SAND2019-3687R

EXASCALE COMPUTING PROJECT

ECP-U-2019-xxx

Nalu-Wind and OpenFAST: A high-fidelity modeling and simulation environment for wind energy

WBS 2.2.2.01, Milestone **ECP-Q2-FY19**

Shreyas Ananthan, NREL
Luigi Capone, ORNL
Marc Henry de Frahan, NREL
Jonathan Hu, SNL
Jeremy Melvin, UTA
James Overfelt, SNL
Ashesh Sharma, NREL
Jay Sitaraman, Parallel Geometric Algorithms LLC
Katarzyna Swirydowicz, NREL
Stephen Thomas, NREL
Ganesh Vijayakumar, NREL
Alan Williams, SNL
Shashank Yellapantula, NREL
Michael Sprague, NREL

March 29, 2019

U.S. DEPARTMENT OF ENERGY | Office of Science

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energys Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nations exascale computing imperative.
This work was authoredin partby the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308.
This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

ECP-U-2019-xxx

ECP Milestone Report
Nalu-Wind and OpenFAST: A high-fidelity modeling and simulation environment for wind energy
WBS 2.2.2.01, Milestone ECP-Q2-FY19

Office of Advanced Scientific Computing Research
Office of Science
US Department of Energy

Office of Advanced Simulation and Computing
National Nuclear Security Administration
US Department of Energy

March 29, 2019

ECP Milestone Report
Nalu-Wind and OpenFAST: A high-fidelity modeling and simulation environment for wind energy
WBS 2.2.2.01, Milestone ECP-Q2-FY19

APPROVALS

Submitted by:

29 March 2019

Michael A. Sprague
ECP-Q2-FY19

Date

Approval:

Thomas Evans
ORNL

Date

REVISION LOG

Version	Creation Date	Description	Approval Date
1.0	2019-03-29	Original	

EXECUTIVE SUMMARY

The goal of the ExaWind project is to enable predictive simulations of wind farms comprised of many megawatt-scale turbines situated in complex terrain. Predictive simulations will require computational fluid dynamics (CFD) simulations for which the mesh resolves the geometry of the turbines and captures the rotation and large deflections of blades. Whereas such simulations for a single turbine are arguably petascale class, multi-turbine wind farm simulations will require exascale-class resources.

The primary physics codes in the ExaWind project are Nalu-Wind, which is an unstructured-grid solver for the acoustically incompressible Navier-Stokes equations, and OpenFAST, which is a whole-turbine simulation code. The Nalu-Wind model consists of the mass-continuity Poisson-type equation for pressure and a momentum equation for the velocity. For such modeling approaches, simulation times are dominated by linear-system setup and solution for the continuity and momentum systems. For the ExaWind challenge problem, the moving meshes greatly affect overall solver costs as reinitialization of matrices and recomputation of preconditioners is required at every time step.

This milestone represents the culmination of several parallel development activities towards the goal of establishing a full-physics simulation capability for modeling wind turbines operating in turbulent atmospheric inflow conditions. The *demonstration simulation* performed in this milestone is the first step towards the "ground truth" simulation and includes the following components: neutral atmospheric boundary layer inflow conditions generated using a *precursor simulation*, a hybrid RANS/LES simulation of the wall-resolved turbine geometry, hybridization of the turbulence equations using a blending function approach to transition from the atmospheric scales to the blade boundary layer scales near the turbine, fluid-structure interaction (FSI) that accounts for the complete set of blade deformations (bending, twisting and pitch motion, yaw and tower displacements) by coupling to a comprehensive turbine dynamics code (OpenFAST). The use of overset mesh methodology for the simulations in this milestone presents a significant deviation from the previous efforts where a sliding mesh approach was employed to model the rotation of the turbine blades. The choice of overset meshes was motivated by the need to handle arbitrarily large deformations of the blade and to allow for blade pitching in the presence of a controller and the ease of mesh generation compared to the sliding mesh approach. FSI and the new timestep algorithm used in the simulations were developed in partnership with the A2e High-Fidelity Modeling project. The individual physics components were verified and validated (V%V) through extensive code-to-code comparisons and with experiments where possible. The detailed V&V efforts provide confidence in the final simulation where these physics models were combined together even though no detailed experimental data is available to perform validation of the final configuration. Taken together, this milestone successfully demonstrated the most advanced simulation to date that has been performed with Nalu-Wind.

The ExaWind team continues to make strides in improving the performance of Nalu-Wind and prepare it for next-generation platforms, in particular, execution on CPU-GPU architectures. Several aspects of the linear solver stack (both in Trilinos and HYPRE) have been ported to GPUs and have shown significant performance gains, compared to the traditional MPI only parallelism. These improvements will be critical in meeting the time per timestep requirements of the ExaWind Challenge Problem. The team has begun the process of converting the discretization operators (*Nalu-Wind kernels*) to be capable of running on GPUs using the Kokkos abstraction layer. The current status and future plans have been documented in this report.

The demonstration simulation is just the first step towards the goal of simulating multiple turbines operating in a wind farm. The team aims to refine and improve the physics models as well as pursue novel algorithms and solver approaches to improve the accuracy and performance of the Nalu-Wind code. Turbulence modeling plays a crucial role in the accuracy of the predictions and will remain a continued focus for the remainder of the project. Bridging the turbulence length scales relevant to the atmospheric boundary layer with that of the blade boundary layer remains an open research question. Another aspect that deserves further attention is the need to run at higher Courant numbers for practical simulations and the accuracy trade-off associated with it in the LES regimes. Development of new solver approaches that can handle the large timestep requirements without sacrificing accuracy will be another area of research for the ExaWind project. These features will be the focus of future milestones.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The ultimate goal of the ExaWind project is to enable scientific discovery through predictive simulations of wind farms comprised of many megawatt-scale turbines situated in complex terrain. Predictive simulations will require computational fluid dynamics (CFD) simulations for which the mesh resolves the geometry of the turbines and captures the rotation and large deflections of blades. Whereas such simulations for a single turbine are arguably petascale class, multi-turbine wind farm simulations will require exascale-class resources [37].

The objective of this milestone is to document the establishment of complete *baseline* set of modeling and simulation capabilities in the ExaWind software stack for body-resolved turbine simulations. As described in the 2015 Atmosphere to electrons strategic planning meetings [15], those capabilities include:

- blade structural dynamics model that includes complicated composite structure and large, nonlinear deflections that can address, e.g., bend-twist coupling,
- blade/nacelle/tower conforming fluid meshes that deform with large blade deflection,
- overset/sliding fluid mesh capabilities that accommodate the rotor rotation, and nacelle yaw,
- fluid meshes that accommodate complex terrain,
- hybrid LES/RANS turbulence modeling, where LES captures the dynamics of wakes and RANS captures sufficiently the boundary layer at the blade surface, and
- coupling of mean flow and turbulence from the meso-scale via numerical weather prediction or precursor atmospheric-boundary-layer simulations.

The primary physics codes in the open-source ExaWind software stack are Nalu-Wind and OpenFAST. Nalu-Wind is an low-Mach-number (acoustically incompressible) computational fluid dynamics (CFD) code. It is a wind-energy-focused variant of the Nalu code that was developed at Sandia National Laboratories (SNL). The Nalu-Wind code is closely tied to the Trilinos libraries, leveraging greatly the Sierra Toolkit (STK), the Kokkos abstraction layer for parallel-performance portability, and the linear-system solver libraries (MueLu, Belos, Tpetra). Through ExaWind, Nalu-Wind is also linked to the HYPRE linear-system solver stack and to the Topology Independent Overset Grid Assembler (TIOGA) for overset meshes. OpenFAST is a whole-turbine simulation code developed at the National Renewable Energy Laboratory (NREL) that includes models for nonlinear deflections of blades, the control system, and the tower.

ExaWind is operated in close collaboration with the Atmosphere to electrons (A2e) High-Fidelity-Modeling (HFM) project, which is supported by the Department of Energy Wind Energy Technologies Offices. Researchers in the ExaWind and A2e-HFM projects have been working since 2016 to implement in Nalu-Wind the necessary modeling capabilities for wind energy, including fluid-structure-interaction coupling with the OpenFAST code. Whereas Nalu was a research code focused on turbulent flow problems that are appropriately modeled with large-eddy simulation (LES) techniques, Nalu-Wind requires a significantly augmented set of models for the complex multi-scale flow physics of wind turbines and wind farms. In this report we describe the complete set of baseline modeling capabilities and the path moving forward. For completeness, we include a description of the FSI modeling efforts that were funded by the A2e-HFM program.

Whereas initial ExaWind modeling efforts focused on a sliding-mesh interface to model wind turbine rotor motion [9], the project has shifted to overset meshes for handling mesh motion and to simplify mesh creation. Advantages of the overset meshing approach include:

- Meshes for each component (e.g., blades, nacelle, tower) can be created largely independently,
- Mesh creation is simplified in that the volume mesh can be readily extruded from the surface mesh of the underlying bode (e.g., blade), and does not need to conform to an outer boundary,
- Provides an avenue to employ Nalu-Wind as a "near-body solver" that is coupled to a structure-grid background solver (e.g., AMReX).

There are additional challenges with overset meshes, including additional search requirements and special handling of constraint equations in the linear systems.

The milestone report is organized as follows. Section 2 describes the milestone and gives an overview of completion accomplishments. Section 3 describes the new overset-mesh capabilities in Nalu-Wind, and Section 4 describes the new time-stepping algorithm that enables stable URANS simulations at large Courant numbers. Section 5 describes the latest URANS and hybrid-URANS-LES capabilities and demonstration results. Section 6 gives an update on the linear-system solver stack including a results from a comparison of segregated- and monolithic-velocity momentum solves. Section 7 describes the new fluid-structure-interaction coupling with OpenFAST (work funded under A2e HFM). Finally, Section 8 describes the status and next steps in preparing the Nalu-Wind software for next-generation platforms (NGP). Concluding remarks are in Section 10.

2. MILESTONE DESCRIPTION

In this section, we provide the approved milestone description and execution plan followed by a brief description of how the milestone was completed. Details regarding completion are included in the following sections.

2.1 DESCRIPTION

This multidisciplinary project to simulate the complex flow physics of whole wind plants embodies a systematic development of the modeling capability as well as computational performance and scalability, required for effective exascale simulations. This milestone builds on the demonstration simulation of a megawatt-class wind turbine completed in FY18, and strives to establish a baseline single-turbine simulation with a complete set of verified modeling capabilities deemed necessary for predictive wind farm simulations, e.g., hybrid RANS/LES turbulence modeling, fluid-structure interaction (FSI). Completing this milestone with FSI will be accomplished through collaboration with the ExaWind sister project A2e High Fidelity Modeling, which is funded under the DOE EERE Wind Energy Technologies Office (WETO). A significant portion of the work in this milestone will be to prepare the Nalu-Wind software stack for GPU-based simulations.

2.2 EXECUTION PLAN

1. Complete implementation and verification of a baseline capabilities, including a hybrid-RANS/LES turbulence model and fluid-structure-interaction.
2. Prototype a segregated momentum solve and evaluate solve-time performance and robustness on baseline ABL simulation, RANS flat-plate simulation, and a turbine simulation with mesh motion.
3. Pursue simulation-time allocation on a GPU-based system and perform preliminary performance studies focused on linear-system solver stack.
4. Demonstrate execution of Nalu-Wind kernel assembly on GPU architectures.
5. Establish baseline performance of turbine simulation with overset mesh capability and create plan for improving on next-generation platform.
6. Establish a Figure of Merit measurement for turbine simulation with a complete set of baseline capabilities and under atmospheric turbulent inflow.

Completion Criteria: Technical report describing the milestone accomplishment as well as a highlight slide summarizing those accomplishments.

2.3 OVERVIEW OF MILESTONE COMPLETION

The following is a concise description of how each of the items in Section 2.2 was satisfied for milestone completion.

1. A complete set of physics models were implemented in Nalu-Wind, including turbulence models (see §5), fluid-structure interaction (see §7, a new time-stepping algorithm that enable the required large time-step sized in RANS regions of the flow (see §4), and overset meshed, which enable the generalized deformations of the fluid mesh (see §3).

2. A segregated-momentum-system solver was implemented and tested in HYPRE, and we demonstrated significant speedup for the ABL, flat-plate, and full-turbine simulations over a monolithic-momentum-system solver (see §6.1).

3. §6.2

4. §8

5. Overset-mesh capability was implemented in Nalu-Wind using a third-party library, TIOGA, that allows Nalu-Wind to handle complex geometries and arbitrary mesh motion. These capabilities were verified and validated on canoncial problems (§3) and rotor simulations (§4.3.1, §9). Section 8.4 documents the NGP plans for the overset mesh capability.

6. We did not establish a Figure of Merit measurement because the ExaWind project shifted from being a KPP-1 project to a KPP-2 project.

3. OVERSET MESH METHODOLOGY

Accurate simulation of wind farm wakes under realistic atmospheric inflow conditions and complex terrain requires modeling a wide range of length and time scales. The computational domain can span several kilometers while requiring mesh resolutions in $\mathcal{O}(10^{-6})$ m to adequately resolve the boundary layer on the blade surface. Overset-mesh methodology offers an attractive option to address the disparate range of length scales; it allows embedding body-conforming meshes around turbine geometries within nested wake-capturing meshes of varying resolutions necessary to accurately model the inflow turbulence and the resulting wake structures. Dynamic overset hole-cutting algorithms permit large relative mesh motions that allow this overlapping mesh structure to track unsteady inflow direction changes, and turbine control changes (yaw, pitch, and blade deformations). The technique is also well suited to combine different mesh strategies, e.g., unstructured boundary-layer-resolving meshes to module complex geometry, and computationally efficient block-structured background meshes (preferably with adaptive-mesh refinement (AMR)) to resolve the wake and the atmospheric boundary layer (ABL) flow field.

Nalu-wind uses TIOGA[1] [2], an open-source, parallel domain-connectivity library, that employs an *implicit hole-cutting* approach to determine the parallel domain connectivity of arbitrary overlapping meshes to model the computational domain of interest. The reader is referred to Sitaraman et al. [33] for a survey of different overset domain connectivity approaches. This section provides a brief overview of the implicit hole-cutting algorithm, details of the integration of TIOGA library with Nalu-Wind, and concludes with some verification and validation studies performed on simple problems to demonstrate the overset-mesh capability within Nalu-Wind.

3.1 IMPLICIT HOLE-CUTTING

Overset domain connectivity is a process to determine where the flow equations are solved and how information is exchanged between the overlapping grids when they overlap in a given point in space. The process tags each node in the collection of computational meshes as *field*, *fringe*, or *hole* nodes. Field nodes are nodes on the mesh where the flow equations are solved, fringe nodes are receptors that receive data from a *donor cell* located in another mesh that solves the flow equations at that point in space, and hole nodes are nodes that either lie within a solid body (and, therefore, cannot have a valid solution) or are entirely bounded by fringe nodes (and, therefore, need not be solved for at a given time-step). Unlike explicit hole-cutting, which require the user to provide inputs on how to perform the domain connectivity, implicit hole-cutting uses some mesh heuristic (the cell volume, for example) to automatically determine field, fringe, and hole nodes for a given

[1] https://github.com/jsitaraman/tioga

set of overlapping meshes. The implicit hole-cutting process implemented in TIOGA can be outlined in the following steps:

Determine hole points In the first step, TIOGA identifies the nodes on computational meshes that lie within solid bodies and cannot have a valid solution. This process will result in a minimal holecut.

Determine fringe points Fringe points (or receptors) are degrees-of-freedom (DOFs), nodes in the case of Nalu-Wind, that have a poor resolution capacity, i.e., not the mesh with the smallest cell volume. The solution at these DOFs are interpolated from *donor elements* from other meshes. Some nodes are tagged as *mandatory receptors* during this step; these are neighbors of hole points, and points on the outer boundary of interior meshes and their immediate neighbors. TIOGA uses an efficient, parallel implementation of alternating digital trees (ADT) to perform the point-in-cell inclusion checks.

The process generates an optimal mesh overlap between participating meshes where the flow equations are solved. Figure 1 shows the process for an example problem of an airfoil with a slat. To ensure good solution quality, it is recommended that the cell volumes are comparable in the transition regions between the near-body and background meshes. Large discrepancies in the cell volume will result in contamination of solution and spurious artifacts at the overset interface.

3.2 OVERSET-MESH SOLUTION PROCEDURE

The solution-exchange process between the intersecting meshes is implemented as a set of constraint equations in the linear system, i.e., the rows of the linear system corresponding to the fringe nodes are modified such that, instead of the terms originating from discrete operators of the terms in the PDE, it contains the interpolation stencil for the solution at the fringe node from the donor cell. These constraints can be written

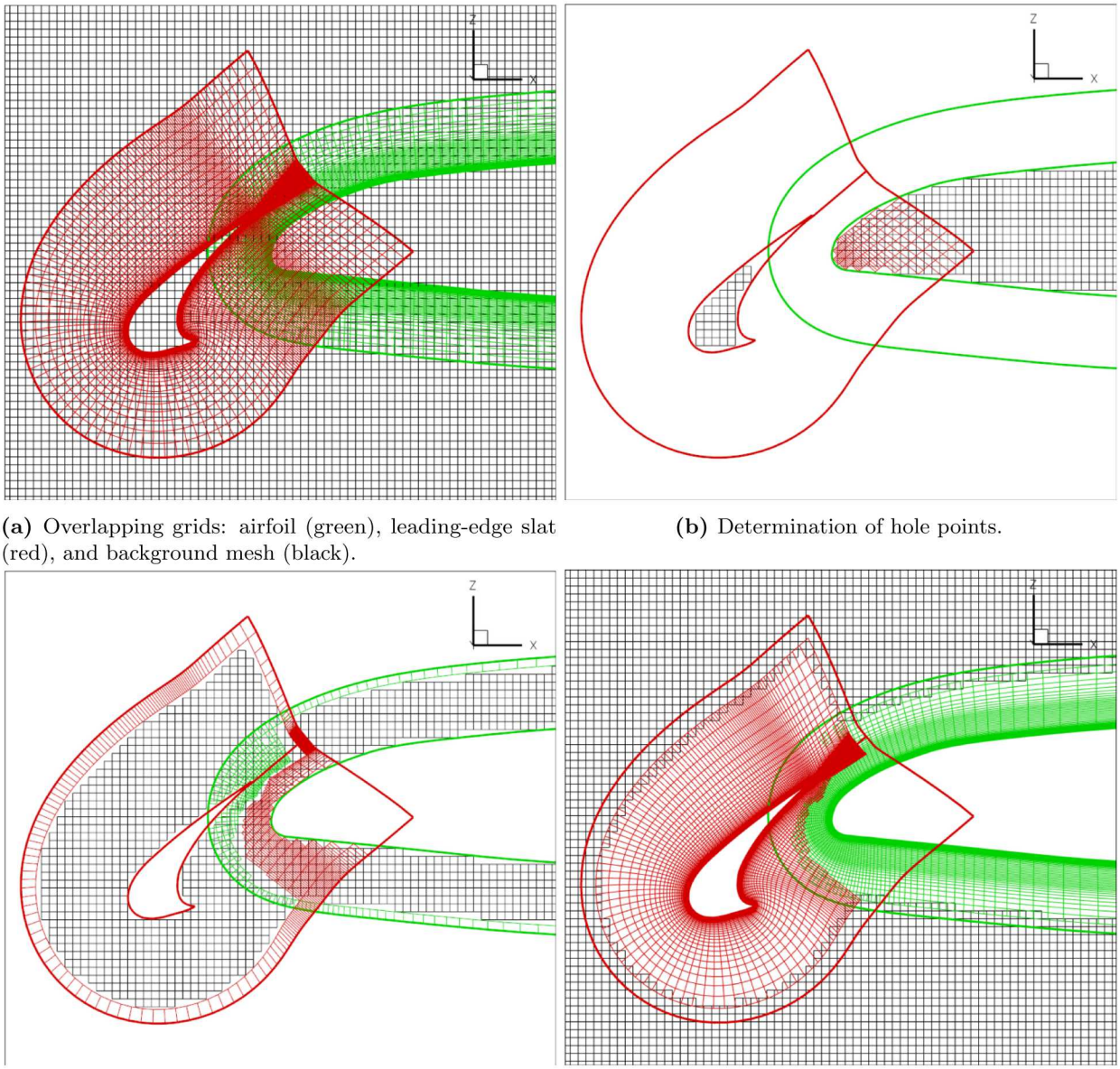
$$\Delta\phi_i^m - \sum_{k=1}^{N^e} w_k^e \Delta\phi_k^n = 0\,, \qquad \sum_{k=1}^{N^e} w_k^e = 1\,, \tag{1}$$

where $\Delta\phi_i^m$ is the solution update to the fringe node on mesh m for a field ϕ, $\Delta\phi_k^n$ is the solution update (determined by the linear system) at nodes ($k = \{1, \ldots, N^e\}$) of the donor element e on mesh n, and w_k^e is the interpolation weight determined by the shape functions of the donor element. The solution is fully coupled and does not require a solution-exchange step after the linear solve.

The introduction of these constraint equations does not appear to affect the convergence rates of the iterative linear-system solvers for the momentum or scalar transport equations. However, it does seem to have a considerable impact on the convergence of the iterative linear-system solver for the pressure Poisson system. Two potential mitigation approaches are being explored. First, a decoupled pressure-Poisson solve approach for which the elliptic solve is performed separately on each of the overlapping meshes and then coupled via iterations for which the solution is exchanged at the fringe nodes. This has the advantage of breaking the global linear system into smaller systems that can be solved in parallel, but suffer the overhead of additional coupling iterations. This approach has been implemented and explored within Nalu-Wind and good convergence has been observed for simple problems such as flow past airfoils. The robustness of this approach has yet to be tested on problems with mesh motion, e.g., rotor simulations. The second option is a constraint-elimination step for which the constraint equations are recast as a restriction operator and the linear system is restricted to only the field degrees-of-freedom. This approach is currently being implemented and will be explored in the future.

3.3 VERIFICATION AND VALIDATION OF OVERSET-MESH METHODOLOGY IN NALU-WIND

The overset implementation in Nalu-Wind was verified and validated on simple test problems that are described here.



(a) Overlapping grids: airfoil (green), leading-edge slat (red), and background mesh (black).

(b) Determination of hole points.

(c) Determination of fringe points.

(d) Final configuration where flow equations are solved.

Figure 1: Schematic showing the steps involved in implicit hole-cutting algorithm during the overset domain connectivity process. Images provided by J. Sitaraman.

3.3.1 *Verification: Heat conduction equation (MMS)*

A mesh convergence study was performed on the heat conduction equation system using the method of manufactured solutions (MMS). The exact solution for the MMS study in a two-dimensional computational domain used was a temperature field described by the equation

$$T = \frac{1}{4}\left(\cos 2\pi x + \cos 2\pi y\right) . \tag{2}$$

Figures 2(a) and (b) show the overset-mesh setup and the exact solution of the problem on a two-dimensional domain. Figure 2(c) shows that the overset meshes are able to demonstrate second-order convergence (same order as the underlying numerical scheme) with one caveat. The separation distance, d, between the fringe points of the two meshes must remain constant as the mesh is refined. This was previously reported by Chesshire and Henshaw [3]. It must be noted that for generally unstructured meshes with arbitrary overlap, the separation distance will not remain constant with mesh refinement and, therefore, the order of accuracy will be less than the order of the underlying numerical discretization scheme.

3.3.2 *Validation: Laminar-flow past cylinder*

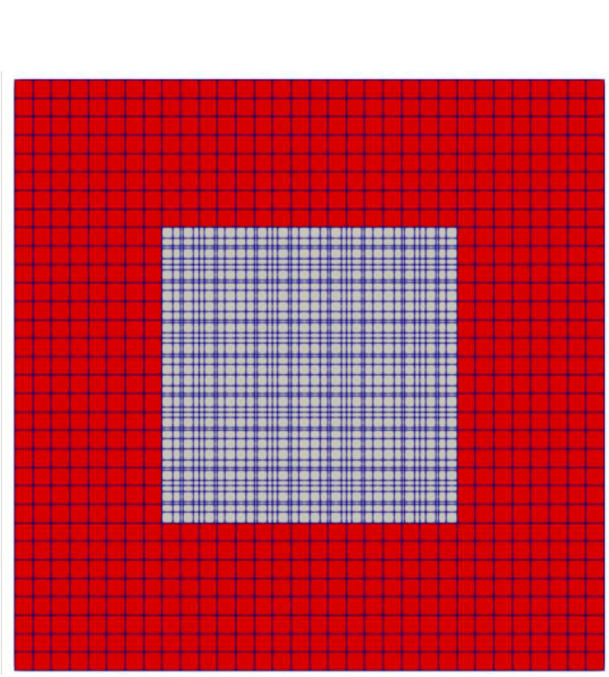
The problem of vortex shedding behind a cylinder in laminar flow conditions were studied using overset meshes in Nalu-Wind and compared to available experimental data [40] in the literature. Two quantities were compared against measurements at various Reynolds numbers (Re): the separation angle, and the Strouhal number $St = fD/U$, where f is the shedding frequency. Figure 3 shows the predictions of these quantities as a function of Reynolds number. A body-conforming mesh around the cylinder was embedded within a background mesh (Fig. 3(a)) to capture the vortex wake structures. The simulations were performed two-dimensional (2D) meshes with one layer of extrusion in the spanwise direction. The flowfield (Fig. 3(b)) shows the distinctive Kármán vortex sheet behind the cylinder. The results agree well with experimental data over the range of Re number studied. The Strouhal number predictions (Fig. 3(c)) at $Re = 200$ show discrepancies from the measurements. However, this is expected as the three-dimensional effects become significant at these Reynolds numbers and cannot be captured by a two-dimensional simulation.

In addition to the verification and validation (V&V) efforts on simple problems with overset meshes, validation was also performed for a 3-D fixed wing and 3-D turbine rotor simulations with mesh motion. These results are described in later sections – see §5 and §4, respectively – which demonstrate that the overset capability implemented using TIOGA in Nalu-Wind is quite robust and is capable of handling the full range of conditions experienced by a wind turbine during normal operation.

