SAND2020-3817PE

A2e High-Fidelity Modeling (HFM) Project

Overview of FY20 Q2 milestone completion:

Actuator Disk Improvements and Hardening

15 April 2020

Philip Sakievich, Robert Knaus, Alan Hsieh, Lawrence Cheung, Myra Blaylock, David Maniaci, Luis 'Tony' Martinez-Tossas, Matthew J Churchfield

NREL is a national laboratory of the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, operated by the Alliance for Sustainable Energy, LLC.

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.

FY20 Q2 Milestone Description

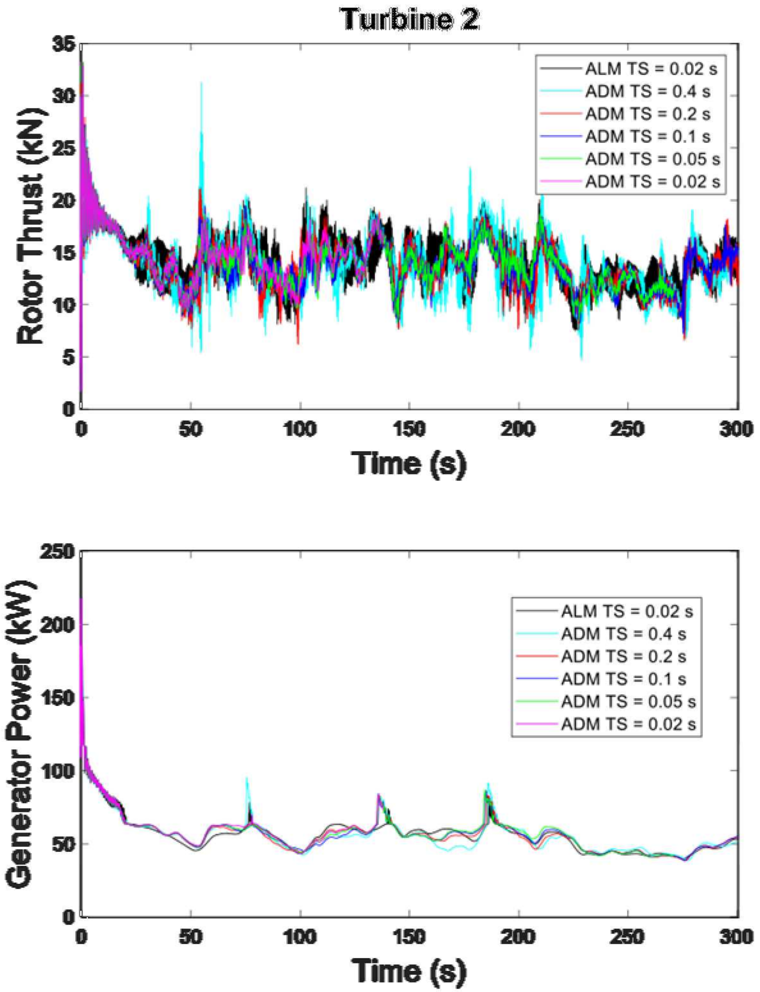
Enhance Nalu-Wind’s actuator disc model through hardening, documenting, stress-testing, verifying, and validating. Existing workflows will be improved by reducing the data output stream, and by making the analysis capabilities more modular and generally better. These model capabilities are needed by other A2e areas, namely Wake Dynamics, AWAKEN, and VV&UQ.

FY20 Q2 Milestone Goals, Background, & Team

- **Main goals in milestone:**
  - Make the actuator disk model (ADM) production ready in Nalu-Wind by improving its scalability and software quality
  - Add additional testing, verification and validation to the actuator models (line and disk) in Nalu-Wind and update the code documentation with these results
  - Document performance changes resulting from the code refactor
  - Outline the performance capabilities of the actuator disk model in simulations with coarse meshes and large time-steps relative to the actuator line model
- **Background:**
  - The actuator disk capability in Nalu-Wind is relatively new, and has not been utilized very much despite being one of the most heavily used turbine models in the community. This gap needs to be closed as Nalu-Wind's user base grows
  - A scalability issue under mesh refinement for the ADM was identified during the preliminary scoping of this milestone that is due to the fundamental implementation of the actuator models in Nalu-Wind
  - The scalability issue coupled with the NGP conversion for the code base necessitated a complete refactor of the actuator code base within Nalu-Wind
- **Milestone Team:**
  - SNL: Philip Sakievich (lead), Robert Knaus, Alan Hsieh, Lawrence Cheung, Myra Blaylock, David Maniaci
  - NREL: Luis 'Tony' Martinez-Tossas, Matthew J Churchfield

