SAND2019-7486R

ECP-U-2019-xxx

High-order methods for wind turbine simulations and moving to next-generation platforms

WBS 2.2.2.01, Milestone ECP-Q3-FY19

Robert Knaus

June 28, 2019

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative. A portion of this research was supported by the U.S. Department of Energy Wind Energy Technologies Office.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This report followed the Sandia National Laboratories formal review and approval process (SAND2019-TBD). As such, the technical report is suitable for unlimited release.

ECP-U-2019-xxx

ECP Milestone Report
High-order methods for wind turbine simulations and moving to next-generation platforms
WBS 2.2.2.01, Milestone ECP-Q3-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

June 28, 2019

ECP Milestone Report
High-order methods for wind turbine simulations and moving to next-generation platforms
WBS 2.2.2.01, Milestone ECP-Q3-FY19

APPROVALS

Submitted by:

Michael Sprague

29 March 2019

Michael A. Sprague
ECP-Q3-FY19

Date

Approval:

Thomas Evans
ORNL

Date

REVISION LOG

Version	Creation Date	Description	Approval Date
1.0	2019-06-28	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 an effort to increase the fidelity of Nalu-Wind at a fixed resolution through the implementation of a tensor-product based, matrix-free high order scheme. High order finite element methods have increased local work per datum communicated and have the potential to provide significantly more accurate solutions at a fixed number of degrees of freedom. Previous to this milestone, Nalu-Wind had an arbitrary order Control Volume Finite Element Method discretization as a solver option, but it required too much memory and was too slow to be of practical use. The work in this milestone addresses these issues by first implementing an implicit, high order solver that only partially assembles the global system. This reduces the memory footprint of the high-order scheme by orders of magnitude for higher polynomial orders. Second, a faster, tensor-product based method for evaluating the action of the left-hand side was implemented. This reduces the amount of computational work required by the scheme and dramatically enhanced the time-to-solution on example problems.

Finally, this milestone is an evaluation of the value of high order methods in the wind application space. With the enhancements to memory and computational cost, accuracy vs. time-to-solution was evaluated for several resolutions on an underresolved Taylor Green vortex test case. Results show that the high order scheme is cost-competitive with the production low-order schemes in Nalu-Wind, being moderately more expensive than the production edge-based vertex centered finite volume scheme. The evaluation of accuracy on the test case shows a potential benefit to high order at the highest resolution while not deteriorating accuracy on the lowest tested resolution. More work is needed to show value in the wind application, but positive strides have been made.

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 [12].

In a wind farm, each turbine generates an extremely long-lived wake that stretches many turbine diameters downstream, interacting with the atmospheric boundary layer and potentially other turbines and their wakes. Crucial to modeling these wakes is the accurate transport of the vortical motions induced by the turbine blades as they spin—effectively a wave propagation problem. High-order finite-element-type methods are attractive to ExaWind due to their enhanced ability to resolve wave propagation and their favorable computational intensity to established second-order accurate methods.

This milestone represents an evaluation for including a high-order, unstructured -grid discretization in Nalu-Wind. In particular, this milestone describes and evaluates a matrix-free implementation of a tensor-product based finite element method on a simple turbulent flow with accuracy versus time-to-solution as the key benchmark. The test case is an intentionally under-resolved turbulent flow. Full resolution of the Navier-Stokes equations is an intractable problem at the scale of a wind-farm, and even at exascale, models that account for our inability to fully resolve the dominantly nonlinear, "subgrid" dynamics are necessary. These models account for an ever smaller portion of the total energy of the flow as resolution improves, providing a convergence of the model-form error that varies depending on the problem and quantity of interest being examined, but is generally of an order below two. It may not be beneficial to invest in numerical methods that converge at an order beyond two, given the overall convergence rate of the simulation will nonetheless remain perhaps at $\mathcal{O}(h^{4/3})$ (for the wall-parallel kinetic energy distribution in a channel [8]). However, there is a particular scale of interest in a wind farm—the diameter of the tip vortex emanating from the turbine blade and traveling persistently a long distance downstream. The physics at this particular scale of interest, with exascale computing power, can be captured by the high-order numerical scheme, increasing the accuracy at a particular resolution albeit not affecting the overall convergence to the statistics of the actual, unfiltered fluid flow.

An efficient high-order implementation is necessary for evaluating such a claim, both in terms of memory usage and performance. In terms of memory, storing all the required connectivities between solution points scales with the polynomial order to the sixth power in 3D. For a mesh with an equivalent number of degrees of freedom, a high-order scheme will take many times more memory than the equivalent second-order scheme: for $p = 8$ it would take 512 times more memory for a structured, hexahedral mesh. This severely limits the amount of work available to a computational node and makes storing the matrix impractical except for $p = 1$ and, questionably, $p = 2$. A matrix-free high-order approach is necessary due to memory considerations alone. In this context, by "matrix-free", we mean storing only a reduced set of matrix entries such that the scaling with polynomial order is broken in terms of memory cost.

In addition to memory cost, forming the left-hand side matrix is relatively expensive. For hexahedral elements, it is possible to factorize out part of the residual evaluation, utilizing the structure of the underlying operations, making computing the action of the left-hand-side matrix scale with p^4 instead of the p^6 implied by the number of non-zero entries. This represents an algorithmic improvement for high-order schemes: the algorithm produces the same result while scaling better with polynomial order. In Nalu-Wind, we exclusively use Krylov-subspace-type iterative solvers to solve our linear systems—mostly GMRES iteration. These methods only require the action of the matrix, which with a careful implementation scales as p^4 [10]. In either case, storing the full-matrix is untenable in terms of computational efficiency and memory usage. To determine the usefulness of an unstructured, high-order discretization in Nalu-Wind, we need an implicit, matrix-free high-order implementation.

The milestone report first will detail the algorithm used to discretize the governing equations. Then, we describe details of its implementation such as the extensive use of explicit vectorization in combination with Kokkos. Finally, we will present our test case with an evaluation of the computational cost of increasing polynomial order in terms of accuracy versus time-to-solution.

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

Continuous higher-order systems require significant memory overhead due to the sparse linear matrix graph size. However, the increase in local work for higher polynomial orders is possibly advantageous on next generation platforms. We will implement and evaluate a matrix-free solver approach for low Mach flow. This will include the implementation and evaluation of matrix-free preconditioning strategies for both the momentum conservation and pressure Poisson systems. The efficiency and scaling with polynomial order will be evaluated on a simple turbulent flow with appropriate solution quality metrics. Alternative discretizations to the high order ”control volume finite element” method will also be evaluated, with evaluation of tools from CEED ECP.

2.2 EXECUTION PLAN

1. Deploy preconditioner coarsening strategies which may require only the P=1 system to be stored.
2. Evaluate matrix-free methods for both advection/diffusion.
3. Implement and evaluate a matrix-free preconditioning strategy for the pressure-Poisson system.
4. Run and analyze full system performance on a simple flow.
5. Explore alternate discretizations to overcome limitations of baseline approach.

Completion Criteria: Technical report describing the milestone accomplishment as well as a highlight slide summarizing those accomplishments.The capability will be available as a solver option for the Nalu-Wind codebase.