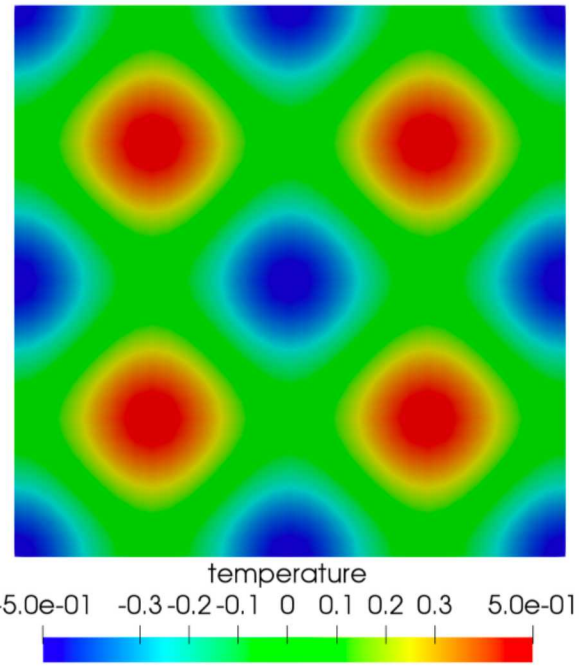
4. TIMESTEP ALGORITHM

Predictions of wind plant performance operating with inflow conditions from an unsteady, turbulent ABL require simulations over large time periods. Consider, for example, the ECP ExaWind challenge problem whose objective is to predict the power output of a wind farm consisting of at least nine NREL 5-MW turbines operating in a $4 \times 4 \times 1$ km domain. With mean inflow velocity of 11.4 m/s at hub-height (90 m), simulating even a single domain transit would require analysts to simulate 500 s of physical time. On the other hand, the maximum timestep is chosen by numerical accuracy and stability concerns, and is quantified as the Courant number C or the Courant-Fredrich-Lewy (CFL) number. The Courant number is calculated as the ratio of the numerical timestep to the advective time across a computational element. Formal mathematical analysis dictates that, for design-order accuracy, the maximum Courant number must be ≤ 1. For boundary-layer resolving, hybrid-RANS/LES simulations of the wind farm, the max Courant number occurs near the trailing edges of the blade tips and for typical meshes, requiring $C = \mathcal{O}(1)$ would restrict the timestep sizes to $\Delta t = 10^{-5}$ s. At such timestep sizes, simulating a single domain transit period of the ExaWind challenge problem would require 50 million timesteps or 578 days of computation assuming one second of compute time per timestep. This renders the computation impractical for any realistic wind farm analysis.

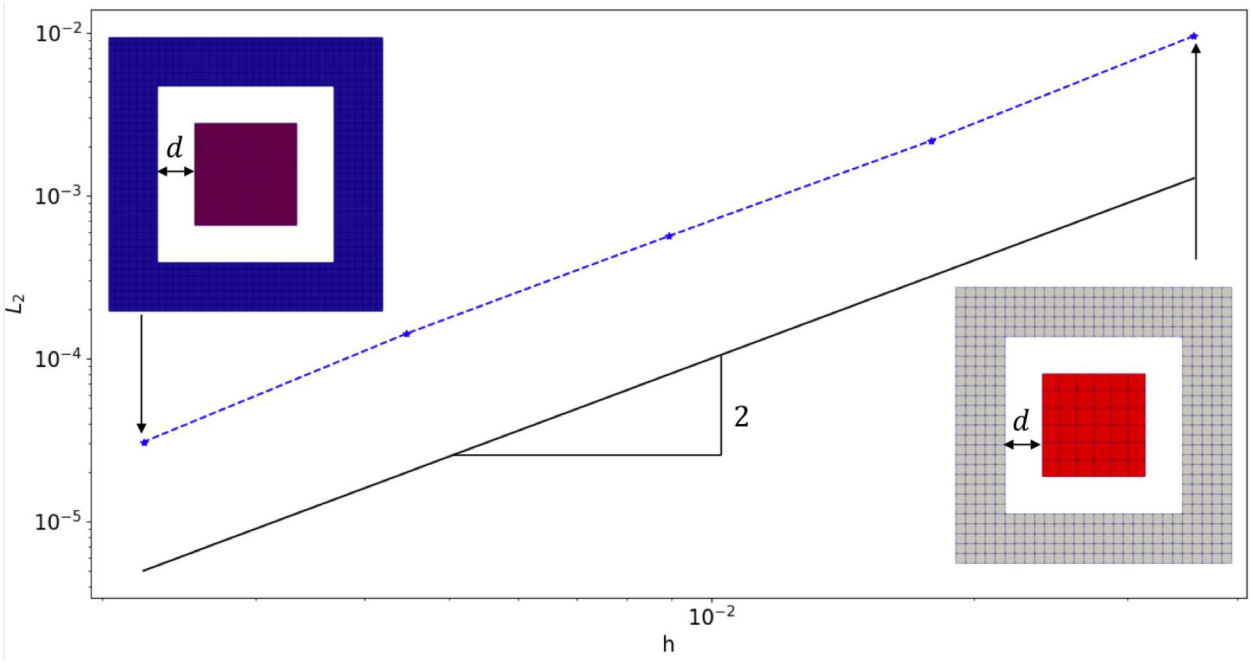
Fortunately, the regions of the most restrictive Courant numbers in the computational domain correspond to those for which turbulence is modeled using a RANS approach. Unlike regions modeled with large-eddy simulation (LES), RANS-regions have modeled(not resolved) all turbulent length and time scales; no grid-dependent turbulence is resolved. Therefore, one can take timesteps much larger than the turbulent timescales



(a) Overset-mesh setup.

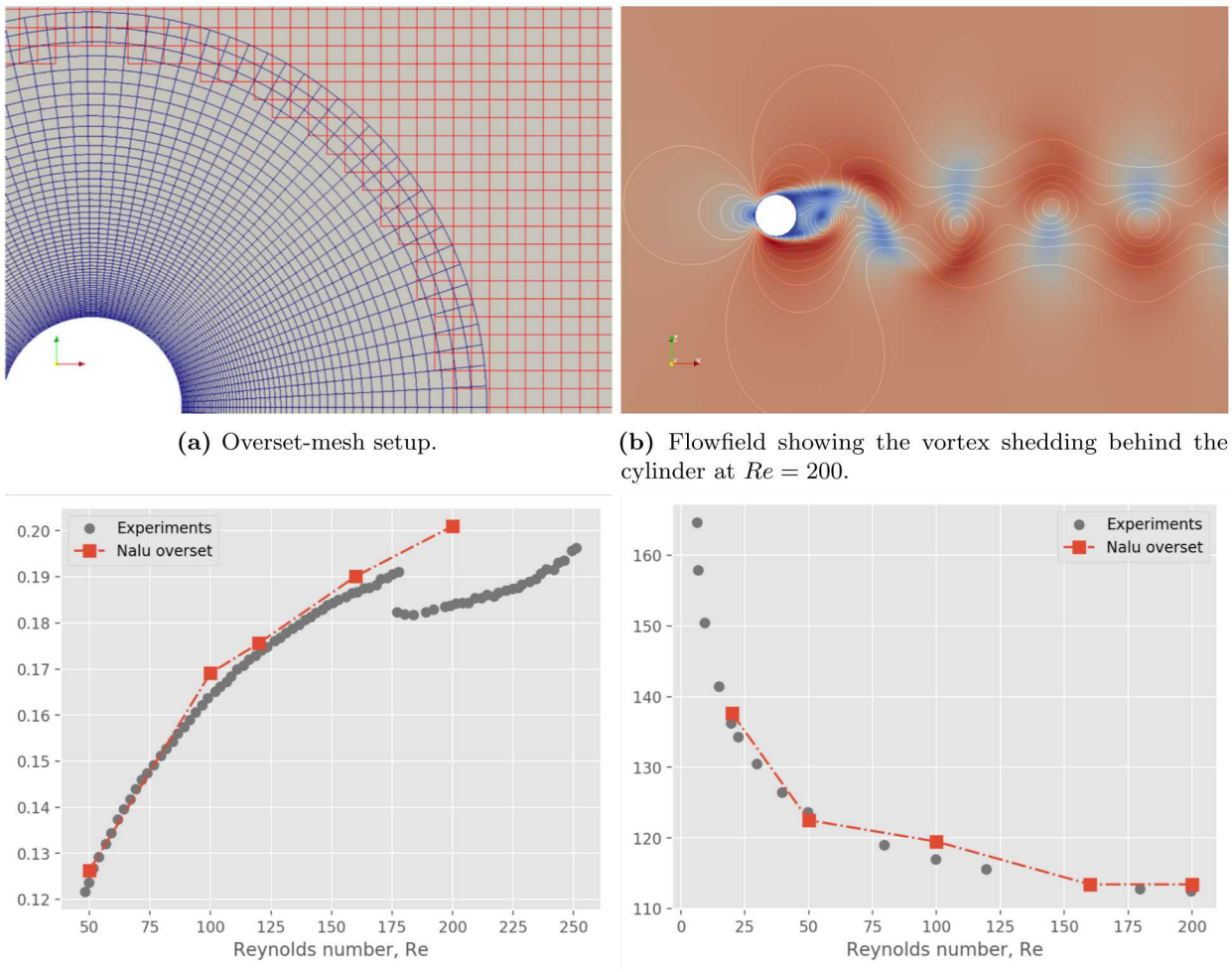


(b) Exact solution.



(c) Convergence with mesh refinement

Figure 2: (a) Overset-mesh setup, (b) the exact solution, and (b) the L_2 error-convergence of the temperature field as a function of mesh resolution for the MMS problem for the heat-conduction-equation system. The overset mesh contains two nested grids such that the inner mesh (gray) has a finer resolution than the background mesh (red).



(a) Overset-mesh setup.

(b) Flowfield showing the vortex shedding behind the cylinder at $Re = 200$.

(c) Strouhal number vs. Reynolds number

(d) Separation angle (deg) vs. Reynolds number

Figure 3: Validation of Nalu-Wind simulations with experimental data [40] for laminar flow past a cylinder.

in this region, while still being able to resolve the timescales critical for the quantities of interest (QoIs). The original timestep algorithm in Nalu, which was inherited by Nalu-Wind, was designed for LES simulations and was not suitable for use in simulations with maximum Courant number $\gg 1$. In partnership with the WETO A2e High-Fidelity Modeling (HFM) project, a new timestep algorithm was implemented in Nalu-Wind to enable stable performance at large timesteps that could enable simulations of the ExaWind challenge problems at reasonable wall clock times.

4.1 MODIFICATIONS TO THE NALU-WIND TIMESTEP ALGORITHM

The original timestep algorithm [8, 26, 10] implemented in Nalu-Wind is described briefly followed by the modifications made to enable large timesteps for hybrid-RANS/LES simulations. Nalu-Wind solves the incompressible form of the Navier-Stokes equations with appropriate turbulence models for closure. The codebase uses an implicit backward-difference-formula (BDF) family of timestep algorithms with an approximate pressure projection algorithm. Here we consider the block matrix form of the incompressible Navier-Stokes equations, and ignore the additional scalar transport equations and focus solely on the

momentum and continuity equations. These can be written

$$\begin{bmatrix} A & G \\ D & 0 \end{bmatrix} \begin{Bmatrix} u \\ p \end{Bmatrix} = \begin{Bmatrix} S_u \\ S_p \end{Bmatrix} , \tag{3}$$

where the matrix A consists of the discrete contribution to the momentum equations from the time derivative, linearized advection, and diffusion terms, and matrices D and G represent the discrete divergence and gradient operators, respectively. S_u and S_p represent the additional right-hand side terms for the momentum and continuity equations. Block factorization of the saddle-point problem yields

$$\begin{bmatrix} A & G \\ D & 0 \end{bmatrix} = \begin{bmatrix} A & 0 \\ D & -DA^{-1}G \end{bmatrix} \begin{bmatrix} I & A^{-1}G \\ 0 & I \end{bmatrix} . \tag{4}$$

Inverting the Schur-complement matrix $M = -DA^{-1}G$ would be costly. Therefore, most solvers use approximate projection schemes [26]. The original Nalu-Wind timestep algorithm used an approximate projection scheme that introduces auxiliary timescale matrices as shown below

$$\begin{bmatrix} A & 0 \\ D & B_1 \end{bmatrix} \begin{bmatrix} I & B_2 G \\ 0 & I \end{bmatrix} \begin{Bmatrix} \Delta u^{n+1} \\ \Delta p^{n+1} \end{Bmatrix} = - \begin{bmatrix} A & 0 \\ DB_4 & B_3 \end{bmatrix} \begin{bmatrix} I & G \\ 0 & I \end{bmatrix} \begin{Bmatrix} u^n \\ p^n \end{Bmatrix} + \begin{Bmatrix} S_u \\ S_p \end{Bmatrix} . \tag{5}$$

The factor $B_1 = -L_1$ defines the linear system for pressure and approximates M, $B_2 = \tilde{\tau} I$ determines the projection time scale, $B_3 = L_2$ and $B_4 = \tilde{\tau}_4 I$. L_1 and L_2 are Laplacian matrices obtained from the discrete form of Gauss divergence theorem and $\tilde{\tau}_i = \tau_i / \rho$, where ρ is the fluid density. The diagonal matrices $\tau = \tau_i = \Delta t I$ are chosen for stabilization [8]. The final system of equations solved at each Picard non-linear iteration step is

$$\mathbf{A}_p \Delta \hat{\mathbf{u}}_p^{k+1} + \mathbf{A}_{nb} \Delta \hat{\mathbf{u}}_{nb}^{k+1} = \mathbf{r}^k - \nabla p^k , \qquad \text{Momentum} \tag{6}$$

$$-\tau \mathbf{L} \Delta p^{k+1} = -\boldsymbol{\nabla} \cdot \left(\rho \hat{\mathbf{u}}^{k+1} \right) + \tau \mathbf{L} p^k - \mathbf{D} \tau \mathbf{G} p^k , \qquad \text{Continuity} \tag{7}$$

$$\mathbf{u}^{k+1} = \hat{\mathbf{u}}^{k+1} - \left(\frac{\tau}{\rho} \right) \nabla \left(\Delta p^{k+1} \right) , \qquad \text{Projection} \tag{8}$$

$$\text{where} \qquad \mathbf{A}_p = -\sum \mathbf{A}_{nb} + \frac{\gamma_1}{\Delta t} \rho \Delta V_p ; \qquad \tau = \frac{\Delta t}{\gamma_1} ; \qquad \Delta \phi^{k+1} = \phi^{k+1} - \phi^k ,$$

where $\mathbf{A}_p$ represents the diagonal term of the A matrix, $\mathbf{A}_{nb}$ are the coefficients of the off-diagonal columns, and $\mathbf{r}^k$ is the residual of the momentum equation system. The last two terms on the right-hand side (RHS) of Eq. 7 play the role of Rhie-Chow interpolation. The above formulation is well suited for unsteady LES flows and has several advantages. For stationary problems, the left-hand side of Eq. 7 is purely geometric, allowing the preconditioner to be computed only once for the entire simulation. However, at large timesteps, this algorithm suffers from two shortcomings: the error due to the stabilization term dominates for steady-state problems, and the momentum system loses diagonal dominance as the viscous terms dominate near the blade boundary layer. The modified algorithm introduces two changes to the base algorithm

1. Change in the projection timescale:

$$\tau = \text{diag}(A)^{-1} = \left[-\sum \mathbf{A}_{nb} + \frac{\gamma_1}{\Delta t} \rho \Delta V_p \right]^{-1} . \tag{9}$$

2. Use of under-relaxation:

$$\phi^{k+1} = \omega \phi^{k+1} + (1 - \omega) \phi^k \tag{10}$$

$$\mathbf{A}_p^{'} = -\frac{\sum \mathbf{A}_{nb}}{\omega} + \frac{\gamma_1}{\Delta t} \rho \Delta V_p = \mathbf{A}_p + \frac{1 - \omega}{\omega} \sum \mathbf{A}_{nb} . \tag{11}$$

Furthermore, the full pressure update (from Eq. 7) is used for the velocity and mass-flux updates, but the pressure solution is under-relaxed at each Picard iteration. The pressure update strategy is similar to the

algorithms implemented in Ellipsys3D[2]. For all the RANS and hybrid-RANS/LES simulations in this report using the new timestep algorithm, the under-relaxation parameter ω is set to 0.7 for momentum and scalar transport equations, and 0.3 for the pressure update. For RANS simulations, upwinding of the advection terms is necessary. All RANS simulations described in this report use linear upwinding, details of which are available in the Nalu-Wind user manual.[3]

It must be emphasized that the robustness of the new timestep algorithm at large timesteps comes at a price. The introduction of underrelaxation in the linear system means that order of accuracy of the discretization scheme is only satisfied upon the convergence of non-linear residuals within each timestep. However, convergence of the non-linear residuals with Picard iterations can be quite slow and would require several thousand sub-iterations to convergence to machine precision. The common practice is to perform a fixed number of Picard iterations and move to the next timestep regardless of how much the non-linear residual dropped over those fixed iterations. While this will not satisfy the formal order of accuracy of the discretization scheme with mesh refinement, tests indicate that this has minimal impact on the QoIs, as will be shown in the later sections. Accelerating the non-linear convergence within each timestep, and exploring techniques to eliminate the need for under-relaxation of the linear systems is an ongoing topic of research for the ExaWind project.

4.2 CODE-TO-CODE VERIFICATION OF THE NEW TIMESTEP ALGORITHM

The ability of the timestep algorithm to generate timestep independent results over a wide range of Courant numbers was studied using the NACA-0012 airfoil benchmark problem available at the NASA turbulence modeling resource[4]. The predictions of the lift (C_l) and drag (C_d) coefficients were compared for an airfoil operating at $\alpha = 12^\circ$ angle of attack in uniform inflow velocity $U_\infty = 15$ m/s. The flow Reynolds number was $Re = 6 \times 10^6$, density $\rho = 1.225$ kg m^{-3}, and molecular viscosity $\mu = 3.0625 \times 10^{-6}$. The $k-\omega$ SST turbulence model was used for turbulence closure. The initial and inflow conditions for the $k-\omega$ SST model are set as $k_\infty = 0.095118$ and $\omega_\infty = 2266.4$.

Both timestep insensitivity and grid convergence characteristics were studied using five different spatial meshes provided at the NASA website. The mesh descriptions are available at the NACA-0012-grids webpage[5] and are reproduced here for completeness. *A series of 5 nested 2D grids are provided. Each coarser grid is exactly every-other-point of the next finer grid, ranging from the finest* 1793×513 *to the coarsest* 113×33 *grid. The finest grid has minimum spacing at the wall of* $y = 4 \times 10^{-7}$*, giving an approximate average* y^+ *between* 0.1 *and* 0.2 *over the airfoil at the Reynolds number run. The grid is stretched in the wall-normal direction, and the clustering is maintained in the wake region. The topology is a so-called "C-grid," with the grid wrapping around the airfoil from downstream farfield, around the lower surface to the upper, then back to the downstream farfield again; the grid connects to itself in a 1-to-1 fashion in the wake. There are 1025 points on the airfoil surface on the finest grid (65 points on the coarsest grid). There are 385 points along the wake from the airfoil trailing edge to the outflow boundary on the finest grid (25 points on the coarsest grid).*

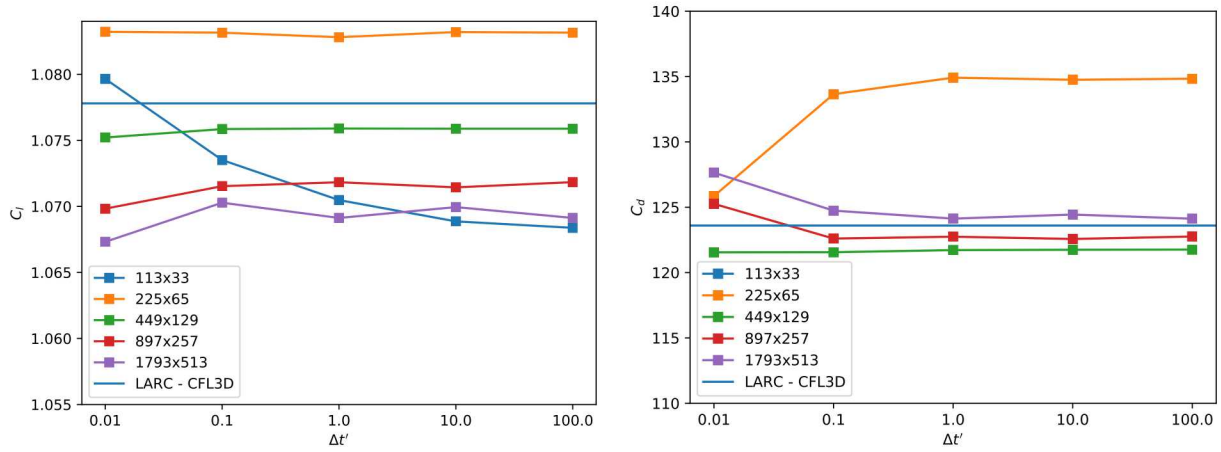
Nalu-Wind results were compared to the `CFL3D` and `FUN3D` results provided on the NASA website. While both `CFL3D` and `FUN3D` are compressible solvers, the simulations were performed at $M = 0.15$ (where M is the Mach number), which permits appropriate comparison of results between compressible and incompressible solvers. The notion of a *characteristic timescale* defined by $\Delta t' = \Delta t U/c$ was introduced to compare the results. The characteristic timescale is used as a metric of flow timescales of interest for wind energy applications. Figure 4 shows the lift and drag-coefficient predictions as a function of the characteristic timestep size for the five different mesh resolutions analyzed in this study. The maximum Courant number ranged from $1-10^6$ for these computations based on the different mesh resolutions and characteristic timestep sizes used. The results show less than 1% variation as a function of timestep size within each mesh analyzed, which demonstrates insensitivity to timestep sizes for the new timestep algorithm. Figure 5 shows the grid convergence of the lift- and drag-coefficient predictions with mesh refinement. For the finest mesh, results from all timestep sizes converge to within 1% and are close to the predictions of `CFL3D` results.

[2] http://www.the-numerical-wind-tunnel.dtu.dk/EllipSys

[3] https://nalu-wind.readthedocs.io/en/latest/source/theory/advectionStabilization.html

[4] https://turbmodels.larc.nasa.gov/naca0012_val_sst.html

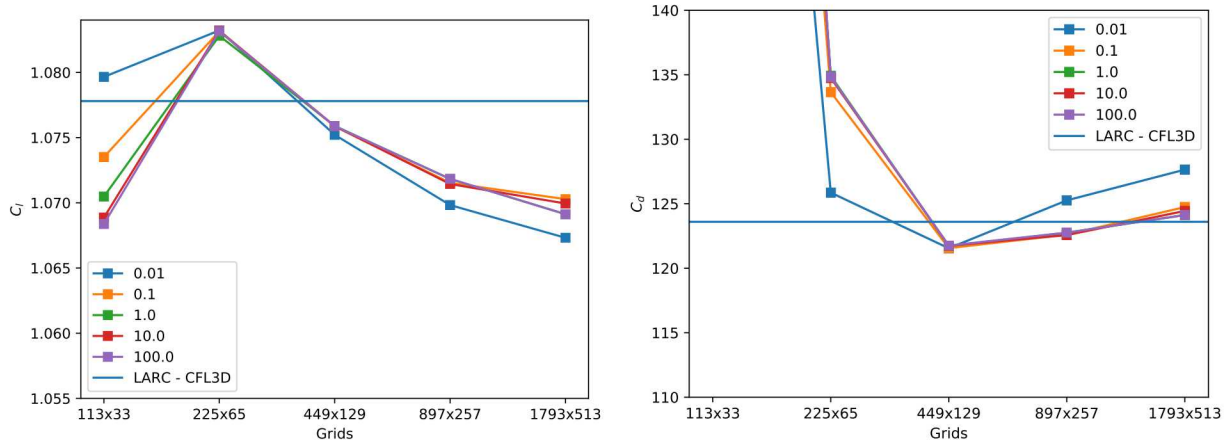
[5] https://turbmodels.larc.nasa.gov/naca0012_grids.html



(a) Lift coefficient as a function of timestep size.

(b) Drag coefficient as a function of timestep size.

Figure 4: Predictions of the lift and drag coefficients for the NACA 0012 airfoil as a function of timestep size for different grid resolutions. $U = 15$ m/s, $\alpha = 12°$, $Re = 6 \times 10^6$.



(a) Lift coefficient as a function of mesh resolution.

(b) Drag coefficient as a function of mesh resolution.

Figure 5: Grid convergence of the lift and drag coefficients for the NACA 0012 airfoil at different timestep sizes. $U = 15$ m/s, $\alpha = 12°$, $Re = 6 \times 10^6$.

4.3 ROTOR SIMULATIONS WITH OVERSET-MESH METHODOLOGY

The modified timestep algorithm and the overset-mesh methodology (§3) were tested on two different rotor configurations: the NREL Phase VI rotor, and the NREL 5-MW rotor. This section details the simulation setup, the results and comparisons to available measurements, and code-to-code comparisons with other simulation results available in the literature.

4.3.1 NREL UAE Phase VI rotor simulations

The Unsteady Aerodynamics Experiment (UAE) Phase VI test turbine [16] was based off a Grumman Wind Stream-33 turbine, a 10 m diameter, stall-regulated turbine. The turbine used for the wind tunnel test had several modifications and was a two-bladed turbine with twisted and tapered blades. Measurements were made in both upwind- and downwind-rotor configurations. The tests were performed in the NASA Ames Research Center's 80×120 ft wind tunnel. Detailed information on the wind tunnel, test turbine, test instrumentation, run conditions and data are available in the NREL report [16].