Advantages of the Actuator Disk Method

- **AD Simulation Results**
  - AD results are consistent regardless of time step
    - ~1% change in generated power between dt = 0.02 and 0.4 seconds
  - Slightly higher generated power for decreasing time step corresponds with previous SOWFA results
  - AD results for upwind turbine are very similar to AL results for generated power (within ~1%)
    - Downwind turbine shows greater discrepancy (~2-3%), likely due to AD smoothing of wake features
  - Main time step constraint for AD model becomes CFL number, instead of $\Delta t_{ALM} = \Delta x / U_{tip}$
    - AD model allows a ~20x increase in time step compared to AL model
  - AD model experienced no errors during a MLMF UQ pilot study for two separate meshes
    - Fifteen samples of five varying turbine input parameters
  - AD model simulation efficiency showed strong dependence on epsilon and grid spacing
    - Potential scalability issue in cached data from actuator calculations



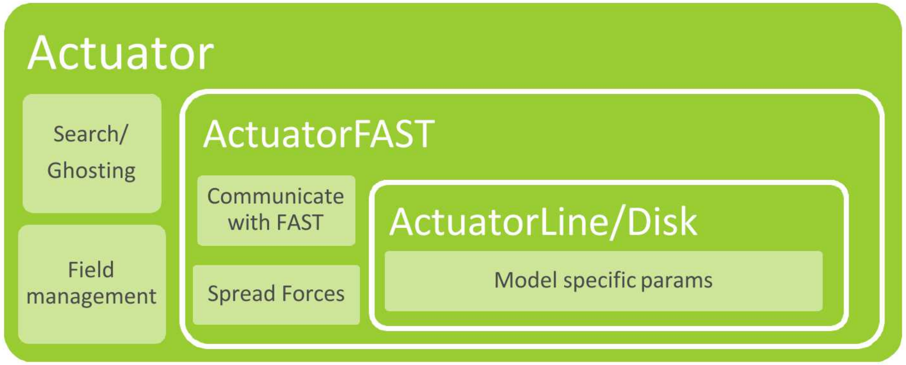
Old ADM Implementation and Scaling Descriptions

Implementation Overview:

- Built around OpenFAST actuator lines
  - ~80-90% + code reuse between ADM/ALM
  - 1 coarse search at start
  - Create additional points
  - Adjust force calculation
- Force from ALM points summed and distributed evenly at each radius
- One line change in input deck to switch from ALM to ADM

Scaling Issue:

- Scaling issue is due to parallel communication pattern
- All elements intersecting the gaussian to a certain radius are ghosted to single rank
- Ghosting causes load and memory imbalance across rank leading to a catastrophic bottle neck under mesh refinement
- Scaling issue is also present for the ALM method, but is less pronounced due to far fewer actuator points



