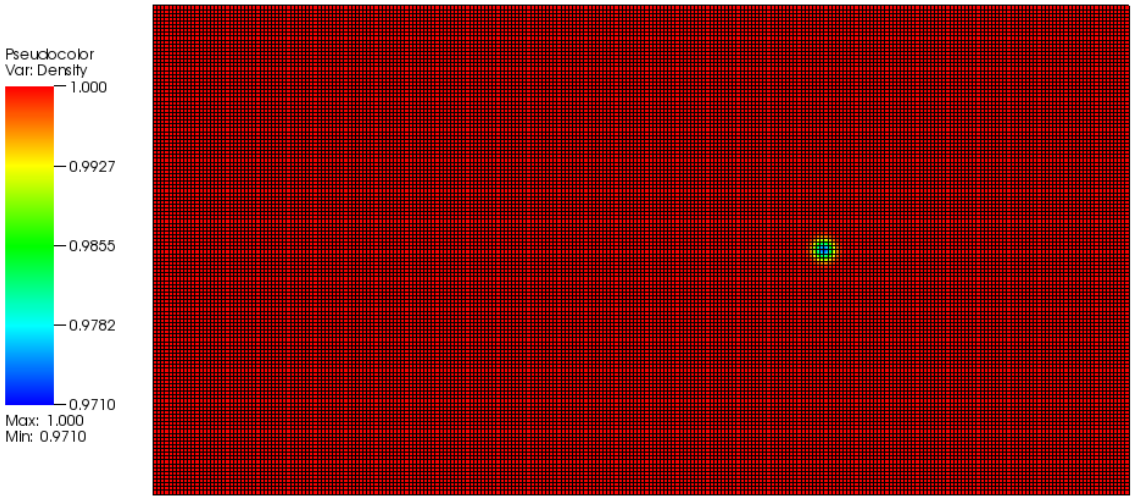
[50]National Center for Atmospheric Research, Boulder, CO, *Yellowstone: IBM iDataPlex System (Climate Simulation Laboratory)*, 2012, http://n2t.net/ark:/85065/d7wd3xhc.

[51]Wissink, A. M., Sitaraman, J., Sankaran, V., Mavriplis, D. J., and Pulliam, T. H., "A multi-code python-based infrastructure for overset CFD with adaptive cartesian grids," AIAA Paper 2008-927, 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2008.

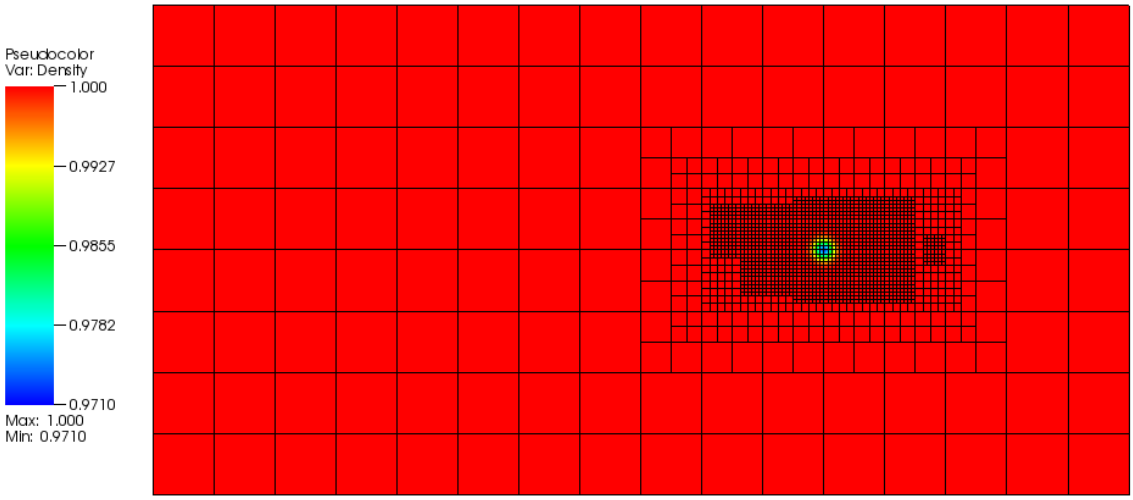
[52]Kopera, M. A. and Giraldo, F. X., "Analysis of adaptive mesh refinement for IMEX discontinuous Galerkin solutions of the compressible Euler equations with application to atmospheric simulations," *Journal of Computational Physics*, Vol. 275, 2014, pp. 92–117.

