A high-order discontinuous-Galerkin octree-based AMR solver for overset simulations

Michael J. Brazell[a], Andrew C. Kirby[b], and Dimitri J. Mavriplis[c]

Department of Mechanical Engineering
University of Wyoming

The goal of this work is to develop a highly efficient off-body solver for use in overset simulations. Overset meshes have been gaining traction in recent years and are being used increasingly to simulate very complex large-scale problems. In particular we focus on a dual-mesh, dual-solver overset approach that combines specialized flow solvers in different regions of the flow domain: near-body and off-body. The near-body flow solver is designed to handle complicated geometry, anisotropic elements, and unstructured meshes. In contrast, the off-body solver is designed to be high-order, Cartesian, and use adaptive mesh refinement (AMR). The high-order discretization used for the off-body solver is based on the discontinuous Galerkin (DG) method. To get the most efficiency out of the method, a Cartesian grid is employed and tensor product basis functions are used in the DG formulation. The dense computational kernels allow this solver to obtain a near-constant cost per degree of freedom for a wide range of p-orders of accuracy. To further enhance the capabilities of the off-body solver, the DG solver in linked to an octree-based AMR library called `p4est`; this gives the ability for h-adaptation via non-conforming elements. p-adaption is also implemented in which each cell can be assigned a different polynomial degree basis. Combined h and p refinement is necessary for overlapping mesh problems where the off-body solver mesh must connect to the low-order near-body solver, since both mesh resolution and order of accuracy must be matched in the overlapping regions. To demonstrate the efficiency, accuracy, and capabilities of the DG AMR flow solver we simulate Ringleb flow and the Taylor-Green vortex problem. Finally, to demonstrate the overset capabilities a NACA 0015 wing case and a NREL PhaseVI wind turbine case are simulated.

I. Introduction

Overset mesh approaches have been used for many years in computational fluid dynamics due to the flexibility they afford for handling complex geometries and simulations involving bodies with relative motion such as store separation, rotorcraft, and wind turbine configurations.[1–4] Most overset mesh applications involve the use of similar mesh components, i.e. either structured meshes overlapped with structured meshes[5] or collections of overlapping unstructured meshes.[6] More recently, frameworks that support combinations of different types of overlapping meshes have been developed. One example is the HELIOS rotorcraft software,[4] which supports arbitrary combinations of unstructured, structured, and Cartesian meshes, along with the different discretizations used on these meshes. Figure 1 shows a dual-mesh dual-solver overset framework used by HELIOS where there is an unstructured near-body mesh and a Cartesian AMR off-body mesh. HELIOS applications have shown how the use of adaptively refined Cartesian meshes in off-body regions can be enabling for capturing wakes and vortices with high levels of accuracy.[4,7,8] High-order accurate discretizations have generated much interest over the last decade for computational aerodynamics due to

[a]AIAA member, Research Scientist, mbrazell@uwyo.edu
[b]AIAA member, Ph.D. Candidate, akirby@uwyo.edu
[c]AIAA Associate Fellow, Professor, mavripl@uwyo.edu

the potential these methods hold for achieving higher accuracy at reduced cost and improved scalability on emerging parallel computer architectures. Naturally, the use of high-order methods in the context of overset mesh applications has been pursued by various authors who have shown that design accuracy can be maintained in an overset mesh framework for high-order discontinuous Galerkin (DG) discretizations.[9–12]

The off-body solver is designed to be highly accurate and efficient. To achieve this, a high-order DG solver is combined with adaptive mesh refinement (AMR). To enable AMR the DG solver is linked to an Octree library called `p4est`.[13–15] Using AMR, the mesh is adapted only where features are detected, which greatly reduces the number of degrees of freedom. The high-order DG AMR solver is designed to be as simple and efficient as possible. It also was developed to simultaneously be capable of 2D and 3D simulations.

In the following sections, the DG AMR discretization and capabilities are described in detail, along with the approach to linking to the `p4est` library and other flow solvers through a common driver interface. The results section contains stand-alone performance of the DG solver, verification of the DG AMR solver using Ringleb flow and the Taylor-Green vortex problem. Lastly, the overset capabilities are demonstrated by simulating a NACA 0015 wing and a NREL PhaseVI wind turbine.

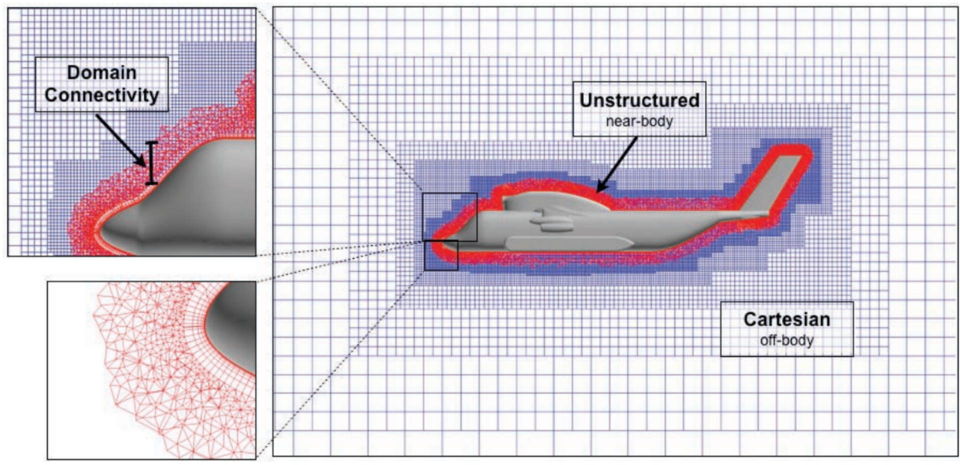


Figure 1: Example of a HELIOS dual-mesh, dual-solver overset approach.[4]

II. Governing Equations

The Navier-Stokes equations govern the dynamics of compressible fluids and are given 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, $H = E + P/\rho$ 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. Einstein notation is used where the subscripts of i and j represent spatial dimensions and have a range of 1 to 3 and the indices of m and n vary over the number of variables.

III. Off-body DG AMR solver formulation

In this section the DG finite element formulation used to solve the Navier-Stokes equations is described. The DG kernel that is responsible for the physics and discretization is referred to as CartDG.[16] To derive the weak form, equation (1) is first multiplied by a test function ϕ_r and integrated over the domain Ω to obtain the weak form:

$$\int_\Omega \phi_r \left(\frac{\partial U_m}{\partial t} + \frac{\partial F_{mi}}{\partial x_i} \right) \mathrm{d}\Omega = 0.$$

Integration by parts is performed and the residual R_{mr} is defined as:

$$R_{mr} = \int_\Omega \left(\phi_r \frac{\partial U_m}{\partial t} - \frac{\partial \phi_r}{\partial x_i} F_{mi} \right) \mathrm{d}\Omega + \int_\Gamma \phi_r F_{mi} \mathrm{n}_i \mathrm{d}\Gamma = 0$$

where ϕ_r are the basis functions and the solution is approximated using $U_m = \phi_s a_{ms}$. The indices r and s run over the number of basis functions. The basis functions used in this formulation are tensor products of 1D Lagrange polynomials. To construct these Lagrange polynomials a nodal basis is formed on the Gauss-Legendre quadrature points. This creates a highly efficient nodal collocated basis function where each basis function equals one at a quadrature point and zero on the remaining quadrature points.

The residual now contains integrals over faces Γ and special treatment is needed for the fluxes in these terms. The advective fluxes are calculated using a Lax-Friedrichs flux.[17] The diffusive fluxes are handled using a symmetric interior penalty (SIP) method.[18,19] For the temporal discretization high-order explicit Runge-Kutta schemes are used including the standard RK4[20] and SSP RK2.[21] These schemes are used on systems of ODE's so the residual formulation is rearranged to be:

$$M_{mrns} \frac{\partial a_{ns}}{\partial t} = \int_\Omega \frac{\partial \phi_r}{\partial x_i} F_{mi} \mathrm{d}\Omega - \int_\Gamma \phi_r F_{mi} \mathrm{n}_i \mathrm{d}\Gamma$$

where M_{mrns} is a mass matrix and the right hand side now contains only spatial residual terms. The mass matrix is inverted to create a system of ODE's. The Lagrange basis functions used in this formulation form a diagonal mass matrix is which makes inversion trivial.

A. Tensor formulation

The main goal of this work is to develop a highly efficient high-order accurate off-body solver. To do this we have made the simplification that the domain is comprised of purely Cartesian hexahedral elements. An efficient set of basis functions for hexahedral elements are tensor product Lagrange polynomials collocated on Gauss-Legendre quadrature points. This formulation is commonly called DGSEM and comes from the combination of the DG discretization and the spectral element method (SEM).[22–25] This allows for many simplifications in both the projection and integration routines. For a standard finite element method with no simplifications an integration and projection routine takes $\mathcal{O}((p+1)^6)$ operations. A solution projection is defined as:

$$U(\xi, \eta, \zeta) = \sum_{r,s,t=0}^{p+1} \psi_{rst}(\xi, \eta, \zeta)\, a_{rst}$$

where an inner product of the basis functions with the solution coefficients requires $\mathcal{O}((p+1)^3)$ work for each quadrature point. In the case that the number of quadrature points is equal to the number of basis functions this equates to $\mathcal{O}((p+1)^6)$ work per cell. Since ψ is a tensor product of one-dimensional basis functions we can expand it to be:

$$\psi_{rst}(\xi, \eta, \zeta) = \ell_r(\xi)\ell_s(\eta)\ell_t(\zeta)$$

where $\ell(\xi)$ is a one-dimensional Lagrange polynomial:

$$\ell_j(\xi) = \prod_{i=0, i\neq j}^{p+1} \frac{(\xi - \xi_i)}{(\xi_j - \xi_i)}$$

where ξ_i are the Gauss-Legendre quadrature points and has the property:

$$\ell_j(\xi_i) = \delta_{ij}$$

at the quadrature points. Therefore, the projection becomes:

$$U(\xi_i, \eta_j, \zeta_k) = \sum_{r,s,t=0}^{p+1} \ell_r(\xi_i)\ell_s(\eta_j)\ell_t(\zeta_k)\, a_{rst}$$

and simplifies to:

$$U(\xi_i, \eta_j, \zeta_k) = \sum_{r,s,t=0}^{p+1} \delta_{ri}\delta_{sj}\delta_{tk}\, a_{rst} = a_{ijk}.$$

The projection work reduces to $\mathcal{O}((p+1)^3)$ operations.

A volume integral has a similar simplification, we begin by looking at one term in the weighted integral form:

$$R^{v1}_{mrst} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \frac{\partial \psi_{rst}}{\partial \xi} F_{m1} \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta$$

where R^{v1}_{mrst} is a single volume residual for a particular element in the ξ-direction. The tensor basis is introduced to give:

$$= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \frac{\partial \ell_r(\xi)}{\partial \xi} \ell_s(\eta)\ell_t(\zeta) F_{m1} \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta.$$

The integrals are approximated using Gauss-Legendre quadrature rules:

$$\simeq \sum_{i,j,k=0}^{p+1} \frac{\partial \ell_r(\xi_i)}{\partial \xi} \ell_s(\eta_j)\ell_t(\zeta_k) F_{m1}(\xi_i, \eta_j, \zeta_k) w_i w_j w_k$$

where w are the quadrature weights. Simplifying further we arrive at:

$$= \sum_{i,j,k=0}^{p+1} \frac{\partial \ell_r(\xi_i)}{\partial \xi} \delta_{js}\delta_{kt} F_{m1}(\xi_i, \eta_j, \zeta_k) w_i w_j w_k$$

and finally due to collocation:

$$= \sum_{i=0}^{p+1} \frac{\partial \ell_r(\xi_i)}{\partial \xi} F_{m1}(\xi_i, \eta_s, \zeta_t) w_i w_s w_t$$

where only a one-dimensional summation remains. The integral is performed for $(p+1)^3$ basis functions and so the work for this particular integral reduces from $\mathcal{O}((p+1)^6)$ to $\mathcal{O}((p+1)^4)$ operations.

A detailed summary of the total work and memory for each operation in the DGSEM method is shown in Ref.[25] The combination of tensor products and collocation creates a large reduction in the total number of floating point operations. However, at lower polynomial degrees the flop rate is higher for the naive approach since the operations can be converted into matrix-matrix products. Therefore, at lower polynomial degrees the naive approach can be faster overall, especially when using linear-algebra libraries such as BLAS. This polynomial degree crossover point will depend on the hardware and the implementation and is thoroughly studied in Ref.[26] A demonstration of this will be shown in the next section in a discussion of the performance of CartDG.

B. CartDG stand-alone performance

CartDG is designed for high computational efficiency. To achieve this, high-order finite-element discretizations are preferred in order capitalize on the dense computational kernels associated with these discretizations, while taking advantage of matrix-matrix multiplication implementations when possible. The discretization is based on a nodal, collocated tensor basis embedded in a Cartesian coordinate system.