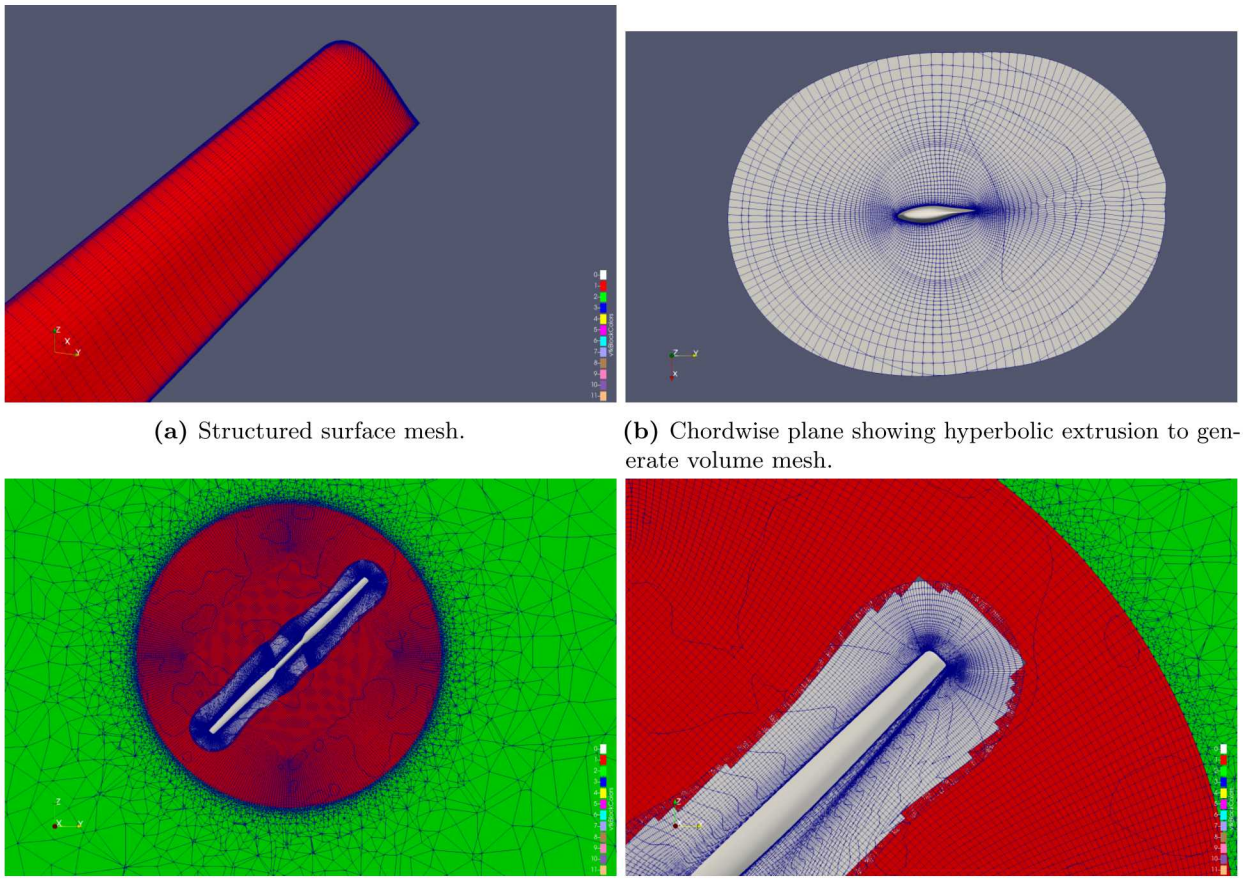
For the Nalu-Wind validation study, simulations were performed to match the run conditions of the Test Sequence H, the upwind baseline configuration. In this configuration, the blades were rigid (i.e., no teeter of the two-bladed configuration) with zero blade coning. The blade pitch was set at 3° and the rotor was run at a fixed speed of 72 rpm for all wind speeds. Simulations were performed for the zero-yaw condition at six different wind speeds: 5, 7, 10, 13, 15, and 20 m/s respectively.

The turbine geometry was simplified for the simulation by ignoring the tower, nacelle, and the aerodynamic interference effects arising from the instrumentation near the rotor hub. Furthermore, the hub section of the two-bladed rotor was idealized as a cylindrical connecting rod that joined the two blades in the computational model. The blade surface geometry was modeled using structured mesh spanwise and chordwise directions (Fig. 6a) and the volume mesh was generated using hyperbolic extrusion to about two chords (Fig. 6b). The wall-normal spacing was set to 10^{-5} m for the first point off the wall ($y^+ \approx 1$). The near-body mesh is embedded in a structured, hexahedral element-only, cylindrical wake-capturing mesh with an O-H (or "butterfly") topology. The cylindrical mesh extended half a diameter upstream and five diameters downstream. The section of the cylindrical mesh around the near-body mesh had constant spacing in the flow direction up to half a diameter upstream and downstream, and stretching was introduced in the flow direction further downstream. The wake-capturing mesh was embedded inside a fully-unstructured mesh that covered the rest of the domain that had extended five diameters upstream, and ten diameters downstream and in lateral directions (Fig. 6c). The mesh resolution was controlled carefully within the wake-capturing mesh to ensure good quality transition (comparable cell volumes) from the blade mesh to the wake mesh (Fig. 6d). The final mesh is composed of 9.9 million elements (7.3 million nodes) of which 3.4 million elements are in the near-body mesh. The meshes were generated with Pointwise, a commercial meshing software.

Simulations were performed in a fixed reference frame and the rotation of the rotor blades was simulated by rotating the near-body mesh at each timestep. This requires re-computation of the overset domain connectivity and the reinitialization of the linear systems and preconditioners at each timestep. Calculations used the $k-\omega$ SST RANS turbulence model and the best practices detailed in the airfoil computation section (§4.2) were used for the rotor simulation. A fixed timestep size was chosen such that the rotor blade would rotate $\Delta\psi = 0.25°$ per timestep ($\Delta t = 5.7869 \times 10^{-4}$), and one rotor revolution would require 1440 timesteps to complete. At least twelve rotor revolutions were simulated at each wind speed to achieve statistical convergence of the integrated thrust and power for the turbine before comparison with experiments. At steady state, the 0.25° timestep resulted in a maximum Courant number of ≈ 200 at the trailing edges of the blade tips. Computations were performed on NREL's Eagle system[6], on 360 MPI ranks (10 compute nodes). On 360 MPI ranks, one rotor revolution was completed in approximately 5 hours when compiled with Intel compiler suite with Intel MPI. For the Phase VI mesh, the overall time spent in overset domain connectivity calculations is less than 10% of the total run time.

Figure 7a shows the flowfield (Q-criterion and vorticity contours) after 12 rotor revolutions for the Phase VI rotor operating at 7 m/s wind speed. The background mesh is able to capture the tip-vortex structure about five rotor revolutions before the grid coarsening in the wake-capturing mesh dissipates the tip vortices. Capturing the vortex pairing, Kelvin-Helmholtz instabilities, and the eventual breakdown of the tip vortices

[6] https://www.nrel.gov/hpc/eagle-system-configuration.html



(a) Structured surface mesh.

(b) Chordwise plane showing hyperbolic extrusion to generate volume mesh.

(c) Overset-mesh setup showing near-body, wake, and background mesh.

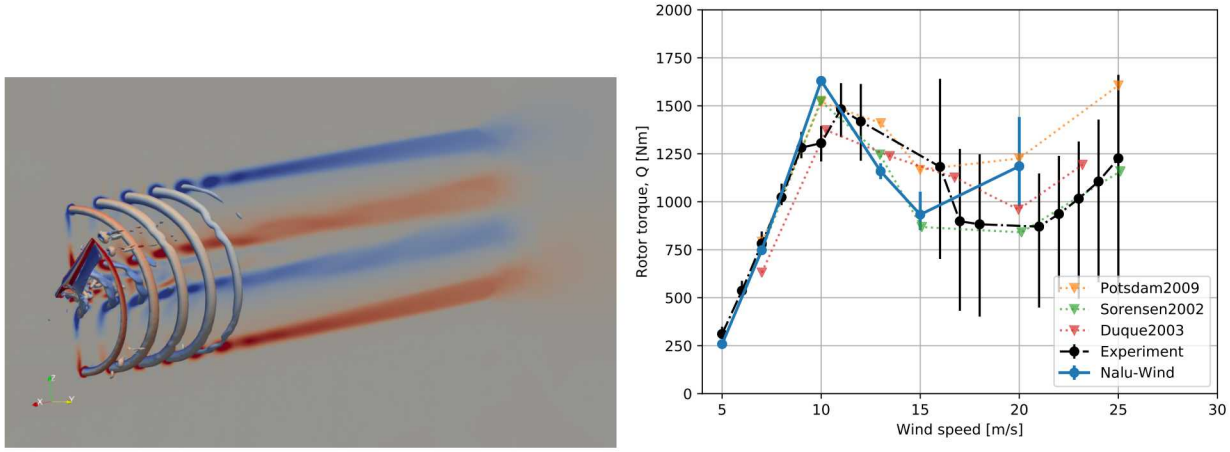
(d) Zoomed view of the overset hole-cut mesh near the blade.

Figure 6: Overset-mesh setup for the NREL Phase VI turbine simulation with Nalu-Wind.

would require higher resolutions in the wake mesh and will be part of future studies. However, the flow visualization shows that the overall physics of the flowfield is captured with the current mesh setup.

Figure 7b shows the rotor-torque predictions as a function of wind speed in comparison to measurements (shown in black) and other simulations from literature [34, 12, 29]. The black vertical bars about the measurements indicate the variation in the measurements over the duration that data was collected. The torque predictions show good agreement with measurements and other computational results for the low wind speeds. In this regime, the flow is mostly attached across the entire blade span and the experimental measurements show very little deviation from the mean values. At higher wind speeds (> 10 ms^{-1}), the measurements show significant variation; this is due to flow separation and stall in the inboard sections of the blade. In this regime, there is greater mismatch between Nalu-Wind computed torque values and the experimental data, as well as other computed results. Next we will examine the sectional performance data to better understand the reasons for the mismatch.

Figures 8 and 9 show the comparison of the chordwise variation of the pressure coefficient (C_p) at four different spanwise locations for two different wind speeds – 7 and 10 m/s. At 7 m/s, the computed results show good agreement with the measurements at all the spanwise stations where experimental data is available. This is not surprising since the flow is attached across most of the blade span and the computational models as well as the mesh used are able to capture the flow characteristics quite accurately. At 10 m/s, however, the computational results significantly overpredict the peak suction pressure at the leading edge for the inboard sections of the rotor blade. The computational models are unable to capture the flow separation characteristics observed in the experimental results. Sørensen [35] reported that a laminar-turbulent transition



(a) Flowfield at U=7 m/s.

(b) Torque vs. wind speed.

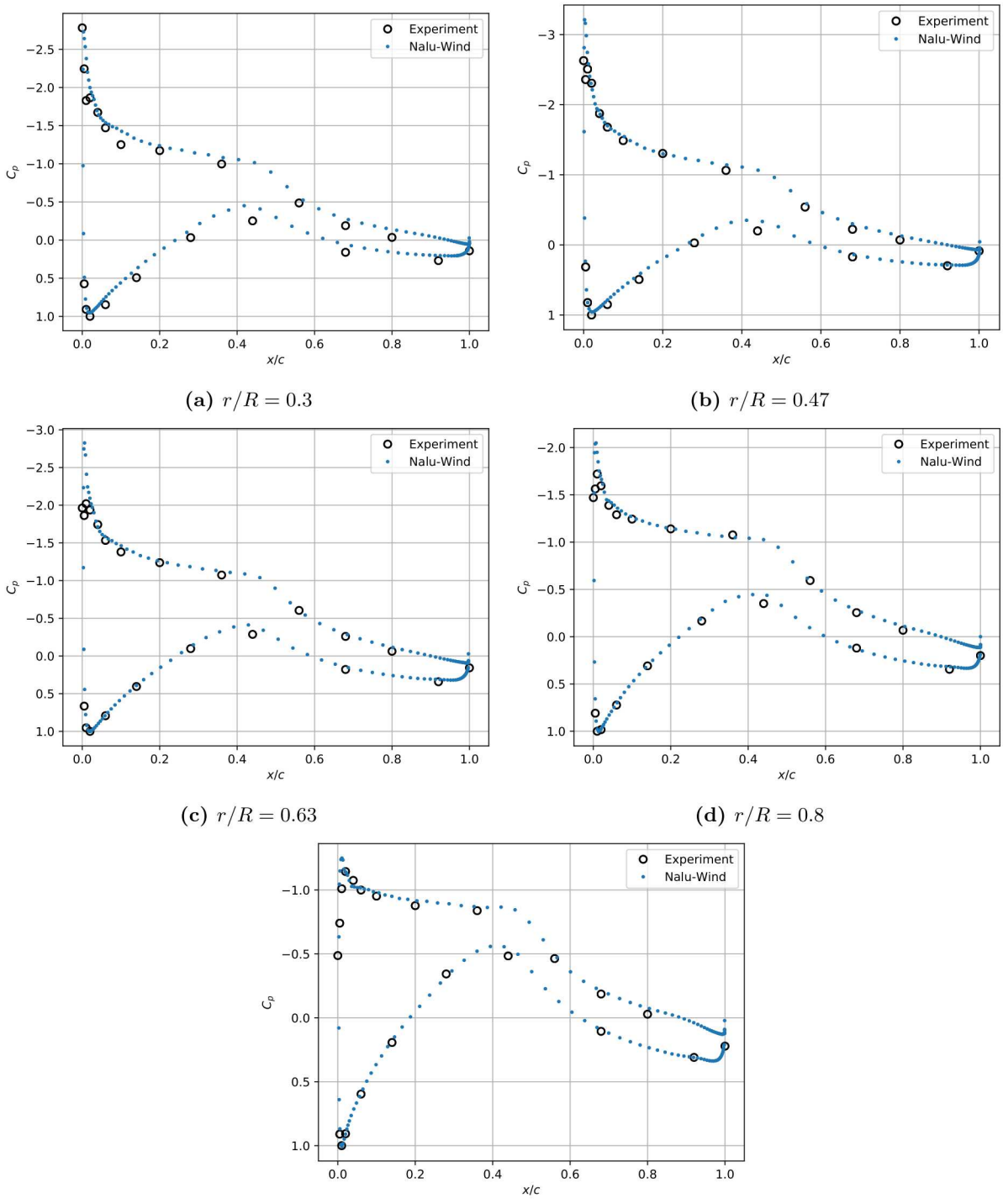
Figure 7: Isocontours of the Q-criterion and vorticity contours for the NREL Phase VI rotor operating at 7 m/s; rotor torque predictions as a function of wind speed and comparison to measurements [16]. Computational results from other researchers from published literature [34, 12, 29].

model improves the prediction of the leading-edge stall at the inboard sections for these wind speeds. Currently, Nalu-Wind does not have any laminar-turbulent modeling capabilities and this will be the subject of future research efforts. Further research is also needed to better understand the ability of the Nalu-Wind models to capture the flow separation and reattachment under the influence of three-dimensional effects such as spanwise flow, etc.

4.3.2 NREL 5-MW rotor simulations

The Phase VI results (§4.3.1) show that the Nalu-Wind code is capable of predicting the performance characteristics of a wind turbine in uniform inflow using the new timestep algorithm and overset-mesh methodology. However, the test turbine used in the Phase VI experiment is very different from the modern day megawatt-scale turbines. The rotor diameters of the turbines in operation today are an order of magnitude larger than the 10m test turbine, thus operating at much higher Reynolds numbers ($\mathcal{O}(10^7)$) than the Phase VI turbine ($\mathcal{O}(10^5)$). Furthermore, they operate in a very different region of the atmospheric boundary layer and thus experience a large variation in wind shear and veer characteristics across the rotor disk. Also, modern turbines have a full pitch and RPM controller that changes the operational state as a function of wind speed. In order to demonstrate that the methodology discussed in the previous sections would translate well to the simulation of modern-day, megawatt-scale turbines, one and two NREL 5-MW turbine simulations were performed in uniform inflow with the following objectives: demonstrate the numerical methods are robust enough to simulate full-scale rotor flow, benchmark the performance of overset search algorithms for full-scale rotors, and explore the weak-scaling characteristics (multiple rotors in wind farm) for Nalu-Wind, in particular, the overset search algorithms.

The NREL 5-MW turbine [19] is a 126 m diameter reference turbine, designed for use in research of offshore wind. While no such turbine exists, it is widely used in the wind research community and thus is a good baseline turbine geometry to study the capabilities of the code and perform code-to-code comparisons with other simulations published in the literature. For the purposes of this study, as with the Phase VI rotor, the turbine geometry was simplified by ignoring the tower and nacelle structures, only the three blades and the hub were modeled. Meshing best-practices from the Phase VI turbine study were used to generate the blade surface mesh. In order to transition smoothly to the hub structure, the structured mesh on the blade surface was constructed outboard of the 20% span – see Fig. 10. The sections inboard used unstructured mesh to transition smoothly to the hub mesh. Like the Phase VI simulations, the near-body mesh was embedded in a wake-capturing mesh that extended half a rotor diameter upstream and about



(a) $r/R = 0.3$

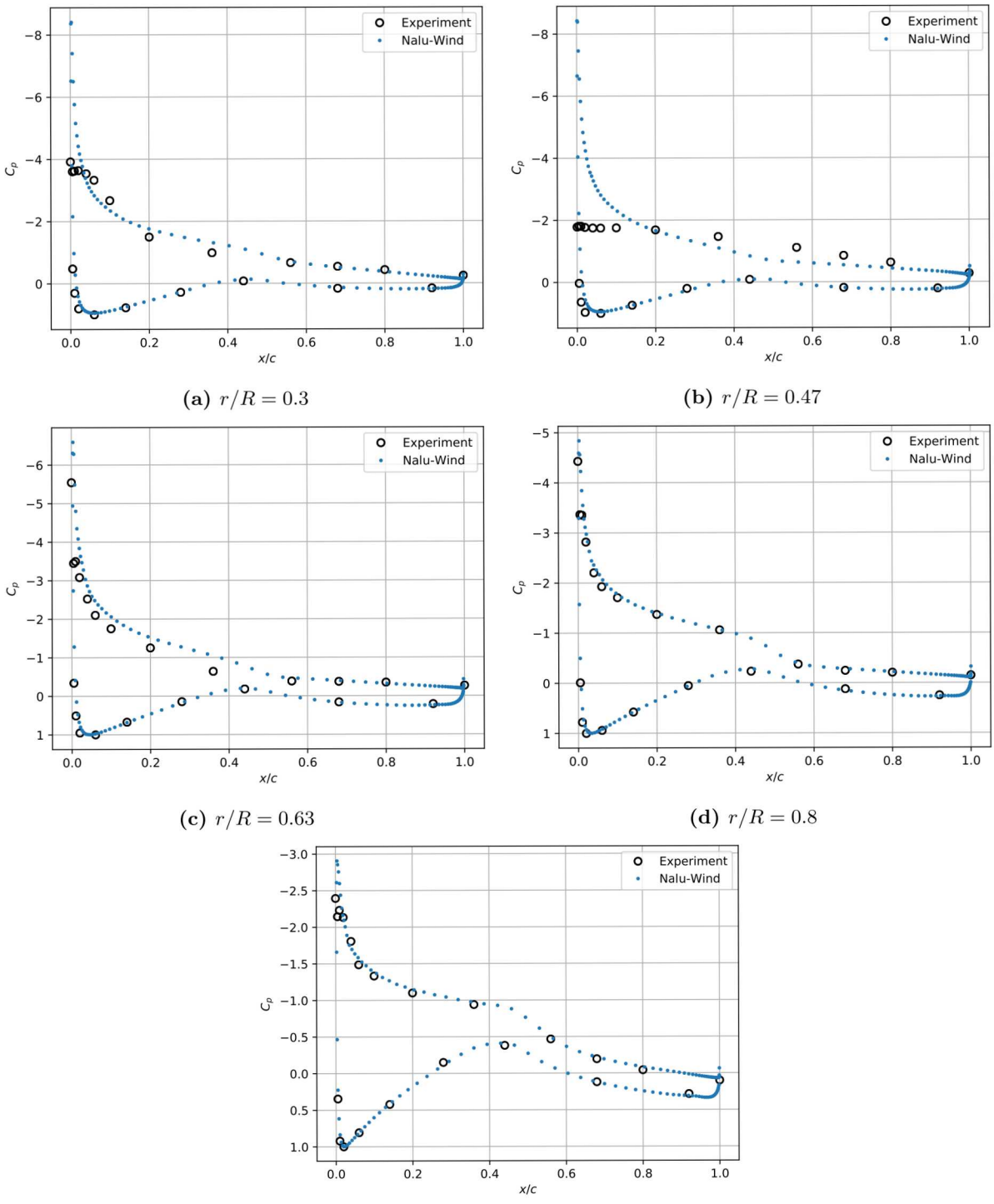
(b) $r/R = 0.47$

(c) $r/R = 0.63$

(d) $r/R = 0.8$

(e) $r/R = 0.95$

Figure 8: Chordwise variation of the surface pressure coefficient (C_p) at various spanwise sections of the rotor blade for the NREL Phase VI rotor operating at $U = 7$ m/s.



(a) $r/R = 0.3$

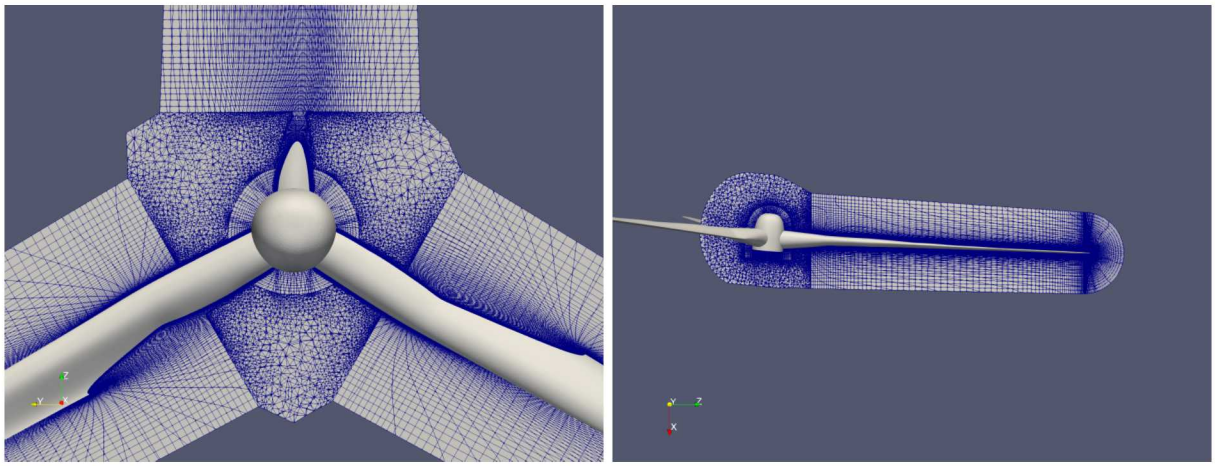
(b) $r/R = 0.47$

(c) $r/R = 0.63$

(d) $r/R = 0.8$

(e) $r/R = 0.95$

Figure 9: Chordwise variation of the surface pressure coefficient (C_p) at various spanwise sections of the rotor blade for the NREL Phase VI rotor operating at $U = 10$ m/s.



(a) Front view of the NREL 5-MW near-body mesh **(b)** Side view of the NREL 5-MW near-body mesh

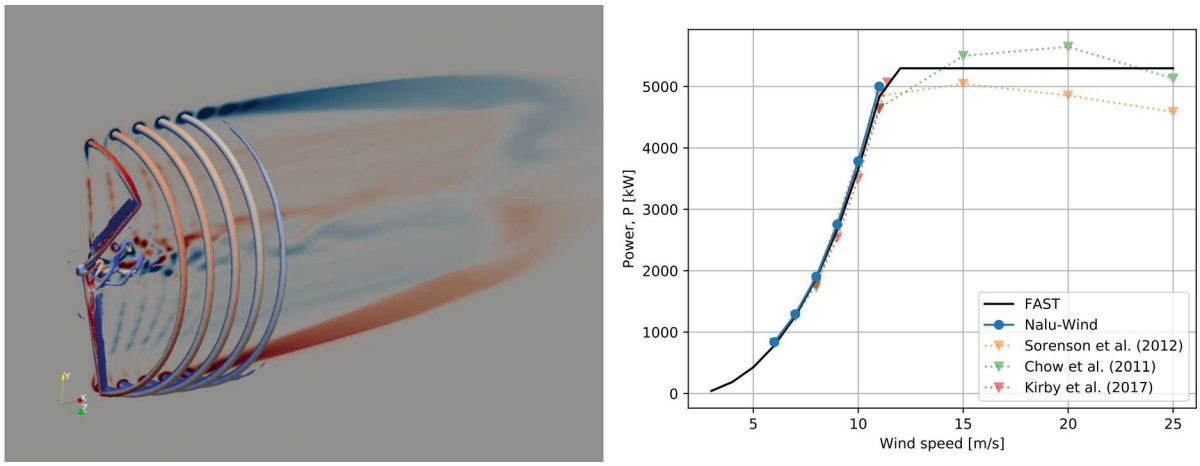
Figure 10: Front and side views of the near-body mesh used to model the NREL 5-MW rotor with Nalu-Wind. The hybrid mesh consists of structured, hyperbolically extruded mesh on the rotor blades and a fully unstructured mesh block around the hub and the hub-blade transition region.

5 rotor diameters downstream. The wake-capturing mesh was enclosed within a fully unstructured mesh that formed the outer domain. The overall computational domain extended 5 rotor diameters upstream, 10 diameters downstream, and 10 diameters in the lateral directions. The mesh contained a total of 38 million elements (23 million nodes), and the near-body mesh contained 7 million elements for all three blades.

Simulations were performed at a fixed timestep size such that the blade rotates 0.25° at each timestep. Unlike the Phase VI rotor simulations, the timestep size varies as a function of wind speed for the 5-MW case because the RPM varies as a function of wind speed. Computations were performed on the NREL Eagle system with 1080 MPI ranks (30 compute nodes). Figure 11a shows the flowfield (isocontours of Q-criterion and vorticity contours) for the NREL 5-MW rotor operating at uniform inflow of $U = 8$ m/s. The qualitiative flow structures are similar to that observed in the Phase VI results with the tip vortex dissipating quickly with coarsening of the mesh in the wake region. Figure 11b shows the NREL 5-MW power curve for Nalu-Wind predictions compared to other simulation results [36, 6, 20] alongwith results from FAST simulation [19]. Nalu-Wind simulations were only performed for wind speeds below rated wind speed (11.4 m/s) because there was no controller active for these simulations. The pitch controller is necessary for simulations above the rated wind speed and will be explored with fluid-structure coupling (§7). Results show good agreement with the other CFD simulations published in literature and provides confidence in the capability of Nalu-Wind simulations to predict the performance of megawatt-scale rotors operating in uniform inflow.