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. The creation of the matrix preconditioner only uses at most the connectivity of connected edges between solution points in the high order mesh. This reduces the memory footprint by orders of magnitude for high polynomial orders.
2. The matrix-free application of a tensor-product, high order generally unstructured control-volume element discretization was implemented and tested on for the momentum equation of the variable-density, acoustically incompressible Navier-Stokes equations. For a fixed number of degrees of freedom, $p = 3$ to $p = 6$ outperformed the production second-order node-centered finite volume scheme in Nalu-Wind for the momentum equation specifically.
3. A matrix-free implementation of the pressure-Poisson equation was implemented, storing only connectivities based on edges between solution points. The preconditioner dramatically reduced the number of linear iterations required to solve the Poisson equation in Nalu-Wind.
4. A Taylor-Green vortex breakdown case was run in Nalu-Wind at a sequence of polynomial orders, evaluating accuracy in prediction of kinetic energy decay rate against total simulation time. At higher resolutions, there appeared to be a benefit in terms of solution accuracy with a relatively small increase in the end-to-end simulation time.
5. A faster quadrature for control-volume finite element was implemented in order to achieve optimal work scaling with polynomial order.

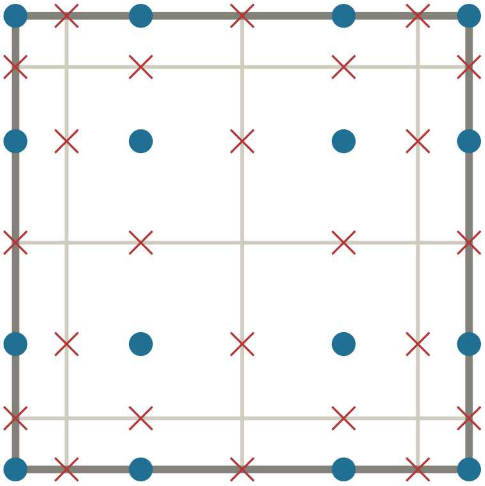


Figure 1: Schematic of the "subcontrol volumes" (grey lines) and integration points used in the $p = 3$ element. Volume integrals are evaluated at the nodes (circles) while surface integrals are evaluated at the flux points (crosses)

3.1 FAST RESIDUAL EVALUTION FOR A CONTROL-VOLUME FINITE ELEMENT SCHEME

Control volume finite element is a finite-volume type discretization that uses a finite-element type assembly. Here, we phrase the method as a Petrov-Galerkin type finite element technique where the test-space consists of a set of piece-wise constant functions taking a value of 1 inside a specially defined "dual cell" surrounding a node in the mesh, and 0 outside (the indicator function of the dual cell). This results in a weak form of the governing equations that is identical to the classic integral form used in node-centered finite volume methods, [1]. In this sense, control-volume finite element is a particular type of node-centered finite volume scheme where the finite-element basis is used to evaluate volume integrals and surface fluxes. However, this phrasing as a Petrov-Galerkin scheme also provides a natural way to extend the method to high-order without requiring a costly reconstruction procedure or special extended connectivity while still providing the finite-volume quality of direct enforcement of local conservation. Some careful consideration of the dual cell definition and nodal locations is necessary for maintaining optimal accuracy [7]. In particular, a basis through the $p + 1$ Gauss-Lobatto-Legendre nodes defined on the element combined with a dual-cell partition defined through the p Gauss-Legendre nodes provides an optimally accurate CVFEM scheme. As an example, we will focus on a scalar advection/diffusion problem.

3.1.1 The integral conservation law for CVFEM

Consider an advection-diffusion equation on a domain Ω,

$$\partial_t (\rho\phi) + \nabla \cdot (\rho \mathbf{u} \phi - \Gamma \nabla \phi) = 0 \text{ on } \Omega, \tag{1}$$

where ϕ, ρ are scalars, $\mathbf{u}$ is a vector, and Γ is a scalar for the purposes of this discussion. We create a hexahedral parition of domain, $\mathcal{T}_h(\Omega)$. For a $K \in \mathcal{T}_h$, we denote T_K as the trilinear the map from the reference element $K_{\text{ref}} = [-1, +1]^3$ to K. Labeling the $(p+2)^3$ "padded Gauss points" as $\{\bar{\xi}_i\}_0^{p+1} = \{-1, \xi_0, \dots, \xi_{p-1}, +1\}$—where $\{\xi_i\}_0^{p-1}$ are the p-point Gauss quadrature abscissae—we can describe the vertices of our "subcontrol volumes" on K as $G_{i,j,k} = T_K(\bar{\xi}_i, \bar{\xi}_j, \bar{\xi}_k)$. Each subcontrol volume is constructed by by connecting the vertices to form hexahedral subdomains on the element as $\overline{G_{i,j,k}G_{i+1,j,k}}, \overline{G_{i+1,j,k}G_{i+1,j+1,k}}, \dots, \overline{G_{i,j,k+1}G_{i,j,k}}$, with i, j, k between 0 and p, so as to construct $(p+1)^3$ hexahedral subcontrol volumes. See Figure 1 for a depiction of the subcontrol volumes on an element in 2D. Similarly, we label the $(p+1)^3$ Gauss-Legendre-Lobatto (GLL) points as $N_{K,ijk} = T_K(\zeta_i, \zeta_j, \zeta_k)$, with $\{\zeta_i\}_0^p$ being the abscissae locations of the $(p+1)$-point GLL quadrature rule. These are our "nodes" over which we'll be constructing Lagrange polynomials. The union of all subcontrol volumes containing a node m forms the dual cell $\Omega_m^\star$ and our "dual mesh" is the set of all the dual cells in the domain, forming a partition $\mathcal{T}_h^\star(\Omega) = \{\Omega_m^\star : 0 \leq m \leq N-1\}$ where N is number of nodes. Our discrete test space then is

$$\mathcal{V}_h = \operatorname{span}\left\{1_{\Omega_m^\star} : \Omega_m^\star \in \mathcal{T}_h^\star(\Omega)\right\}, \tag{2}$$

where $1_{\Omega_m^\star}(\mathbf{x})$ is the indicator function for the dual cell $\Omega_m^\star$—taking a value of 1 if $\mathbf{x}$ is inside the cell and 0 otherwise. ϕ is discretized using the set of piecewise continuous tensor-product polynomials through the GLL nodes of order p, $\mathcal{P}^p(K_{\text{ref}})$,

$$\Phi_h = \left\{\phi \in H^1(\Omega) : \phi|_K \circ T_K \in \mathcal{P}^p(K_{\text{ref}})\right\}. \tag{3}$$

Using these definitions and the Gauss divergence theorem, we can create a "weak form" for an advection-diffusion equation, Eq. 1, for all $\Omega_m^\star$ away from the boundary $\partial\Omega_m^\star \cap \partial\Omega = \varnothing$. Seek a solution $\phi_h \in \Phi_h$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_m^\star} \rho\phi_h \,\mathrm{d}\mathbf{x} + \int_{\partial\Omega_m^\star} (\rho\mathbf{u}\phi_h - \Gamma\nabla\phi_h)\cdot \mathrm{d}\mathbf{s} = 0 \text{ for each } \Omega_m^\star. \tag{4}$$

3.1.2 Definitions

Recall the padded Gauss-Legendre quadrature points $\{\bar{\xi}_i\}_0^{p+1} = \{-1, \xi_0, \ldots, \xi_{p-1}, +1\}$, where $\{\xi_i\}_0^{p-1}$ denoted the p-point Gauss quadrature locations. Denote the $p+2$ Lagrange interpolants through $\{\bar{\xi}_i\}_0^{p+1}$ as $\{d_j\}_0^{p+1}$, then the "histopolation functions" (see *e.g* [5]) are a set of polynomials orthonormal to the test space on the element, written as

$$h_j(\hat{x}) = -\sum_{k=0}^{j-1} d_k'(\hat{x}) \quad \text{so that} \quad \int_{\bar{\xi}_i}^{\bar{\xi}_{i+1}} h_j(x')\,\mathrm{d}x' = \delta_{ij}. \tag{5}$$