[53]Gunney, B. T., Wissink, A. M., and Hysom, D. A., "Parallel clustering algorithms for structured AMR," *Journal of Parallel and Distributed Computing*, Vol. 66, No. 11, 2006, pp. 1419–1430.

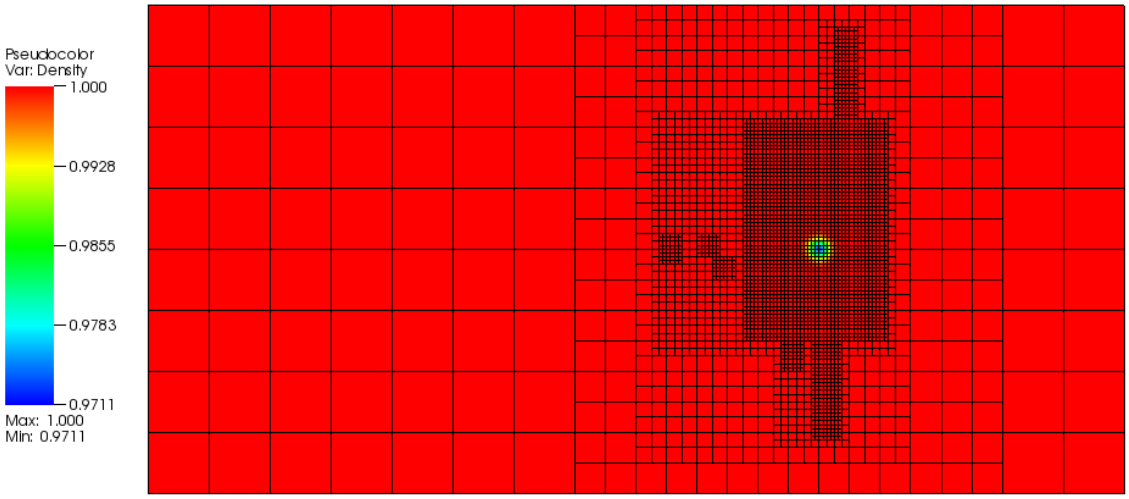
[54]Anderson, R., Arrighi, W., Elliott, N., Gunney, B., and Hornung, R., "SAMRAI Concepts and Software Design," Feb 2013, `https://computation.llnl.gov/project/SAMRAI/download/SAMRAI-Concepts_SoftwareDesign.pdf`.