Finally, to explore the weak scaling characteristics of the Nalu-Wind code with overset mesh methodology, a two rotor simulation of the NREL 5-MW turbine, in uniform inflow, was conducted. The rotors were separated by a distance of five rotor diameters in the streamwise direction such that the downstream turbine operates in the wake of the upstream turbine. It must be noted that the physics of the turbine-wake interaction for the downstream turbine is very different in uniform inflow compared to the interactions observed in wind farms. Furthermore, because there was no controller in the loop, the operational conditions for the downstream turbine was based on best guess and engineering intuition. However, the simulation is still relevant because it provides proof of the ability of Nalu-Wind to scale to the final ExaWind challenge problem, provides valuable insight into the performance characteristics and potential bottlenecks that must be addressed over the remainder of the project.

The simulations were performed such that the front turbine was operating in uniform inflow wind speed of $U = 11$ m/s, very close to the rated wind speed, and the pitch and RPM settings for the second turbine was set to that corresponding to $U = 7$ m/s. From prior engineering experience, this wind speed was guessed to be close to what the inflow velocity in wake would be once the flow has fully evolved. The rotor mesh was *copied and pasted* for the downstream turbine, and the wake-capturing mesh was extended such that



(a) Isocontours of Q-criterion and vorticity contours $U = 8$ m/s

(b) NREL 5-MW power curve

Figure 11: Flowfield of the NREL 5-MW turbine operating in uniform inflow of $U = 8$ m/s, and the power predictions of Nalu-Wind simulations as a function of wind speed compared to other simulations from published literature [36, 6, 20].

it would cover a region half a diameter upstream of the front turbine to about 5 rotor diameters behind the second turbine. The overall domain dimensions were extended in the streamwise direction such that the outflow boundary was 10 rotor diameters behind the downstream turbine. The computational mesh contained approximately 44 million nodes, roughly double the size of the single turbine mesh. Figure 12 shows the flowfield of the two turbine simulation after ≈ 14 revolutions. The qualitative results demonstrate the potential of current Nalu-Wind codebase to simulate multiple wind turbines using the current overset mesh methodology. Future work will focus on extending this capability to include fluid-structure interaction, along with real-time controller capability, and atmospheric turbulent inflow.

5. TURBULENCE MODELING

In this section, we discuss the turbulence model implementations in Nalu-Wind. The codebase has capabilities to use a pure SST RANS model and the hybrid RANS/LES SST-DES model. We being by detailing the model equations and required blending. Next, SST and SST-DES model verification and validation through unit tests, NASA benchmark simulations, and the simulation of a NACA-0015 fixed wing experiment are shown. Finally we present the hybrid RANS-LES turbulence model implemented in Nalu-wind to simulate the operation of wind turbines in a realistic atmospheric boundary layer (ABL).

5.1 RANS AND HYBRID RANS-LES

5.1.1 Description of the models

Nalu-Wind supports the Menter 2003 SST RANS model [25], a model based on blending the k-ω and k-ϵ RANS models to leverage the advantages of the ω treatment near the wall and the ϵ treatment in the free stream. The two transport equations necessary for the SST model are:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k u_j)}{\partial x_j} = P - \beta^* \rho k \omega + \frac{\partial}{\partial x_j}\left((\mu + \sigma_k \mu_t)\frac{\partial k}{\partial x_j}\right), \tag{12}$$

$$\frac{\partial(\rho \omega)}{\partial t} + \frac{\partial(\rho \omega u_j)}{\partial x_j} = \frac{\rho\gamma}{\mu_t} P - \beta \rho \omega^2 + \frac{\partial}{\partial x_j}\left((\mu + \sigma_\omega \mu_t)\frac{\partial \omega}{\partial x_j}\right) + 2(1 - F_1)\frac{\rho \sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, \tag{13}$$

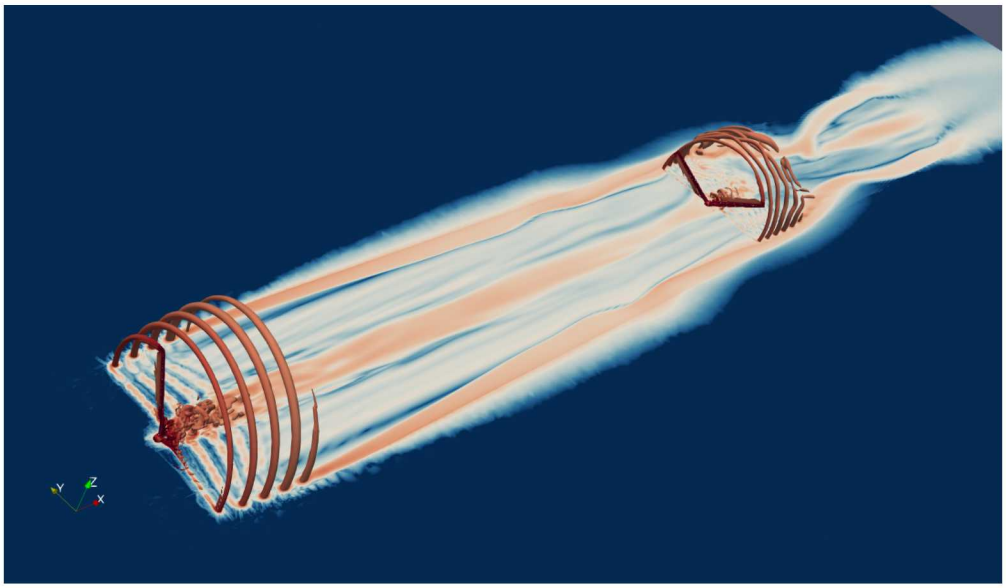


Figure 12: Flowfield showing the turbine-wake interaction for a two-turbine NREL 5-MW simulation in uniform inflow ($U = 11$ m/s).

where ρ is the density, u_i is the velocity in the i direction, μ is the molecular dynamic viscosity, k is the turbulent kinetic energy, ω is the specific dissipation rate, and

$$P = \tau_{ij}\frac{\partial u_i}{\partial x_j} \text{ is the production term,} \tag{14}$$

$$\tau_{ij} = 2\mu_t S_{ij} - \frac{2}{3}\rho k\delta_{ij} \text{ is the shear stress,} \tag{15}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{16}$$

$$S = \sqrt{2S_{ij}S_{ij}}. \tag{17}$$

The model coefficients, i.e., σ_k, σ_ω, γ, and β, are blended as

$$\phi = F_1\phi_1 + (1 - F_1)\phi_2. \tag{18}$$

where $\sigma_{k1} = 0.85$, $\sigma_{k2} = 1.0$, $\sigma_{\omega 1} = 0.5$, $\sigma_{\omega 2} = 0.856$, $\gamma_1 = \frac{5}{9}$, $\gamma_2 = 0.44$, $\beta_1 = 0.075$ and $\beta_2 = 0.0828$. The first blending function is defined as:

$$F_1 = \tanh(arg_1^4), \tag{19}$$

where

$$arg_1 = \min\left(\max\left(\frac{\sqrt{k}}{\beta^*\omega d}, \frac{500\mu}{\rho d^2\omega}\right), \frac{4\rho\sigma_{\omega 2}k}{CD_{k\omega}d^2}\right), \tag{20}$$

$$CD_{k\omega} = \max\left(2\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-10}\right), \tag{21}$$

and d is the nearest distance to the wall. Additionally, the turbulent eddy viscosity is computed as:

$$\mu_t = \frac{\rho a_1 k}{\max(a_1\omega, SF_2)}, \tag{22}$$

where $a_1 = 0.31$ and F_2 is a second blending function:

$$F_2 = \tanh(arg_2^2), \tag{23}$$

$$arg_2 = \max\left(\frac{2\sqrt{k}}{\beta^*\omega d}, \frac{500\mu}{\rho\omega d^2}\right). \tag{24}$$

Finally, it should be noted that a production limiter is used for both the k and ω equations, $\min(P, 10\beta^*\rho\omega k)$.

The baseline hybrid RANS-LES model in Nalu-Wind is the SST-DES model. The key aspect of this model is to relax the RANS model and allow for transient flows away from the pure RANS region. This idea can be translated into the introduction of a minimum length scale in the dissipation term of the turbulent kinetic energy equation:

$$\beta^*\rho k\omega \text{ becomes } \frac{\rho k^{3/2}}{l_{DES}}, \tag{25}$$

where $l_{DES} = \min(l_{SST}, c_{DES}\delta)$, $l_{SST} = \frac{\sqrt{k}}{\beta^*\omega}$, δ is the maximum edge length scale touching a given node, and c_{DES} is the blended DES constants, $c_{DES_1} = 0.78$ and $c_{DES_2} = 0.61$.

An automated test suite was developed for the FY17-Q3 milestone to exercise the core components of the SST and SST-DES turbulence models. This includes tests for the maximum length scale calculation, the turbulent viscosity, and the different source terms in the k and ω transport equations.

5.1.2 Model validation and verification

Though previously discussed in the FY17-Q3 milestone report, we include here for completeness the results of verification and validation of the SST RANS model.

Code-to-code verification of the SST RANS model was performed in a comparison of several standard test cases proposed in NASA's Turbulence Modeling Resource[7]. Figure 13 shows Nalu simulation results of the 2D zero pressure gradient flat plate and the 2D bump-in-channel as well as simulation results from the NASA CFL3D and FUN3D codes. Though both of NASA's codes solve the compressible Navier-Stokes equations and Nalu solves the low-Mach equations, the Mach numbers for these test cases are low enough such that the flow can be considered incompressible. Figure 13a shows the drag coefficient, C_d, as a function of grid size. FUN3D is a vertex centered scheme and uses a normal distance at the wall node close to that used in Nalu. This accounts for the very close agreement between Nalu and FUN3D. For the 2D bump-in-channel, Figure 13b shows the skin-friction-coefficient streamwise distribution, C_f, at $t = 0.5$, where each model has the same mesh (1409×641).

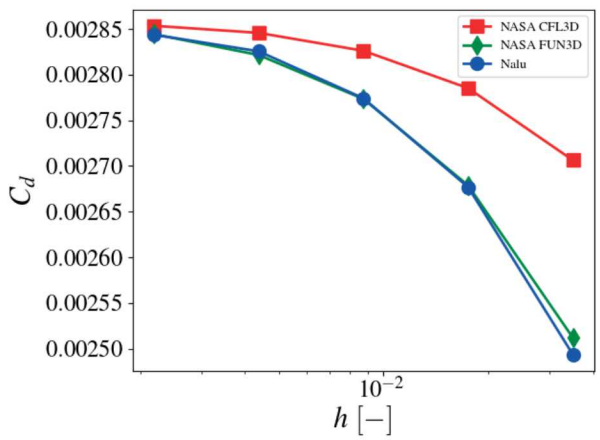
Following NASA's Turbulence Modeling Resource validation test cases, experimental data of turbulent flow over a backward facing step [11] were used to perform validation of the SST RANS model. Figure 14 shows vertical velocity distributions at two locations in the domain as produced by Nalu and CFL3D along with experimental results. Nalu simulation results compare well with the experimental data and published NASA code results.

5.1.3 Simulation of a NACA-0015 fixed wing

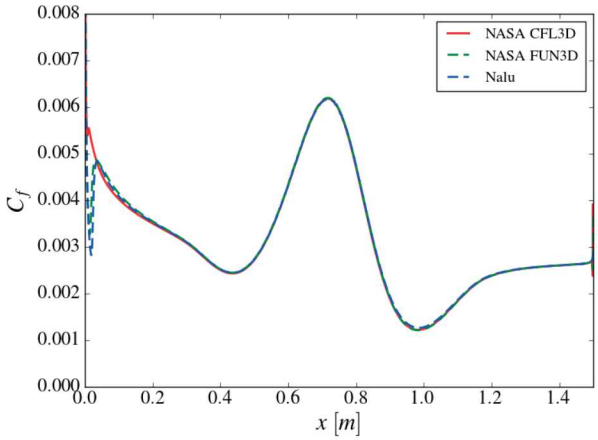
For the FY17-Q3 milestone, we used simulations of the McAlister-Takahashi NACA-0015 experiment [24] to establish baseline Nalu turbulence modeling, scaling performance, and solver capabilities. Several challenges were noted in the milestone report surrounding mesh sizes and the comparisons between the numerical simulations and the experimental data. We decided to revisit this case to leverage the extensive additions to the capabilities of Nalu-Wind, i.e., the new overset and time-stepping capabilities discussed in this report and establish this case as a baseline for future turbulence modeling improvements.

Simulation description For the comparisons in this report, we use a fixed NACA-0015 wing (untwisted, untapered, unswept) in unbounded flow (the tunnel walls from the experiments are neglected). The simulations were performed for a wing at 12° angle of attack, a 1 m chord length, denoted c, 3.3 aspect ratio, i.e., $s = 3.3c$,

[7] https://turbmodels.larc.nasa.gov

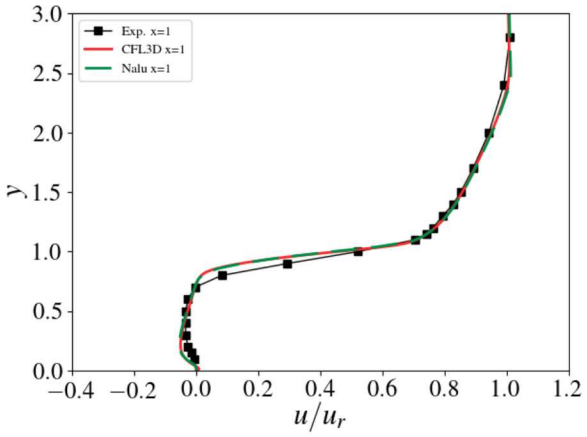


(a) The drag coefficient for the 2D zero pressure gradient flat plate as a function of element size.

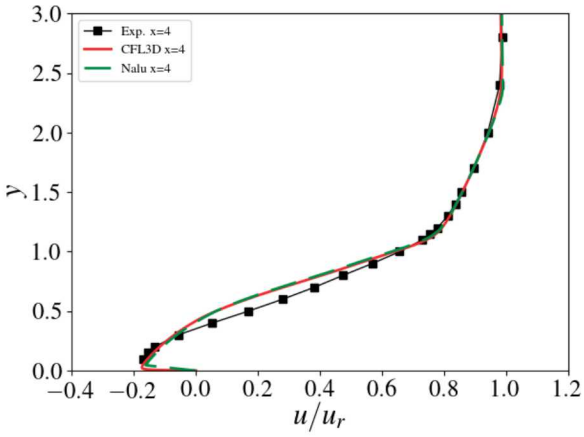


(b) Skin-friction-coefficient distribution along the 2D bump in channel flow at t = 0.5 (1409×641 mesh).

Figure 13: Code-to-code comparison between Nalu and NASA codes for the flat plate (left) and the bump in channel (right) using the SST RANS model.



(a) $x = 1$ behind the step.



(b) $x = 4$ behind the step.

Figure 14: Comparison of velocity profiles from Nalu using the SST RANS model, the NASA CFL3D code, and experimental data [11] at different downstream locations.

and a square wing tip. The inflow velocity is $u_\infty = 46$ m/s, the density is $\rho_\infty = 1.225$ kg/m^3, and the dynamic viscosity is $\mu = 3.756 \times 10^{-5}$ kg/(m s), leading to a Reynolds number, $Re = 1.5 \times 10^6$. A half-span wing is simulated and the chord is aligned in the x direction. The inflow velocity is rotated accordingly to ensure the appropriate angle of attack. The domain geometry extends to 10 chords upstream, 20 chords downstream, and 15 chords in the span. Symmetry boundary conditions are imposed in the spanwise direction. The bottom and inlet are set to inflow boundary conditions. The top and outlet are set to open boundary conditions. The initial conditions and inflow boundary conditions for the SST model variables are set according to $k_\infty = \frac{3}{2} I^2 u_\infty = 0.69$, where $I = 0.1$ is the turbulence intensity, and $\omega_\infty = 5u_\infty/c = 230$ 1/s. The final time for the simulations is 40τ, where $\tau = \frac{c}{u_\infty}$ is a non-dimensional chord flow-through time, at which point the wing forces and wake statistics are converged. For the SST-DES simulations, the initial condition is the converged SST solution.

Meshes We used Pointwise, a commercial meshing software[8], to generate multiple hexahedral-dominant meshes. The wing surface is discretized with the quadrilaterals and the wing tip is discretized with anisotropic tetrahedral extrusion (see Figure 15a). The wall-normal spacing along the wing is $10^{-5}c$, ensuring $y+ < 1$ along the wing. The mesh spacing is $5 \times 10^{-4}c$ at the leading edge and $3 \times 10^{-3}c$ at the trailing edge, with 160 points in the streamwise direction for both the upper and lower wing surface. The mesh spacing is $0.003c$ at the wing root and wing tip with 50 points in the spanwise direction. The wing surface mesh is then hyperbolically extruded to approximately $4c$ away from the wing (see Figure 15b). In the near-wake region, an additional hexahedral mesh is positioned to capture the wake at various angle of attacks. The domain far from the wing and wake regions are filled with wedges and pyramids. The resulting mesh contained 2.8 million nodes. Going forward, we will refer to this as the BASE mesh.

Additionally, because of the importance of capturing wake dynamics, we studied the effect of adding a refined tip-vortex block, using the new overset-mesh capability in Nalu-Wind, as shown in Figure 15a. This mesh was constructed of hexahedrons with an O-H grid (or "Butterfly") topology and extended $7c$ downstream. Two different grid resolutions were investigated, the first with 54 points in the vortex core (OVS-low), the second with 80 points in the vortex core (OVS-high). The overset blocks increased the node count by $700,000$ and 2.3 million nodes, respectively. Even with the overset tip-vortex blocks, the meshes in this study are significantly smaller than those generated for the FY17-Q3 milestone (those ranged from 35 million to 224 million nodes). This enabled quick turnaround times on computing clusters and the ability to rapidly evaluate different configurations, solvers, and models.

Results For this milestone report, we focus on the SST and SST-DES turbulence models and conduct multiple simulations using the meshes described above: SST with BASE, OVS-low and OVS-high, as well as SST-DES with OVS-low and OVS-high. The quantities of interest (QOI) used to compare with the experimental values are the lift coefficient,

$$c_l = \frac{2L}{\rho_\infty u_\infty^2 A}, \tag{26}$$

where L is the lift force and $A = 3.3c$ is the wing area, the drag coefficient,

$$c_d = \frac{2D}{\rho_\infty u_\infty^2 A}, \tag{27}$$

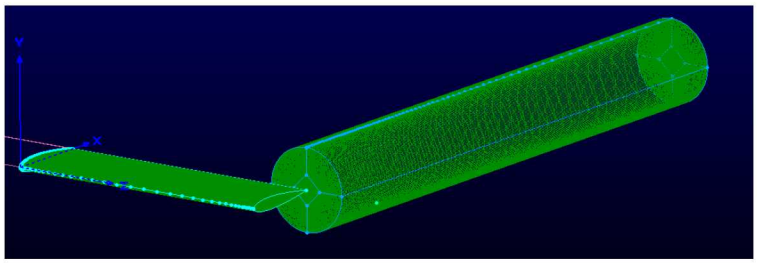
where D is the drag force, and the pressure coefficient,

$$c_p = \frac{2p}{\rho_\infty u_\infty^2} \tag{28}$$

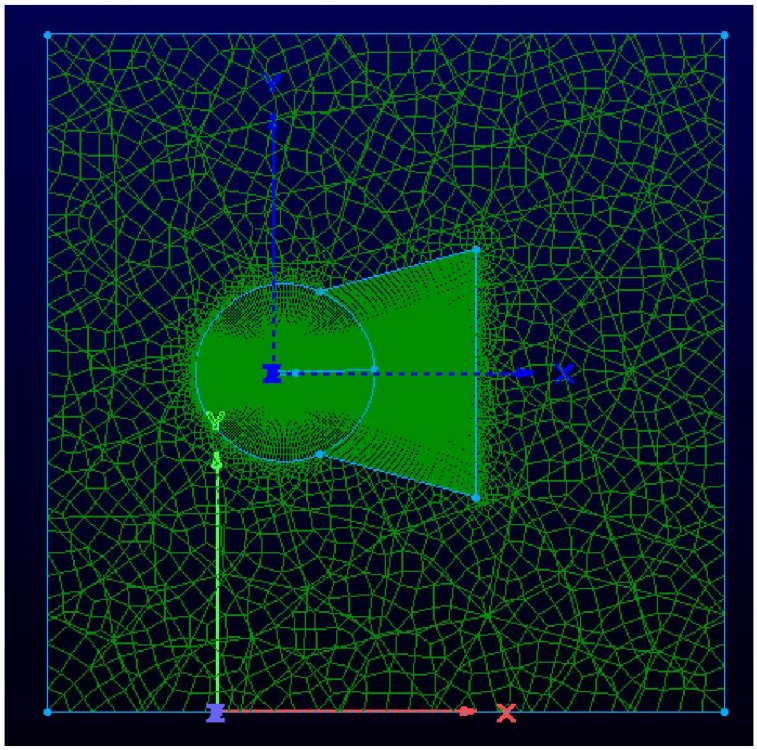
with p, the pressure, and u_x and u_y, the axial and tangential velocities across at several locations along the wing span and in the vortex core at several downstream locations.

For all models and meshes, the converged lift coefficient is $c_l = 0.9$ and the experimental value is 1.05, indicating that the simulations underpredict the lift forces by about 10%. The converged drag coefficient is $c_d = 0.056$ compared to an experimental value of 0.05.

[8] http://www.pointwise.com

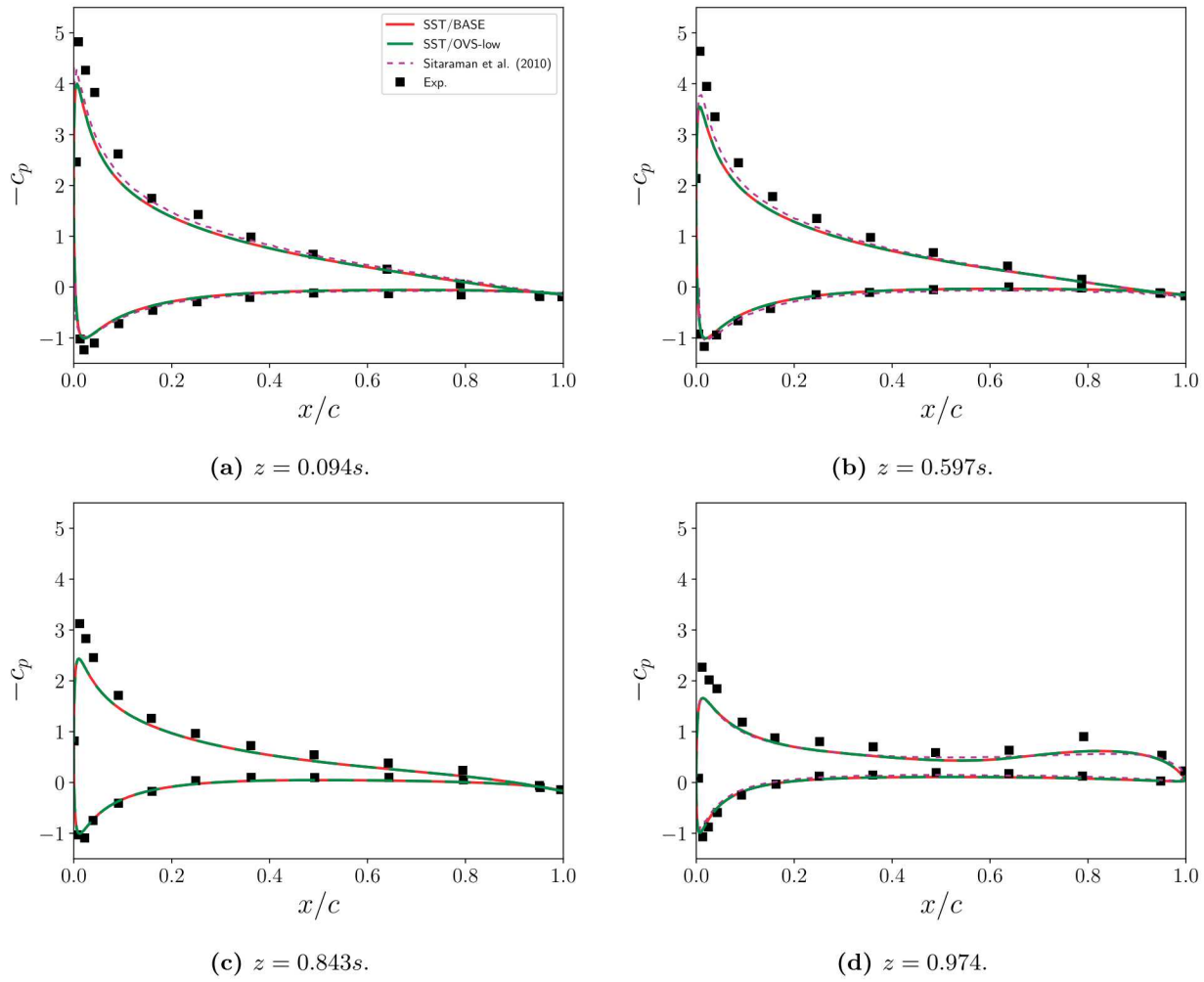


(a) Mesh on the wing and overset tip-vortex block.



(b) Background mesh.

Figure 15: Meshes for the NACA-0015 wing.



(a) $z = 0.094s$. **(b)** $z = 0.597s$.

(c) $z = 0.843s$. **(d)** $z = 0.974$.

Figure 16: Pressure coefficient, c_p, at different locations along the wing span. Solid red: SST / BASE; dashed green: SST / OVS-low; dotted magenta: Sitaraman et al. [33]; Black squares: experimental data [24].