Actuator

Search/ Ghosting

Field management

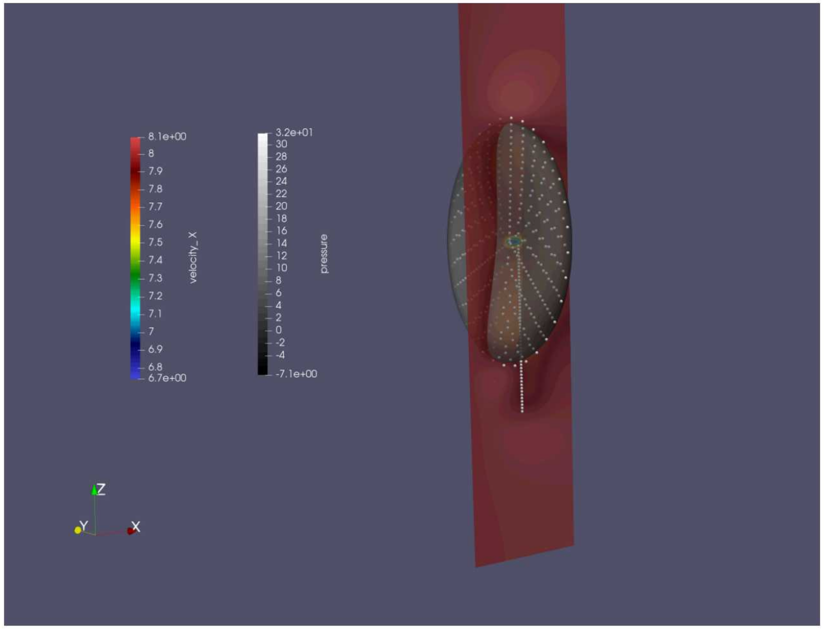
ActuatorFAST

Communicate with FAST

Spread Forces

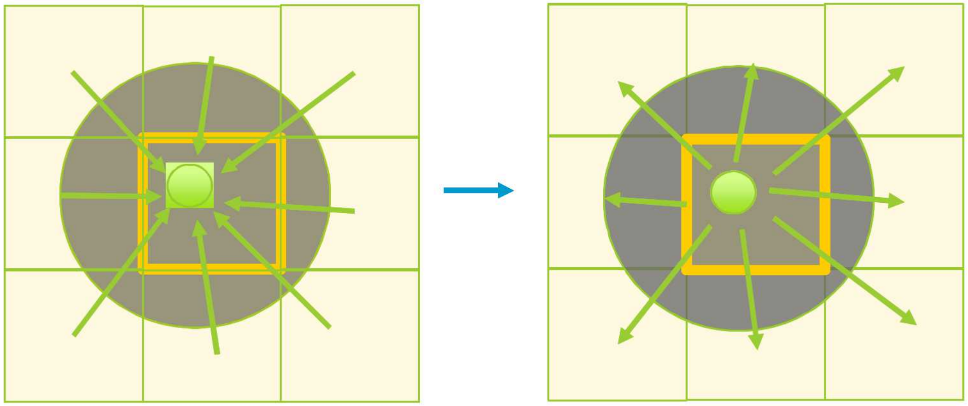
ActuatorLine/Disk

Model specific params



Block diagram of code description (top) and a sample image of ADM displaying actuator points, an iso-surface of the actuator force and contour of velocity (bottom)