(a) p=2, fixed mesh

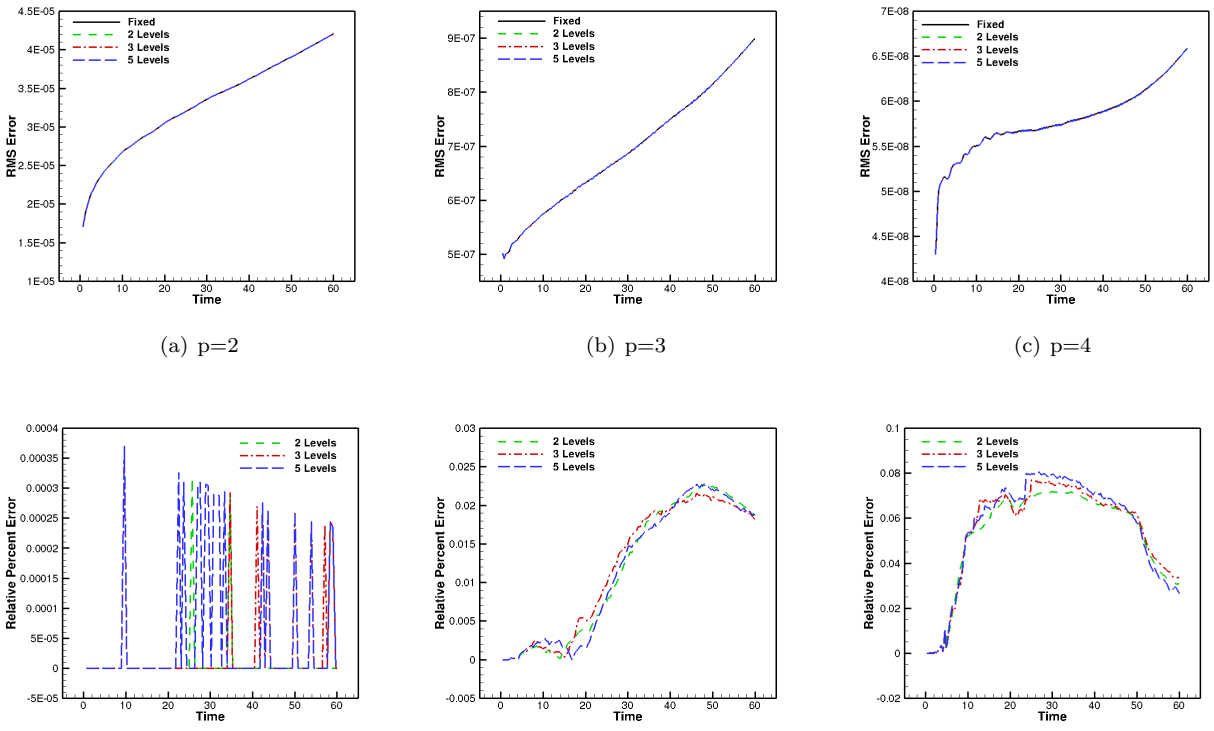


(b) p=2, five refinement levels



(c) p=4, five refinement levels

Figure 15. Density on fixed and SAMR grids for isentropic vortex convection at the final solution time.



(a) p=2 (b) p=3 (c) p=4

(d) p=2 (e) p=3 (f) p=4

Figure 16. Computed error levels for the convection of an isentropic vortex problem using various refinement levels.

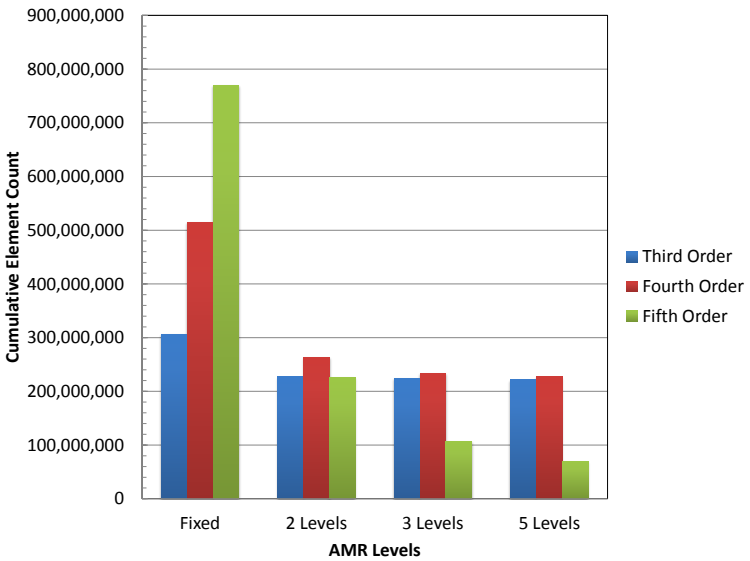
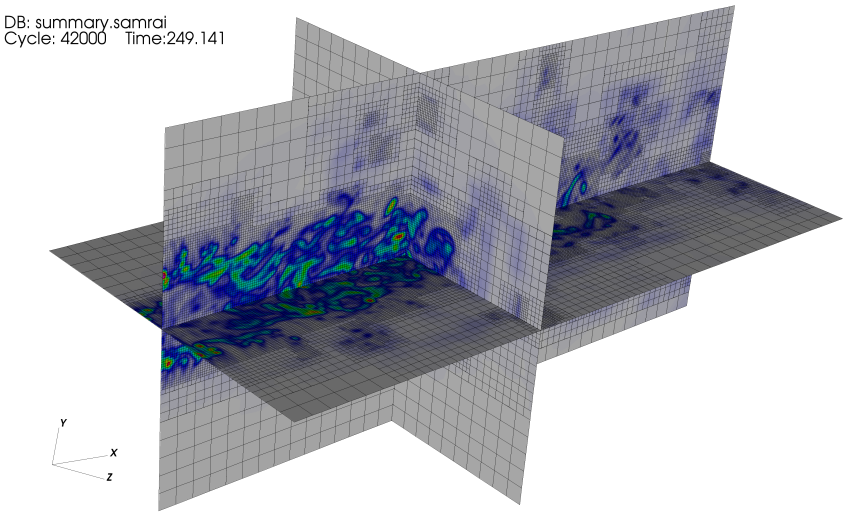
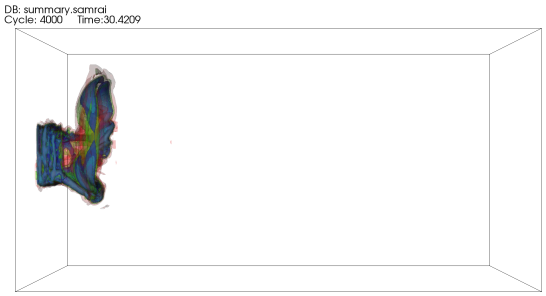