Figure 16 shows the pressure coefficient at various locations along the wing span for the SST model (the SST-DES yields similar results along the wing). Throughout the wing span, the pressure coefficient is underpredicted compared to the experimental results. The simulation results, however, are similar to previously published results of a fixed wing in unconstrained flow which used the Spalart-Allmaras RANS model and adaptive mesh refinement to resolve the wake [33]. It is expected that the underpredictions result from improper modeling of blockage effects due to the tunnel walls (as the simulations conducted here use a symmetry boundary and a double inflow/outflow configuration).

Axial and tangential velocities across the vortex core are shown in Figure 17. The axial velocity shows a velocity deficit in the vortex far from the tip ($x > 3c$). This is because the simulations exhibit an adverse pressure gradient along the vortex, Figure 18. The tangential velocity is underpredicted throughout the downstream locations, with the prediction error increasing as the vortex propagates further downstream. The use of the hybrid SST-DES model yields improved predictions of the tangential velocities when compared to the SST model, though still falls below the experimental data. Improvement is also found with increased resolution in the tip-vortex region. This can be observed by comparing the velocity deficit in the SST simulations using both the BASE (solid red) and OVS (low: dash-dot blue, high: dotted orange) meshes in Figure 17c. The SST model simulations are converged at the OVS-low mesh resolution as no change is observed between the OVS-low and OVS-high simulations with the pure SST RANS model. However, for

the SST-DES model, the results using the OVS-high mesh show improvement in QOIs, which is most likely attributed to the decreased numerical dissipation in the higher resolution simulation.

Numerical upwinding was used throughout the simulation domain for both SST and SST-DES simulations. Though this has little impact on the SST simulations, this is likely detrimental in the LES regions of the SST-DES simulations. Initial approaches to remove upwinding in the SST-DES simulations resulted in an increase of solution time by two orders of magnitude, due to the loss of diagonal dominance in the momentum matrix and the need for a smaller time step. A proposed solution is being investigated by the linear solvers team that will potentially alleviate this issue in the future.

Previous work [33], though still underpredicting, captures the tangential velocity more accurately, most likely because of the use of adaptive mesh refinement and high order numerical methods in the wake, reducing the artificial dissipation. Several authors who have performed simulations of this case and compared to the experimental results have also observed underpredictions of c_l and c_p and the lack of acceleration in the wake [18, 33]. These effects have been attributed to blockage effects and will be investigated in future work through the development of meshes which include experimental geometry, such as tunnel walls, to provide a comparison between unconstrained and constrained flow.

We will also will pursue the use of this benchmark study to evaluate and compare turbulence model development, including but not limited too, rotation-curvature corrections for more accurate vortex transport and alternate hybrid models, such as Time-Averaged Model Split (TAMS) [14]. Finally, this case provides a setup to investigate the use of high order methods for capturing phenomena of interest such as the wake transport.

5.2 TURBULENCE MODELING FOR WIND TURBINES OPERATING IN THE ATMOSPHERE

Wind turbines typically operate in highly turbulent inflow conditions in the lower part of the atmospheric boundary layer. Atmospheric turbulence contains flow fluctuations created by different physics compared to the boundary layer turbulence on the wind turbine blades and the wake. Nalu-wind uses the 1-equation k_{sgs} model [27] to simulate the flow in the ABL using Large Eddy Simulation (LES). The RANS and hybrid RANS-LES turbulence modeling approaches described in Section 5.1 are used to simulate the turbulent boundary layer on the wind turbine and the wake under typically uniform inflow conditions. The two turbulence modeling approaches are blended together to create a unified hybrid RANS-LES model to simulate a wind turbine operating in the atmosphere following an approach similar to the one used by Vijayakumar et al. [39].

The unified hybrid RANS-LES model uses a zonal blending approach and assumes that the resolved velocity field transitions to the Reynolds averaged velocity field at a specified location. Similarly, the SGS kinetic energy field transitions to the total turbulent kinetic energy at the same location. The 1-equation k_{sgs} LES model solves a subgrid-scale (SGS) kinetic energy transport equation:

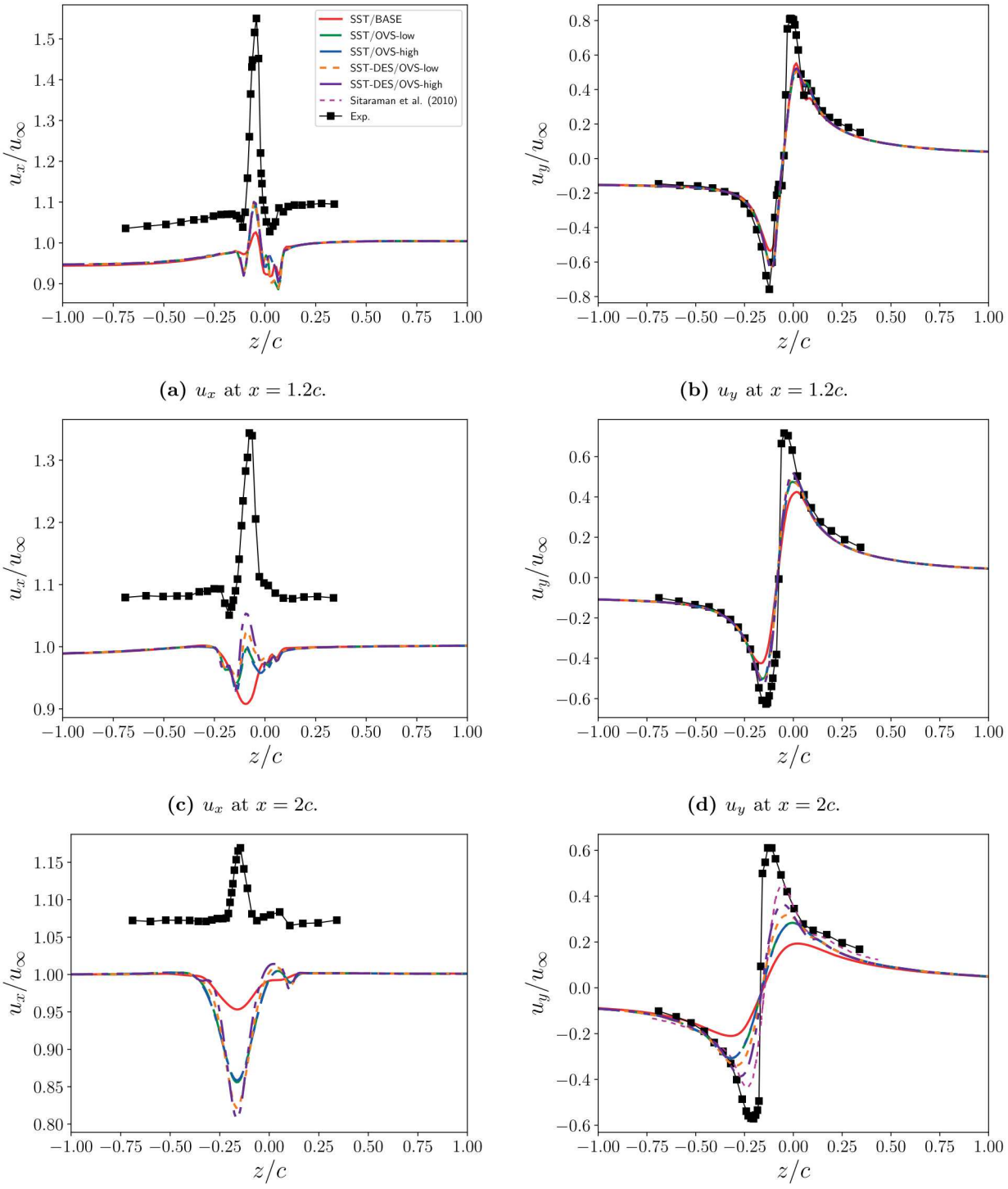
$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k u_j)}{\partial x_j} = P_k + P_b - D_k + \frac{\partial}{\partial x_j}\left(\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right), \tag{29}$$

$$P_k = \tau_{ij}\frac{\partial u_i}{\partial x_j}, \tag{30}$$

$$P_b = \beta\frac{\mu_t}{Pr}g_i\frac{\partial T}{\partial x_i} \tag{31}$$

$$D_k = \rho C_\varepsilon \frac{k^{3/2}}{\Delta}, \tag{32}$$

where $\Delta = V^{1/3}$ is a measure of the filter width, here taken to be the cube root of the volume of the cell, V, and P_b is a buoyancy source term [30]. The SGS kinetic energy, k, is used to calculate the turbulent viscosity, $\mu_t = C_{\mu_\varepsilon}\rho\Delta k^{1/2}$, with $C_\varepsilon = 0.845$ and $C_{\mu_\varepsilon} = 0.0856$. Equation 29 is very similar to the total turbulent kinetic energy equation in the SST model equations, Eq. 12. The unified hybrid RANS-LES model blends the diffusion and dissipation terms between the two to create a hybrid kinetic energy equation. The turbulent viscosity calculation is also blended between the two models along with a calculation of consistent ω in LES regions. The blending function used is $f_B = 0.5 + 0.5\tanh((loc - d)/smth)$, where d is the distance from



(a) u_x at $x = 1.2c$.

(b) u_y at $x = 1.2c$.

(c) u_x at $x = 2c$.

(d) u_y at $x = 2c$.

(e) u_x at $x = 5c$.

(f) u_y at $x = 5c$.

Figure 17: Velocity across the vortex core at different downstream locations. Solid red: SST / BASE; dashed green: SST OVS-low; dash-dotted blue: SST OVS-high; dotted orange: SST-DES with OVS-low; dash-dot-dotted purple: SST-DES OVS-high; dotted magenta: Sitaraman et al. [33]; Black squares: experimental data [24].

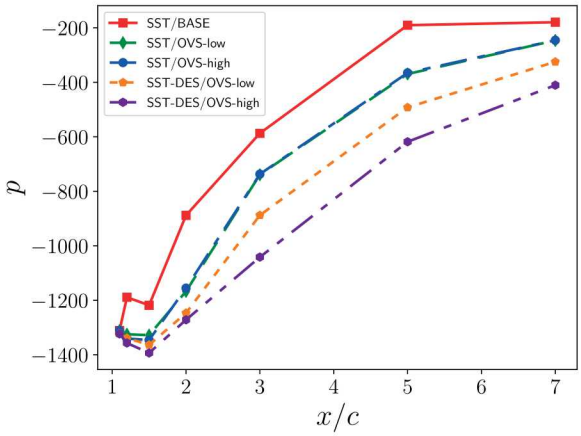


Figure 18: Pressure in the vortex core as a function of downstream location. Red squares: SST / BASE; green diamonds: SST / OVS-low; blue circles: SST / OVS-high; orange pentagons: SST-DES / OVS-low; purple hexagons: SST-DES / OVS-high.

the turbine, *loc* is a parameter controlling the location of the blending transition, and *smth* is a parameter determining how smooth the transition is. This form implies that as the turbine is approached, $f_B \to 1$ and, thus, the SST model is activated, while $f_B \to 0$ will activate the k_{sgs} model.

The blended turbulent viscosity is calculated as:

$$\mu_t = f_B \left(\frac{\rho a_1 k}{\max(a_1 \omega, S f_2)} \right) + (1 - f_B) \left(C_{\mu_\varepsilon} \Delta k^{1/2} \right). \tag{33}$$

The form of the dissipation term in both models is blended to produce:

$$D_k = f_B \left(\beta^* \rho k \omega \right) + (1 - f_B) \left(\rho C_\varepsilon \frac{k^{3/2}}{\delta} \right). \tag{34}$$

The 1-equation k_{sgs} model does not contain a transport equation for specific dissipation rate ω. Hence a representative value of ω is calculated from the k_{sgs} model in the LES region as:

$$\omega = f_B \omega + (1 - f_B) \frac{\sqrt{k}}{C_{\mu_\varepsilon} \rho \Delta}. \tag{35}$$

The highly nonlinear source terms in the transport equation for ω are blended out to zero in the LES region to aid numerical convergence as:

$$\frac{\partial(\rho\omega)}{\partial t} + \frac{\partial(\rho\omega u_j)}{\partial x_j} = \frac{\partial}{\partial x_j} \left((\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right) + f_B \left(\frac{\rho\gamma}{\mu_t} P - \beta \rho \omega^2 + 2(1 - F_1) \frac{\rho \sigma_{\omega 2}}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} \right). \tag{36}$$

Ongoing work is being pursued to analyze the sensitivity of the loads and the flowfield around the turbine to the parameters of the blending constants. Other approaches to blending the SST and k_{SGS} models in Nalu-Wind have been investigated by Matt Barone, based on the work by Baurle et. al [1] and extensions to SST-k_{SGS} blending by Lynch et al. [23] and Sánchez-Rocha et al. [32]. Future work will test a variety of these blending approaches along with the development of a more formal approach to the ABL–near body model interaction using the hybrid TAMS model.

6. LINEAR-SYSTEM SOLVER STACK

6.1 SEGREGATED MOMENTUM SOLVER

The baseline Nalu-Wind implementation uses a *monolithic* linear system to solve the momentum equation, i.e., the three components are solved together in a 3×3 block matrix per degree-of-freedom. The matrix,

therefore, includes cross terms that denote the linearization of the $u_i u_j$ terms in the momentum equations. It is common practice in incompressible codes (e.g., Nek5000[9] and OpenFOAM[10]) to ignore the cross coupling and solve the components as separate linear systems. This *segregated* approach has the advantage of solving a linear system that is 3× smaller than the monolithic approach, thereby incurring lower MPI communication overhead. To explore the potential benefits of the segregated approach, a segregated momentum solver, using HYPRE, was implemented in Nalu-Wind. The implementation used a single `HYPRE_IJMatrix` instance to store the LHS for all three components of the momentum system, and three separate `HYPRE_IJVector` instances to store the RHS of the individual components. The system was solved by three separate invocations to the HYPRE GMRES solver for each component.

The performance of the two approaches, monolithic and segregated, were benchmarked on three problems relevant to wind applications: a precursor simulation for the neutral atmospheric boundary layer (9 million nodes), the NASA flat plate[11] (1.2 million nodes), and the NREL 5MW G1 mesh [21] (761 million nodes). In this approach, a preconditioned GMRES iteration is applied separately for each of the velocity components (u, v, w). The momentum preconditioner is one sweep of the Hypre-BoomerAMG l_1 symmetric Gauss-Seidel smoother. The pressure solver is GMRES(20) with the full C-AMG V-cycle. A comparison of run times for the segregated approach versus the monolithic approach are given below

Mesh	MPI ranks	time steps	Monolithic (s)	Segregated (s)
ABL	360	15,000	24,959	19,934
Flat plate	108	100	175	146
5MW G1	16K	10	705	427

Table 1: Comparison of monolithic and segregated solvers.

The ABL and flat-plate simulations were performed on the NREL Eagle computer with Intel Skylake processors. The Hypre-BoomerAMG segregated solver stack is 25% more efficient for the ABL test. For the flat-plate, the segregated solve results in a 20% faster integration. In the case of the NREL 5MW G1 simulation on NERSC Cori, the segregated solver results in a 65% faster integration rate.

Efforts are underway to implement the segregated solver in the Trilinos solver stack. Trilinos has `MultiVector` which allows the solution of all three linear systems simultaneously. Compared to the current HYPRE implementation, this could yield further performance gains because it eliminates three separate calls to the linear solver for each solve.

6.2 COMMUNICATION-OPTIMAL GMRES IN HYPRE SOLVER STACK: PRELIMINARY RESULTS AND PATH FORWARD.

A low-synchronization variant of the MGS-GMRES iterative solver algorithm has been implemented in the Hypre-BoomerAMG solver stack [38]. The solver now supports both distributed-memory MPI communication and fine-grain thread parallelism through multi-GPU acceleration. This new algorithm is based on the inverse compact WY modified Gram-Schmidt algorithm with lower triangular solve and is backward stable. Depending on the version and the implementation, our low-synchronization GMRES requires either one or two global inner products while performing Gram-Schmidt orthogonalization.

The key operations implemented on the GPU are the mass inner-product $V^T w$ kernel and the mass AXPY, $z = w - V H_{1:i,i}$. We employ multi-vector MPI_Allreduce for $V^T w$. The mass IP kernel has been benchmarked at over 400 GigaFlops on the NVIDIA Volta V100 GPU. Within the Hypre-BoomerAMG preconditioner, the l_1 Jacobi smoother has been implemented for the GPU. Our implementation employs GPU device memory for these kernels.

A Hypre-BoomerAMG linear solver mini-app has been written to test linear systems based on the Nalu-Wind pressure solver and can read matrices output by the model. The multi-MPI, multi-GPU solver has been coupled to a C-AMG V-cycle preconditioner based upon an l_1 Jacobi smoother for the GPU. The GMRES solver components and smoother employ GPU device memory, versus unified, to achieve high execution

[9] https://nek5000.mcs.anl.gov
[10] https://github.com/OpenFOAM/OpenFOAM-6
[11] https://turbmodels.larc.nasa.gov/flatplate_val.html

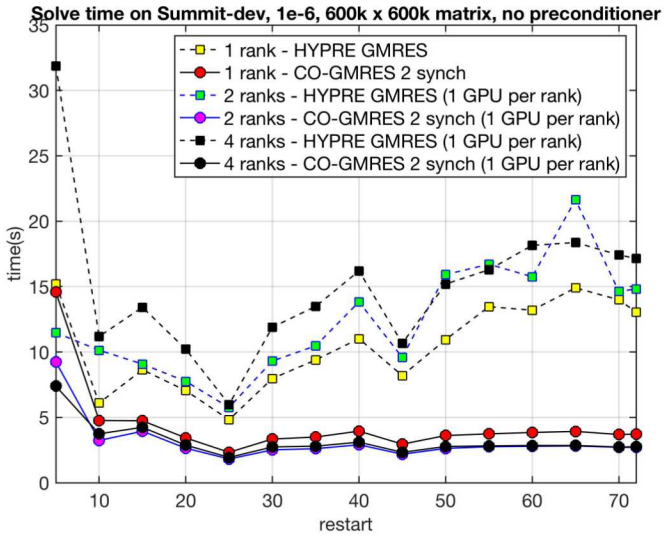


Figure 19: Comparison between scaling of regular Hypre-GMRES and low-synchronization version (CO-GMRES with 2-synchronizations). The tests were performed on one node of the Summit-dev (OLCF) computational cluster equipped with NVIDIA Pascal P100s.

speeds. Both the ABL mesh with 9M DOF and NREL 5MW G0 mesh with 95M DOF have been tested. For the ABL 9M mesh described earlier, a 10× speed-up on a Volta V100 GPU was observed and compared to the Intel Haswell processors on NERSC Cori. This comparison is based on the Nalu-Wind solver running on Cori with 360 MPI ranks (run time 0.7 sec) versus the mini-app run on Eagle using 16 MPI-ranks across eight nodes with two GPU's per node (run time 1.5 sec).

6.2.1 Scaling with the increasing size of Krylov space

Because GMRES needs to store the entire Krylov subspace it builds, the storage requirements grow with the number of iterations. The number of flops required to add a new vector to the existing Krylov subspace grows quadratically with the number of iterations. Thus, GMRES is usually restarted, with typical restart value ranging from 5 to 100. However, there are two issues with restarting. First, it slows down the convergence. Second, some problems might require a high restart value to converge at all. Hence, a GMRES version for which the time-to-solution is either independent of the restart or grows very slowly with an increase of the restart value is highly desirable.

Figure 19 displays the results of a scaling study performed for a small, dimension 600K matrix on one node of the OLCF Summit-dev machine using multiple MPI ranks. For this particular matrix it takes at least 1650 GMRES iterations to converge without a preconditioner, hence the cost of Gram-Schmidt orthogonalization dominates the computation.

In our study, each MPI rank is associated with its own GPU. The vertical axis shows time-to-solution of the linear system, which stays constant with increasing size of the Krylov subspace. If we increase the number of ranks, the time decreases. Scaling stalls at 4 ranks and this can be attributed to the small size of the problem i.e., for 4 ranks the communication takes more time than the computation. Thus, only a marginal improvement in performance is observed. In contrast, the time to solution for standard GMRES increases with the increase of Krylov subspace size, which agrees with the theory, and does not scale well.

Figure 20 shows the scaling results for a matrix coming from NREL 5MW simulation (dimension 95M). In this case, we run 2000 iterations of GMRES and 2000 iterations of low-synchronization GMRES, and measure the time for the solve and the time for Gram-Schmidt. No preconditioner is applied in this case. Again, the cost of Gram-Schmidt is dominating the computational cost.

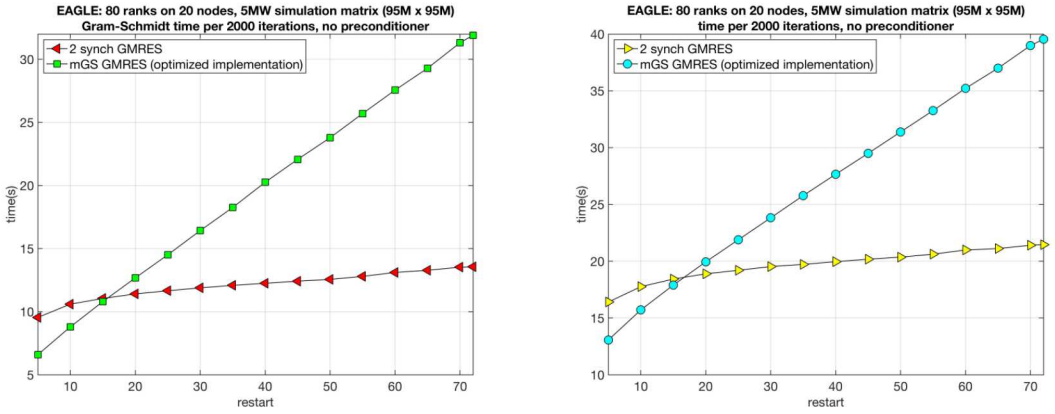


Figure 20: Comparison between scaling of regular Hypre-GMRES and low-synchronization version (CO-GMRES with 2-synchronizations). The tests were performed on 20 nodes of the Eagle (NREL) computational cluster equipped with NVIDIA Volta V100s. Left: solve time plotted against size of the Krylov space. Right: Gram-Schmidt time plotted against size of the space.

6.2.2 Scaling with the number of cores

The final tests were performed on the Eagle (NREL) computational cluster using a matrix coming from the Atmospheric Boundary Layer (ABL) problem. The matrix size is 9M. We apply GMRES preconditioned with an AMG V-cycle based on the l_1-Jacobi smoother implemented in device memory. Low-synchronization GMRES requires 38 iterations to converge, hence the solver is not restarted. The Gram-Schmidt orthogonalization times are shown in Figure 21. The total run-time on 16 MPI-ranks across eight nodes with two GPU's per node was 1.5 sec.

6.2.3 Lessons learned and path forward

While performing testing for the NREL 5MW 95M matrix on 64 MPI ranks, we obtained better results for the standard GMRES, as implemented in Hypre (which was run on GPU in unified memory). Our current implementation of the l_1-Jacobi smoother requires copying vectors from unified memory to device memory and when the matrix size is as large as 95M, the copying starts to be a bottleneck. Even the decrease in the cost of Gram-Schmidt orthogonalization does not mitigate the preconditioner cost.

In addition, parallel scaling in a complex code (such as the GMRES solver and preconditioner) is limited by the component that scales the worst. In our case, this is represented by the preconditioner application. In our current implementation, the unified memory used in Hypre forces us to copy the vectors to the unified memory and back; additionally unified memory is much slower than the device memory. Hence, we plan to work with the Hypre team to re-factor the AMG preconditioner (or at least, the relevant components) using device memory.

We must emphasize the importance of NVIDIA Multi Process Service (MPS) for performance. This service allows multiple MPI ranks to use a single GPU at the same time, without pipelining the calls. If this service is not configured properly, the scaling is limited; the best results are achieved when the number of GPUs in a node equals the number of ranks assigned to the node. Figure 22 illustrates the difference in performance for Gram-Schmidt with and without MPS enabled. The matrix used in this test is the same as for the test results shown in Figure 19. A speedup of 2.5 is achieved by enabling MPS.

As predicted in the Milestone report for FY18-Q3, low-synchronization GMRES scales reasonably well and performs well on modern computational clusters equipped with both CPUs and GPUs. The testing allowed us to identify new directions in our research, which are: coding the existing Hypre-BoomerAMG preconditioner for device memory, and finding alternative AMG algorithms and preconditioners that parallelize better on GPUs.

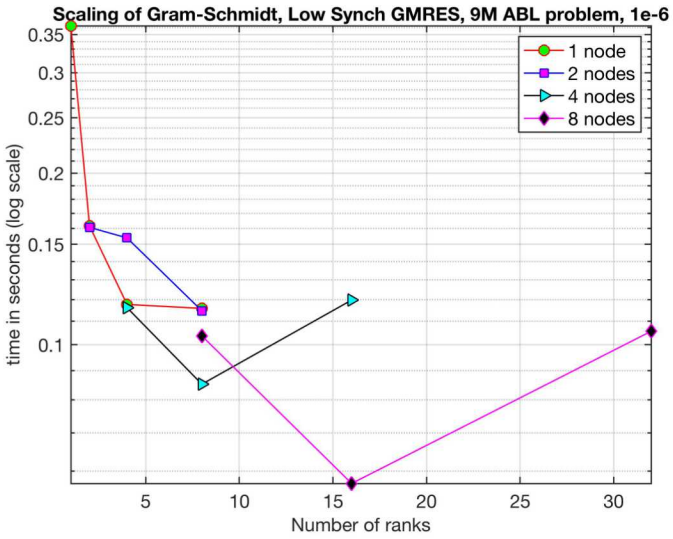


Figure 21: Scaling of low-synchronization GMRES. The tests were performed on 8 nodes of the Eagle (NREL) computational cluster equipped with NVIDIA Volta V100s. Note log scale on y-axis.