We define matrices W, I and D that are used for the integration, interpolation, and derivative operations respectively using the Gauss points and the GLL points $\{\zeta_i\}_0^p$,

$$\begin{gathered}\widetilde{I}_{ij} = \ell_j(\xi_i),\ \widetilde{D}_{ij} = \ell_j'(\xi_i),\\ W_{ij}^{-1} = h_j(\zeta_i),\ I_{ij} = \ell_j(\zeta_i) = \delta_{ij} \text{ and } D_{ij} = \ell_j'(\zeta_i).\end{gathered} \tag{6}$$

Here the W, I and D are square $(p+1)\times(p+1)$ matrices. I is the identity matrix due to the Krönecker-δ property of the Lagrange interpolants. W, I, and D could be modified to be evaluated at points other than the GLL nodes in which case I would be a full matrix, but that is not done here. The tilde overscript denotes matrices evaluated at the dual cell surfaces and they are rectangular, $p \times (p+1)$

Finally, we will define a $(p+1)\times p$ differencing or incidence matrix as

$$\widetilde{\Delta} = \begin{pmatrix} -1 & 0 & \ldots & 0 & 0 \\ +1 & -1 & 0 & \ldots & 0 \\ 0 & & \ddots & & \vdots \\ \vdots & \ldots & 0 & +1 & -1 \\ 0 & 0 & \ldots & 0 & +1 \end{pmatrix} \tag{7}$$

This compactly represents the equal and opposite fluxes "entering" and "leaving" a dual cell and results in a telescoping flux form for the conserved quantities.

With these definitions, we have for the time term

$$(Mass)\,\partial_t(\rho\phi)_{\ell mn} = \sum_p\sum_q\sum_r W_{\ell p}W_{mq}W_{nr}\,(\det\mathbf{J}_{pqr})\sum_j \gamma_j\rho_{pqr}^{n+1-j}\phi_{pqr}^{n+1-j} \tag{8}$$

where $\{\gamma_j\}_1^3$ are the coefficients for BDF2, divided by the timestep. For clarity, we will describe advection and diffusion separately. In practice, they are combined into an "advection-diffusion" kernel since some operations are shared. First, we define a mass flux term to be

$$\dot{m}_{pqr}^{\hat{x}} = \left(\sum_i \tilde{I}_{pi}\,(\rho_{iqr}\mathbf{u}_{iqr})\right)\cdot\mathbf{A}_{pqr}^{\hat{x}} \tag{9}$$

which is stored at each of the $dp(p+1)^{d-1}$ flux points inside the element. $\mathbf{A}$ is the area vector of the dual cell surface evaluated at the flux point, $\mathbf{A}^{\hat{x}}_{pqr} = \mathbf{J}_{pqr}\cdot\mathbf{e}_{\hat{y}} \times \mathbf{J}_{pqr}\cdot\mathbf{e}_{\hat{z}} = (\det \mathbf{J}_{pqr})\,\mathbf{J}^{-T}_{pqr}\mathbf{e}_{\hat{x}}$, where $\mathbf{J} = \partial_{\hat{\mathbf{x}}}\mathbf{x}$ is the element Jacobian matrix from the reference coordinates $\hat{\mathbf{x}}$ to the physical coordinate $\mathbf{x}$. $\mathbf{e}_{\hat{x}}$ is the unit-vector aligned with the $\hat{x}$ coordinate of the reference element.

$$
\begin{aligned}
\left[(\mathrm{Adv})^{\hat{x}}\phi\right]_{lmn} &= \sum_p\sum_q\sum_r \widetilde{\Delta}_{\ell p} W_{mq} W_{nr}\, \dot{m}^{\hat{x}}_{pqr} \sum_i \widetilde{I}_{pi}\phi_{iqr} \\
\left[(\mathrm{Adv})^{\hat{y}}\phi\right]_{lmn} &= \sum_p\sum_q\sum_r W_{\ell p} \widetilde{\Delta}_{mq} W_{nr}\, \dot{m}^{\hat{y}}_{pqr} \sum_j \widetilde{I}_{qj}\phi_{pjr} \\
\left[(\mathrm{Adv})^{\hat{z}}\phi\right]_{lmn} &= \sum_p\sum_q\sum_r W_{\ell p} W_{mq} \widetilde{\Delta}_{nr}\, \dot{m}^{\hat{z}}_{pqr} \sum_k \widetilde{I}_{rk}\phi_{pqk} \qquad (10)
\end{aligned}
$$

where we have used $\widetilde{\Delta}$ for compactness of notation but we never explicitly use an undivided difference matrix. For $\ell \neq 0$ and $\ell \neq p+1$, the term is computed

$$
\left[(\mathrm{Adv})^{\hat{x}}\phi\right]_{lmn} = \sum_q W_{mq} \sum_r W_{nr} \left[\dot{m}^{\hat{x}}_{\ell qr} \sum_i \widetilde{I}_{\ell i}\phi_{iqr} - \dot{m}^{\hat{x}}_{(\ell-1)qr} \sum_i \widetilde{I}_{(\ell-1)i}\phi_{iqr} \right]. \qquad (11)
$$

and we add "boundary" terms for $\ell = 0$ and $\ell = p+1$. For diffusion, the metric term is

$$
\mathbf{G}^{\hat{x}}_{pqr} = \left(\sum_i \tilde{I}_{pi}\Gamma_{iqr}\right)\mathbf{J}^{-T}_{pqr}\cdot\mathbf{A}_{pqr} = \tilde{\Gamma}_{pqr}\left(\det\mathbf{J}_{pqr}\right)\left(\mathbf{J}^T\mathbf{J}\right)^{-1}_{pqr}\mathbf{e}_{\hat{x}} \qquad (12)
$$

with an index-shifted form for the other reference directions. For the reference element with $\Gamma = 1$, $G^{\hat{x}_i}_{\hat{x}_j} = \delta_{ij}$. This assumes Γ is a scalar quantity. For a tensor-form like what is needed for the $M^{4/3}$ turbulence model [6], then Γ would need to be computed with the metric. With this definition, the diffusion term is

$$
\begin{aligned}
\left[(\mathrm{Diff})^{\hat{x}}\phi\right]_{lmn} = &\sum_p\sum_q\sum_r \widetilde{\Delta}_{\ell p} W_{mq} W_{nr} \\
&\left(G^{\hat{x}}_{x,pqr}\sum_i \widetilde{D}_{pi}\phi_{iqr} + G^{\hat{x}}_{y,pqr}\sum_i\sum_j \tilde{I}_{pi}D_{qj}\phi_{ijr} + G^{\hat{x}}_{z,pqr}\sum_i\sum_j \tilde{I}_{pi}D_{rk}\phi_{iqk}\right), \qquad (13)
\end{aligned}
$$

with a similar, but index shifted form for the other reference directions. More compactly we write these operators as

$$
\begin{aligned}
\begin{pmatrix} S^{\hat{x}} & S^{\hat{y}} & S^{\hat{z}} \end{pmatrix} &= \begin{pmatrix} \tilde{\Delta}\otimes W\otimes W & W\otimes\tilde{\Delta}\otimes W & W\otimes W\otimes\tilde{\Delta} \end{pmatrix} \\
\begin{pmatrix} \tilde{I}^{\hat{x}} & \tilde{I}^{\hat{y}} & \tilde{I}^{\hat{z}} \end{pmatrix} &= \begin{pmatrix} \tilde{I}\otimes I\otimes I & I\otimes\tilde{I}\otimes I & I\otimes I\otimes\tilde{I} \end{pmatrix} \\
\begin{pmatrix} \tilde{\mathbf{D}}^{\hat{x}} & \tilde{\mathbf{D}}^{\hat{y}} & \tilde{\mathbf{D}}^{\hat{z}} \end{pmatrix} &= \begin{pmatrix} \tilde{D}\otimes I\otimes I & D\otimes\tilde{I}\otimes I & D\otimes I\otimes\tilde{I} \\ \tilde{I}\otimes D\otimes I & I\otimes\tilde{D}\otimes I & I\otimes D\otimes\tilde{I} \\ \tilde{I}\otimes I\otimes D & I\otimes\tilde{I}\otimes D & I\otimes I\otimes\tilde{D} \end{pmatrix} \qquad (14)
\end{aligned}
$$

where $\otimes$ denotes the Krönecker product.