The mass matrix of a nodal, collocated basis forms a diagonal, static matrix meaning that the mass matrix does not change over the duration of the simulation. Many implementations of the DG method involve a matrix multiplication of the inverted mass matrix and the spatial residual vector. Since the inverse mass matrix is static over the duration of the simulation, it can be pre-multiplied into the basis functions that are used to construct the spatial residual vector therefore saving a matrix-vector multiplication at every residual evaluation.

By restricting the discretization to a Cartesian coordinate system, further simplifications can be introduced. The element Jacobian which is used to transform the solution in physical space to reference space is constant for every cell with cell width δx. Therefore, the element Jacobian is constant and can also be pre-multiplied into the basis functions used to construct the spatial residual vector. Additionally, since the normal vectors in a Cartesian coordinate system only contain one component, e.g. n=(1,0,0), specialized routines for flux calculations are constructed for each coordinate direction that eliminate floating point operations related to the orthogonal coordinates.

Advanced vectorization and data alignment techniques are employed throughout the implementation. At higher polynomial degrees it is advantageous to rearrange the solution order in memory. Traditionally, for column-major memory storage, the solution vector is stored by ordering fields first, followed by modes and elements, where the fields are the conservative flow variables and the modes are the polynomial modes. However, in order to perform single-instruction-multiplie-data (SIMD) vectorization of flux calculations, reordering the storage to modes, fields and elements is needed. When this action is performed, vectorization allows for either two, four, or eight flux evaluations to be performed in parallel with the Intel AVX, AVX-2, or AVX-512 vector instructions, respectively.

To demonstrate the computational efficiency of CartDG, several performance statistics for a wide range of polynomial degrees are provided. The sustained peak performance of the inviscid and viscous residual evaluations is shown in Figure 2 on a Intel Core i7-5960X Haswell-E processor with eight CPU cores, clock speed of 3GHz and 4GB of memory per core. For reference, the TAU benchmark for this processor is 5.25 seconds with a theoretical peak performance of 384 GFLOPS. As seen from the figure, the tensor-product based FEM formulation achieves roughly 12% sustained compute peak performance for evaluation of the residual of the full Navier-Stokes equations (viscous residual) at high polynomial degree. Figure 3 demonstrates the time per degree of freedom for inviscid and viscous residual evaluations. Overall, the time required for a viscous residual evaluation is higher than for an equivalent inviscid residual evaluation. This is because there are more operations required for the viscous residual evaluation. For higher polynomial degrees, the time per degree of freedom becomes approximately constant for both inviscid and viscous residuals. The performance suffers at low polynomial degrees because the number of floating point operations is very low compared to the number of bytes transferred from DRAM required for the calculation. This severely limits the performance because the problem becomes communication bound meaning the performance is limited by the communication bandwidth of the CPU. Figure 4 shows the sustained peak performance of the tensor-based DG method in comparison to a general FEM formulation on the NWSC-1 Yellowstone supercomputer using 128 Intel Xeon E5-2670 cores with a clock speed of 2.6Ghz and 2GB per core memory. The general FEM implementation is able achieve nearly 65% sustained peak performance whereas the tensor-product formulation achieves 10% of peak performance. However, the general FEM formulation requires significantly more floating point operations, most of which are multiplication and addition of zeros for a nodal collocated basis. This is demonstrated in Figure 5 comparing the time per degree of freedom for a viscous residual evaluation. The cost of the general FEM formulation is nearly 10 times larger than that required by the tensor-product formulation at high polynomial degrees. The only polynomial degree for which the general FEM formulation is more efficient is p=0 which corresponds to first-order solution accuracy. However, this polynomial degree is never used in simulations because of its severe inaccuracies and large numerical dissipation.

C. Coarsen and refine operators

Coarsen and refine linear operators are responsible for transferring solutions between coarse and fine approximations.[15] In this work, we allow h-adaption which modifies the mesh size by a factor of two as well as p-adaption which modifies the polynomial degree within a cell. We define h-coarsening as transferring children cells to a parent cell, for example in three dimensions this turns eight hexahedral elements into one hexahedron, whereas h-refining is the reverse of this procedure. Lowering the polynomial degree of a hexahedron will be referred to as p-coarsening and increasing the polynomial degree as p-refining. In the case that we want to apply an h-coarsening to a set of children with different polynomial degrees we search for the maximum polynomial degree of the children and perform a p-refinement on all of the children first.

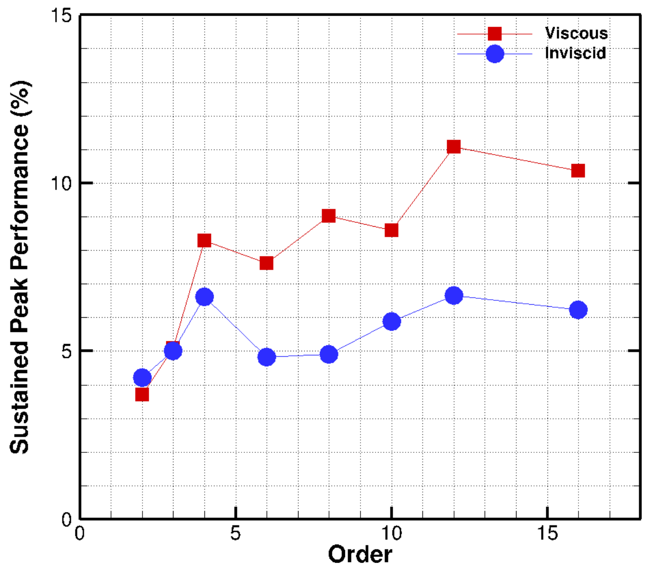


Figure 2: Sustained peak performance for $p = 1 - 15$ tensor product, inviscid and viscous

We can then apply a fixed polynomial degree h-coarsening operator to the children. This is an algorithmic simplification and more complicated coarsen and refine operators can be derived that perform this operation in one step. First we will derive the p refine and coarsen operators and then the h variants in one dimension.

The p coarsen and refine operators are derived by minimizing the norm:

$$E(a_p) = \|\phi_p^T a_p - \phi_q^T a_q\|_{L_2}$$

where $E(a_p)$ denotes the norm of the difference between the two representations of the solution, a_p are the solution coefficients for the new polynomial degree p and its corresponding basis ϕ_p. The solution coefficients a_q are from the original cell of a polynomial degree q and its corresponding basis ϕ_q. To minimize $E(a_p)$ we use the weighted integral method. This is performed by multiplying by ϕ_p and integrating over the domain and setting it equal to zero:

$$\int_{-1}^{1} \phi_p \phi_p^T a_p \mathrm{d}\xi - \int_{-1}^{1} \phi_p \phi_q^T a_q \mathrm{d}\xi = 0$$

where the left hand side becomes a mass matrix M_{pp}

$$M_{pp} a_p = \int_{-1}^{1} \phi_p \phi_q^T a_q \mathrm{d}\xi$$

and inverting the mass matrix leads to the new solution coefficients. In our case the mass matrix is diagonal and this step is trivial. Notice that we have not defined the polynomial degree p or q as this derivation works for both refining and coarsening. The transfer operator is therefore defined as:

$$T_{pq} = M_{pp}^{-1} \int_{-1}^{1} \phi_p \phi_q^T \mathrm{d}\xi$$

where T is a rectangular matrix that has $(p+1)^d$ rows and $(q+1)^d$ columns. This transfer operator is conservative and can be verified by:

$$\int_{-1}^{1} \phi_p T_{pq} a_q \mathrm{d}\xi = \int_{-1}^{1} \phi_q a_q \mathrm{d}\xi.$$

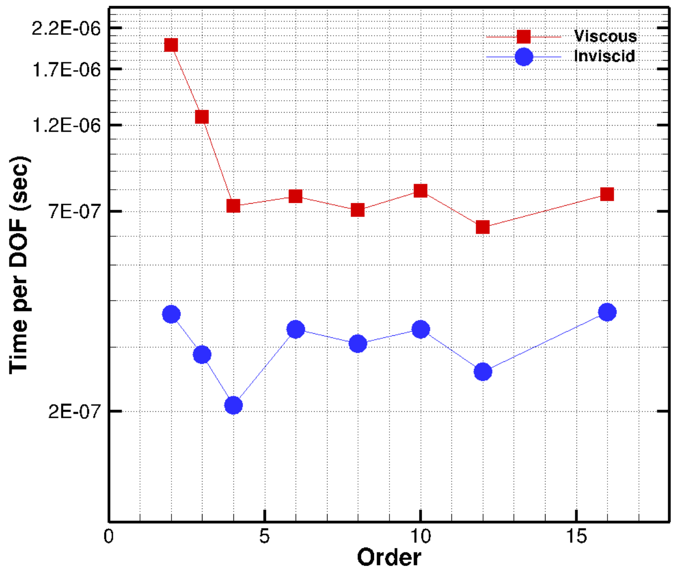


Figure 3: Time per degree of freedom for $p = 1 - 15$ tensor product, inviscid and viscous

The h coarsen and refine operators are derived in a similar way except there are multiple cells and therefore multiple integrals. The coarsen operator transfers the solution from the 2^d children to the parent using a similar minimization process as above. In one-dimension two children cells are transferred to one parent cell as depicted in Figure 6.

The h-coarsen operator C in one dimension is defined as

$$C = [G_1 \;\; G_2]$$

where

$$G_1 = M^{-1}\int_{-1}^{0} \phi\phi_1^T \mathrm{d}\xi \quad \mathrm{G}_2 = \mathrm{M}^{-1}\int_{0}^{1} \phi\phi_2^{\mathrm{T}} \mathrm{d}\xi$$

are block matrices. In general C is a rectangular matrix that has $(p+1)^d$ rows $2^d(p+1)^d$ columns. The left and right child basis functions are defined as:

$$\phi_1(\xi) = \phi(2\xi + 1) \quad \phi_2(\xi) = \phi(2\xi - 1)$$

which are translations of the parent basis functions. The parent mass matrix is defined as

$$M = \int_{-1}^{1} \phi\phi^T \mathrm{d}\xi$$

and the left and right children mass matrices simplify to become:

$$M_{11} = M_{22} = \frac{1}{2^d} M$$

since they are just $\frac{1}{2^d}$ fraction of the whole cell. The h-refine operator moves the solution coefficients from the parent to the children cells which turns into a pure injection and is depicted in Figure 7.

The h refine operator R in one dimension is defined as:

$$R = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$$

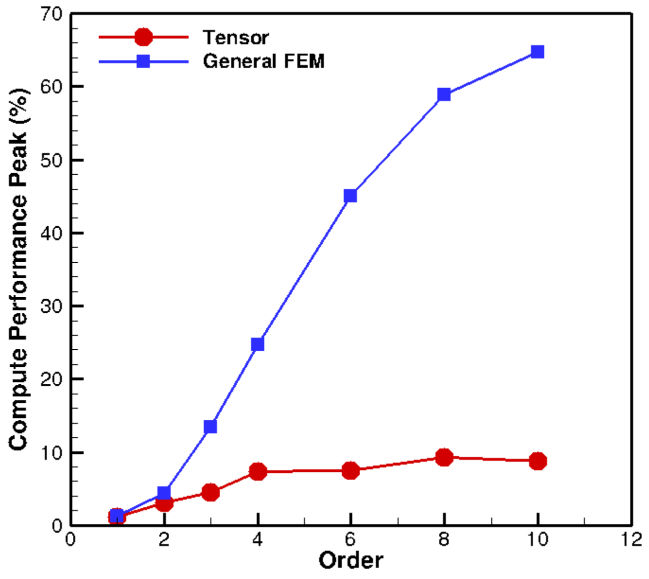


Figure 4: Tensor basis and general FEM performance peak for $p = 1 - 15$

where

$$P_1 = M_{11}^{-1} \int_{-1}^{0} \phi_1 \phi^T \mathrm{d}\xi, \quad \mathrm{P}_2 = \mathrm{M}_{22}^{-1} \int_{0}^{1} \phi_2 \phi^{\mathrm{T}} \mathrm{d}\xi.$$

In general R is a rectangular matrix that has $2^d(p+1)^d$ rows and $(p+1)^d$ columns. It follows that applying a refine and then a coarsen operator should return the original coefficients:

$$a = CRa$$

which is equivalent to the property that:

$$CR = I.$$

It is also important to maintain conservation with these operations. The h-coarsen operator holds this conservation property that the sum of the average cell quantity of the children cells equals the average cell quantity of the coarsened parent:

$$\int_{-1}^{0} \phi_1^T a_1 \, \mathrm{d}\xi + \int_{0}^{1} \phi_2^T a_2 \, \mathrm{d}\xi = \int_{-1}^{1} \phi^T \left(G_1 a_1 + G_2 a_2 \right) \mathrm{d}\xi.$$

The h-refine operator holds the conservation property that the average cell quantity of the parent equals the average quantities of the sum of the children cells:

$$\int_{-1}^{1} \phi^T a \, \mathrm{d}\xi = \int_{-1}^{0} \phi_1^T P_1 a \, \mathrm{d}\xi + \int_{0}^{1} \phi_2^T P_2 a \, \mathrm{d}\xi.$$

This process is extended to two and three dimensions using the tensor basis. This is a general formulation and some simplifications can occur depending on the chosen set of basis functions. For example the p-refine and coarsen operators turn into a matrix of ones and zeros for a Legendre basis.