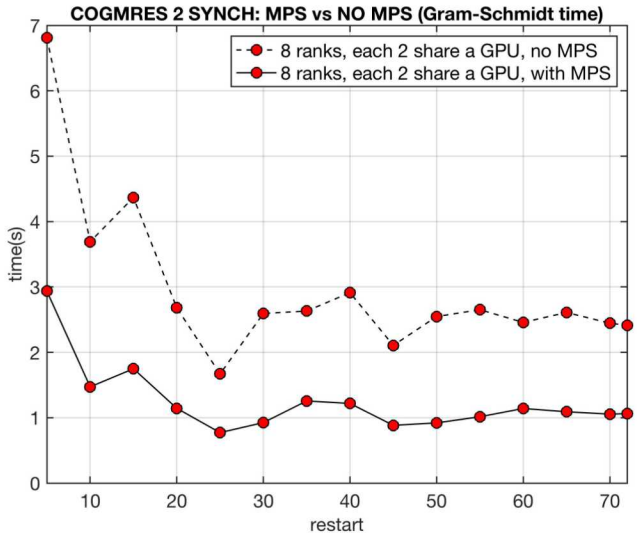


Figure 22: Comparison between the performance of Gram-Schmidt on a cluster with NVIDIA MPS turned on and off. The tests were performed on one node of Summit-dev (OLCF) computational cluster.

7. FLUID-STRUCTURE INTERACTION

Modern wind turbines use highly flexible, aeroelastically tailored blades along with an active control system to maximize power capture and reduce fatigue loads. Modeling the response of the blades, tower and the turbine as a whole to incoming atmospheric turbulence is essential to the accurate calculation of the loads on a turbine in a wind farm. The ECP ExaWind challenge problem is to predict the power output of a wind farm with multiple wind turbines operating in a realistic atmosphere. To achieve this objective, Nalu-wind is coupled to OpenFAST, a whole-turbine simulation code that includes models for nonlinear deflections of blades, tower and an active control system.

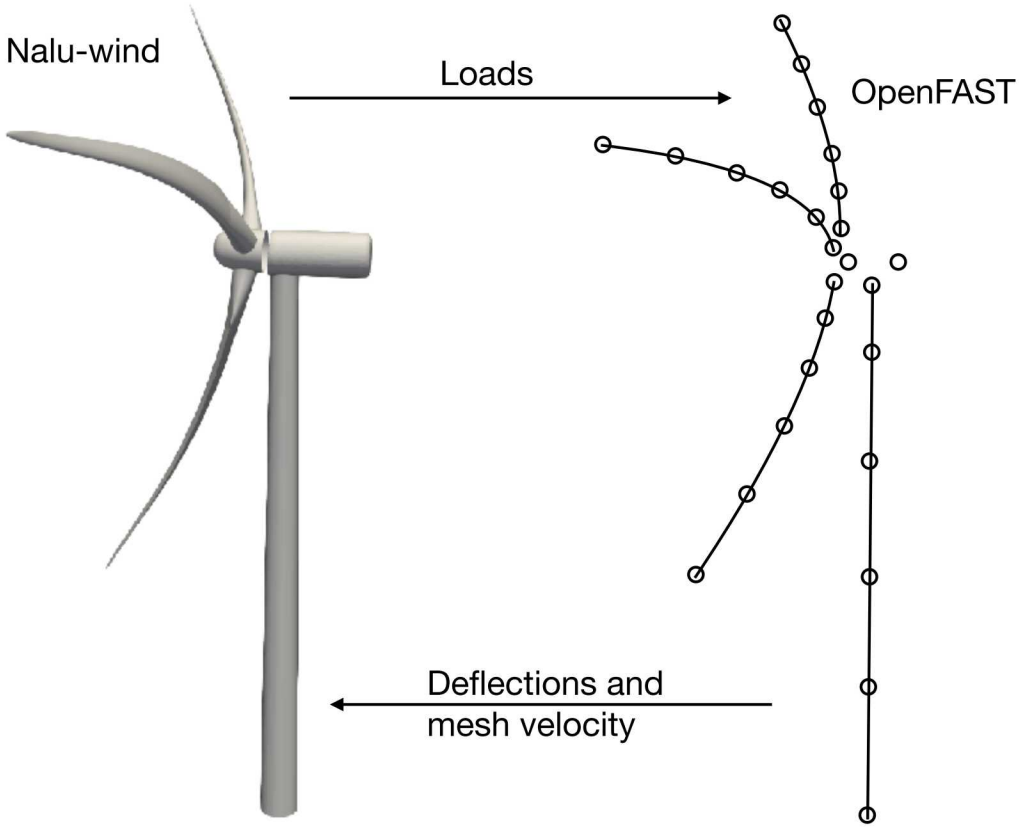


Figure 23: Overview of fluid-structure interaction framework in nalu-wind coupled to OpenFAST

Figure 23 gives an overview of the fluid-structure interaction framework in Nalu-wind coupled to OpenFAST. Nalu-wind and OpenFAST are loosely-coupled with independent time-stepping schemes. Nalu-wind provides the loads on the blades, nacelle and tower to OpenFAST. OpenFAST computes the response of the turbine as a whole to the input loads and provides the deformations and velocities to Nalu-wind. OpenFAST models the blades and the tower as slender beams along with point masses for the nacelle and hub. Optionally the blades can be represented using geometrically exact beam theory that allows for six degrees of freedom at all nodes. However, Nalu-wind resolves the geometry of the blade and tower surface. We have implemented surface-line and line-surface mapping algorithms to transfer the loads and deflections between Nalu-wind and OpenFAST. The mapping algorithms work in parallel across several processors and will be described later in this section.

Figure 24 describes the conventional-serial-staggered algorithm for fluid-structure ineraction by Lesoinne and Farhat [22]. We use the above scheme along with a specified number of "outer" or nonlinear iterations to couple Nalu-wind and OpenFAST. The choice of the number of nonlinear iterations as well as a convergence criterion will be developed in the future for wind-energy problems.

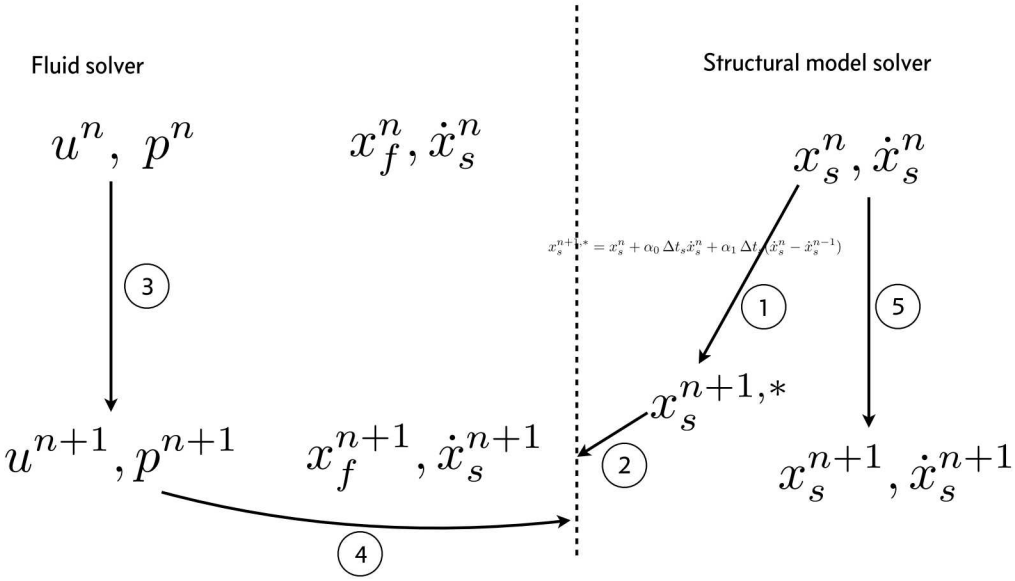


Figure 24: Conventional Serial Staggered scheme for Fluid-structure interaction from Lesoinne and Farhat [22].

7.1 MAPPING ALGORITHMS

Figure 25 describes the line-surface displacement mapping algorithm to transfer the displacements and velocities from OpenFAST to Nalu-wind. OpenFAST uses 3 degrees of freedom (DOF) in translation and rotation to describe the full displacement of each mesh node. The mapping algorithm assumes that airfoil cross sections remain rigid to convert the 6-DOF displacements in OpenFAST into 3-DOF translational displacements for Nalu-wind. Similarly, the 6-DOF translational and rotational velocity in OpenFAST is converted into a 3-DOF translational velocity in Nalu-wind. Figure 26 describes the coordinate system and the procedure used to perform this conversion. Each node on the CFD mesh is mapped to nearest OpenFAST line element through line-normal projection at initialization in the reference configuration. This mapping is then stored and used throughout the simulation.

While Figure 25 shows the mapping of the displacements to the surface of the blades, the mapping procedure is extended to the entire volume mesh block surrounding each turbine part. The fluid-structure interaction simulations are designed to be run with overset meshes where each mesh part will move independent of the other parts.

Figure 27 describes the conservative surface-line mapping algorithm to transfer the loads from Nalu-wind to OpenFAST. The surface stresses are converted to point loads at CFD (Nalu-Wind) nodes on the blade/tower surface mesh as shown in figure 28. The CFD mesh node to OpenFAST line element mapping for displacements is reused for the loads. The point load at each CFD mesh point is distributed between two nodes of an OpenFAST line element in a variationally consistent manner. The 3-DOF loads on a Nalu-wind surface mesh node is converted to 6-DOF forces and moments on the neighboring OpenFAST line-mesh element.

We verified the mapping of loads and displacements between Nalu-wind and OpenFAST using a simple

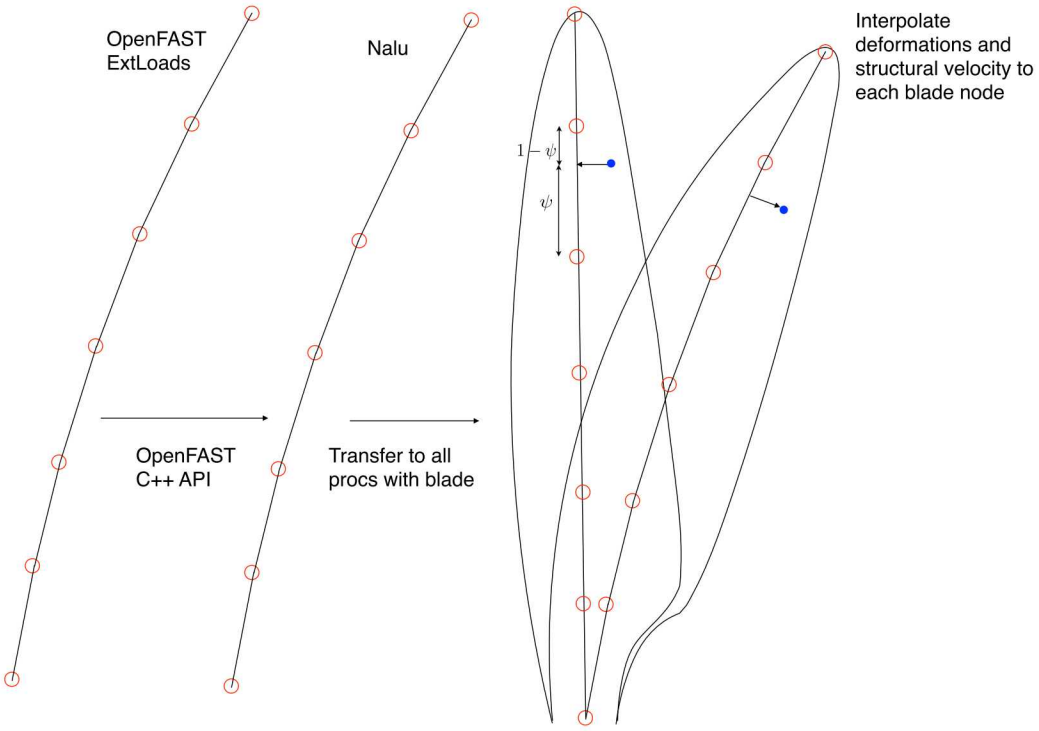


Figure 25: Illustration of displacement mapping algorithm

turbine model with square-cross-section blades. The geometry-resolved CFD mesh in Nalu-wind is consistent with the OpenFAST model of the dummy turbine. We populated the OpenFAST mesh with a prescribed displacement $\boldsymbol{d}$ and rotation in the local frame of reference along with a prescribed translational velocity $\boldsymbol{v}$ and rotational velocit $\boldsymbol{\omega}$ as shown in equations 37-40 where r is the radial position on the blade.

$$\boldsymbol{d} = 15.0 \sin(\Omega t) \sin\left((r/r_{\max})^2\right) [1.0, 1.0, 1.0] \tag{37}$$

$$\text{Rotation} = 45.0^\circ \sin\left((r/r_{\max})^2\right) [1.0, 1.0, 1.0] \tag{38}$$

$$\boldsymbol{v} = 3.743 \tan\left((r/r_{\max})^2\right) [1.0, 1.0, 1.0] \tag{39}$$

$$\boldsymbol{\omega} = 6.232 \sin\left((r/r_{\max})^2\right) [1.0, 1.0, 1.0] \tag{40}$$

The displacements and velocities are mapped from the OpenFAST mesh to the Nalu-wind CFD mesh. The error in the mapped displacements and velocities are computed against the direct application of the analytical displacements and velocities to the CFD mesh. Figure 29 shows that the L_2 norm of the error in mapping converges at a second-order rate as the OpenFAST mesh is refined. We also verified that the total force and moment about the hub in OpenFAST matched that computed on the Nalu-Wind CFD surface mesh.

7.2 DEMONSTRATION

We ran a fluid-structure interaction simulation using Nalu-wind coupled to OpenFAST model of the NREL 5-MW turbine in uniform inflow of 8 m/s in a small domain. The OpenFAST model is set to start at an initial rotor speed of 12 rotations per minute that is higher than the design rotor speed for the inflow conditions. Figure 30 shows the time history of the rotor speed and generator power from the FSI simulation. Figure 31 shows the time history of the flap-wise deflection of blade 1 along with a visualization of the deflection of

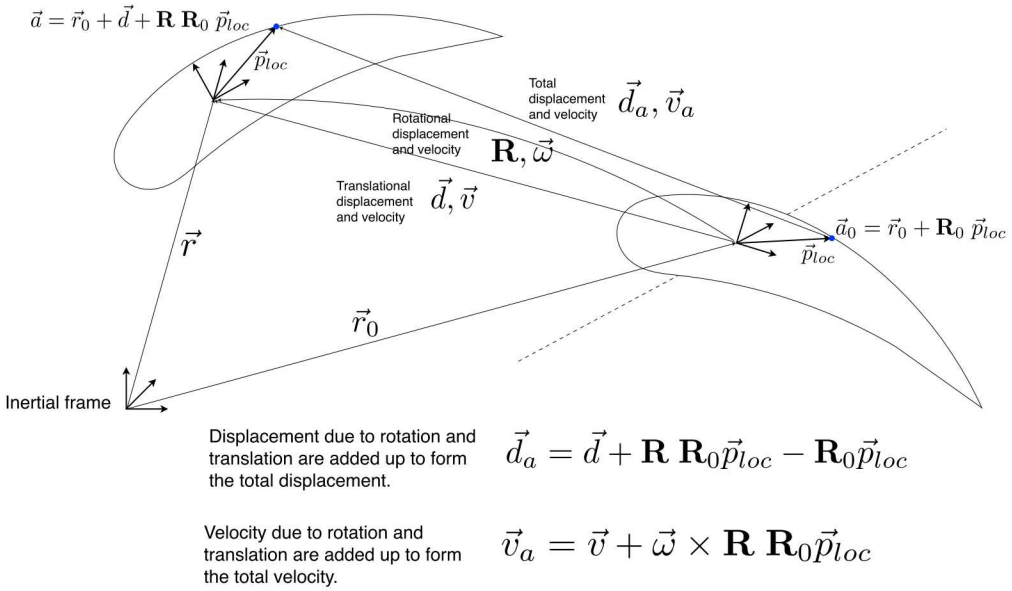


Figure 26: Displacement mapping algorithm - Coordinate system

the blade surface mesh. Future work will focus on validation of the FSI capability as a part of IEA task 29 project.

8. MOVING TO NEXT-GENERATION ARCHITECTURES: STATUS AND PLANS

Our primary next-generation-platform effort is focused on porting the Nalu-Wind code to GPU-accelerated platforms. There are several significant algorithms in Nalu-Wind, including edge-based and element-based algorithms, as well as mesh-motion and mesh-oversetting algorithms.

We have chosen to focus on the element-based algorithms first. Within this path through the code, the computational expense is mostly due to assembly of linear systems (which involves traversing the mesh and executing computational kernels at each element, then adding contributions to sparse matrix and vector objects), and then setup and execution of the linear-system solvers provided by Trilinos or Hypre. Additionally, there are "outer iterations" associated with the non-linear solve, and also operations involving geometric search and ghosting of mesh elements to handle the needs of moving-mesh simulations, through mesh overset operations, etc.

Since using GPUs means using separate memory spaces, code which is converted to run on GPU depends on data being copied to the GPU. Thus if the program is only partially converted, then data must be repeatedly copied back and forth from the CPU host to the GPU accelerator. The ramification of this is that performance won't be very good until we have completed the conversion of entire phases of the computation, to minimize the expense of data copying.

To obtain good GPU performance for a static case with no mesh motion, we need to complete the first 3 items in the following table.

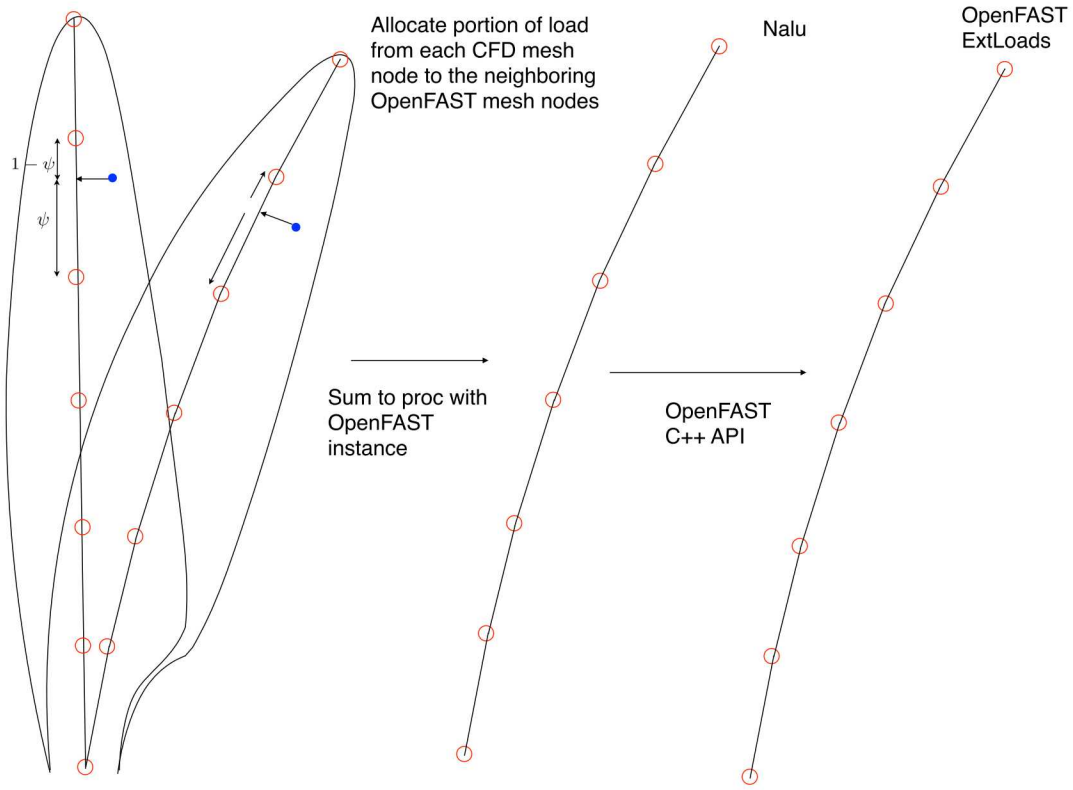


Figure 27: Illustration of load mapping algorithm

Code Area	Est. Completion
1. Assembly a): Mesh traversal, Kernel algorithms	In progress, FY19, Q2/Q3
2. Assembly b): matrix/vector contributions, solve/precond	Up next, FY19, Q3/Q4
3. Non-linear solver, coordination, etc	FY20, Q1
4. Non-element-based algorithms	FY20, Q1/Q2
5. Mesh motion: ghosting, graph initialization, etc	FY20, Q3/Q4

To obtain good GPU performance for a case involving mesh motion, we also need to complete the items in 5. It is important to note that the non-element-based algorithms are not dependent on the element-based algorithms, and GPU-conversion of those algorithms could be performed at the same time, depending on person-resources.

We are following a rigorous approach of converting the code "piece by piece", adding unit-tests at each step to establish confidence and to prevent regression and side-effects.

8.1 STK AND NGP

Nalu-wind is utilizing Sandia's STK-Mesh libraries for unstructured-mesh storage and manipulation, and the related NGP-Mesh for utilizing unstructured-mesh data on the GPU.

The conversion of mesh-motion capabilities such as ghosting etc., is less certain (higher risk), and has a stronger dependence on resource constraints including resources of Sandia's STK team.

The Nalu-Wind project is already utilizing threaded (OpenMP) architectures, such as KNL (Intel's Knights Landing). Previous reports have outlined improvements in performance of linear-system assembly, as well as solve times. Some issues remain, such as sub-optimal thread-scaling for certain problems with

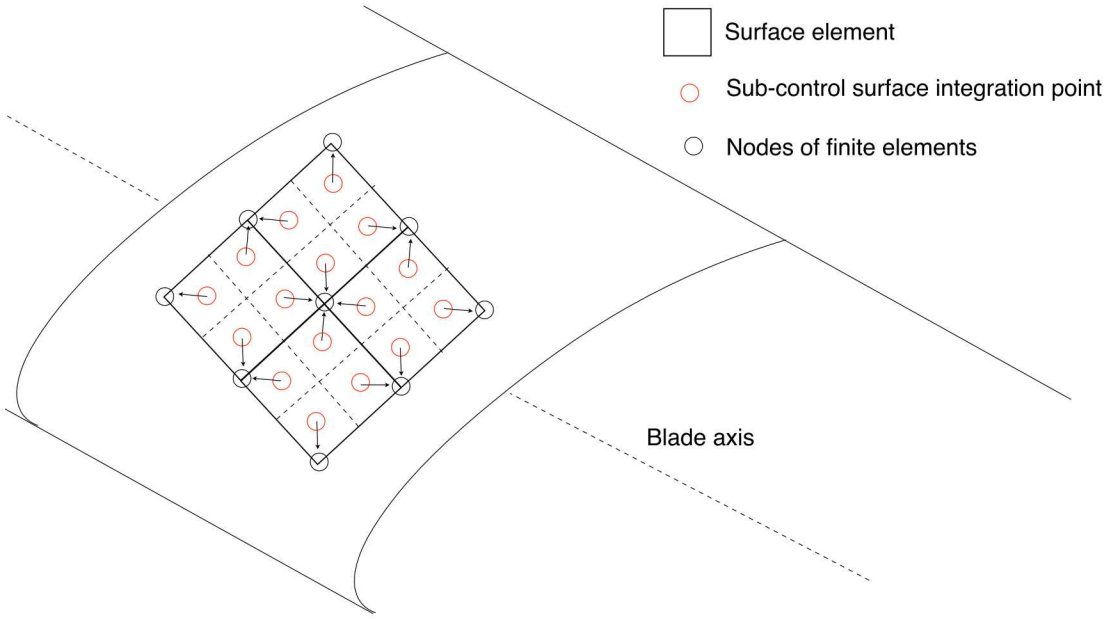


Figure 28: Assembly of loads on each surface element into point-loads on the nodes.

mixed-topology meshes. We are pursuing strategies involving mesh-coloring to enable us to avoid the use of atomic updates during thread- parallel assembly. (Without coloring, atomic updates are required in order to avoid write- conflicts among threads operating on neighboring elements.)

8.2 TRILINOS AND NGP

The Trilinos solver stack is designed purposely to be able to leverage Kokkos [13] to achieve portable and high performing shared memory parallelism. In the following discussion, it is important to keep in mind that all Trilinos code has been or will be written using Kokkos, *not raw CUDA*. Thus, the Trilinos solver stack will be portable between CPUs, NVIDIA GPUs, and future accelerators such as those planned for Aurora.

The momentum linear system in Nalu-Wind simulations is solved via GMRES preconditioned by symmetric Gauss-Seidel. The pressure linear system is solved with GMRES preconditioned by algebraic multigrid (AMG). In contrast to the simple momentum preconditioner, the AMG preconditioner has a nontrivial setup phase.
The main computational kernels for the AMG preconditioner setup are as follows.

1. **Aggregation** Aggregation is the process of grouping fine level unknowns, where each group forms a coarse level unknown. Grouping starts with the selection of so-called root degrees-of-freedom (DOFs) such that they are distance-two from each other. Once the root DOFs are chosen, then each remaining non-root DOF is grouped with a single root DOF, based on matrix-stencil connectivity.

   Current status: Root node selection is NGP ready in Kokkos-Kernels via a distance-two maximal independent set algorithm. Aggregation is NGP ready in MueLu.

2. **Creation of Interpolation Matrices** The interpolation matrix (often called a "smoothed prolongator") is created in a two-step process. In the first step, an initial "tentative" prolongator is created. The main computational kernel, which can often be bypassed for scalar systems, is a sequence of small local QR decompositions. In the second step, the tentative prolongator is improved with a single step of damped Jacobi, which entails a sparse matrix-matrix multiply and sparse matrix-matrix addition.

   Current status: The interpolation matrix construction is NGP ready, with the exception of the local QR decompositions. However, these can skipped for scalar systems. Batched versions of the QR decompositions will be ready in Kokkos-Kernels in FY20.