Figure 17. Convection of an isentropic vortex using various refinement levels: total space-time elements.

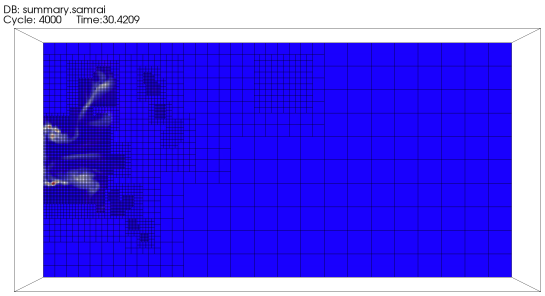


(a) Vorticity contours: step 4000

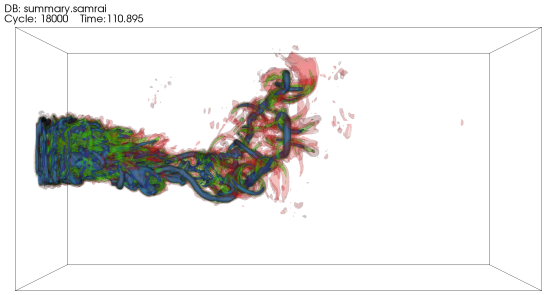
Figure 18. Flow over a cube (Re = 1000): AMR grid colored by vorticity magnitude



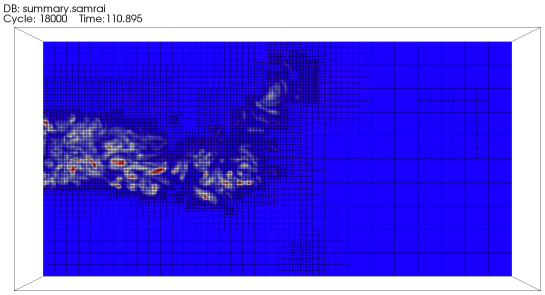
(a) Vorticity contours: step 4000



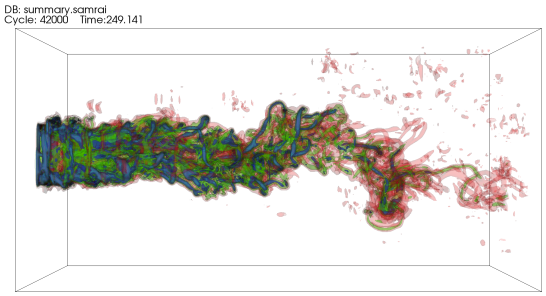
(b) AMR grid colored by vorticity magnitude: step 4000



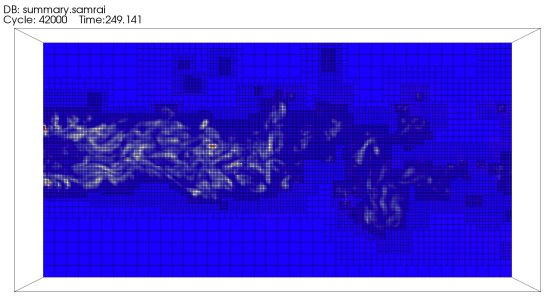
(c) Vorticity contours: step 18000



(d) AMR grid colored by vorticity magnitude: step 18000



(e) Vorticity contours: step 42000



(f) AMR grid colored by vorticity magnitude: step 42000

Figure 19. Three-dimensional flow over a cube: Re = 1000.