D. p-adaption strategy

One objective of this work is to enable the use of high-order discretizations in off-body regions due to the fact that high-order methods are more effective than second-order accurate methods in terms of accuracy and

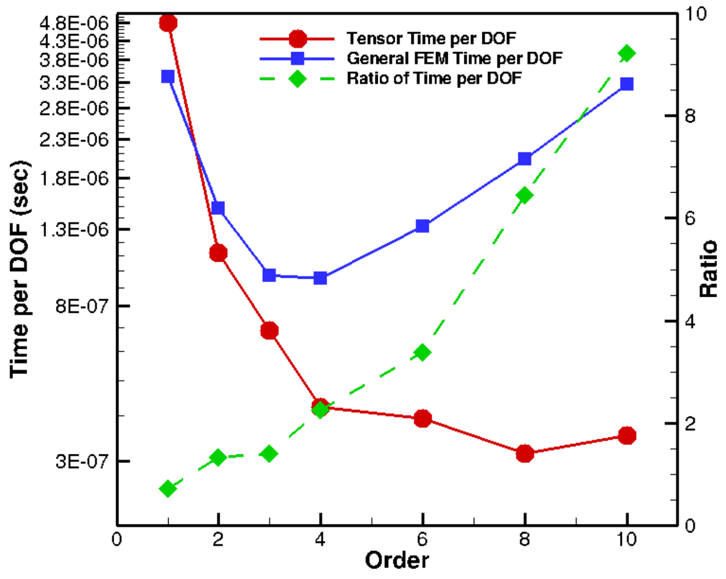


Figure 5: Time per degree of freedom for $p = 1 - 15$ tensor product and general FEM, inviscid and viscous

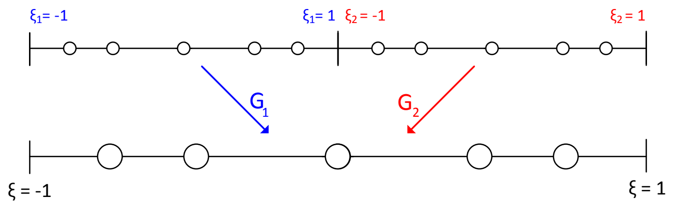


Figure 6: Visual depiction of an h-coarsen operator.

efficiency on a per degree-of-freedom basis. Since we are targeting applications with moderate Mach numbers and small temporal scales, we also chose to employ an explicit time-stepping scheme exclusively for the off-body solver. However, for explicit schemes high-order methods have a CFL time-step limit that becomes more restrictive with increasing polynomial degree. In overlapping mesh regions between the near-body and off-body grids, similar mesh resolution must be maintained in order to avoid the generation of orphan cells where solution interpolation cannot be maintained. This resolution matching requirement results in the generation of small cell sizes in overlap regions. For high-order discretizations, the combination of small cell size and high order accuracy becomes expensive both in terms of the overall number of degrees of freedom, as well as the small explicit time-step limit which is governed by the smallest cell with the highest order of accuracy in the entire off-body region. For these reasons, it is necessary to include a simple p-adaption strategy for the off-body solver that favors low polynomial degrees and fine meshes where the off-body is in close proximity to the near-body, and high polynomial degrees away from the near-body. The p-adaption strategy for the off-body solver can be summarized with these four rules:

- Refine and match the off-body resolution both in mesh size and accuracy with that of the near-body mesh in close proximity to the overlap region of the overset grids.
- Increase the polynomial degree in the off-body region as quickly as possible without creating a more restrictive time step by simultaneously increasing the mesh size and raising the polynomial degree of the discretization;

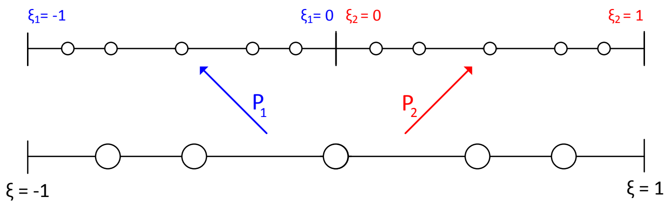


Figure 7: Visual depiction of an h-refine operator.

- Refine the mesh to flow features using the highest possible polynomial degree;
- Refine elements to stay ahead of propagating flow features. This is implemented by refining any neighboring cells that share a face with a cell that has been tagged for flow feature refinement.

To detect a flow feature we use gradients of the solution and compute vorticity at the quadrature points. If the vorticity magnitude at a quadrature point is greater than a tolerance τ then the cell is tagged for refinement. The vorticity criteria is defined as:

$$\omega_c = \|\Omega\|_F$$

where Ω is a vorticity tensor defined as:

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right)$$

and $\|\cdot\|_F$ is the Frobenius matrix-norm. We also have an option for detecting flow features using Q-criterion which is defined as:

$$Q_c = \frac{1}{2}\left(\|\Omega\|_F - \|S\|_F\right)$$

where S is the strain tensor defined as:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right).$$

E. Implementation of Octree AMR library `p4est`

The AMR library `p4est` is an octree based approach which is highly scalable.[13,14] The `p4est` library was used as part of the project that won the 2015 ACM Gordon Bell prize,[27] where the library was used to scale up to 1.5×10^6 processors with 6.02×10^8 degrees of freedom. Along with the impressive scalability of the `p4est` library we chose it due to the simplicity of the interface (written entirely in C language) and its natural fit with finite-element methods.

The `p4est` library handles all of the Octree storage which is distributed in parallel. The name of the library comes from the ability to handle forests of Octrees. The library provides some high level routines that use callback functions to manipulate and move data. The main `p4est` routines are "refine", "coarsen", "balance", "partition", and "iterate". "Refine" and "coarsen" routines recursively adapt a mesh based on callback functions the user writes. The "balance" routine ensures that the new adaption pattern obeys a 2:1 refinement criteria. The "partition" routine moves all of the data associated with the tree in parallel to equally distribute the work on each processor. The "iterator" provides a way to loop over cells and faces using callback functions. There are also options for edge and node callback functions which are useful for continuous methods.

IV. Overset framework

This section gives brief overviews of the driver, the near-body flow solver, and the overset mesh connectivity library that are used in conjunction with the off-body solver described in this work. All of these specialized individual components are combined in the dual-mesh dual-solver overset framework.

A. Driver

In a multi-solver framework a main driving routine is needed to coordinate and run each individual solver component. The driver in this work is similar to the multi-solver framework developed by Wissink et al. in Ref[28] with some notable differences. In this previous work, a Python interface is used to dynamically load different solvers, where each individual flow solver is typically run in sequence across all processors. This has the advantage of automatically load balancing across solver types but has some disadvantages. Each flow solver has a different amount of computational work and scales differently and in many instances concurrent execution of the different solvers on the optimal number of processors results in more effective resource utilization. Also, high-order implicit solvers have a very large memory footprint and running each solver on the same group of processors may have significant implications for memory requirements.

The driver in this work is written in C, handles multiple coding languages (C, C++, and Fortran) in the component solvers, and performs parallel communication through MPI. It has the flexibility of running flow-solvers sequentially on the same set of processors or concurrently on different processor groups. The solvers used in this work are self contained and only communicate with each other through the TIOGA overset mesh assembler. TIOGA performs both implicit hole cutting on all meshes and executes the fringe cell interpolation between overlapping meshes. All flow solvers and other codes that are controlled by the driver will be referred to as modules. These modules can include near-body and off-body flow solvers, overset domain connectivity assemblers, as well as other disciplinary solvers such as structural solvers, mesh deformation solvers, and even atmospheric boundary layer solvers to provide turbulent inflow conditions for wind turbine simulations.[7,29]

B. Unstructured near-body finite volume solver: NSU3D

NSU3D is an unstructured mesh multigrid Unsteady Reynolds-averaged Navier-Stokes (URANS) solver developed for high-Reynolds number external aerodynamic applications. NSU3D employs a second-order accurate vertex-based discretization, where the unknown fluid and turbulence variables are stored at the vertices of the mesh, and fluxes are computed on faces delimiting dual control volumes, with each dual face being associated with a mesh edge. This discretization operates on hybrid mixed-element meshes, generally employing prismatic elements in highly stretched boundary layer regions, and tetrahedral elements in isotropic regions of the mesh. A single edge-based data structure is used to compute flux balances across all types of elements. The single-equation Spalart-Allmaras turbulence model[30] as well as a standard $k - \omega$ two-equation turbulence model[31] are available within the NSU3D solver.

The NSU3D solution scheme was originally developed for optimizing convergence of steady-state problems. The basic approach relies on an explicit multistage scheme which is preconditioned by a local block-Jacobi preconditioner in regions of isotropic grid cells. In boundary layer regions, where the grid is highly stretched, a line preconditioner is employed to relieve the stiffness associated with the mesh anisotropy.[32] An agglomeration multigrid algorithm is used to further enhance convergence to steady-state.[33,34] The Jacobi and line preconditioners are used to drive the various levels of the multigrid sequence, resulting in a rapidly converging solution technique.

For time-dependent problems, a second-order implicit backwards difference time discretization is employed, and the line-implicit multigrid scheme is used to solve the non-linear problem arising at each implicit time step. NSU3D has been extensively validated in stand-alone mode, both for steady-state fixed-wing cases, as a regular participant in the AIAA Drag Prediction workshop series,[35] as well as for unsteady aerodynamic and aeroelastic problems,[36] and has been benchmarked on large parallel computer systems.[37]

C. Overset Mesh Assembly: TIOGA

The overset grid connectivities are handled through the TIOGA interface (Topology Independent Overset Grid Assembler).[38] The TIOGA overset grid assembler relies on an efficient parallel implementation of Alternating Digital Trees (ADT)[39] for point-cell inclusion tests. Multiple grids are loaded in parallel and TIOGA computes the IBLANKing information required by the flow solver, as well as the cell donor-receptor information. We use the notation IBLANK which is an array of integers used to identify cell type in the overset framework. In this notation a fringe cell has IBLANK value of: IBLANK $= -1$, a hole cell: IBLANK $= 0$, and a field cell: IBLANK $= 1$. A hole cell occurs when an outer mesh overlaps an inner mesh with an enclosed surface (such as a geometry component). This hole cell does not contain a donor cell and is usually eliminated from the discretization. This is allowed because a collection of hole cells needs to be completely surrounded by fringe cells. This in turn means that the field cells only interact with fringe cells. If a field cell neighbors a hole cell then the overset connectivity fails. In order to interface TIOGA with a high-order method several callback functions need to be supplied. This allows high-order interpolation which is required to maintain overall high-order solution accuracy.[10]

V. Results

There are four sets of results that demonstrate the performance, efficiency, and capability of the DG AMR solver. The first two consist of the simulations of Ringleb flow and the Taylor-Green vortex to test the stand-alone capabilities. The last two utilize overset meshes and are solved in conjunction with a finite-volume near-body solver NSU3D for flow over a NACA 0015 wing and a NREL PhaseVI wind turbine.

A. Ringleb flow mesh resolution study

Ringleb flow is a two dimensional exact solution to the Euler equations.[40] This exact solution can be used to test the order of accuracy of the discretization through the use of a mesh resolution study. Figure 8 shows contours of density on a grid with a refined block in the middle of the domain.

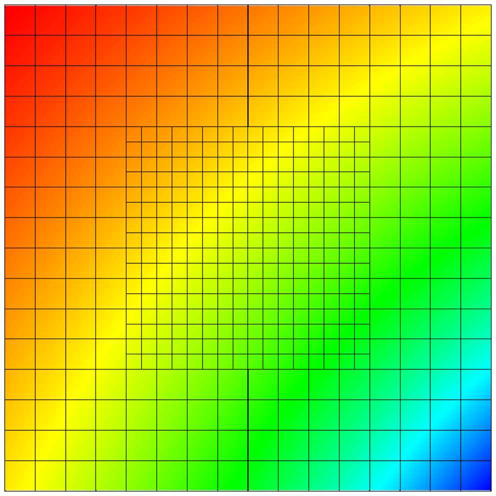


Figure 8: Contours of density for Ringleb flow with a refined block in the middle of the domain.