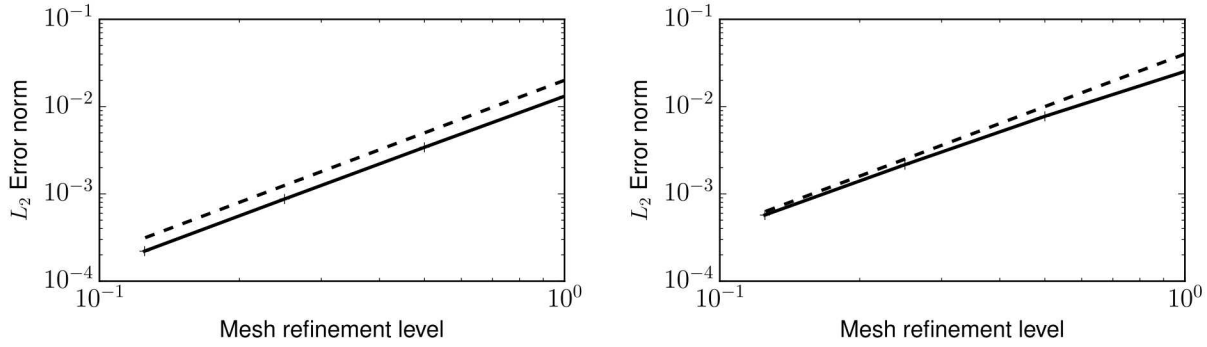


Figure 29: L2 error of the CFD (a) mesh displacement and (b) mesh velocity as a function of OpenFAST output-mesh element size. The dashed line shows second-order convergence.

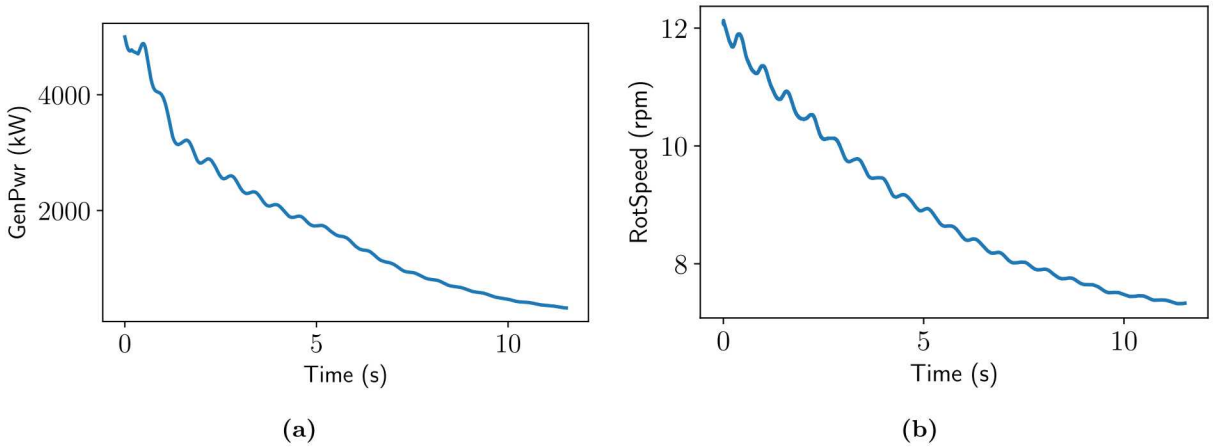


Figure 30: Time history of (a) rotor speed and (b) generator power from FSI simulation of NREL-5MW turbine in uniform inflow.

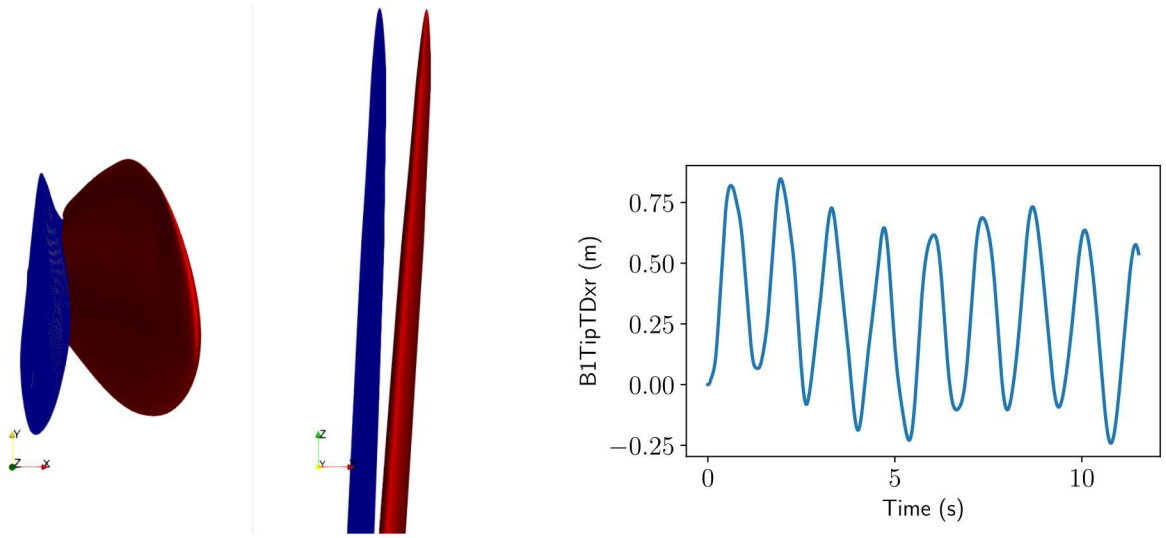


Figure 31: (a) Visualization of blade deflection at $t = 0.6s$ and (b) time history of x displacement (flap-wise deflection) of the tip of blade 1 from FSI simulation of NREL-5MW turbine in uniform inflow.

3. **Creation of Restriction Matrices** For the Nalu-Wind pressure matrix, the restriction matrix is the transpose of the interpolation matrix. Restriction can either be applied implicitly, or explicitly by transposing the interpolation matrix.

   Current status: The explicit transpose of the interpolation matrix is NGP ready.

4. **Creation of Coarse Grid Matrices** A coarse grid matrix A_{l+1} is formed with a sparse triple matrix product operation, $A_{l+1} = R \times A_l \times P$, where R and P are the restriction and prolongation matrices, respectively. The triple matrix product requires two sparse matrix-matrix products (SPGEMMs). The first SPGEMM is $A_l \times P$, and the second SPGEMM is $R \times (A_l \times P)$. Each SPGEMM has a communication phase and a computation phase.

   Current status: The shared-memory computation phase is supplied by Kokkos-Kernels [7] and is NGP ready. The communication phase is largely complete, and any remaining gaps will be finished by FY20/Q2.

5. **Rebalancing of Coarse Grid Matrices** Each coarse matrix in the AMG hierarchy has fewer rows and nonzeros than the previous fine matrix. This leads to coarse levels that are communication bound. Additionally, the coarse matrix may be more imbalanced than the previous fine matrix. MueLu's AMG setup has a rebalancing phase that redistributes the coarse matrix to a subset of processors. This phase also ensures that all participating ranks have an approximately equal number of nonzeros.

   Current status: The calculation of the mapping of rows to new partition number is provided by Zoltan2 and is currently done on the CPU. The Zoltan2 team plans to have a base version of the multijagged algorithm for GPUs ready in FY20.

6. **Creation of All-but-Coarse-Level Smoothers** The main smoothers used are (symmetric) Gauss-Seidel, ℓ_1 Gauss-Seidel, ℓ_1 Jacobi, and Chebyshev (polynomial). ILU(k) (incomplete factorization) smoothing is also available, although we've found it unnecessary.

   Current status: The main setup kernel in Gauss-Seidel (standard and $\ell - 1$ variants) is coloring of unknowns, where unknowns of the same color can be updated independently from each other. This kernel is implemented in Kokkos-Kernels and NGP-ready. Chebyshev requires only an estimate of a largest eigenvalue, and this is NGP ready. Jacobi has no heavy setup requirements. ILU(k) setup

is NGP ready. It is available in Kokkos-Kernels and is based on the iterative method proposed by Chow [5, 28].

7. creation of coarsest-level solver MueLu will typically use a sparse-direct solve on the coarsest AMG level. It is possible, however, to use an iterative solver (Jacobi, Gauss-Seidel, Chebyshev, Krylov, or ILU(k)). Using such a method will degrade algorithmic convergence.

   Current status: An iterative triangular solve as proposed by Chow[4] is available in Kokkos-Kernels. Trilinos also has the aforementioned NGP-ready iterative solvers.

The application of GMRES with AMG or Gauss-Seidel preconditioning, i.e., solving the linear system, relies on the following kernels:

1. **Sparse matrix-vector multiplication** This is a main kernel in both GMRES and in the application of the AMG preconditioner.

   Current status: This is provided by Tpetra and Kokkos-Kernels and is NGP-ready.

2. **multivector norms and inner products** This kernel is primarily used in GMRES.

   Current status: This is provided by Kokkos-Kernels and is NGP-ready.

3. **triangular solve** This kernel is necessary for either smoothing with ILU(k) or performing a direct solve of the coarest matrix in the AMG preconditioner.

   Current status: An iterative triangular solve as proposed by Chow[4] is available in Kokkos-Kernels.

Finally, the PEEKS project (Milestone STMS11-8) is actively developing communication avoiding and pipelined solvers within Trilinos. For more details, see [17].

8.3 HYPRE AND NGP

The solution of linear systems using the Hypre-BoomerAMG solver stack from LLNL consists of several distinct phases. These may be accelerated through the introduction of CUDA kernels for NVIDIA GPUs such as the Volta V100. The Nalu-Wind and Exawind team is collaborating with the HYPRE team at LLNL in order to implement these different solver phases efficiently on GPU accelerators. For a linear system $Ax = b$ based on the Nalu-Wind finite volume discretization, the coefficient matrix must first be assembled, where the element mesh and sparse matrix are distributed across the MPI ranks.

For a Hypre sparse matrix, an MPI rank will contain a certain number of rows, represented in a diagonal block. Matrix elements with column indices shared with other MPI ranks are contained in an off-diagonal block. Matrix assembly adds together local mesh element contributions to these blocks as well as those from other processors. Within an MPI rank, the current design of matrix assembly is embodied in the Hyper-BoomerAMG function 'addToValues'. For GPU threaded assembly, issues such as how individual threads can update row elements must be addressed. In addition, we must examine how to handle the situation when additional elements are added to the blocks owned by an MPI rank. In order to reduce preconditioner set-up time, the matrix assembly may be pipelined with the strength of connection matrix construction in HYPRE-BoomerAMG.

In order to prepare the Nalu-Wind solver stack(s) for next-generation platforms (NGP) and GPU accelerators, several algorithm changes may be required. Our initial experiences with the Volta GPU on the ORNL SummitDev and NREL Eagle machines have demonstrated much higher performance is possible by employing GPU 'device' memory as opposed to 'unified' GPU-CPU memory which relies on memory page migration.

Our Hypre-BoomerAMG solver stack will be based on the GPU implementation of the Low-Synchronization MGS-GMRES, AMG V-cycle and smoothers. The specific areas where further work is required are described below

1. Both the momentum and pressure linear systems are solved using a preconditioned GMRES iterative solver for $Ax = b$. The momentum system matrix is nonsymmetric and currently requires a single level preconditioner such as symmetric Gauss-Seidel or incomplete ILU factorizations. In both cases

a sparse lower and upper triangular solver is required in the preconditioner and represents the main computational cost together with matrix-vector multiplication. These are both suitable for GPU acceleration. However, modifications to the conventional CPU algorithms are required.

(a) Matrix-vector multiplies may employ the NVIDIA 'cuSparse' library with Compressed Sparse Row (CSR) storage. Indirect addressing can affect the speed (e.g. Hypre splits CSR into on MPI rank local A_{diag} with on rank columns and A_{offd} with non-local columns). Optimal storage formats for sparse matrices copied to the GPU memory may lead to increased speed and could be explored.

(b) Low-Synch GMRES based on inverse compact WY modified Gram-Schmidt have demonstrated 10× speed-up on Volta GPU compared to the Intel Haswell processors on NERSC Cori. The basic operations on the GPU are the $V^T w$ kernel and MAXPY: $z = w - V\, H_{1:i,i}$. We employ multi-vector MPI_Allreduce for $V^T w$.

2. Momentum Preconditioner(s)

The current momentum system preconditioner is based on either an l_1 symmetric Gauss-Seidel relaxation or an incomplete ILU factorization with sparse lower-upper triangular solves. These can be accelerated as follows on the GPU:

(a) Employ the traditional level-scheduled triangular solve implemented in the 'cuSparse' library by NVIDIA. The algorithm is limited by the available fine-grain parallelism.

(b) Implement the triangular solve as a Jacobi iteration following the work of Edmond Chow, [4]. Convergence can be fast and blocking strategies further enhance the convergence rate.

(c) The resulting iteration is equivalent to a Neumann polynomial preconditioner.

3. Pressure Preconditioner(s) and the AMG V-cycle

The classical Ruge-Stuben (C-AMG) coarsening algorithms are implemented. In order to increase the speed of the C-AMG preconditioner on the GPU, both the set-up phase and components of the V-cycle hierarchy must be accelerated. The set-up phase is particularly important for Nalu-Wind because the matrices are re-assembled and preconditioners are re-computed every time step. The set-up consists of the coarsening algorithms to construct the coarser level matrices at each level of the V-cycle. The Hypre team has begun or plans to accelerate the following set-up algorithm components

(a) Determine strength of connection matrix.

(b) Parallel Multiple Independent Set (PMIS) graph algorithms.

(c) Interpolation to form the prolongation P and restriction R matrices.

(d) $A_c = RAP$ matrix products to form the coarse matrices

4. Pressure preconditioner V-cycle:

The basic components of the V-cycle are as follows:

(a) Each level of the V-cycle requires sparse matrix-vector products to apply the prolongation and restriction operators. These are crucial for speed as they dominate the cost due to their high operation counts.

(b) We may also find the optimal storage format for these matvec's in the V-cycle. This promotes coalesced memory fetches on the GPU.

(c) Residual $r = b - A_c x$ computation as a matrix-vector multiply and subtraction.

(d) The application of the smoother must be accelerated. Currently only the l_1 Jacobi smoother is available on the GPU. This smoother is less effective than symmetric Gauss-Seidel and leads to higher GMRES iteration counts.

(e) Therefore, Gauss-Seidel smoothers can be implemented on the GPU either with a lower-triangular solver type algorithm (recurrence) or an alternative is to implement these by using Edmond Chow's idea [4] create a type of Neumann polynomial smoother for the GPU.

The focus would be placed on accelerating the most critical and time-consuming AMG components described above. Initially, these are the *RAP* products and smoothers. However, set-up costs are important for the Nalu-wind application and need to be addressed. Our time horizon is at least one year to complete these steps, with the smoothers and matrix-matrix products taking priority.

8.4 TIOGA: TRANSITION TO NGP

Extending simulation algorithms to General Purpose Graphical Processing Units (GPGPU) is central in the pursuit of development of exascale applications. The search algorithm in TIOGA incurs the biggest expense and provides the biggest opportunity for acceleration on GPU based systems. Re-implementation of all of TIOGA algorithms to execute on GPUs is a tedious endeavor involving a large code base. In order to facilitate this process, a smaller mini-application that executes the search algorithm alone is created. The mini-app created is named PIFUS[12] which stands for Parallel Interpolation For Unstructured Sets. PIFUS is created to be used as a sand box for testing, verifying and assessing capabilities of GPUs. The data that TIOGA takes as input includes the coordinates of mesh points and the connectivity graph of elements. In PIFUS, this is simplified further and the connectivity graph is omitted. Instead PIFUS facilitates interpolation from just a point cloud of data to another point cloud of data with no notion of elements or nodal connectivity. This in itself is a powerful capability and can be leveraged for obtaining extracts from large data sets.

At the core of PIFUS is an alternating digital tree (ADT) based search algorithm that is very similar to the one implemented in TIOGA. Instead of a cell containment search, PIFUS locates 8 closest points (one in each octant) around each target point. These 8 points are used to construct an interpolation basis for each target point. Several interpolation techniques such as (a) Radial Basis Functions (RBF) (b) Kriging and (c) Weighted least squares are implemented. The ADT algorithm was enhanced to be octant constrained using a specialized divide-and-conquer technique that eliminates tree nodes based on octant visibility. Location of 8 closest donor points that surround the target point results in a convex interpolation scheme where no extrapolation is necessary.

The major operations in PIFUS are tree-construction, search and interpolation. Tree-construction is a sequential process and is not easily amenable to many-core GPU application. Tree-construction however incurs lesser computational cost compared to the search process itself. Currently both search and interpolation are implemented to be GPU (device) compatible in PIFUS with the tree construction still performed on the CPU (host). Searching of the alternating digital tree is usually accomplished using a recursive function. Recursions are however not supported by CUDA on device specific functions. Therefore, the recursive search routine had to be rewritten using a stack based approach that maintains a stack that tracks all the child nodes that need to be tracked after a particular level of the tree is complete. The non-recursive implementation was first verified against the standard recursive implementation on the CPU. Then the same code was rewritten to be executable on the GPU. Figure 32 shows the performance comparison of the search algorithm executing on GPUs compared to CPU (host). The computations were performed on the Summitdev system at the Oakridge National Laboratory. Summitdev is an eary access system that is one generation removed from OLCF's system, Summit. The host (CPU) calculations were performed with 20 MPI ranks running 8 OpenMP threads, and the device (GPU) calculations were performed on 1 NVIDIA Tesla P100 card. Figure shows a near linear increase in the search time with increasing number of target search points. Computations on GPU maintain a steady $\approx 1.7\times$ speedup over the CPU timings.

9. FULL-PHYSICS SIMULATION OF A SINGLE NREL 5MW TURBINE OPERATING IN ABL INFLOW

The simulations and results discussed in the previous sections focused on particular physics aspects to demonstrate the capability of the models implemented in Nalu-Wind to accurately predict the various phenomena relevant to a wind turbine operating in turbulent inflow conditions. The demonstration simulation for the FY19-Q2 milestone combines all the different physics models in one simulation that is the closest representation of conditions and responses experienced by an isolated turbine operating in flat terrain. The generation of the appropriate inflow conditions using a precursor simulation is first discussed, the details of

[12] https://github.com/jsitaraman/pifus

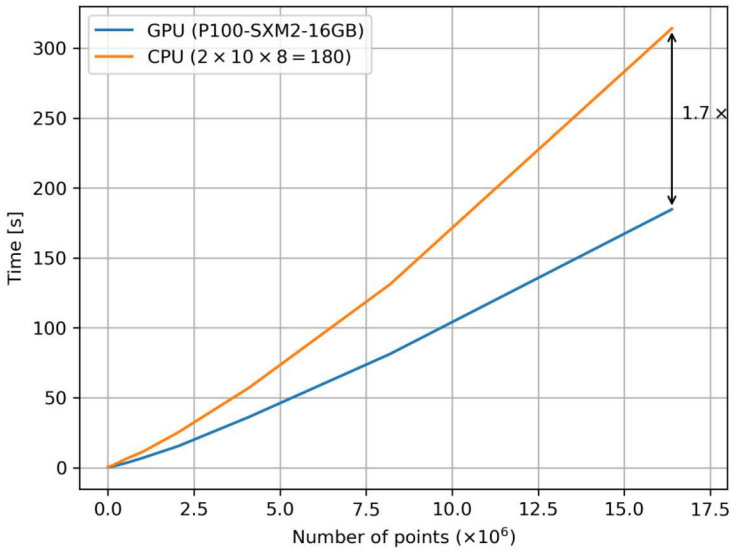


Figure 32: Comparison of execution times on GPGPU vs. CPU (MPI+OpenMP) for point cloud search algorithm as a function of the target search points. Computations performed on ORNL's SummitDev system where the CPU computations used both sockets (20 MPI ranks each with 8 OpenMP threads), and the GPU computations used one NVIDIA Tesla P100 card.

the computational domain and mesh configuration is presented next. This is followed by the details of the simulation setup and the preliminary results from the full-physics simulation and the next steps based on the observations.

9.1 ABL PRECURSOR SIMULATION

Simulations of a wind turbine in operating conditions requires the presence of inflow turbulence. In prior Nalu-Wind simulations [21], this was represented through the use of an external homogeneous isotropic turbulence (HIT) database superimposed onto a background uniform flow to meet a specified turbulent intensity. Initial conditions and a consistent HIT inlet boundary condition were used to represent and maintain the turbulence. To facilitate a more realistic representation of atmospheric turbulence, the capability to utilize a precursor atmospheric boundary layer (ABL) simulation has now been developed for Nalu-Wind.

The demonstration discussed in this section, consists of an NREL-5MW turbine overlaid on a background $1 \times 1 \times 1$ km ABL mesh, with a uniform 10m spacing. To perform a precursor ABL simulation for use on this mesh, a uniform 10m ABL simulation was conducted on a domain measuring $3 \times 3 \times 1$ km. The ABL precursor was run using neutral stratification (i.e. 0 Km/s ground heat flux) and an initial mean velocity of 8 m/s with perturbations seeded near the terrain. A body force source term is added to the momentum equations to maintain a 8 m/s wind speed at hub height ($z = 90$m for the NREL 5MW), oriented at a compass direction of 230°. The temperature profile is initialized to a constant 300K up to a height of 650m, where it is linearly increased to 308K at a height of 750m and then a constant temperature gradient of 0.003K/m is imposed above 750m. Periodic boundary conditions are used for the lateral boundaries, the terrain wall boundary is based on Monin-Obukhov theory [31] using a rough-wall surface with a roughness length of $z_0 = 0.15$m for implementation of the ABL wall function and the upper boundary uses a slip wall with a specified normal temperature gradient of 0.003K/m.

The precursor simulation is run for 20000s, at which point the flow has reached a quasi-equilibrium state. Figure 33 details the converged statistics in the precursor simulation. From the left frame, it is clear that the mean velocity exhibits a vertical shear up to around 700m where the effect of the capping inversion can be seen. In addition, the expected 8m/s wind speed at hub height is observed. The right frame shows the mean velocity variances (solid line) and the range of variances (shaded region) as a function of elevation. Stronger

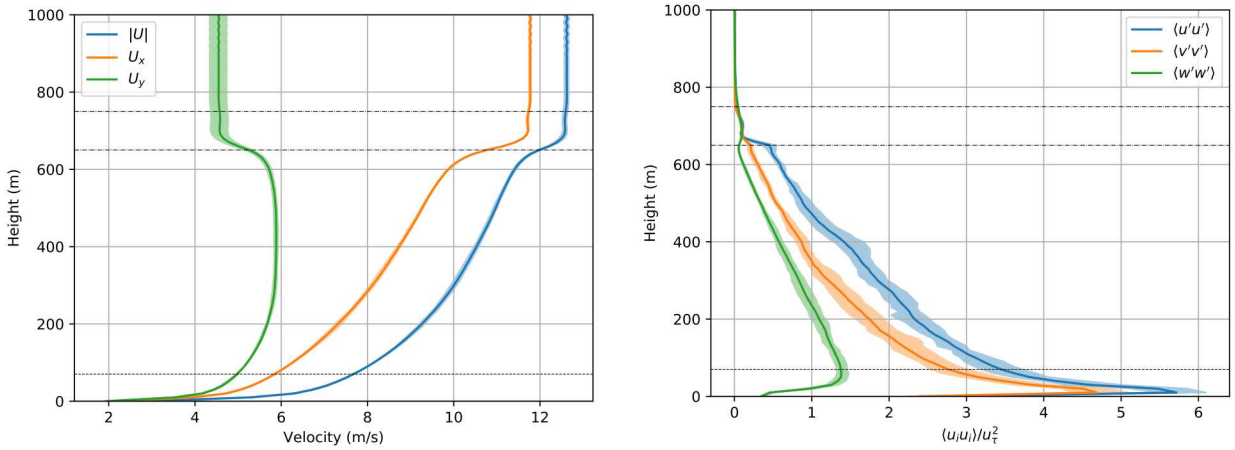


Figure 33: Velocity statistics from precursor ABL simulation (a) mean velocity profile (b) velocity variances.

turbulence is observed near the surface with an approximate turbulent intensity of 10.8% at hub height.

Once the precursor simulation has reached quasi-equilibrium, the $t = 20000$s flow field is extracted and used as initialization for the demonstration simulations. The precursor simulation is then restarted and the flow field on the "south" and "west" boundary planes is extracted at every timestep to be used as inflow boundary conditions for the turbine simulation. The ABL flow field is interpolated onto the turbine mesh by determining a bounding box of nodes on the precursor mesh around each turbine mesh point and linearly interpolating. The time dependent precursor inflow boundary conditions are specified every 0.5s (the timestep used in the precursor simulation) and intermediate time boundary conditions are found by linearly interpolating in time at each grid point. This precursor method allows the incoming flow field to the turbine to match conditions observed at the same point in space and time as found in the precursor ABL simulation.

9.2 TURBINE IN ABL INFLOW SIMULATION

The computational domain for the turbine simulation was a $1 \times 1 \times 1$ km box such that the inflow planes ("south" and "west") aligned with the precursor mesh used to generate the turbulent inflow profiles. The ABL mesh (Fig. 34a) was a structured Cartesian mesh with a uniform mesh resolution of 10 m. The outflow planes ("north" and "east") were modeled with open boundary (zero-gradient) conditions. The "terrain" and "top" boundary conditions were the same as for the precursor simulations.

All-hexahedron (HEX) body-conforming meshes were generated around the individual blades, nacelle, and tower components of the turbine – see Fig. 34b. The lessons learned from the meshes and simulation results of the NREL Phase VI (Sec. 4.3.1) and NREL 5MW (Sec. 4.3.2) were used to inform the final mesh configuration for the blade meshes for the demonstration simulation – see Fig. 34c. Much care was taken to ensure an all-HEX mesh around the aerodynamic surfaces to ensure optimal solution quality for the number of grid points used to generate the mesh. As in Sec. 4.3.2, the first point in the wall normal direction was set to 10^{-5} m on the blades. The nacelle and tower meshes used a coarser wall normal spacing. A close-up view of the nacelle and tower meshes is shown in Fig. 34d. The lower part of the tower mesh (up to ≈ 30 m) was extruded several layers in the radial direction to match the resolution of the background ABL mesh. Further, due to the use of wall functions at the "terrain" boundary, the cell span in the axial direction of the tower at the terrain-tower interface was set to five meters.