3.1.3 Quadrature modification

Previously, as the test function is discontinuous over the element, a Gauss-quadrature method was used *per control volume* to achieve the correct accuracy for the polynomial order. That is, for each subcontrol volume, for an even number polynomial order, $(p/2+1)^3$ Gauss-points were used in a particular continuous section of the test-space in order to integrate volumes (with a simular rule being used for face quadratures). As there are p^3 sections of the subcontrols volumes in each element, this resulted in a quadrature scheme with $(p+1)^3(p/2+1)^3 \sim p^6$ quadrature points. While this quadrature is more accurate than the current quadrature, the work scaling made it impractical. The new quadrature overcomes this scaling and produces a method whose work scales as p^4 (with sum factorization) while maintaining the order of accuracy for the overall scheme.

3.1.4 Governing equations

Nalu-Wind uses a stabilized, equal-order interpolation scheme for pressure and velocity. A spectrally vanishing pressure dissipation term is added that damps spurious pressure modes. The following equations are understood as the interior element-wise contributions to the left- and right hand sides that are assembled solved, with periodic boundary conditions for the purposes of this paper, for timestep n with an outer Picard iteration over the index k,

- $\frac{1}{\tau} M \left(\rho^{n,k} \odot \mathbf{u}^{\star} \right) + \sum_i S^{\hat{x}_i} \left[\dot{m}^{\hat{x}_i} \odot \tilde{I}^{\hat{x}_i} \mathbf{u}^{\star} - 2 \left(\tilde{I}^{\hat{x}_i} \mu \right) \odot \mathbb{S}^{\hat{x}_i} \odot \mathbf{A}^{\hat{x}_i} \right] = -M \left(\overline{\mathbf{Gp}}^{n,k} + \sum_j \gamma_j \rho^{n-j} \odot \mathbf{u}^{n-j} \right)$

  where $\mathbb{S}^{\hat{x}_i} = \frac{1}{2} \left(\mathbf{J}^{-T,\hat{x}_i} \odot \tilde{\mathbf{D}}^{\hat{x}_i} \mathbf{u} + \left(\mathbf{J}^{-T,\hat{x}_i} \odot \tilde{\mathbf{D}}^{\hat{x}_i} \mathbf{u} \right)^T \right) - \frac{1}{3} \operatorname{trace} \left(\mathbf{J}^{-T,\hat{x}_i} \odot \tilde{\mathbf{D}}^{\hat{x}_i} \mathbf{u} \right) \mathbf{I}$
- $\sum_i S^{\hat{x}_i} \left[\mathbf{G}^{\hat{x}_i} \odot \tilde{\mathbf{D}}^{\hat{x}_i} p^{n,k+1} \right] = \frac{1}{\tau} \sum_i S^{\hat{x}} \left[\mathbf{A}^{\hat{x}_i} \odot \tilde{I}^{\hat{x}_i} \left(\rho^{n,k} \odot \mathbf{u}^{\star} - \tau \overline{\mathbf{Gp}}^{n,k} \right) \right] - \frac{1}{\tau} M \left(\sum_j \gamma_j \rho^{n-j} \right)$
- update $\dot{m}^{\hat{x}_i} = \mathbf{A}^{\hat{x}_i} \odot \tilde{I}^{\hat{x}_i} \left(\rho^{n,k} \odot \mathbf{u}^{\star} - \tau \overline{\mathbf{Gp}}^{n,k} \right) - \tau \mathbf{G}^{\hat{x}_i} \odot \mathbf{D}^{\hat{x}_i} p^{n,k+1}$
- $M \overline{\mathbf{Gp}}^{n,k+1} = \sum_i S^{\hat{x}_i} \left[\mathbf{A}^{\hat{x}_i} \odot p^{n,k+1} \right]$
- $\mathbf{u}^{n,k+1} = \mathbf{u}^{\star} - \tau \left(\frac{1}{\rho^{n,k}} \right) \odot \left(\overline{\mathbf{Gp}}^{n,k+1} - \overline{\mathbf{Gp}}^{n,k} \right)$
- Solve transport equations for EOS and update density if applicable
- if tolerance is met, update $\mathbf{u}^{n+1} = \mathbf{u}^{n,k+1}$, and $p^{n+1} = p^{n,k+1}$, $\rho^{n+1} = \rho^{n,k+1}$ (15)

where γ are the coefficients for the backwards differencing-type time-integration (BDF2 is used exclusively herein) divided by the time step and τ is the "projected time scale", $\tau = \gamma_1^{-1} = \frac{2\Delta t}{3}$ for a constant timestep. The symbol $\odot$ is the Hadamard product. In the $p = 1$ case, $M = W \otimes W \otimes W \det \mathbf{J}$ can be lumped and becomes the typical Green-Gauss gradient for finite volume schemes—the nodal gradient is averaged over the dual volume.

$$\int_{\Omega_i^{\star}} \overline{\mathbf{Gp}} \, \mathrm{d}\mathbf{x} = \int_{\partial \Omega_i^{\star}} p \, \mathrm{d}\mathbf{s}. \tag{16}$$

For higher order, a sufficiently accurate projected nodal gradient is required such that the pressure stabilization error vanishes at the appropriate rate—$\left| \tilde{I}^{\hat{x}_i} \tau \overline{\mathbf{Gp}} - \tau \mathbf{J}^{-T,\hat{x}_i} \odot \mathbf{D}^{\hat{x}_i} p \right| < Ch^p$—and does not affect the order of accuracy of the overall scheme. This requires inversion of the mass matrix, a "reconstruction" of the gradient from the volume-averaged gradient. This is a consequence of the particular pressure stabilization technique used by Nalu-Wind and is not required by the CVFEM discretization itself. For this effort, the overall pressure-splitting scheme and stabilization is unmodified in the high-order, matrix-free approach compared to the default approach in Nalu-Wind. More details are available in the FY19Q2 milestone and in [9, 3].

When solving the equations, we assemble the right-hand side in δ-form from Eq. 15. Additionally for the momentum, the off-diagonal components of the block matrix resulting from gradients in the viscosity are dropped, creating a "split" momentum evaluation of one left-hand side and three right-hand sides. The full variable viscosity right-hand side, however, is kept even in the constant viscosity case. All test cases herein use constant density and constant viscosity, though some verification testing is done using full variable property case.