Explanation of Code Refactor

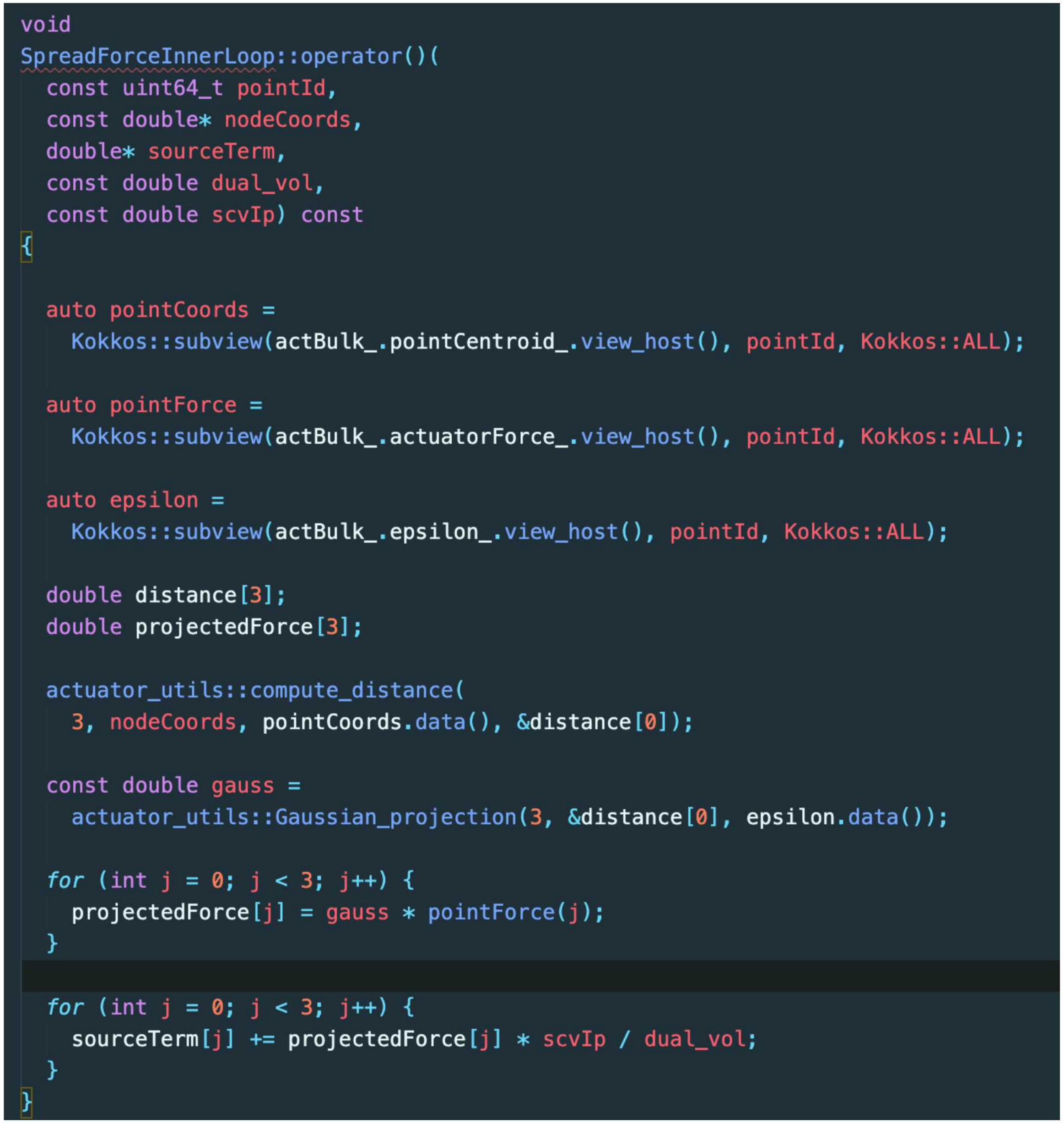


- Initial algorithm was unscalable
  - All cells influenced by a point communicated to a unique owner
- New algorithm distributed with all-reduce communication for actuator data
  - Even load balancing
  - Local only search

Explanation of Code Refactor

- New code broken into chunks that can be unit tested
- Code designed to be extendible for models that don't rely on OpenFAST
- Integration point-based assembly
- Keep old code in place for testing and deprecation
- Changes made to OpenFAST as well

Code Example



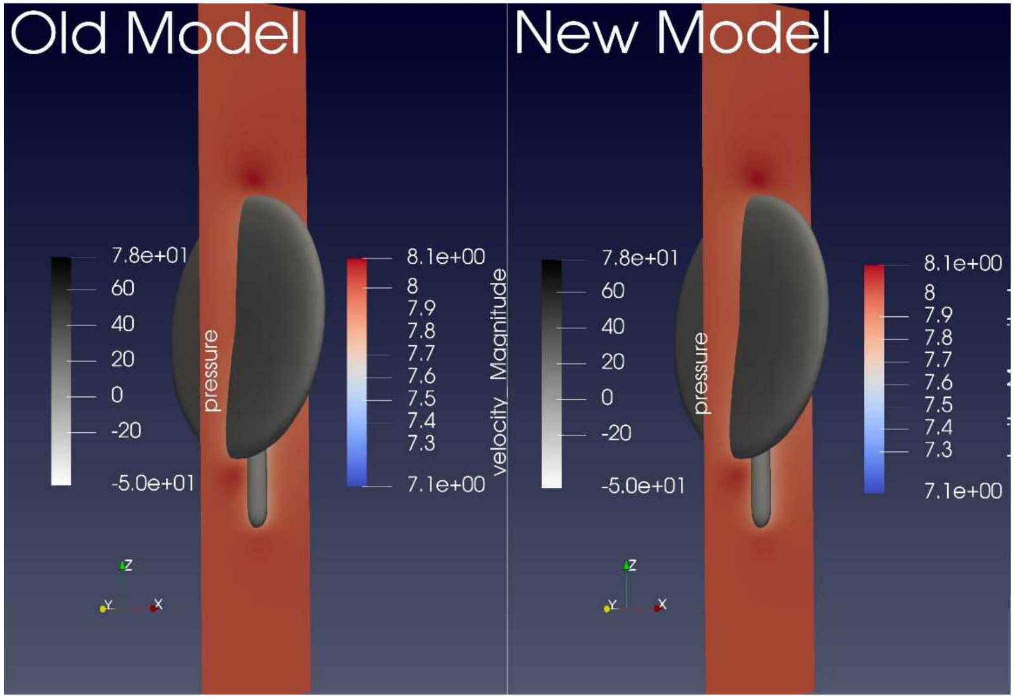
```
void
SpreadForceInnerLoop::operator()(
  const uint64_t pointId,
  const double* nodeCoords,
  double* sourceTerm,
  const double dual_vol,
  const double scvIp) const
{

  auto pointCoords =
    Kokkos::subview(actBulk_.pointCentroid_.view_host(), pointId, Kokkos::ALL);

  auto pointForce =
    Kokkos::subview(actBulk_.actuatorForce_.view_host(), pointId, Kokkos::ALL);

  auto epsilon =
    Kokkos::subview(actBulk_.epsilon_.view_host(), pointId, Kokkos::ALL);

  double distance[3];
  double projectedForce[3];

  actuator_utils::compute_distance(
    3, nodeCoords, pointCoords.data(), &distance[0]);

  const double gauss =
    actuator_utils::Gaussian_projection(3, &distance[0], epsilon.data());

  for (int j = 0; j < 3; j++) {
    projectedForce[j] = gauss * pointForce(j);
  }

  for (int j = 0; j < 3; j++) {
    sourceTerm[j] += projectedForce[j] * scvIp / dual_vol;
  }
}
```