This case is used to determine if the correct order of accuracy is obtained when there are non-conforming elements. A mesh resolution is performed by refining the mesh uniformly in each direction. The refined block in the middle of the domain is also refined to maintain an element size corresponding to one half of the coarse element size. Figure 9 shows the L_2 error for density for polynomial degrees p=1, 3, and 5 for two level AMR meshes and a variable polynomial degree two level AMR mesh. Even order polynomial degrees are also tested and perform similarly to odd orders, although these cases are removed for brevity. The variable polynomial degree cases have a polynomial degree of p on the inner fine portion of the mesh and $p+1$ on the outer coarse portion of mesh. Reference lines are also provided for each polynomial degree using the function $C\Delta h^{p+1}$, where h is the cell size and the slope of $p+1$ gives the design order of accuracy. This case demonstrates that the design order of accuracy is obtained in the presence of hanging nodes or non-conforming elements within the mesh. It also shows that order of accuracy is dominated by the lowest polynomial degree as expected. However, having a higher polynomial degree on the coarser region of the mesh translates the curves downward resulting in overall higher accuracy.

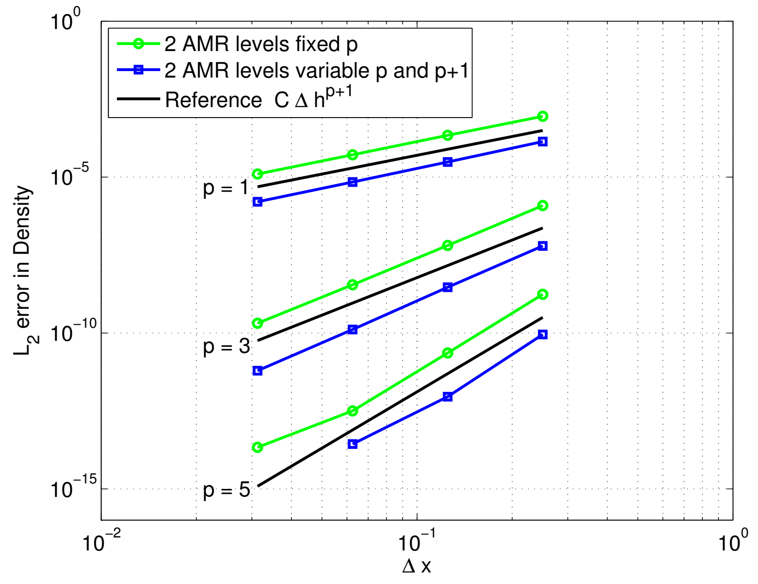


Figure 9: Mesh resolution study for Ringleb flow. L_2 error of density for $p = 1, 3, 5$ (second, fourth, and sixth order accuracy). Circle symbols are cases run with a single polynomial degree. Square symbols are for mixed polynomial degrees and as expected maintains the accuracy of the lower polynomial degree.

B. Taylor-Green vortex

The Taylor-Green vortex problem is a standard test case to study Direct Numerical Simulation (DNS) of turbulent flow in a periodic box.[41] The initial condition is defined analytically and is given as:

$$u = V_0 \sin\left(\frac{x}{L}\right) \cos\left(\frac{y}{L}\right) \cos\left(\frac{z}{L}\right),$$

$$v = -V_0 \cos\left(\frac{x}{L}\right) \sin\left(\frac{y}{L}\right) \cos\left(\frac{z}{L}\right),$$

$$w = 0,$$

$$p = p_0 + \frac{\rho_0 V_0^2}{16}\left(\cos\left(\frac{2x}{L}\right) + \cos\left(\frac{2y}{L}\right)\right)\left(\cos\left(\frac{2z}{L}\right) + 2\right)$$

where $L = 1$, $\rho_0 = 1.0$, $V_0 = 0.1$, and $P_0 = 1/\gamma$. For this case the Reynolds number is $Re = 1600$, the ratio of specific heats is $\gamma = 1.4$, the Prandtl number is $Pr = 0.71$, and the Mach number is $Ma = 0.1$. The domain size is $= \pi L \leq x, y, z \leq \pi L$ and the boundary conditions are periodic in all directions. The purpose of this study is to confirm that the viscous terms are implemented correctly in the new framework and to see how the results respond to AMR. Three polynomial degrees ($p = 1, 3, 7$) with AMR and without AMR (single grids) are simulated. Single grids include a fine mesh and a medium mesh corresponding to 256 degrees of freedom and 128 degrees of freedom respectively in each direction, where the degrees of freedom are $n_e(p + 1)$ and n_e denotes the number of elements in one dimension. The AMR cases have three levels of refinement where the coarse level has 64 degrees of freedom in each direction and the medium and fine levels correspond to the resolution of the corresponding single grid cases. All cases are performed at uniform fixed p-orders of accuracy, and compared with static coarse, medium and fine mesh solutions using the same p-order discretizations. The adaptive refinement process is based on a simple tagging criterion, where cells are refined if they contain a vorticity magnitude ω_c, as defined previously, greater than a specified level τ. Three tagging thresholds are examined in this study: $\tau = 0.3$, 0.5, or 1.0. After all the cells are tagged for refinement, neighboring cells sharing a face with a tagged cell are also tagged in order to provide a buffer region of refined cells. Each cell that is tagged is then refined recursively until it reaches the finest level. If a cell is not tagged for vorticity or is not a neighbor of a tagged cell then it is coarsened recursively until it reaches the coarsest level or until is borders on a refined cell, thus limiting cell resolution jumps to 2:1. It is important to realize that this tagging refinement criterion does not constitute an error estimate.

Rather we follow this simple feature-based approach which has proven to be successful for capturing vortical structures for rotorcraft and wind energy problems particularly in the HELIOS software framework,[42] with the understanding that a more fundamental error-based approach should be considered in future work.

The computed dissipation of kinetic energy, the number of degrees of freedom, the total CPU-hours, and the L_2 errors are shown in Figure 10 for the $p = 1$ simulations, Figure 11 for the $p = 3$ simulations, and Figure 12 for the $p = 7$ simulations. The dissipation in these cases is obtained by calculating the change of kinetic-energy at each time step:

$$\epsilon = -\frac{\mathrm{d}E_k}{\mathrm{d}t}$$

where E_k is the kinetic energy. For the Taylor-Green vortex problem as the simulation progresses the large scales are converted into smaller scales, which in turn converts kinetic energy into thermal internal energy. Therefore a simple way to measure the rate at which dissipation occurs is to take a time derivative of kinetic energy. The L_2 error is a running error over the time history of dissipation computed as:

$$E_t = \left(\sum_{j=0}^{t} \left(\epsilon_j^{fine} - \epsilon_j \right)^2 \right)^{1/2}$$

where ϵ^{fine} is the dissipation from the fine grid run and ϵ is the dissipation for the particular case under consideration. By computing the error in this manner, we assume that the finest grid solution at the given order of accuracy represents the highest accuracy attainable by the corresponding AMR solutions. In other words, the ideal AMR solution would provide the same accuracy or overall solution as a run performed on the finest uniform resolution available in the AMR process, but with lower cost due to the use of coarser cells in unrefined regions of the domain.

It is important to note that this is a difficult case for an adaptive mesh refinement approach to demonstrate significant computational savings over a fixed mesh approach because of the fact that the turbulence breaks down into uniformly small scales throughout the domain as the simulation progresses. The major savings for AMR occur in the beginning when the scales are large and at the end when the small scales have been mostly converted into internal energy. This makes it difficult to compare savings over a fixed grid due to the isotropic turbulent behavior and the arbitrary end point.

For the $p = 1$ case the highest threshold tagging (1.0) took the least amount of CPU-hours, followed by the static medium mesh case, the remaining AMR cases, while the fine mesh simulation required the most CPU-hours. With a tagging threshold of 0.3 and 0.5 the AMR cases were more accurate than the medium mesh case and cheaper in terms of CPU-hours than the fine mesh case. For the $p = 3$ case the static medium mesh case took the least amount of CPU-hours, the AMR cases fell in the middle, and the static fine mesh case took the most CPU-hours. With a tagging threshold of 0.3 the AMR case was more accurate than the static medium mesh case and cheaper in terms of CPU-hours than the static fine mesh case. For the $p = 7$ case the static medium mesh case took the least amount of CPU-hours, the AMR cases fell in the middle, and the static fine mesh case took the most CPU-hours. With a tagging threshold of 0.3 and 0.5 the AMR cases were more accurate than the static medium mesh case and cheaper in terms of CPU-hours than the static fine mesh case. The most accurate AMR case is the low tagging threshold of 0.3 and polynomial degree of $p = 7$, and although not as accurate as the fine mesh, is still about 20,000 CPU-hours cheaper.

A series of snapshots are shown in Figure 13 for a single slice located at $z = 1$ in the xy-plane. The solution evolves in the natural reading order of left to right and then top down. Vorticity magnitude contours are shown along with the AMR Octree mesh. Although this is a high-order simulation the visualization is performed only using linear interpolation. As the vorticity magnitude increases the mesh refines to those regions. Later in the simulation dissipation causes the vorticity to weaken and the mesh begins to coarsen. A volume rendering of this case is shown in Figure 14. The opacity is based on vorticity and the color on density.

Overall the AMR tracks the fine mesh dissipation reasonably well at reduced cost. The lower the tagging threshold the closer the AMR matches the fine mesh simulation. Also interesting to note is the maximum number of degrees of freedom shifts as the vorticity tagging threshold is decreased. For a tagging threshold of 1.0 it occurs near $t = 10$, for a threshold of 0.5 it occurs near $t = 12 - 13$, and for a threshold of 0.3 it occurs near $t = 14 - 18$ depending on the polynomial degree. Surprisingly, the AMR simulations start out with larger error than the fine mesh solutions, due to the coarse mesh levels used starting at the initial time. This indicates that either a finer initial mesh or a better refinement criterion is required in order to minimize the

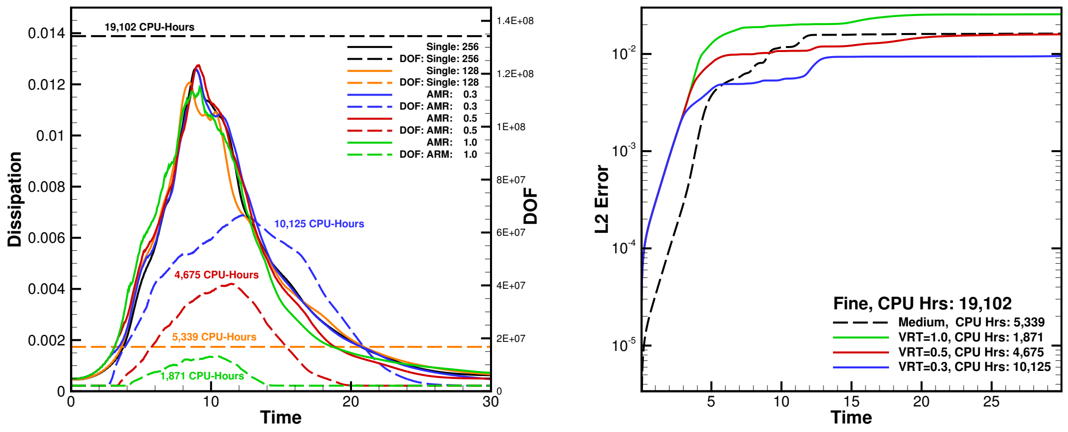


Figure 10: Taylor-Green vortex dissipation and degrees of freedom (left), L_2 error compared to fine mesh (right), polynomial degree: $p = 1$, single level medium and fine meshes, and three level AMR meshes with three different levels of vorticity tagging.

error at the initial times of the simulation. Additionally, the relative efficiency of the AMR cases compared to the static mesh cases improves for longer simulation times, due to the additional dissipation of the smallest scales for longer simulation times.

C. NACA 0015 wing

The first overset simulation consists of the study of flow over a wing based on a NACA 0015 airfoil. The NACA 0015 wing has been studied experimentally by McAlister and Takhashi.[43] Computational studies have been performed by Wissink,[42] Sitaraman and Baeder,[44] and by Hariharan and Sankar.[45]

For this case we demonstrate the capability of simulating unsteady, turbulent flow over a rectangular square-tipped lifting wing. The wing is based on a constant cross-section NACA0015 airfoil, and has a finite span of 6.6 chords. For this case the Mach number is 0.1235, the angle of attack is $\alpha = 12°$, and the Reynolds number is 1.5 million (based on chord length c).