3.1.5 Implementation details and code structure

Nalu-Wind is built heavily on Sandia's Trilinos package (`stk` and `Kokkos` in particular) and uses either the `Tpetra`-solver stack or LLNL's `hypre` solver stack. The initial implementation is focused on the `Tpetra`-solver stack, using `Tpetra`'s datastructures and a custom `Tpetra::Operator<>` to implement the scheme. At a high level, Nalu-Wind implements an `apply` function that computes the action of the left-hand side matrix on an input Tpetra::MultiVector<>, either of 1-column (for the pressure-Poisson) or up to 3 columns for the momentum equation and projection equation, code snippet 1. As Nalu-Wind uses a shared-node abstraction for parallel assembly, there is a "import" stage that performs the off-processor gathering of data according to the internal maps of the Tpetra::Importer<> class. Similar, there is an export stage that adds the element contributions of shared-but-not-owned nodes on a particular process to the node-owner's process. This is the same essentially as what occurs for sparse mat-vec operation that Nalu-Wind currently performs.

```
void apply(const Tpetra::MultiVector<double, int, long>& ownedSolution,
  Tpetra::MultiVector<double, int, long>& ownedRHS,
  Teuchos::ETransp trans = Teuchos::NO_TRANS,
  double alpha = 1.0,
  double beta = 0.0
  ) const final
{
  // ... error checking ...
  auto sln = ownedAndSharedSlnCached_;
  auto sharedRHS = sharedRHSCached_;

  sln->doImport(ownedSolution, *importer_, Tpetra::INSERT);
  const auto xOwnedAndShared = sln->getLocalView<ExecSpace>();

  ownedRHS.putScalar(0.);
  auto yOwned = ownedRHS.getLocalView<ExecSpace>();

  sharedRHS->putScalar(0.);
  auto yShared = sharedRHS->getLocalView<ExecSpace>();

  mfInterior_.compute_linearized_residual(maxOwnedRowLid_, maxSharedNotOwnedRowLid_,
      xOwnedAndShared, yOwned, yShared);
  mfBoundary_.compute_linearized_residual(maxOwnedRowLid_, maxSharedNotOwnedRowLid_,
      xOwnedAndShared, yOwned, yShared);
  ownedRHS.doExport(*sharedRHS, *exporter_, Tpetra::ADD);
}
```

Listing 1: Matrix-free evaluation

After the import stage, the matrix free operator calls an "interior" and "boundary" operator, implemented currently using a templated policy pattern. These operators compute for instance the interior element contributions of linearized momentum equation in the Navier-Stokes equations, and the "boundary" operator either overwrites that contribution on the boundary to implement a strong-Dirichlet condition or adds to it to implement for instance an open condition.

3.1.6 SIMD element fields

The core algorithm works on a set of Kokkos::View datastructures termed "SIMD element fields" collectively. These are elemental field data or element geometric data for n-elements, where n is the SIMD-length for double-precision numbers (8 for avx512 on skylake). The explicit vectorization is performed using the stk::simd library, which implements a floating-point type that stores n-values and provides overloaded mathematical operations. The Kokkos team is currently implementing a similar, GPU-friendly vectorization abstraction. The matrix-free operators perform gathers of required field data, such as density or velocity, into these SIMD-element fields, as well perform metric computations to determine the volume (scaled by density, possibly) evaluated at dual cell volumes or the "mapped area"—$\mathbf{G}$, from Eq 12—evaluated at dual cell surfaces. Other calls required for the method are the "corrected mass flux"—the mass flux with a correction due to the pressure-dissipation stabilization used and a computation of the area ($\dot{m}$ in Eq 15). In all these methods, the elements are considered to be multilinear boxes without curvature. The geometric computations are done every timestep, despite the constant-in-time mesh, but take only a small fraction of total simulation time according to profiling data.

Time per apply

Order	Auto-vectorized	Explicitly vectorized	Speed-up
$p = 1$	230. ms	148. ms	1.56
$p = 3$	93.5 ms	62.3 ms	1.50
$p = 7$	99.4 ms	57.9 ms	1.72
$p = 8$	108. ms	58.8 ms	1.83

Table 1: Effect of explict outer-loop vectorization on the matrix-free apply with avx512 instructions of a skylake machine

From some rudimentary tests, there's a large performance benefit for the solve of around 50-80% from explicit outer-loop vectorization, as seen in Table 1. The effect is lower than what was seen in the FY17Q4 milestone, however, so more investigation is warranted.

3.1.7 Interior Operator

```
template <typename OwnedViewType, typename SharedViewType>
void compute_linearized_residual(
   global_ordinal_type maxOwnedRowLid,
   global_ordinal_type maxSharedNotOwnedLid,
   OwnedViewType xv,
   OwnedViewType yowned,
   SharedViewType yshared) const
 {
   const auto range = HostRangePolicy(0, entityOffsets_.extent_int(0));
   Kokkos::parallel_for("continuity_linearized_residual", range, KOKKOS_LAMBDA(const int index) {
     const auto delta = gather_delta<p>(index, entityOffsets_, xv);
     const auto delta_view = nodal_scalar_view<p, DoubleType>(delta.data());
     add_element_rhs_to_local_tpetra_vector<p>(
       index,
       maxOwnedRowLid,
       maxSharedNotOwnedLid,
       entityOffsets_,
       continuity_element_residual<p>::linearized_residual(index, mapped_area_, delta_view),
       yowned,
       yshared
     );
   });
 }
```

Listing 2: Interior evaluation

`DoubleType` in this context on the host is a `stk::simd::Double`, an explicit vectorization type that stores n-doubles with overloaded vector operations, see the FY17Q4 milestone for more details. Here, "mapped_area_" is the pre-computed element metric $\mathbf{G}$ (Eq 12) for the n-elements that are being operated on. The residual operator calls the function that performs elemental-action of the laplacian, which is summed into the `Tpetra::MultiVector<>` used by the `Belos::PseudoBlockGmres` solver. A block size of one for the pressure Poisson equationm, three for momentum and the projected nodal gradient equation with a specialized algorithm for evaluating three right-hand sides.

```
template <int p, typename Scalar>
void scalar_laplacian_rhs(
  const CVFEMOperators<p, Scalar>& ops,
  const scs_vector_view<p, Scalar>& metric,
  const nodal_scalar_view<p, Scalar>& scalar,
  nodal_scalar_view<p, Scalar>& rhs)
```

```
{
  static constexpr int n1D = p + 1;

  nodal_vector_array<Scalar, p> work_grad_phi;
  auto grad_phi_scs = la::make_view(work_grad_phi);

  nodal_scalar_array<Scalar, p> work_integrand;
  auto integrand = la::make_view(work_integrand);

  ops.scs_xhat_grad(scalar, grad_phi_scs);
  for (int k = 0; k < n1D; ++k) {
    for (int j = 0; j < n1D; ++j) {
      for (int i = 0; i < p; ++i) {
        integrand(k,j,i) =
           metric(XH, k, j, i, XH) * grad_phi_scs(k, j, i, XH)
         + metric(XH, k, j, i, YH) * grad_phi_scs(k, j, i, YH)
         + metric(XH, k, j, i, ZH) * grad_phi_scs(k, j, i, ZH);
      }
    }
  }
  ops.integrate_and_diff_xhat(integrand, rhs);
  // ... other directions ...
}
```

Listing 3: Element Residual

where the "CVFEMOperators" performs tensor-contractions on the SIMD-elemental data.

While the code is not compatible with GPUs at the point in time, it heavily uses `Kokkos` with some abstractions being used from Nalu-Wind's on-going GPU conversion. The next step for this effort would be to use Kokkos's performance-portability infrastructure to run on the GPU as the NGP effort has now finalized a design for Nalu-Wind on the GPU.

3.1.8 Verification

Multiple levels of verification were used. The tensor-product operators are tested with patch tests to ensure proper order of accuracy for the integration, interpolation and differentiation operators in Eq 14. Each "kernel" operator has an independent, unit-level verification test that is executed nightly. A smooth trigonometric "method-of-manufactured solutions"(MMS) source term is devised using Mathematica and implemented in the Nalu-Wind unit tests. For example, an MMS source for the variable-density, variable-viscosity momentum advection-diffusion "kernel" is applied to a single element. The polynomial order is increased until the kernel is able to satisfy the MMS to a tolerance of 10^{-10}. The order at which this occurs varies but generally a much higher order (*e.g.* $p = 22$ for the pressure Poisson test) is necessary for the element to spectrally resolve the MMS than what would be used in practice. The matrix-free portion was compared to the `CrsMatrix` implementation and showed agreement.