Summary of Additional Testing

- Unit testing
  - Unit tests added to all major work units (24+ new unit tests)
  - New unit tests also cover OpenFAST API, and parsing to ensure user input errors receive specific Nalu-Wind generated error messages
  - Prior implementation only had 2 unit tests for peripheral calculations
- Regression tests
  - Fixed wing test
  - Anisotropic Gaussian ALM
- Verification & Validation
  - Momentum disk
  - Full scale ABL
  - Fixed wing

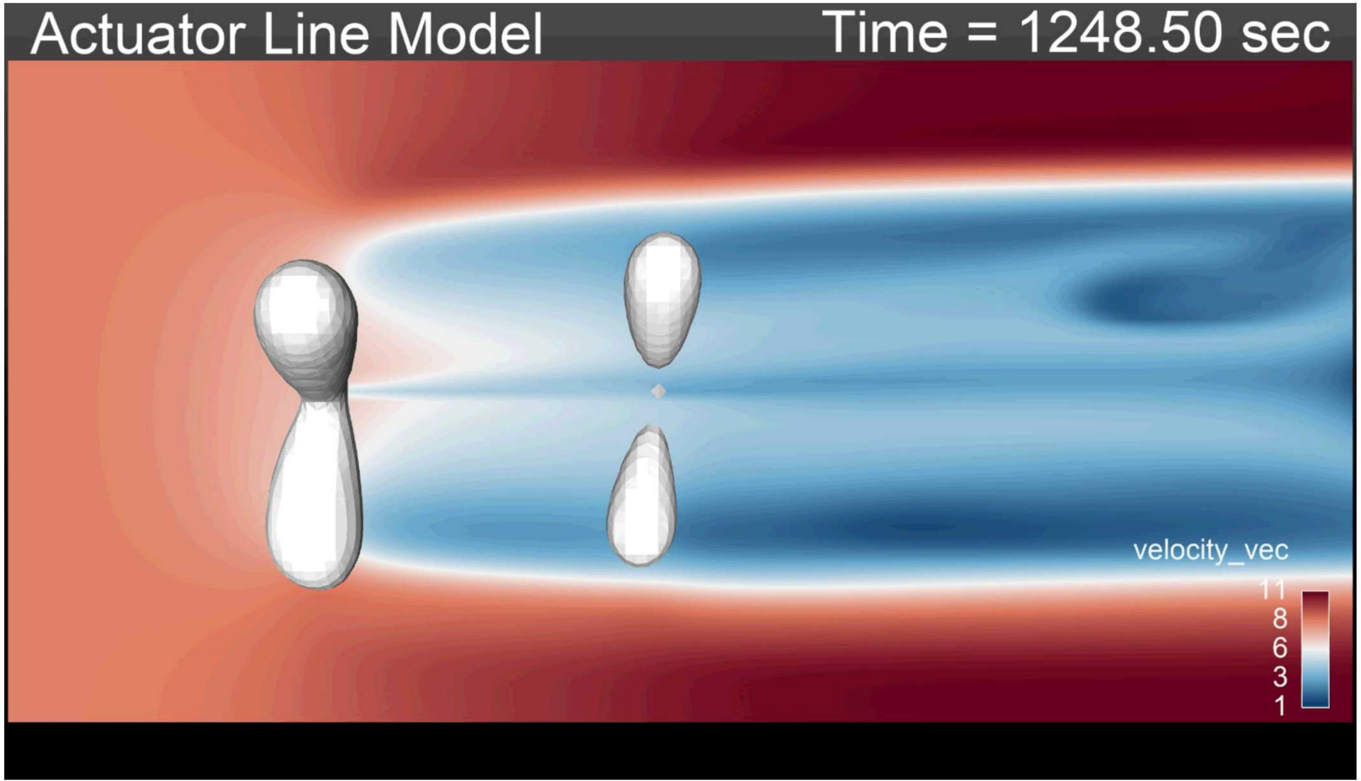
Old vs New Model: Single Turbine

- Resulting actuator source differs by less than 1e-6 in magnitude
- The new ADM is ~20% faster for a single turbine case running on 4 MPI ranks
- Similar results are also seen for the ALM
- More detail on timing in the V&V cases