The rotor and nacelle mesh system is embedded in a structured, cylindrical wake-capturing mesh with an O-H ("butterfly") topology. The wake-capturing mesh of uniform resolution ($= 0.6$ m) was extended 0.5 radii upstream and downstream of the rotor structure. The mesh was further extruded another 1.25 diameters downstream of the turbine. An unstructured buffer region was used to extend the outer boundaries of the wake-capturing mesh to match the resolutions of the background ABL mesh. The entire system (turbine and

wake-capturing mesh) was translated such that the turbine base was located 300 m inboard from the south and west inflow faces of the ABL domain. Further, the entire system was rotated 40° counterclockwise about the $z-$ axis such that the turbine faced south-west into the incoming wind direction. The final mesh was comprised of approximately 28 million elements (24 million nodes), each blade mesh contained 3.35 million elements.

The initial conditions for the turbine simulation were mapped from a precursor simulation (§9.1) at the end of 20,000 seconds. The inflow conditions were mapped to the "south" and "west" boundaries from a time-varying inflow database that was dumped starting from 20,000 s with the precursor mesh. Figure 35 shows the flow field after 1250 time steps. The simulation is in the very initial stages and the turbine wake has not yet developed fully, therefore, the wake deficit is not evident in the flow field visualizations. The results show that the various physics models are working together as expected. The interaction of the ABL with the refined near-body meshes create a sudden change in turbulent scales and this deserves further attention. Future research will focus on the optimal settings for a blending model between the atmospheric $k-$SGS and near-body $k-\omega$ SST models. The OpenFAST outputs show that the blade is deforming as expected (see Fig. 36) under the influence of the CFD aerodynamic loads, and the controller is adjusting the RPM based on the instantaneous values of rotor torque. The simulations highlight the need for a better initialization strategy when simulating turbines in ABL inflow. The current approach of abruptly introducing a rotor in ABL flow introduces a large initial transient that pings the structural model and can result in instabilities and modes that take a lot of time to damp out. An alternative approach could be to blend a converged rotor simulation in uniform inflow with ABL inflow as the starting solution. Further work is needed to determine the optimal starting solution that minimizes the impact of initial transients in the solution.

10. CONCLUDING REMARKS AND NEXT STEPS

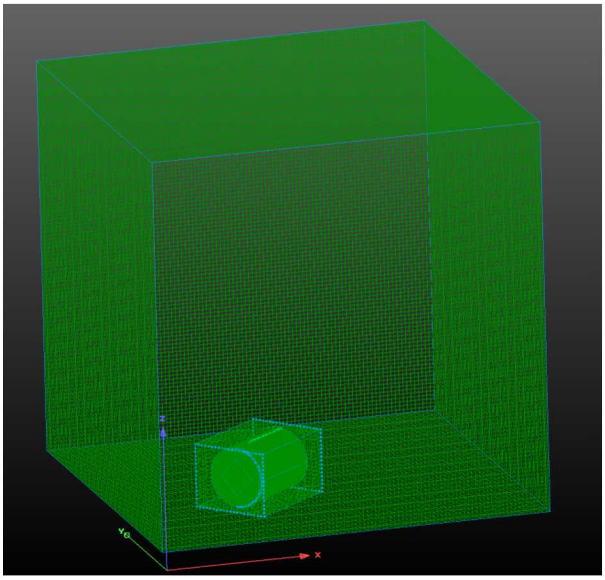
This milestone successfully demonstrated Nalu-Wind's ability to simulate a megawatt-scale turbine in turbulent atmospheric boundary layer inflow conditions. In partnership with the A2e High-Fidelity Modeling project, the simulation included a fluid-structure interaction capability by coupling to a comprehensive turbine simulation code (OpenFAST) as well as using a new timestep algorithm that enables large timesteps ($CFL \gg 1$) in the RANS regime. The simulations documented in this report also use overset-mesh methodology in production runs with great success. Overset-meshes allow the generation of high-quality, hex-dominant meshes around the aerodynamic surfaces that can be embedded within background meshes well suited for capturing wakes and ABL characteristics. The report also documents in detail the verification and validation studies performed to evaluate the validity of the models implemented within Nalu-Wind.

The major accomplishments in this milestone effort and future work are summarized below:

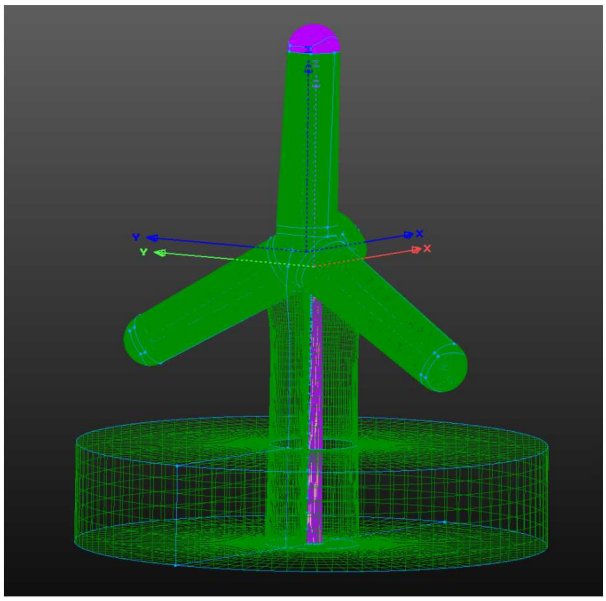
1. In partnership with the A2e HFM project, a new timestep algorithm was implemented that enables execution at large timesteps and high aspect ratio meshes necessary for hybrid RANS/LES simulations. The new algorithm was verified with code-to-code comparisons on the NACA 0012 2-D airfoil problem, the NREL Phase VI and 5-MW rotors. Results from the Phase VI simulation were also validated against available experimental data. Results show good agreement for both the integrated quantities, e.g., torque, and section aerodynamic quantities of interest, e.g., pressure coefficient as a function of chord at multiple spanwise locations.

2. The new timestep algorithm relies on under-relaxation of the linear system to recover diagonal dominance of the linear system at large timesteps. The resulting system satisfies the order of accuracy of the underlying discretization system only upon the convergence of the non-linear residuals with sufficient Picard iterations within a timestep. The simulations in this report, however, only performed a fixed number of Picard iterations per timestep without regard to the actual convergence of the non-linear residuals at each timestep. The tradeoff between accuracy of the underlying numerical scheme and the time-to-solution concerns remain an active area of research for the ExaWind team. We continue to explore approaches to improve the solver such that the need for underrelaxation can be minimized.

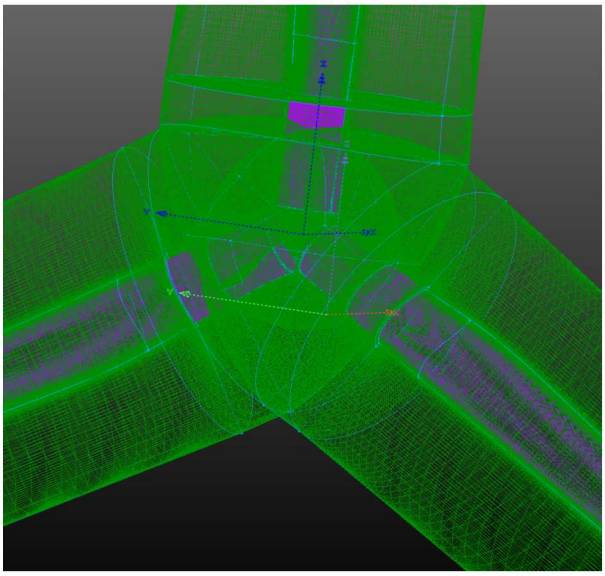
3. The implementation of the $k-\omega$ SST model was verified on canonical problems available on the NASA turbulence modeling website. Validation studies were conducted for a 3-D NACA 0015 wing. Results indicate that the simulations under-predict the peak suction pressure at almost all spanwise



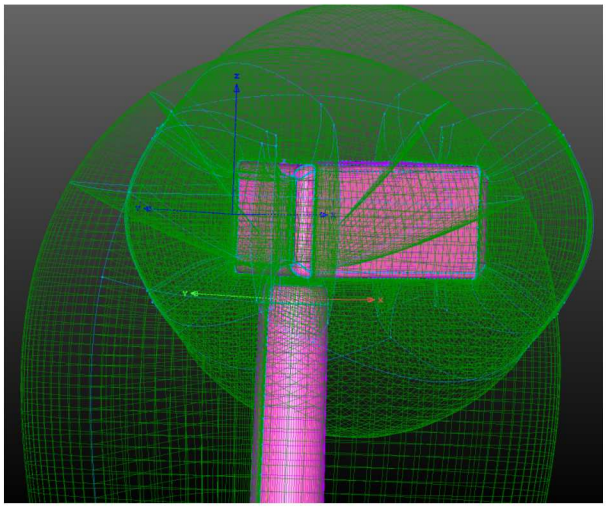
(a) ABL and wake-capturing mesh setup



(b) Mesh setup to model the full-turbine structure

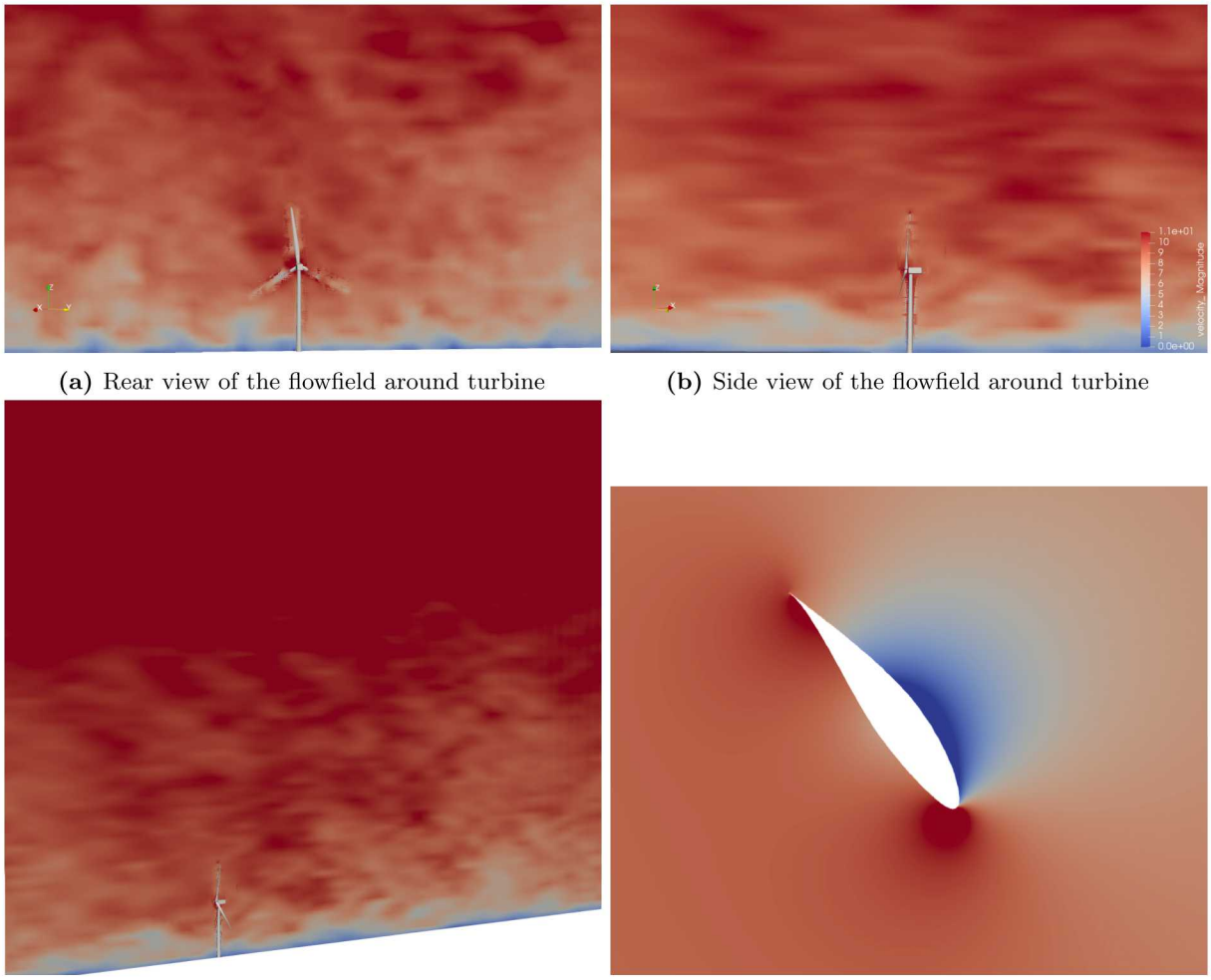


(c) Close-up of the individual blade meshes and their overlap near the root region



(d) Close-up of the all-HEX nacelle and tower meshes

Figure 34: Mesh for the simulation of turbine in a neutral atmospheric boundary layer (a) blades interface (b) nacelle-tower interface (c) full turbine mesh (d) full turbine mesh in the atmospheric boundary layer mesh.



(a) Rear view of the flowfield around turbine

(b) Side view of the flowfield around turbine

(c) Full-boundary layer profile on a slice parallel to streamwise direction

(d) Flowfield around the blade section at a span of 0.8R

Figure 35: Flowfield of the NREL 5-MW turbine operating in turbulent, neutral atmospheric boundary layer inflow.

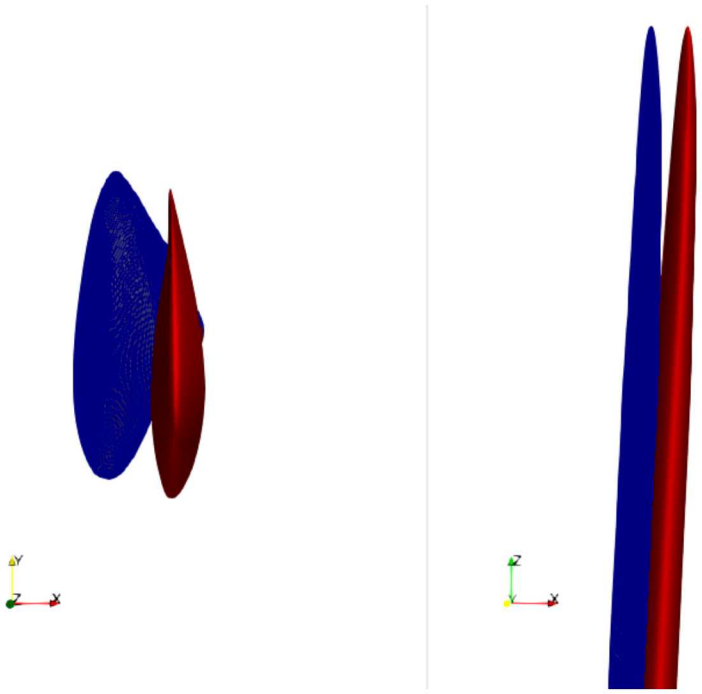


Figure 36: Comparison of the deformed blade geometry for the reference blade compared to the initial undeformed state for the NREL 5-MW rotor operating under turbulent ABL inflow.

stations. Deficits were also observed in the peak-to-peak values of the tangential velocities along the tip vortex core. The deficiencies are attributed to the lack of the wind tunnel walls in the simulations and, therefore, the inability to capture the blockage effect experienced by the flow in the experiment. Future simulations will include wind tunnel walls to study the effect of including the blockage effect on the numerical predictions. The 3-D wing problem will provide an important baseline to compare the new hybrid RANS/LES models developed at UT Austin for the FY19 Q4 milestone.

4. The RANS and hybrid RANS/LES simulations used upwinding to stabilize the advective terms in the simulation. Upwinding is undesirable in the LES regimes and reduces the accuracy of the overall simulation in the hybrid runs. The team continues to explore solver approaches that will eliminate the need to resort to upwinding for stabilizing the advective terms.

5. Fluid-structure interaction capabilities were added to Nalu-Wind by coupling with the OpenFAST code. The FSI coupling algorithm was verified to be second order with mesh refinement, and the algorithm was demonstrated on a rotor in a uniform inflow simulation to verify the implementation.

6. A segregated momentum solver was implemented using the HYPRE solver stack and tested on three different problems relevant to the wind application. The segregated approach showed significant speedup compared to the monolithic approach for momentum solves in Nalu-Wind. Work is underway to implement a segregated monolithic solver in the Trilinos stack.

7. Execution of the HYPRE solver stack on GPUs demonstrated that significant performance gains are possible by using device memory instead of relying on the unified memory (UVM) architecture. The studies also showed that NVIDIA's Multi-Process Service (MPS) is critical when multiple MPI ranks are operating on the same GPU. The research identified preconditioners as the primary bottleneck for performance on GPUs and future research will focus on identifying preconditioner/smoother approaches that perform well on GPUs.

8. The full-physics simulation of the NREL 5-MW turbine operating in turbulent ABL was demonstrated successfully, however, the results must be considered preliminary at this point. The simulation indicates

several areas in need of further research and improvement: 1. the blending of the atmospheric turbulence model with the near-blade DES model needs more parametric studies to determine the best settings for the transition between the two models, 2. the fluid-structure interaction capability must be verified in uniform inflow to ensure that the models predict the desired QoIs accurately with the coupled simulation, 3. the mesh resolution and the impact of the overset interpolations on the turbulent structures in the turbine meshes need further attention.

REFERENCES

[1] R. A. Baurle, C.-J. Tam, J. R. Edwards, and H. A. Hassan, *Hybrid simulation approach for cavity flows: Blending, algorithm, and boundary treatment issues*, AIAA Journal, 41 (2003), pp. 1463–1480.

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

[3] G. Chesshire and W. D. Henshaw, *Composite overlapping meshes for solution of partial differential equations*, Journal of Computational Physics, 90 (1990), pp. 1–64.

[4] E. Chow, H. Anzt, J. Scott, and J. Dongarra, *Using Jacobi iterations and blocking for solving sparse triangular systems in incomplete factorization preconditioning*, Journal of Parallel and Distributed Computing, 119 (2018), pp. 219–230.

[5] E. Chow and A. Patel, *Fine-grained parallel incomplete LU factorization*, SIAM Journal on Scientific Computing, 37 (2015), pp. C169–C193.

[6] R. Chow and C. P. van Dam, *Verification of computational simulations of the NREL 5 MW rotor with a focus on inboard flow separation*, Wind Energy, 15 (2012), pp. 967–981.

[7] M. Deveci, C. Trott, and S. Rajamanickam, *Multithreaded sparse matrix-matrix multiplication for many-core and GPU architectures*, Parallel Computing, 78 (2018), pp. 33–46.

[8] S. Domino, *A comparison between low-order and higher-order low-Mach discretization approaches*, in Studying Turbulence Using Numerical Simulation Databases - XV, P. Moin and J. Urzay, eds., Stanford Center for Turbulence Research, 2014, pp. 387–396.

[9] S. Domino, *Design-order, non-conformal low-Mach fluid algorithms using a hybrid CVFEM/DG approach*, Journal of Computational Physics, 349 (2018), pp. 331–351.

[10] S. P. Domino, *Sierra low-Mach module: Nalu theory manual 1.0*, Tech. Rep. SAND2015-3107W, Sandia National Laboratories, 2015.

[11] D. Driver and H. Seegmiller, *Features of a reattaching turbulent shear layer in divergent channel flow*, AIAA Journal, 23 (1985).

[12] E. P. N. Duque, M. D. Burklund, and W. Johnson, *Navier-Stokes and comprehensive analysis performance predictions of the NREL Phase VI experiment*, Journal of Solar Energy Engineering, 125 (2003), pp. 457–467.

[13] H. C. Edwards, C. R. Trott, and D. Sunderland, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, Journal of Parallel and Distributed Computing, 74 (2014), pp. 3202 – 3216. Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[14] S. Haering, T. A. Oliver, and R. D. Moser, *Towards a predictive hybrid RANS/LES framework*, in AIAA Scitech 2019 Forum, 2019, p. 0087.

[15] S. W. Hammond, M. A. Sprague, D. Womble, and M. Barone, *A2e High Fidelity Modeling: Strategic Planning Meetings*, Tech. Rep. NREL/TP-2C00-64697, National Renewable Energy Laboratory, 2015.

[16] M. M. Hand, D. A. Simms, L. J. Fingerish, D. W. Jager, J. R. Cotrell, S. Schreck, and S. M. Larwood, *Unsteady Aerodynamics Experiment Phase VI: Wind tunnel test configurations and available data campaigns*, Tech. Rep. NREL/TP-500-29955, National Renewable Energy Laboratory (NREL), December 2001.

[17] M. Hoemmen, I. Yamazaki, E. Boman, and J. Dongarra, *Communication-avoiding and pipelined Krylov solvers in Trilinos.* SIAM Conference on Computational Science and Engineering, Spokane, Washington, USA, 2019.

[18] C.-T. Hsiao and L. L. Pauley, *Numerical study of the steady-state tip vortex flow over a finite-span hydrofoil*, J. Fluids Eng., 120 (1998), p. 345.

[19] J. Jonkman, S. Butterfield, W. Musial, and G. Scott, *Definition of a 5-MW reference wind turbine for offshore system development*, Tech. Rep. NREL/TP-500-38060, National Renewable Energy Laboratory, 2009.

[20] A. C. Kirby, M. Brazell, Z. Yang, R. Roy, B. Reza Ahrabi, D. Mavriplis, J. Sitaraman, and M. K. Stoellinger, *Wind farm simulations using an overset hp-adaptive approach with blade-resolved turbine models*, in 23rd AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, 2017/10/20 2017.

[21] M. Lawson, J. Melvin, S. Ananthan, K. Gruchalla, J. Rood, and M. Sprague, *Blade-resolved, single-turbine simulations under atmospheric flow*, Tech. Rep. NREL/TP-5000-72760, National Renewable Energy Laboratory, 2019.

[22] M. Lesoinne and C. Farhat, *Higher-order subiteration-free staggered algorithm for nonlinear transient aeroelastic problems*, AIAA Journal, 36 (1998), pp. 1754–1757.

[23] E. C. Lynch and M. J. Smith, *Hybrid rans-les turbulence models on unstructured grids*, in 38th Fluid Dynamics Conference and Exhibit, 06 2008.

[24] K. W. McAlister and R. K. Takahashi, *NACA 0015 wing pressure and trailing vortex measurements*, Tech. Rep. NASA-A-91056, National Aeronautics and Space Administration, AMES Research Center, Moffett Field, CA, 1991.

[25] F. R. Menter, M. Kuntz, and R. Langtry, *Ten years of industrial experience with the SST turbulence model*, in Turbulence, Heat and Mass Transfer 4, K. Hanjalic, Y. Nagano, and M. Tummers, eds., Begell House, Inc., 2003, pp. 625–632.

[26] C. D. Moen and S. P. Domino, *A review of splitting errors for approximate projection methods*, in 16th AIAA Computational Fluid Dynamics Conference, no. 2003-4236, Orlando, FL, 23-26 June 2003, AIAA.

[27] C.-H. Moeng, *A large-eddy-simulation model for the study of planetary boundary-layer turbulence*, Journal of the Atmospheric Sciences, 41 (1984), pp. 2052–2062.

[28] A. Patel, E. Boman, S. Rajamanickam, and E. Chow, *Cross platform fine grained ilu and ildl factorizations using kokkos*, in Center for Computing Research Summer Research Proceedings 2015, A. M. Bradley and M. L. Parks, eds., no. SAND-2016-0830, 2015, pp. 159–177.

[29] M. Potsdam and D. Mavriplis, *Unstructured mesh CFD aerodynamic analysis of the NREL Phase VI rotor*, in Aerospace Sciences Meetings, American Institute of Aeronautics and Astronautics, Jan. 2009.

[30] W. Rodi, *Turbulence models and their application in hydraulics - a state of the art review*, NASA STI/Recon Technical Report A, 81 (1980).

[31] M. A. S. and O. A. M., *Basic laws of turbulent mixing in the surface layer of the atmosphere*, Tr. Akad. Nauk SSR Geophiz. Inst., 24 (1954), pp. 163–187.

[32] M. Sánchez-Rocha, M. Kirtas, and S. Menon, *Zonal hybrid rans-les method for static and oscillating airfoils and wings*, in 44th AIAA Aerospace Sciences Meeting and Exhibit, 01 2006.

[33] J. Sitaraman, M. Floros, A. Wissink, and M. Potsdam, *Parallel domain connectivity algorithm for unsteady flow computations using overlapping and adaptive grids*, J. Comput. Phys., 229 (2010), pp. 4703–4723.

[34] N. N. Sørensen, J. A. Michelsen, and S. Schreck, *Navier-Stokes predictions of the NREL Phase VI rotor in the NASA AMES 80 ft x 120 ft wind tunnel*, Wind Energy, 5 (2002), pp. 151–169.

[35] N. N. Sørensen, *CFD modelling of laminar-turbulent transition for airfoils and rotors using the* $\gamma - re_{\Theta_t}$ *model*, Wind Energy, 12, pp. 715–733.

[36] N. N. Sørensen and J. Johansen, *Upwind, aerodynamics and aero-elasticity, rotor aerodynamics in atmospheric shear flow*, in 2007 European Wind Energy Conference and Exhibition, Milan, Italy, 7–10 May 2008.

[37] M. Sprague, S. Boldyrev, P. Fischer, R. Grout, W. G. Jr., and R. Moser, *Turbulent flow simulation at the Exascale: Opportunities and challenges workshop*, tech. rep., U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, 2017. Published as Tech. Rep. NREL/TP-2C00-67648 by the National Renewable Energy Laboratory.

[38] K. Swirydowicz, J. Langou, S. Ananthan, U. Yang, and S. Thomas, *Low synchronization GMRES algorithms*, CoRR, abs/1809.05805 (2018).

[39] G. Vijayakumar, J. Brasseur, A. W. Lavely, B. Jayaraman, and B. Craven, *Interaction of atmospheric turbulence with blade boundary layer dynamics on a 5MW wind turbine using blade-boundary-layer-resolved CFD with hybrid URANS-LES*, in 34th Wind Energy Symposium, American Institute of Aeronautics and Astronautics, 10 2016.

[40] C.-H. Wu, C.-Y. Wen, R.-H. Yen, and M.-C. Weng, *Experimental and numerical study of the separation angle for flow around a circular cylinder at low Reynolds number*, Journal of Fluid Mechanics, 515 (2004), pp. 233–260.