In addition to the unit-level verification tests, an integrated MMS is used. Source terms for forcing a manufactured solution of

$$\begin{aligned} u &= +\cos(2\pi x) sin(2\pi y)\sin(2\pi z)/2 \\ v &= -\sin(2\pi x)\cos(2\pi y)\sin(2\pi z) \\ w &= +\sin(2\pi x)\sin(2\pi y)\cos(2\pi z)/2 \end{aligned} \tag{17}$$

were derived and applied to the matrix-free scheme. Optimal convergence is achieved, with higher convergence at even orders of $p + 2$ given the cubical mesh, inline with previous verification efforts that also show the accuracy degrading back to $p + 1$ on general meshes.

For a smooth problem with very tight error tolerances, the advantage of high-order is clear from Table 2: extrapolating the $p = 1$ result, it would require over a billion nodes to achieve the same error as the $p = 4$ refinement with two million nodes. In direct numerical simulation, where we are concerned with creating

# elements	# rows	L^2 error	rate	# elements	# rows	L^2 error	rate
$p = 1$				$p = 2$			
32^3	32^3	1.18×10^{-3}	—	16^3	32^3	4.05×10^{-4}	—
64^3	64^3	4.58×10^{-4}	1.98	32^3	64^3	3.26×10^{-5}	3.64
128^3	128^3	1.15×10^{-4}	2.00	64^3	128^3	2.13×10^{-6}	3.93
$p = 3$				$p = 4$			
10^3	30^3	3.89×10^{-5}	—	8^3	32^3	5.70×10^{-6}	—
20^3	60^3	1.55×10^{-6}	4.65	16^3	64^3	8.98×10^{-8}	5.99
40^3	120^3	7.90×10^{-8}	4.29	32^3	128^3	1.54×10^{-9}	5.87

Table 2: Convergence of velocity for a manufactured soluton, Eq. 17

reliable, highly accurate databases, this argument may apply. In the wind application space, there are many sources of error and we are rarely interested in driving the spatial numerical error to 10^{-9}. While the usefulness is more complicated than the MMS would imply, it may still be useful in wind area by providing highly accurate propagation of resolved flow features.

3.2 PRESSURE POISSON EQUATION PRECONDITIONER

The preconditioner is created by Trilinos's `MueLu` [11] package using a sparse, approximate representation of the high order system. First, an equivalent low-order representation of the Laplacian was used, using the low-order basis to form a Laplacian for each of p^3 subelements of the the high-order element. For an orthogonal element with an aspect ratio of α, we can write the CVFEM stiffness matrix as

$$K_{\text{orth,element}} = (W \otimes \tilde{\Delta}\tilde{D}) + \alpha^{-2}\left(\tilde{\Delta}\tilde{D} \otimes W\right) \tag{18}$$

and with some algebra, we can determine that the Laplacian matrix for $p = 1$ ceases to be an M-matrix when the aspect ratio is sufficiently large, $\alpha > \sqrt{3}$ with a similar result in 3D ($\alpha_1^2 + \alpha_2^2 > 3$). For an element with nodes at the Gauss-Lobatto quadrature points, the M-matrix condition is violated for the sub-element matrices if $p > 2$. As we will be using a smoothed-aggregation multigrid preconditioner, providing a monotone system was suggested as perhaps providing a better preconditioner. To that end, we use a lumped mass matrix (W is identity) to form the low-order preconditioner. The Laplacian then remains an M-matrix and the contributions from nodes not connected by a common edge are zero. The approximate Laplacian that we provide to `MueLu` only contains connectivity information for the connected edges of the equivalent low-order mesh, increasing the sparseness of the matrix. However, we have yet to test the edge reduction on a highly skewed mesh. All tests are performed using the same `MueLu` parameters. Generally, we see in Table 3 using the sparsified full CVFEM matrix or the edge matrix to generate the multigrid preconditioner does not have a large effect on iterative convergence generally except for at relatively high p where perhaps the lack of monotonicity of the CVFEM stiffness matrix is degrading the quality of `MueLu`'s preconditioner. Both options are available, but due to increased sparsity and sometimes superior performance, the edge-based sparse system was used herein.

4. TAYLOR GREEN VORTEX BREAKDOWN

The Taylor-Green vortex breakdown, in this case at $Re = 1600$, is a well-studied test for high-order schemes [13]

$$u(x, y, z) = +u_0 \sin(x)\cos(y)\cos(z) \tag{19}$$

$$v(x, y, z) = -u_0 \cos(x)\sin(y)\cos(z) \tag{20}$$

$$w(x, y, z) = 0 \tag{21}$$

$$p(x, y, z) = 1/16\left(\cos(2x)\cos(2y)\left(\cos(2z) + 2\right)\right) \tag{22}$$

Preconditioner matrix	Linear iterations	Preconditioner matrix	Linear iterations
$p = 1$ Matrix-free		$p = 2$ Matrix-free	
None	60	None	71
Full Matrix	5	Sparse CVFEM	8
Sparse Edge	8	Sparse Edge	9
$p = 3$ Matrix-free		$p = 8$ Matrix-free	
None	86	None	129
Sparse CVFEM	8	Sparse CVFEM	15
Sparse Edge	8	Sparse Edge	9

Table 3: Effect of Matrix-free preconditioner on average GMRES iterations (first 40 solves) to solve the Poisson problem to 10^{-4} of its initial preconditioned residual norm, 144^3 Taylor-Green problem

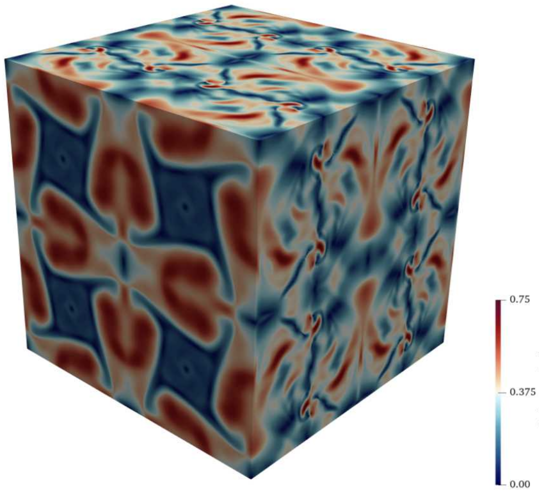
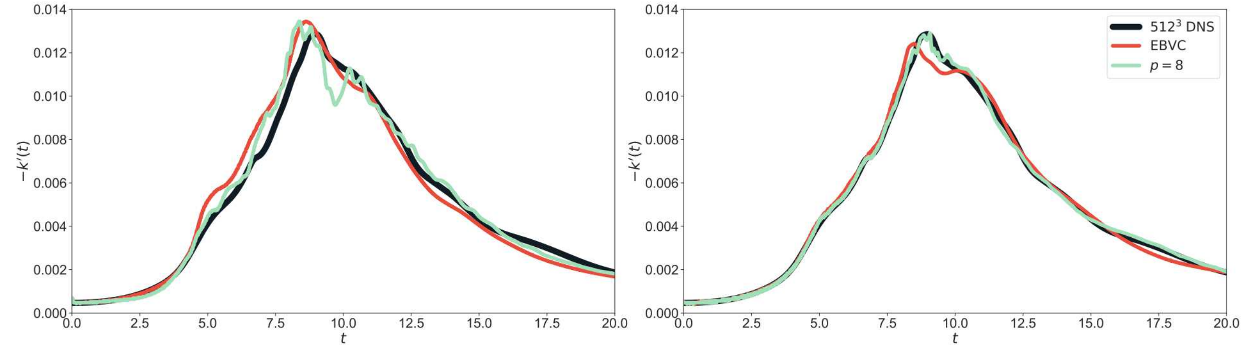