NSU3D is used on the near-body solver and the AMR DG solver is used on the off-body solver. The overset mesh configuration is shown in Figure 15. This case is used to demonstrate the ability to combine a node-based finite-volume discretization with a cell-based, high-order, DG discretization in an overset framework, and to accurately capture wing-tip vortices. NSU3D solves the unsteady RANS equations closed by the Spalart-Allmaras with Rotation Correction (SA-RC) turbulence model.[46] NSU3D employs a steady-state implicit solver on a mesh with 4.4×10^6 nodes. The off-body AMR solver runs explicitly with RK4 at a non-dimensional time step of $\Delta t u_\infty / c = 0.036$. Figure 16 shows iso-contours of vorticity equal to 1.0 and 3.0 and slices of AMR grids all the way to a location 48 chords downstream.

Three ranges of polynomial degrees are studied in this case for the off-body DG discretization. The first is uniform $p = 1$, the second is $p = 2$ to attach to the near-body and $p = 3$ in the wake, and the third is $p = 3$ to attach to the near-body and grows to $p = 5$ in the wake. Off-body refinement occurs when vorticity magnitude w_c is greater than a tolerance of $\tau = 0.4$. The non-linear residual convergence and lift and drag coefficients are shown in Figure 17 for only the $p = 3 - 5$ case but show similar trends to the other cases. A residual reduction of four orders of magnitude for the near-body solver is obtained although full convergence to steady-state is inhibited due to the time dependent nature of the off-body flow solver. The lift and drag time histories exhibit small oscillations and a moving average with a window of 10,000 non-linear near-body solution updates is used to estimate the lift and drag as shown inTable 1.

The lift and drag for each of these cases shown in Table 1 are also compared to experimental data[43] and HELIOS simulation results taken from Ref.[28] The HELIOS simulations used the same near-body resolution as the current results. HELIOS has improved these results recently[47,48] but at the same time modified the near-body mesh so we compare to HELIOS results from Ref.[28]

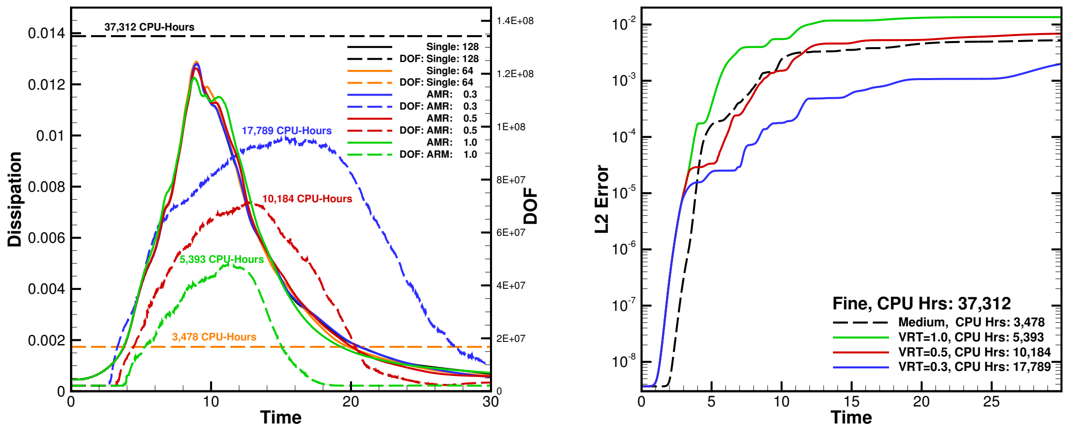


Figure 11: Taylor-Green vortex dissipation and degrees of freedom (left), L_2 error compared to fine mesh (right), polynomial degree: $p = 3$, single level medium and fine meshes, and three level AMR meshes with three different levels of vorticity tagging.

Table 1: Coefficient of lift and drag for NACA 0015 at $\alpha = 12°$ and $Re = 1.5 \times 10^6$

	Experiment	HELIOS	$p = 1$	$p = 2 - 3$	$p = 3 - 5$
C_L	1.04	0.91950	0.91790	0.91895	0.91881
C_D	0.049	0.0568	0.06223	0.06013	0.06019

Figure 18 shows wake profiles downstream from the wing at 1, 2, 4, and 6 chords. The three polynomial degree ranges are shown and the higher-polynomial degrees resolve the wakes more accurately compared to the experimental data. There is room for improvement but since there is not a large difference when moving from the $p = 2 - 3$ cases to the $p = 3 - 5$ cases we conclude that more resolution is needed on the near-body in order to improve the wakes.

Figure 19 shows the wake profiles of our best case ($p = 3 - 5$) compared to HELIOS and the experimental data at 2, 4, and 6 chords. Also, shown is a comparison between HELIOS and our overset case with $p = 3 - 5$ at 12 chords downstream. Our results show slightly better vortex resolution compared to the HELIOS results taken from Ref.[28]

D. NREL PhaseVI

A second set of overset simulations is performed on a NREL PhaseVI wind turbine. This is a standard test case for wind turbines due to the large amount of data collected in a full-scale wind tunnel experiment conducted on an instrumented wind turbine in the NFAC wind-tunnel at NASA Ames. The PhaseVI wind turbine has two blades with a radius of 5.029 m and a tip pitch of 3°. The turbine rotates at a rate of 72 rpm and has a Reynolds number of 2.5×10^6. NSU3D is used on the near-body and the AMR DG solver is used on the off-body. The near-body and off-body meshes are shown in Figure 20.

The near-body mesh contains 3.1×10^6 nodes and the surface mesh is shown in Figure 21. Two sets of simulations are studied; the first examines the accuracy of the simulations by performing a sweep over velocities and comparing the power and thrust to experimental data and simulations performed previously by other CFD solvers. The second demonstrates how the number of degrees of freedom responds to different polynomial degrees in the downstream wake region.

Figure 22 shows the number of degrees of freedom per time step for three simulations, excluding the near-body degrees of freedom since they are constant. The three simulations are at a velocity of 10 m/s but for different polynomial degrees in the wake. All cases attach to the near-body grid with $p = 1$ cells

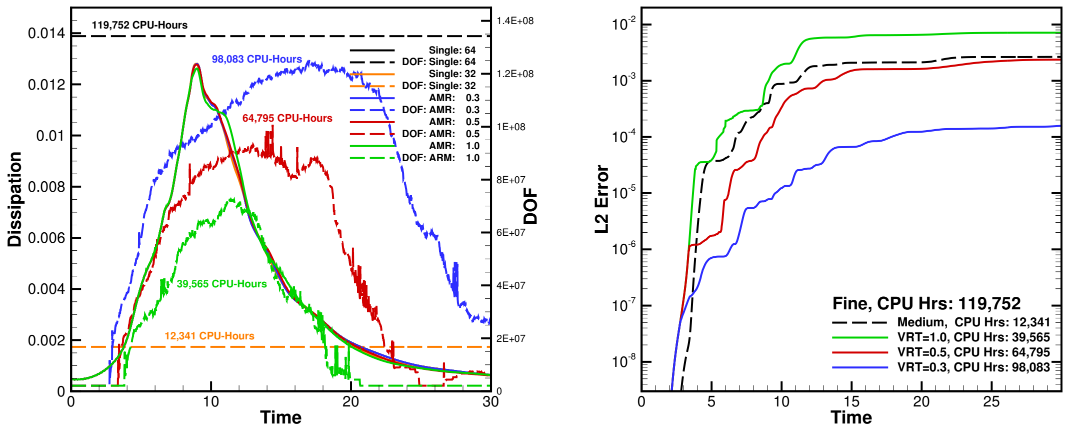


Figure 12: Taylor-Green vortex dissipation and degrees of freedom (left), L_2 error compared to fine mesh (right), polynomial degree: $p = 7$, single level medium and fine meshes, and three level AMR meshes with three different levels of vorticity tagging.

and the polynomial degree grows to $p = 1$, 2, or 4. In the wake region a base mesh consisting of 5 levels of refinement accounts for the relatively large number of degrees of freedom seen in the $p = 4$ case in the beginning of the simulation. This can be ignored since we are more interested in the increase of degrees of freedom or the rate of change created as the wake of the wind turbine convects downstream. All cases are run for 24 hours of wall-clock time on 576 cores (256 near-body and 320 off-body flow solver cores) and over that time frame more time steps are obtained for the $p = 4$ solution compared to the cases run with lower polynomial degrees. Also, the $p = 2$ case exhibits a similar trend and more time steps are obtained compared to the $p = 1$ case. More time steps are achieved for the higher polynomial degrees for two main reasons. The first is that the computational time per degree of freedom goes down slightly for higher polynomial degrees. Additionally, the increase in degrees of freedom is greater for the lower polynomial degrees. This may not hold if the simulation was continued further because there is a cross over point with $p = 1$ and $p = 2$ near time step 1300. To better understand these results a quartic polynomial is fitted to each case and the derivative of that polynomial is shown in Figure 22(b). This helps explain how the degrees of freedom are added to the simulation over time. For the $p = 1$ case the number of degrees of freedom is always increasing albeit more slowly over time. This could occur due to the wake dissipating over time and loss of resolution downstream from the turbine. The $p = 2$ case shows an opposite trend and the number of elements increase more rapidly than linearly until near the end of the simulation. The $p = 4$ case is also growing quicker than linearly but at a more reasonable increase. These results are qualitative in nature, and a more quantitative study is required to determine the accuracy for each case and to determine whether the decrease in degrees of freedom but increase in polynomial degree results in a more accurate simulation.

Figure 23 shows the off-body mesh used for the $p = 1$ case and the $p = 4$ case. The mesh is colored using polynomial degree and the $p = 4$ case only uses $p = 1 - 3$ polynomial degrees close to the near-body. The $p = 1$ case requires more cells to capture and track the wake compared to the $p = 4$ case. More cells in the off-body puts a burden on both the `p4est` library and the overset connectivity library TIOGA. Figure 24 shows contours of velocity magnitude for the $p = 1$ case and the $p = 4$ case with the mesh over-layed. The $p = 1$ wake remains stable as the dissipation keeps the vortices from breaking down. The wake in the $p = 4$ case breaks down into smaller scales. The plotting resolution on the $p = 4$ case is not fine enough to resolve the solution in each cell since we only plot $(p + 1)^d$ points in each cell.

To test the accuracy of the overset framework, 9 simulations are carried out with varying free-stream velocity. The off-body mesh uses a polynomial degree of $p = 1$ to attach to the near-body mesh and $p = 4$ in the wake. The domain size is 1000 meters in each direction and contains 10 Octrees in each direction and a maximum of 11 levels of refinement. The maximum level of refinement is only used in the $p = 1$ region while a maximum level of 8 is used in the $p = 4$ wake region. We study velocities 7 m/s to 15 m/s and report integrated forces. After 8 rotor revolutions of the wind turbine the forces are averaged and post processed to obtain power and thrust. These are shown in Figure 25 and compared to experimental data and previously reported computational results.[49] Good agreement is seen for velocities 7-10 m/s for all flow solvers and experimental data. For velocities 11-15 m/s our overset results appear to delay blade stalling compared to the HELIOS results and the experiment. Power in this range is overpredicted by approximately 10-20%, however our overset results are significantly more accurate than the stand-alone Overflow and NSU3D results. Conversely, thrust in the 11-15 m/s range is overpredicted by about 10%, which is similar to the Overflow results.

VI. Conclusions

In this work we have developed a high-order off-body DG AMR solver. This solver uses an Octree based AMR library called `p4est` which provides h-adaption capabilities. In addition the DG AMR solver incorporates a p-refinement capability which allows each cell to have a variable polynomial degree. The combined hp-refinement strategy allows for a highly efficient off-body solver for use in an overset framework. We have demonstrated the feasibility, accuracy, and capabilities of this off-body solver on two stand-alone cases which include Ringleb flow and Taylor-Green vortex. Also, we have demonstrated the ability to simulate two overset simulations. The first is a static overset NACA 0015 wing which compares well to experimental results and HELIOS. The second is a dynamic overset simulation of a PhaseVI wind turbine. Both of these overset cases show the added benefit of incorporating high-order accuracy in the off-body solver by increasing the accuracy and efficiency of the overall overset framework. Future work will focus on the development of an error-based refinement criterion for combined hp-refinement in wake regions and the validation of this approach for LES type simulations.

VII. Acknowledgments

This work was supported in part by ONR Grants N00014-14-1-0045 and N00014-16-1-2737 and by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under Award DE-SC0012671. Computer time was provided by the NCAR-Wyoming Supercomputing Center (NWSC) and University of Wyoming Advanced Research Computing Center (ARCC). The second author was supported in part by the NSF Blue Waters Graduate Fellowship as part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993).

References

[1] Murphy, K. J., Buning, P. G., Pamadi, B. N., Scallion, W. I., and Jones, K. M., "Overview of Transonic to Hypersonic Stage Separation Tool Development for Multi-Stage-To-Orbit Concepts," *AIAA Paper 2004-2595*.

[2] Brown, D. L. and Henshaw, W. D., "Overture: An Object-Oriented Framework for Solving Partial Differential Equations on Overlapping Grids," *Object Oriented Methods for Interoperable Scientific and Engineering Computing, SIAM*, 1999, pp. 245–255.