Old vs New Model: Multi-turbine

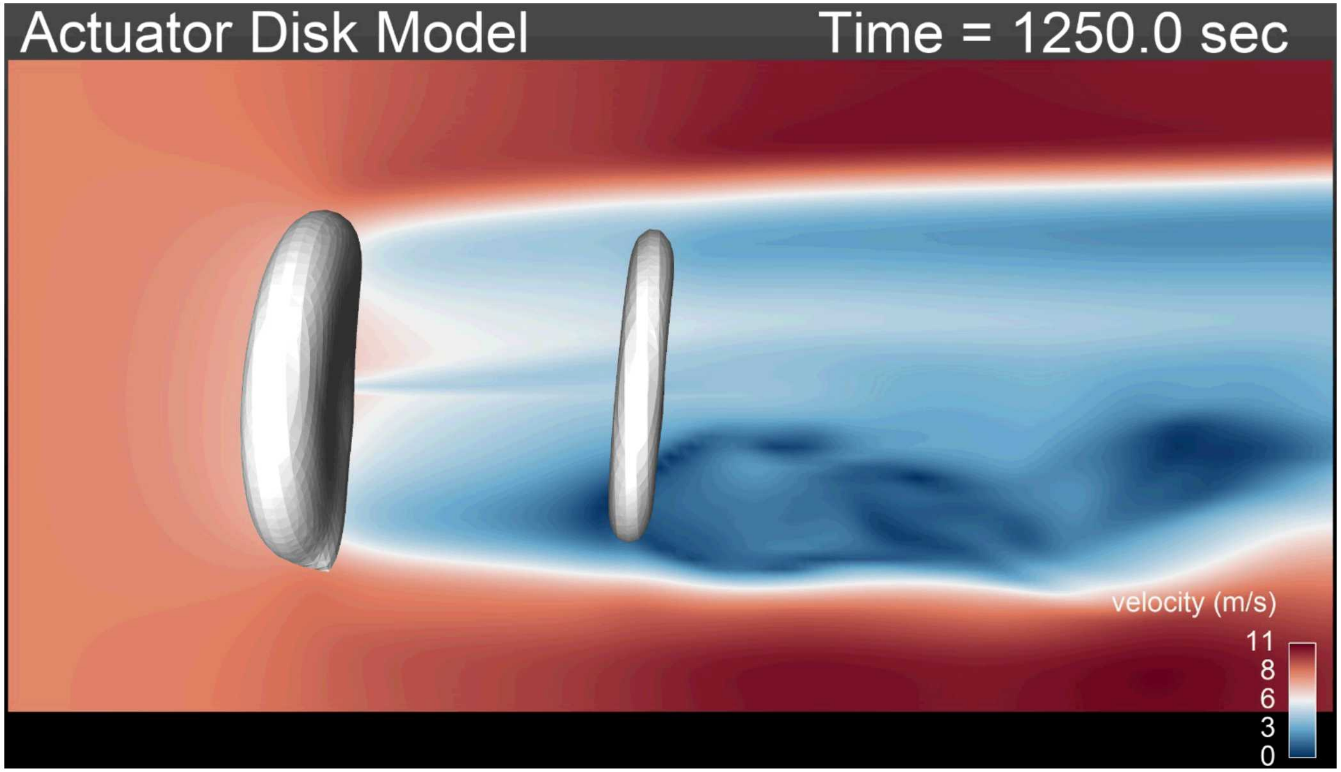
- New model can run multi-turbines with both ALM and ADM
- ~25% faster for two ALM turbines on 32 processors



- Steady 8m/s inflow
- NREL 5MW Turbines
- White represents actuator region

Old vs New Model: Multi-turbine

- "(Red) Apples to (Green) Apples", ADM is slower than ALM (Same force points/blade = $O(n^2)$ more for ADM)
- With 4x bigger time step and 15 force points/blade, ADM is faster than ALM



- Steady 8m/s inflow
- NREL 5MW Turbines
- White shows ADM

Old vs New Model: Multi-turbine

- Mesh has 524,288 elements ran on 32 processors
  - 16k elements/processor
- ALM: 50 force points/blade; dt = 0.125 s; 10k steps
- ADM: 15 force points/blade; dt = 0.5 s; 2500 steps

Model	Walltime to finish	Walltime secs per iter	% Speed up
ALM -old	13:15	4.77	
ALM -new	9:40	3.48	27%
ADM -new	4:10	6	69%