Figure 2: The Taylor-Green vortex breakdown case at $t = 15$, for $p = 8$ with 24^3 elements



(a) 96^3 nodes

(b) 192^3 nodes

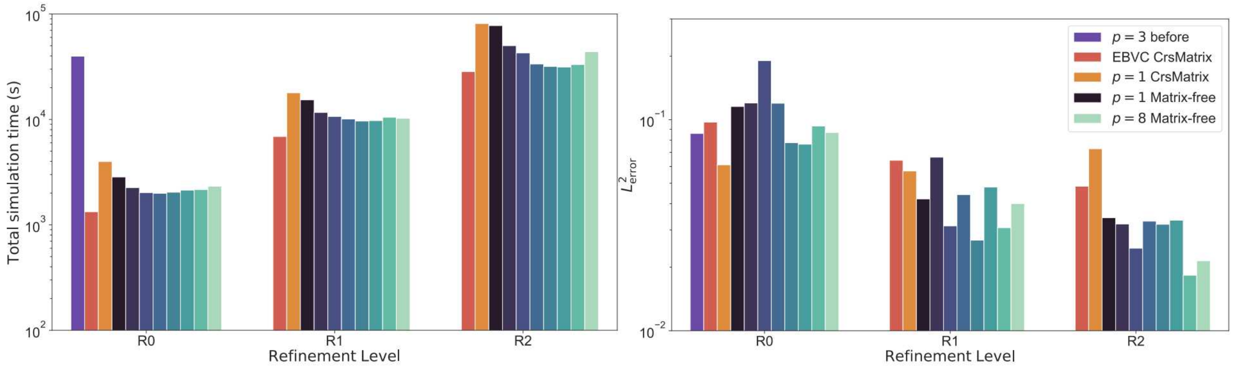
Figure 3: Comparison to the DNS result between the low-order (EBVC) and high-order($p = 8$) schemes at two different resolutions

The initial condition breaks down and forms a turbulent-like flow with the classic Kolmogorov 5/3 decay of the energy spectrum. This provides a relatively easy case to set-up and compare the effect of polynomial order on accuracy for underresolved simulations. Note that because these cases are underresolved, there is no asymptotic convergence to be expected. From other studies [13, 2], a resolution of 256^3 is necessary for convergence to begin for this problem. While the method converges at the expected $p + 1$ rate for resolved problems, in our application we are typically only marginally resolved—we resolve to the effective Nyquist frequency and rely on the turbulence model to correctly account for the energy transfer between the resolved and unresolved fluid motion. Because the effect of the turbulence model is reduced at a physics-based rate below $\mathcal{O}(h^2)$, the overall asymptotic convergence benefit of high-order (beyond a second-order discretization) is not relevant. However, a turbulent flow contains many motions at a large range of scales, the most important of which are captured adequately by the numerical mesh and may benefit from a high-order discretization. To this end, we examine an underresolved turbulent flow and look directly at the accuracy of the scheme scheme at a few resolutions versus the overall time to solution.

This type of study is inspired by [4], in which stabilization is shown to play a critical role in achieving good accuracy and stability at low resolutions. The cases herein do not use any stabilization beyond the pressure dissipation necessitated by the collocated scheme

4.1 DETAILS

Meshes of $(96/p)^3$, $(144/p)^3$, $(192/p)^3$ elements—so that the number of nodes is kept constant—were used and compared with a pseudo-spectral method DNS using 512^3 modes [13]. Since the number of nodes isn't exactly matched for $p = 5$ and $p = 7$, a normalization factor is applied for the timing data—however, the normalization factor is close to one (94% in the worst case). The 96^3 case is significantly underresolved while the 192^3 is marginally underresolved. We look at the rate of integrated kinetic energy decay—the dissipation—as our quantity of interest. A constant timestep of 0.025 was used for all the 96^3 cases resulting in a peak Courant number of approximately 0.5 for the $p = 1$ cases. A constant of four outer Picard iterations was used. Subsequent resolutions have the time step reduced proportionally. Each case was run to a non-dimensional time of 20, matching the end of the DNS data provided. The L^2 error in the integrated quantity relative to the DNS result was computed over the entire time horizon of the data. All errors are normalized by the L^2 norm of the DNS result over the same time window (a constant, same for all case). No turbulence model is used in this case, nor is any stabilization beyond Nalu-Wind's pressure-dissipation scheme. Timings are averaged over two runs for the $R0$ case with the variability between the two being low. Subsequent resolutions are not averaged in timings. The matrix-free branch of Nalu-Wind was compiled with `-xHost -O3` with `intel-19.0.2`. Each case was kept to a fixed number of processors—a resource limited case.



(a) Total simulation time

(b) L^2 integrated error

Figure 4: A comparison between accuracy and error for an underresolved Taylor-Green vortex decay problem

4.2 RESULTS

The time-to-solution for the new matrix-free implementation is roughly independent of polynomial order at a fixed number of nodes (see Figure 4 and Figure 5, Table 4 and Table 5). While the amount of work scales as Np, where N is the total number of nodes, more of the mesh points are collected together into memory-coherent elements. Beyond that, the computation is likely memory-boundary bound. For conventional architectures, we observe that the time-to-solution is not strongly a function of polynomial order, with $p = 1$ performing the worst out of the eight matrix-free runs. Note that the $p = 1$ cases are run identically to the high-order cases, but there is opportunity for specialization (some of the matrices are just a constant value for $p = 1$) The full matrix can also be stored to generate the preconditioning matrix, but that's not done here. The discretization for the $p = 1$ matrix free is different from the "default" scheme in Nalu-Wind

In terms of accuracy, for this particular problem, it seems that high-order only provides a real benefit at the highest resolution tested. As the simulation becomes resolved, the highest order schemes will eventually pull away in terms of accuracy at a given resolution considering the enhanced rate of convergence, see Table 2. In these underresolved simulations, there may be a similar trend where there is no particular benefit to the high order scheme if one is too underresolved, but as the resolution increases, the large scale motions become well-resolved by the scheme despite there still being small residual motions that cannot be captured. On the other hand, the underresolved cases are not particularly polluted by aliasing-type errors seen with higher-order scheme and the overall run times are only marginally greater than the edge-based vertex-centered finite volume scheme (EBVC, also called "node-centered finite volume"). Implementing an effective stabilization has been shown to greatly improve accuracy on these underresolved flows for high order [4], and it's an avenue that should be pursued in the future. While LES cases are not "well"-resolved, the turbulence model ideally provides an appropriate amount of dissipation such that the flow can be resolved by the numerical scheme, bring us closer to the 192^3 result. In practice, the turbulence model will either over- or underestimate dissipation leading to some underresolved areas of the flow.

The timing for the high-order scheme would also be improved by evaluating different pressure stabilization strategies that do not require a separate linear solve. At the higher resolution, the matrix-free scheme would be faster than the EBVC scheme if not for the projected nodal gradient equation which is an artifact of extending the low-order pressure stabilization procedure to the high order system.

5. CONCLUDING REMARKS AND NEXT STEPS

Compared with the previous implementation, the new implementation is roughly 20 times faster for $p = 3$ while using 5 times less memory. The benefit only becomes greater with higher polynomial orders. The milestone has taken the high-order code in Nalu-Wind into a position where the speed is, in one case, within 10% of our optimized node-centered finite volume scheme on a per-degree of freedom basis. This allows us to evaluate the efficacy of high order on our problems of interest.