[3] Noack, R., "SUGGAR: a general capability for moving body overset grid assembly," *AIAA paper 2005-5117*.

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

[5] Buning, P. G., Gomez, R. J., and Scallion, W. I., "CFD Approaches for Simulation of Wing-Body Stage Separation," *AIAA Paper 2004-4838*.

[6] Biedron, R. T. and Thomas, J. L., "Recent Enhancements to the FUN3D Flow Solver for Moving-Mesh Applications," *AIAA Paper 2009-1360*.

[7] Sitaraman, J., Mavriplis, D. J., and Duque, E. P., "Wind Farm simulations using a Full Rotor Model for Wind Turbines," *AIAA Paper 2014-1086*, 2015/05/29 2014.

[8] Sankaran, V., Sitaraman, J., Wissink, A., Datta, A., Jayaraman, B., Potsdam, M., Mavriplis, D., Yang, Z., O'Brien, D., Saberi, H., et al., "Application of the Helios Computational Platform to Rotorcraft Flowfields," *AIAA paper 2010-1230*.

[9] Galbraith, M. C., Benek, J. A., Orkwis, P. D., and Turner, M. G., "A Discontinuous Galerkin Chimera scheme," *Computers & Fluids*, Vol. 98, No. 0, 2014, pp. 27 – 53.

[10] Brazell, M. J., Sitaraman, J., and Mavriplis, D. J., "An overset mesh approach for 3D mixed element high-order discretizations," *Journal of Computational Physics*, Vol. 322, 10 2016, pp. 33–51.

[11] Zhang, B. and Liang, C., "A Simple, Efficient, High-Order Accurate Sliding-mesh Interface Approach to FR/CPR Method on Coupled Rotating and Stationary Domains," *AIAA Paper 2015-1742*, 2015/06/01 2015.

[12] Merrill, B. E., Peet, Y. T., Fischer, P. F., and Lottes, J. W., "A spectrally accurate method for overlapping grid solution of incompressible Navier–Stokes equations," *Journal of Computational Physics*, Vol. 307, 2 2016, pp. 60–93.

[13] Burstedde, C., Wilcox, L. C., and Ghattas, O., "`p4est`: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees," *SIAM Journal on Scientific Computing*, Vol. 33, No. 3, 2011, pp. 1103–1133.

[14] Isaac, T., Burstedde, C., Wilcox, L. C., and Ghattas, O., "Recursive algorithms for distributed forests of octrees," *SIAM Journal on Scientific Computing*, Vol. 37, No. 5, 2015, pp. C497–C531.

[15] 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.

[16] Kirby, A. C., Mavriplis, D. J., and Wissink, A. M., "An Adaptive Explicit 3D Discontinuous Galerkin Solver for Unsteady Problems," 2015, AIAA Paper 2015-3046, 22nd AIAA Computational Fluid Dynamics Conference, Dallas, TX, June 2015.

[17] Lax, P. D., "Weak solutions of nonlinear hyperbolic equations and their numerical computation," *Communications on Pure and Applied Mathematics*, Vol. 7, No. 1, 1954, pp. 159–193.

[18] Hartmann, R. and Houston, P., "An Optimal Order Interior Penalty Discontinuous Galerkin Discretization of the Compressible Navier-Stokes Equations," *J. Comput. Phys. (USA)*, Vol. 227, No. 22, 2008/11/20, pp. 9670 – 85.

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

[20] Butcher, J. C., "Numerical methods for ordinary differential equations," 2016.

[21] Gottlieb, S. and Shu, C.-W., "Total variation diminishing Runge-Kutta schemes," *Mathematics of computation of the American Mathematical Society*, Vol. 67, No. 221, 1998, pp. 73–85

[22] Black, K., "Spectral element approximation of convection–diffusion type problems," *Applied Numerical Mathematics*, Vol. 33, No. 1, 2000, pp. 373–379.

[23] Cockburn, B. and Shu, C.-W., "The Runge–Kutta Discontinuous Galerkin Method for Conservation Laws V," *Journal of Computational Physics*, Vol. 141, No. 2, 1998, pp. 199–224.

[24] 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.

[25] 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, 5 2012, pp. 86–93.

[26] Bolis, A., Cantwell, C., Kirby, R., and Sherwin, S., "From h to p efficiently: optimal implementation strategies for explicit timedependent problems using the spectral/hp element method," *International journal for numerical methods in fluids*, Vol. 75, No. 8, 2014, pp. 591–607 1097–0363.

[27] Rudi, J., Malossi, A. C. I., Isaac, T., Stadler, G., Gurnis, M., Staar, P. W. J., Ineichen, Y., Bekas, C., Curioni, A., and Ghattas, O., "An Extreme-scale Implicit Solver for Complex PDEs: Highly Heterogeneous Flow in Earth's Mantle," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, ACM, New York, NY, USA, 2015, pp. 5:1–5:12.

[28] Wissink, A., Sitaraman, J., Sankaran, V., Mavriplis, D., and Pulliam, T., "A Multi-Code Python-Based Infrastructure for Overset CFD with Adaptive Cartesian Grids," *AIAA Paper 2008-927*, 2015/05/27 2008.

[29] Kirby, A. C., Brazell, M. J., Wang, Z., Roy, R., Ahrabi, B. R., Mavriplis, D. J., Stoellinger, M. k., and Sitaraman, J., "Wind Farm Simulations Using an Overset hp-Adaptive Approach with Blade-Resolved Turbine Models," AIAA Paper 2017-...., 23rd AIAA Computational Fluid Dynamics Conference, Denver, CO., June 2017.

[30] Spalart, P. R. and Allmaras, S. R., "A one-equation turbulence model for aerodynamic flows," *La Recherche A erospatiale*, Vol. Vol. 1, 1994, pp. 5–21., No. 5-21, 1994.

[31]Wilcox, D. C., “Reassessment of the scale-determining equation for advanced turbulence models,” *AIAA journal*, Vol. 26, No. 11, 1988, pp. 1299–1310.

[32]Mavriplis, D. J., “Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes,” *Journal of Computational Physics*, Vol. 145, No. 1, 1998, pp. 141–165.

[33]Mavriplis, D. and Venkatakrishnan, V., “A unified multigrid solver for the Navier-Stokes equations on mixed element meshes,” *International Journal of Computational Fluid Dynamics*, Vol. 8, No. 4, 1997, pp. 247–263.

[34]Mavriplis, D. and Pirzadeh, S., “Large-scale parallel unstructured mesh computations for 3D high-lift analysis,” *AIAA Paper 1999-537*, 2015/06/01 1999.

[35]Mavriplis, D. J., “Third Drag Prediction Workshop Results Using the NSU3D Unstructured Mesh Solver,” *Journal of Aircraft*, Vol. 45, No. 3, 2015/06/01 2008, pp. 750–761.

[36]Yang, Z. and Mavriplis, D. J., “Higher-Order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes,” *AIAA Journal*, Vol. 45, No. 1, 2007, pp. 138–150.

[37]Mavriplis, D. J., Aftosmis, M. J., and Berger, M., “High Resolution Aerospace Applications using the NASA Columbia Supercomputer,” *International Journal of High Performance Computing Applications*, Vol. 21, No. 1, 2007, pp. 106–126.

[38]Brazell, M. J. and Mavriplis, D. J., *High-Order Discontinuous Galerkin Mesh Resolved Turbulent Flow Simulations of a NACA 0012 Airfoil (Invited)*, American Institute of Aeronautics and Astronautics, 2015/05/27 2015.

[39]Bonet, J. and Peraire, J., “An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems,” *International Journal for Numerical Methods in Engineering*, Vol. 31, No. 1, 1991, pp. 1–17.

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

[41]Taylor, G. I. and Green, A. E., “Mechanism of the Production of Small Eddies from Large Ones,” Vol. 158, No. 895, 1937, pp. 499–521.

[42]Wissink, A., “An Overset Dual-Mesh Solver for Computational Fluid Dynamics,” 2012, 7th International Conference on Computational Fluid Dynamics, Paper ICCFD7-1206, Hawaii.

[43]McAlister, K. W. and Takahashi, R., “NACA 0015 wing pressure and trailing vortex measurements,” Tech. rep., DTIC Document, 1991.

[44]Sitaraman, J. and Baeder, J. D., “Evaluation of the wake prediction methodologies used in CFD based rotor airload computations,” 2006, AIAA Paper 2006-3472, AIAA 24th Conference on Applied Aerodynamics, Washington, DC.

[45]Hariharan, N., “Rotary-Wing Wake Capturing: High-Order Schemes Toward Minimizing Numerical Vortex Dissipation,” *Journal of aircraft*, Vol. 39, No. 5, 2002, pp. 822–829.

[46]Shur, M. L., Strelets, M. K., Travin, A. K., and Spalart, P. R., “Turbulence Modeling in Rotating and Curved Channels: Assessing the Spalart-Shur Correction,” *AIAA Journal*, Vol. 38, No. 5, 2017/04/26 2000, pp. 784–792.

[47]Sitaraman, J., Floros, M., Wissink, A., and Potsdam, M., “Parallel domain connectivity algorithm for unsteady flow computations using overlapping and adaptive grids,” *Journal of Computational Physics*, Vol. 229, No. 12, 2010, pp. 4703 – 4723.

[48]Kamkar, S. J., Wissink, A. M., Sankaran, V., and Jameson, A., “Feature-driven Cartesian adaptive mesh refinement for vortex-dominated flows,” *Journal of Computational Physics*, Vol. 230, No. 16, 7 2011, pp. 6271–6298.

[49]Potsdam, M. and Mavriplis, D. J., “Unstructured Mesh CFD Aerodynamic Analysis of the NREL Phase VI Rotor,” *AIAA Paper 2009-1221*.

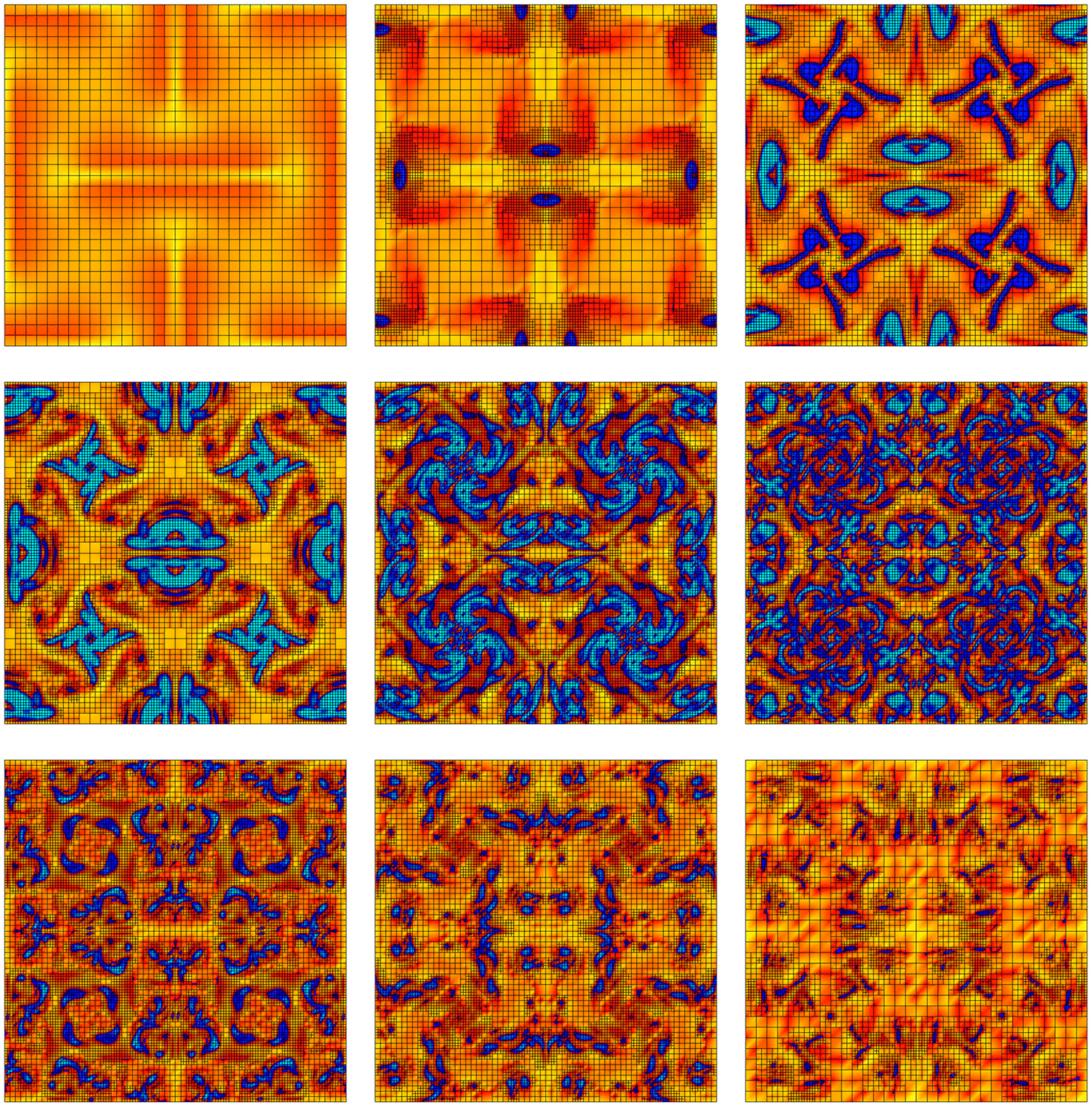


Figure 13: Snapshots of the Taylor-Green Vortex with three levels of AMR. Contours of vorticity magnitude. Time history goes in order of natural reading order.