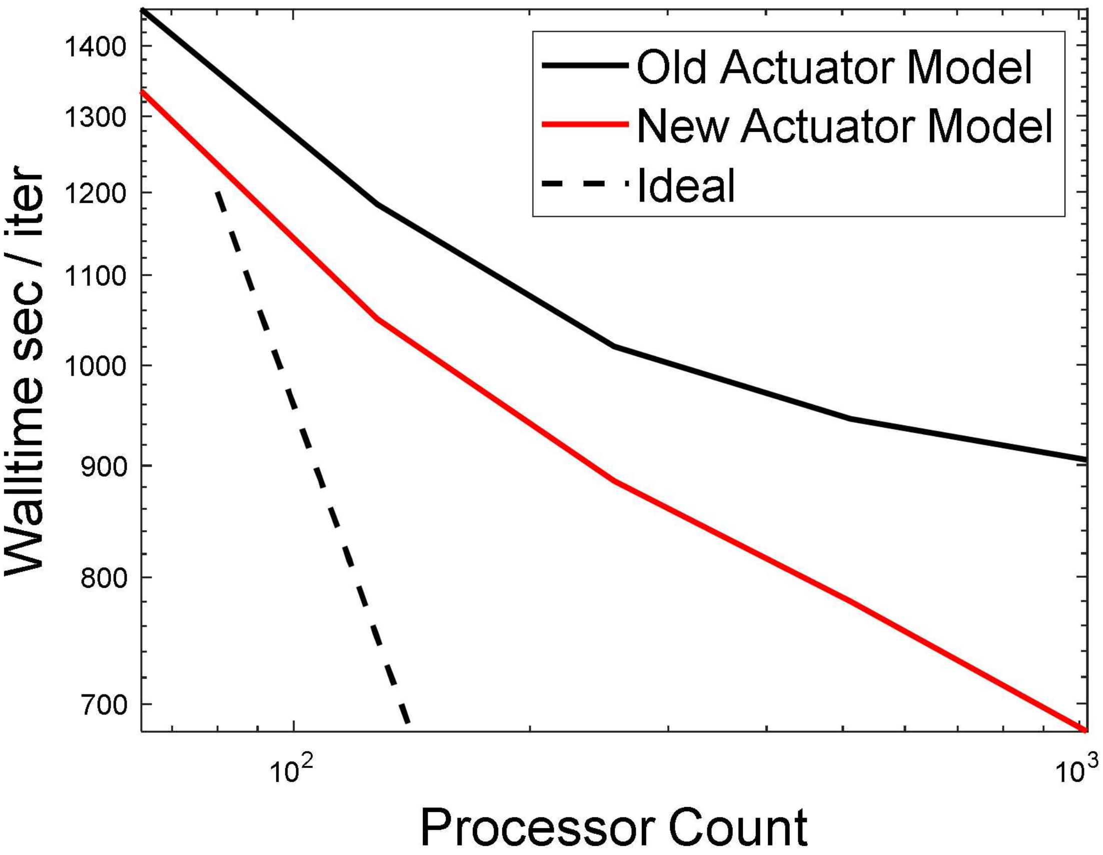
Old vs. New Model: Full ABL Simulations

- Full-scale ABL simulation comparisons using old and new actuator models; everything else constant
- Neutral ABL, U_∞=8.69 m/s, 3x3x1 km domain, single V27 turbine over 10 sim. minutes
- Generally observe significant decreases in walltime per iteration for both actuator disk/line models
- New actuator model at lowest tested resolution saw increase in walltime per iteration; currently under investigation

Turbine Model	# Mesh Elements	# Elements / Proc	Old Act Timing (Walltime secs / iter)	New Act Timing (Walltime secs / iter)	% Change
Actuator Disk	10.5e6	1.37e4	25.6	29.9	+16.8
Actuator Disk	11.8e6	1.54e4	76.9	72.6	-5.6
Actuator Disk	22.5e6	2.93e4	914.3	591.2	-35.3
Actuator Line	11.8e6	1.54e4	166.6	136.1	-18.3

Old vs. New Model: Full ABL Simulations

- Strong scalability comparison of new/old actuator disk models
- 22.5e6 element mesh; 2.93e4 elements/proc
- New model shows improved timing and scaling compared to old model
- Future planned studies
  - Similar strong scalability comparison for actuator line model
  - Do more timing comparisons at different mesh resolutions
  - Match actuator points between actuator line/disk models



Fixed Wing Validation Problem

Create a simple problem which can be used for quick validation of actuator computations

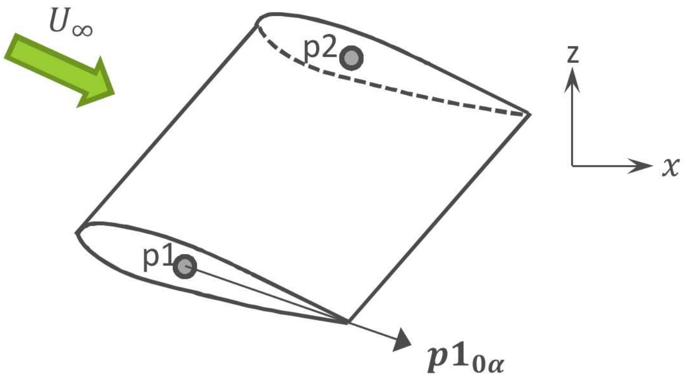
- Choose fixed wing with simple physics
  - Rigid body -- no aero-elastic coupling
  - No blade motion
  - No FAST dependencies