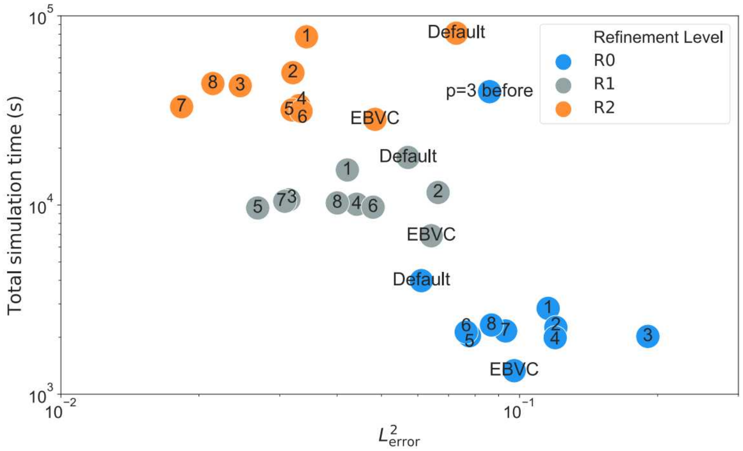


Figure 5: Time vs. Accuracy scatter plot for all cases

*R*0

	PNG	MOM	CONT	TOTAL	MOM/EBVC	CONT/EBVC	TOTAL/EBVC
$p = 1$	4.46×10^2	1.13×10^3	9.69×10^2	2.84×10^3	1.50	2.19	2.13
$p = 2$	3.86×10^2	7.48×10^2	9.18×10^2	2.25×10^3	0.99	2.07	1.69
$p = 3$	3.65×10^2	6.50×10^2	8.28×10^2	2.02×10^3	0.86	1.87	1.52
$p = 4$	3.62×10^2	6.38×10^2	8.21×10^2	1.99×10^3	0.85	1.86	1.50
$p = 5$	3.75×10^2	6.70×10^2	8.21×10^2	2.04×10^3	0.89	1.85	1.53
$p = 6$	3.87×10^2	7.07×10^2	8.58×10^2	2.13×10^3	0.94	1.94	1.60
$p = 7$	3.96×10^2	7.72×10^2	8.23×10^2	2.16×10^3	1.03	1.86	1.62
$p = 8$	4.04×10^2	8.94×10^2	8.50×10^2	2.32×10^3	1.19	1.92	1.74
$p = 3$ before	1.20×10^4	1.73×10^4	7.85×10^3	3.97×10^4	23.0	17.7	29.9
Default	NA	2.50×10^3	1.09×10^3	3.98×10^3	3.32	2.46	2.99
EBVC	NA	7.52×10^2	4.43×10^2	1.33×10^3	1	1	1

Table 4: Assorted timings (in seconds) for the matrix-free method for 96^3 node meshes (32 processors). PNG: Projected nodal gradient assembly + Solve time . MOM: Momentum assembly and solve time. CONT: Continuity assembly and solve time. TOTAL: Overall simulation time

*R*2

	PNG	MOM	CONT	TOTAL	MOM/EBVC	CONT/EBVC	TOTAL/EBVC
$p = 1$	1.28×10^4	3.04×10^4	2.21×10^4	7.75×10^4	1.81	2.38	2.73
$p = 2$	9.15×10^3	1.73×10^4	1.93×10^4	5.01×10^4	1.03	2.08	1.76
$p = 3$	8.35×10^3	1.39×10^4	1.71×10^4	4.28×10^4	0.83	1.85	1.51
$p = 4$	8.02×10^3	1.09×10^4	1.22×10^4	3.35×10^4	0.65	1.32	1.18
$p = 5$	7.62×10^3	1.02×10^4	1.17×10^4	3.18×10^4	0.61	1.26	1.12
$p = 6$	7.90×10^3	9.87×10^3	1.17×10^4	3.13×10^4	0.59	1.26	1.10
$p = 7$	7.97×10^3	1.10×10^4	1.20×10^4	3.31×10^4	0.66	1.29	1.17
$p = 8$	8.58×10^3	1.74×10^4	1.57×10^4	4.39×10^4	1.04	1.69	1.55
Default	NA	4.85×10^4	2.34×10^4	8.10×10^4	2.90	2.53	2.85
EBVC	NA	1.68×10^4	9.27×10^3	2.84×10^4	1	1	1

Table 5: Assorted timings (in seconds) for the matrix-free method for 192^3 node meshes (32 processors). PNG: Projected nodal gradient assembly + Solve time . MOM: Momentum assembly and solve time. CONT: Continuity assembly and solve time. TOTAL: Overall simulation time

The Nalu-Wind application will likely being using a hybrid discretization to solve windfarm problems, with a “background” discretization being used for the region of the flow resolved with large-eddy simulation and the second-order node-centered finite volume scheme (EBVC) being used in the Reynolds-Averaged Navier-Stokes region where solution is much more difficult and the problem is dominated by model-form error. The matrix-free high-order CVFEM discretization could be used in concert with the overset mesh capability (see FY19Q2, for example) to model the atmospheric boundary layer and turbine wakes, where high order may be advantageous. To that end, this work would need to be extended to use the overset mesh capability in Nalu-Wind in order to achieve a mixed order discretization.

Next Steps:

1. Pursue alternative pressure stabilizations for higher order
2. Evaluate performance of high order on a turbulent channel flow configuration
3. Run a study of an atmospheric boundary layer with the high order scheme
4. Extend to work with mixed order through overset
5. Evaluate performance on GPU-based platforms
6. Look at the performance of the preconditioner on significantly skewed meshes

While the benefit for high-order in our application has not been demonstrated yet, the comparable cost to our second-order, optimized finite volume scheme is promising.

REFERENCES

[1] T. Barth and M. Ohlberger, *Finite volume methods: Foundation and analysis*, Encyclopedia of Computational Mechanics, (2004).

[2] L. Diosady and S. Murman, *Case 3.3: Taylor green vortex evolution*, in Case summary for 3rd International Workshop on Higher-Order CFD Methods, 2015.

[3] 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.

[4] G. J. Gassner and A. D. Beck, *On the accuracy of high-order discretizations for underresolved turbulence simulations*, Theoretical and Computational Fluid Dynamics, 27 (2013), pp. 221–237.

[5] M. Gerritsma, *Edge functions for spectral element methods*, in Spectral and High Order Methods for Partial Differential Equations, Springer, 2011, pp. 199–207.

[6] S. Haering and R. D. Moser, *Resolution-induced anisotropy in les*, arXiv preprint arXiv:1812.03261, (2018).

[7] Y. Lin, M. Yang, and Q. Zou, *L ˆ2 error estimates for a class of any order finite volume schemes over quadrilateral meshes*, SIAM Journal on Numerical Analysis, 53 (2015), pp. 2030–2050.

[8] A. Lozano-Durán and H. J. Bae, *Error scaling of large-eddy simulation in the outer region of wall-bounded turbulence*, Journal of Computational Physics, 392 (2019), pp. 532–555.

[9] C. Moen and S. Domino, *A review of splitting errors for approximate projection methods*, in 16th AIAA Computational Fluid Dynamics Conference, 2003, p. 4236.

[10] S. A. Orszag, *Spectral methods for problems in complex geometrics*, in Numerical methods for partial differential equations, Elsevier, 1979, pp. 273–305.

[11] A. Prokopenko, J. J. Hu, T. A. Wiesner, C. M. Siefert, and R. S. Tuminaro, *MueLu user's guide 1.0*, Tech. Rep. SAND2014-18874, Sandia National Laboratories, 2014.

[12] M. Sprague, S. Boldyrev, P. Fischer, R. Grout, W. Gustafson 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.

[13] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, et al., *High-order cfd methods: current status and perspective*, International Journal for Numerical Methods in Fluids, 72 (2013), pp. 811–845.