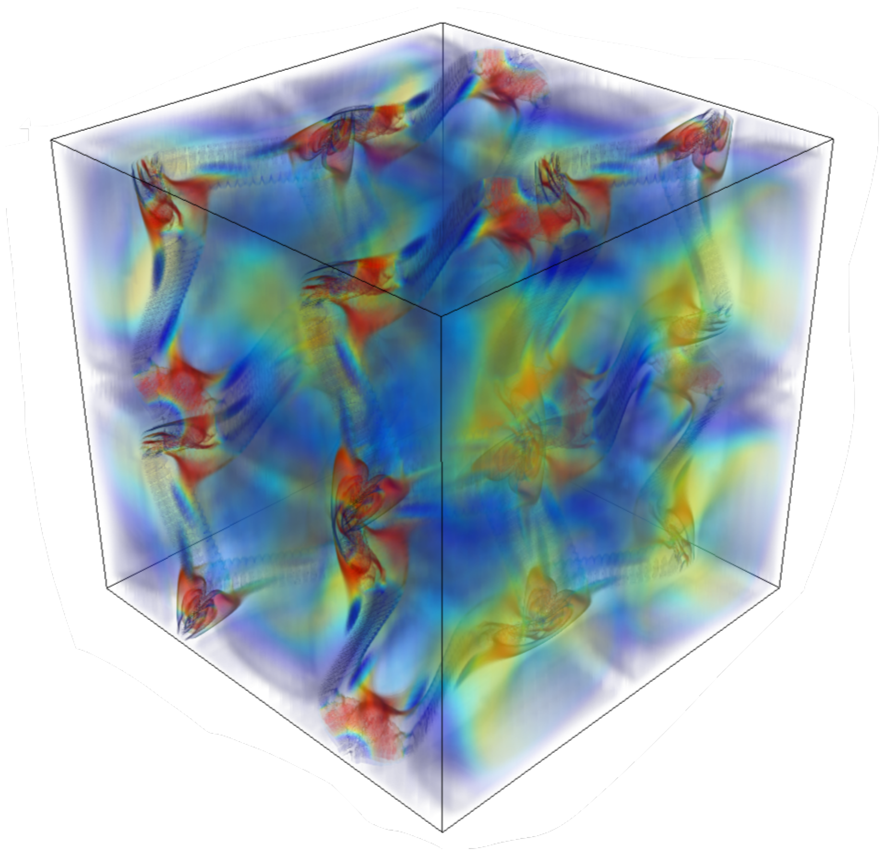


Figure 14: Taylor-Green vortex volume rendering, fourth order accuracy ($p = 3$), opacity based on vorticity and colored by density.

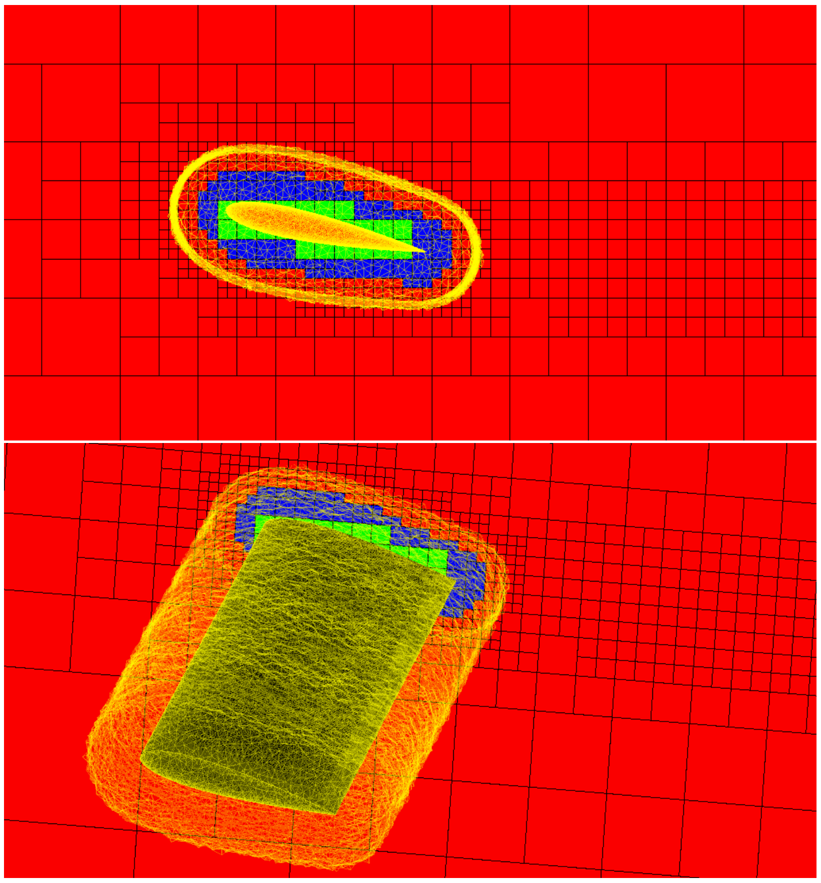


Figure 15: Overset meshes for the near-body and off-body NACA 0015 wing.

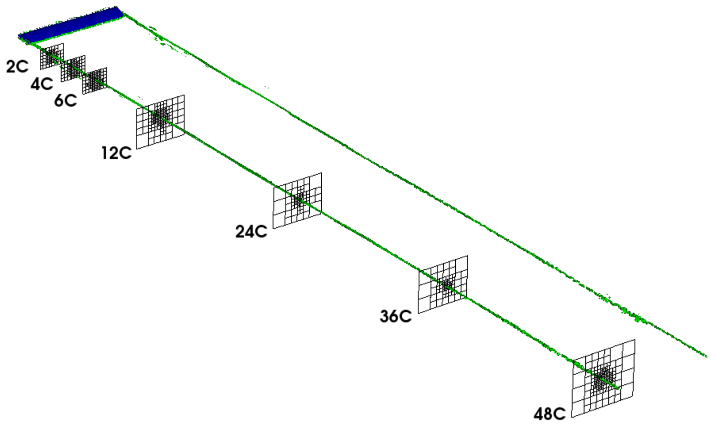


Figure 16: Iso contours of vorticity on NACA 0015 wing for $p = 2 - 3$.

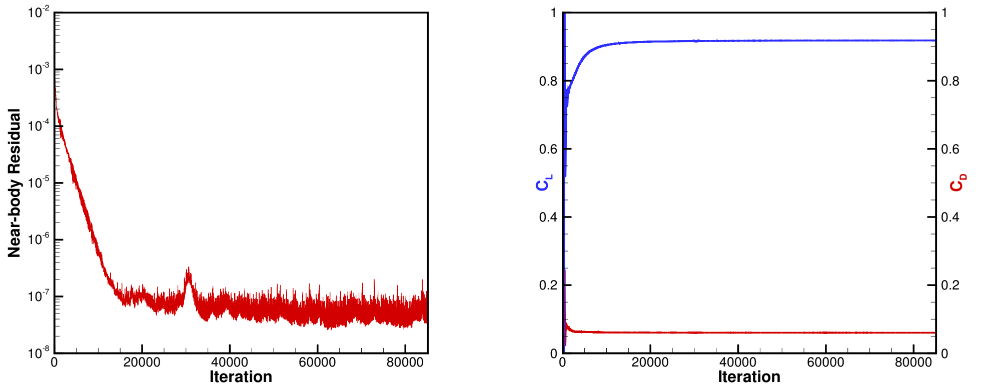
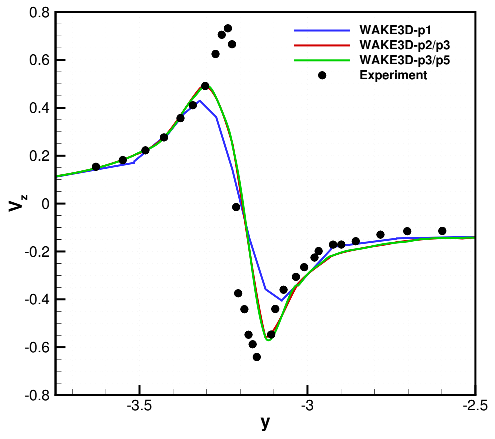
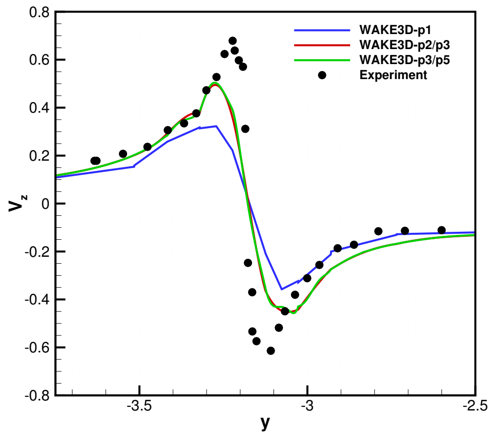


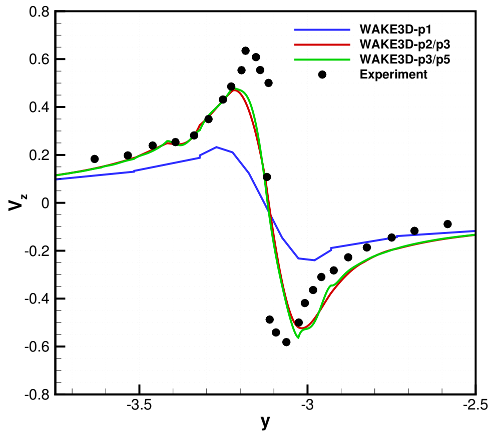
Figure 17: Residual convergence (left) and coefficient of lift and drag convergence for NACA 0015 wing for $p = 3 - 5$.



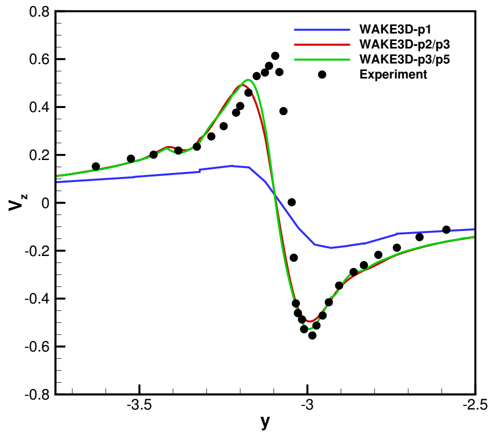
(a) 1 chord downstream



(b) 2 chords dowstream

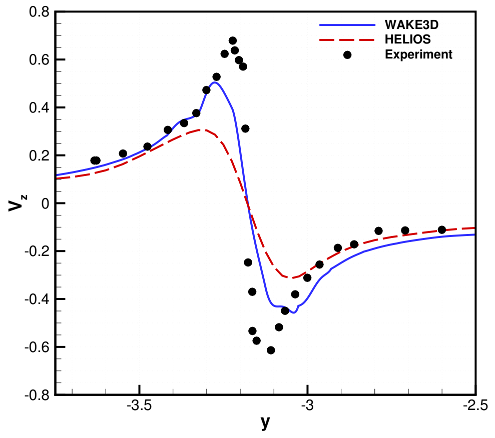


(c) 4 chords downstream

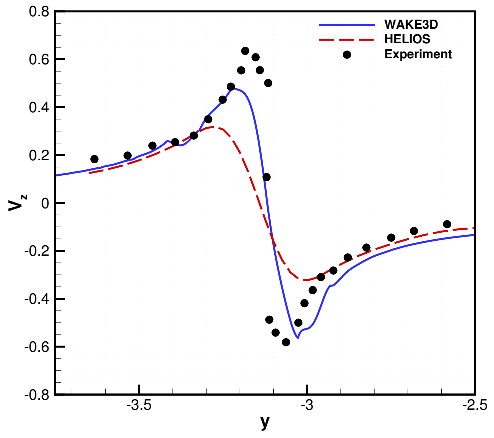


(d) 6 chords downstream

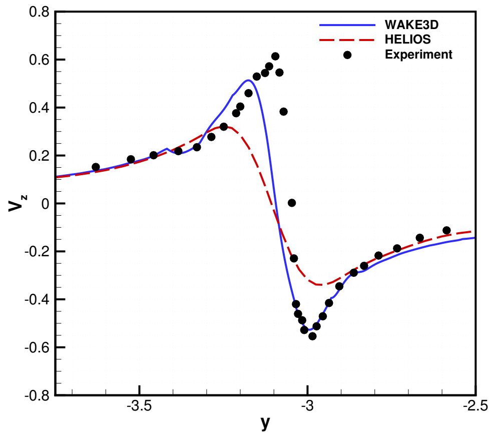
Figure 18: Velocity wake profile downstream from wing for polynomial degrees $p = 1$, $p = 2-3$, and $p = 3-5$.