- Airfoil aerodynamics computed from polar tables
  - Lift and drag at each section defined by

$$L = \frac{1}{2}\rho U^2 C_L(\alpha) A_{blade}$$
$$D = \frac{1}{2}\rho U^2 C_D(\alpha) A_{blade}$$

  - Forces calculation can be verified by hand

Can build additional validation cases from fixed wing problem



Input parameters

```
actuator:
   type: ActLineSimple
   search_method: stk_kdtree
   search_target_part: Unspecified-2-HEX
   n_simpleblades:  1
   n_turbines_glob: 0
   Blade0:
     num_force_pts_blade: 10       # Number of blade stations
     epsilon: [3, 3, 3]
     p1: [0, -4, 1]                # Start of blade
     p2: [0,  4, 1]                # End of blade
     p1_zero_alpha_dir: [1, 0, 0]  # Direction of zero AOA at p1
                                   #    measured from LE to TE
     chord_table: [1.0, 1.0, 1.0]  # Chord at each blade station.
     twist_table: [0.0]            # Twist at each blade station.
     aoa_table: [-180, 0, 180]     # AOA for polar table
     cl_table:  [-19.739,0,19.739] # CL's for polar table
     cd_table:  [0]                # CD's for polar table
```

Fixed Wing Validation Problem

Using a simple 2D flat plate, with linear lift curve:

$$C_L = 2\pi\alpha$$
$$C_D = 0$$

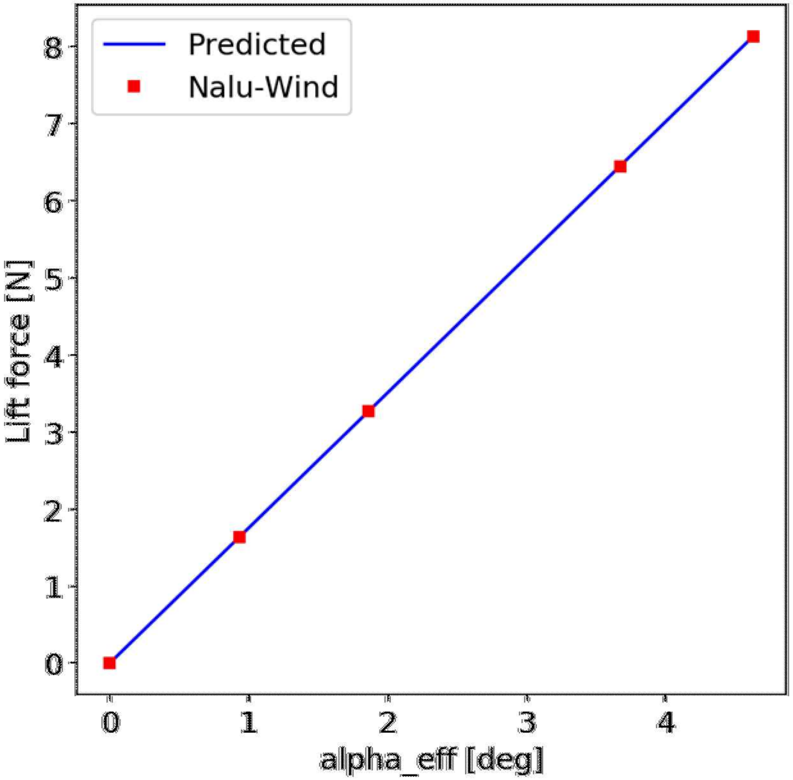
Isotropic force spreading

$$\varepsilon = 3$$

Blade parameters

- Chord = 1m
- Span = 8m
- 20 blade sections

All lift errors < 0.1%



Documentation

Documentation needs to be formalized and archived:

- Phil for source code/input parameters (Doxygen currently)
- Tony and Lawrence for fixed wing
- Myra, Alan and David summary of multi-turbine and full scale ABL
- All documents should be added to users manual and/or V&V manual

Conclusions and Next Steps

- New code is performing well, 25% speedup for some cases
- More robust scalability, memory scaling issues resolved
- Large improvement in testing: 24 unit tests added
- Workflow improvements, mainly guardrails
- Initial V&V efforts completed, but more in progress
- Next Steps
  - Deprecate and remove legacy code
  - Move as much execution to GPU as possible, add GPU testing
  - Face average (Rhie-Chow-esque) interpolation for ALM/ADM methods
  - Investigate actuator point and timestep dependencies
  - Additional formal documentation and V&V tests