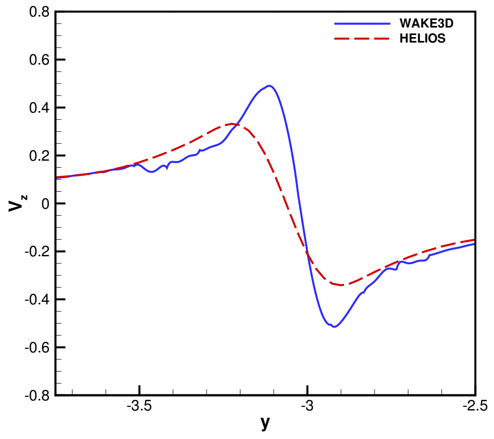
(a) 2 chords downstream



(b) 4 chords dowstream



(c) 6 chords downstream



(d) 12 chords downstream

Figure 19: Velocity wake profile downstream from wing for polynomial degrees $p = 3 - 5$ and comparison to HELIOS.

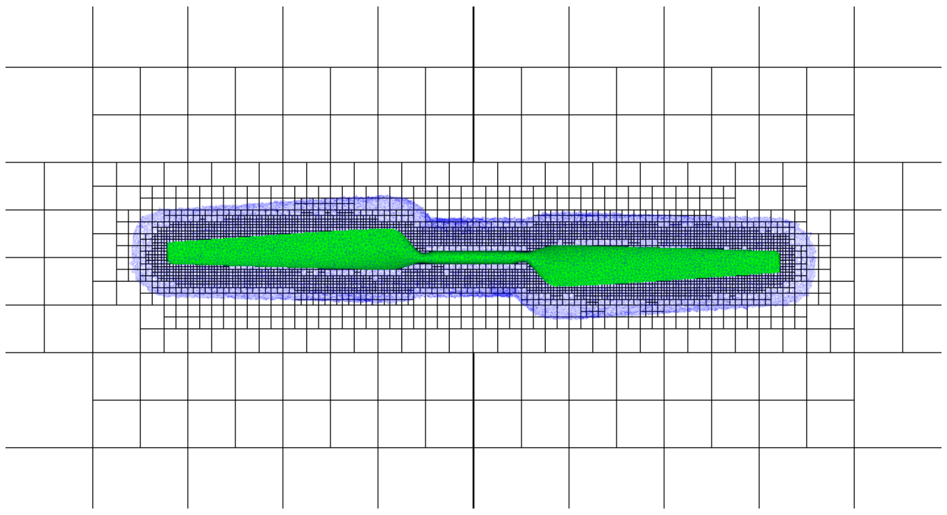


Figure 20: Overset mesh configuration for near-body and off-body for the PhaseVI wind turbine.

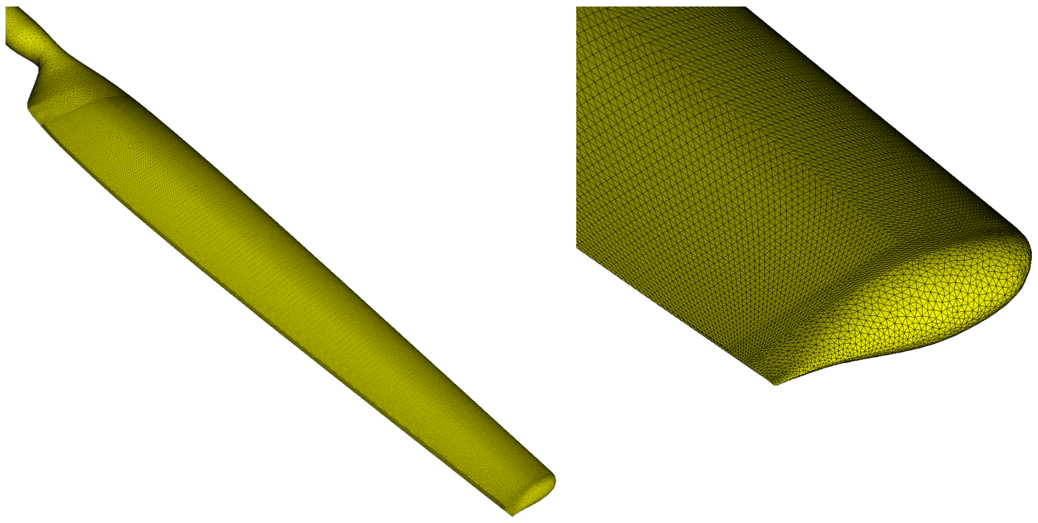


Figure 21: Near-body surface mesh for the PhaseVI wind turbine.

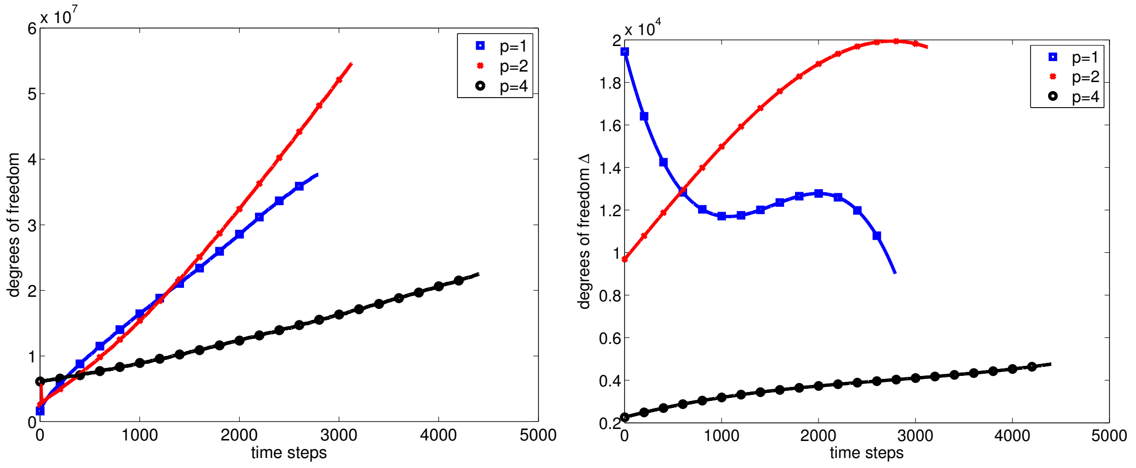


Figure 22: Degrees of freedom per time step (left) and derivative of quartic fit (right) for PhaseVI wind turbine with freestream velocity 10 m/s.

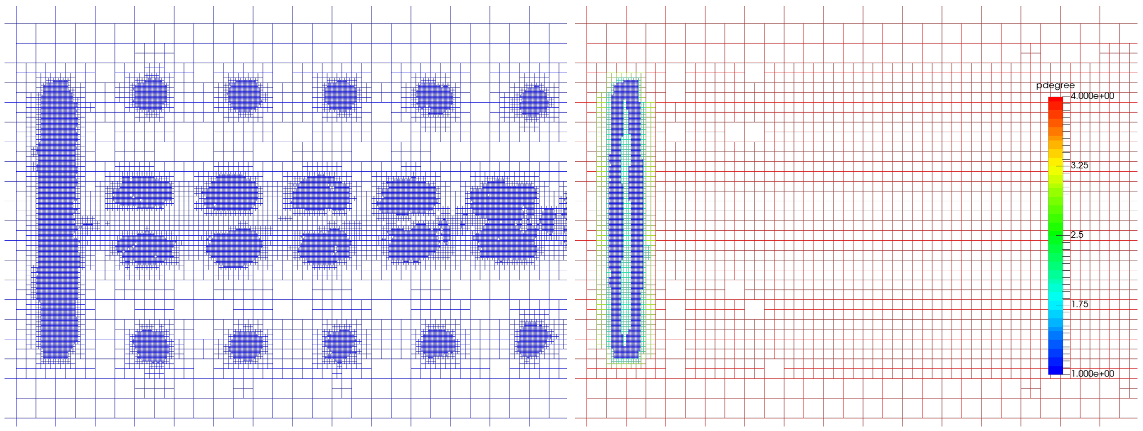


Figure 23: PhaseVI wind turbine off-body mesh for $p = 1$ case (left) and $p = 4$ case (right) colored by polynomial degree.

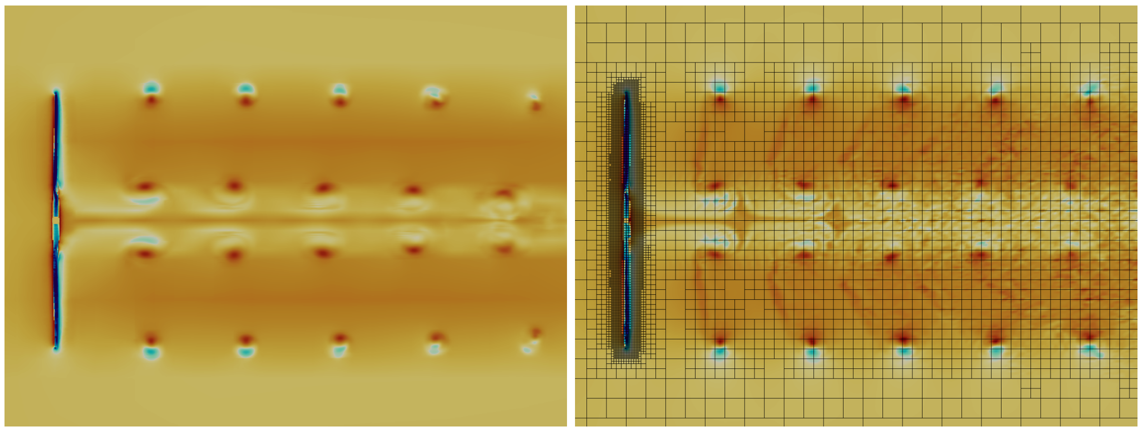


Figure 24: Contours of velocity magnitude for the PhaseVI wind turbine. Left image is $p = 1$ case and the right image is $p = 4$ case with mesh overlayed.

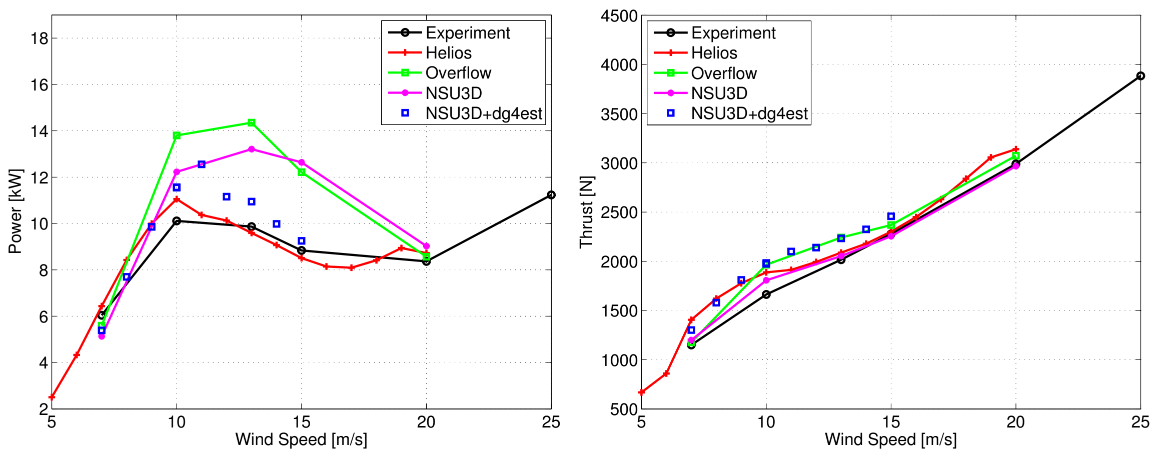


Figure 25: NREL PhaseVI wind turbine power sweep (left) and thrust sweep (right).