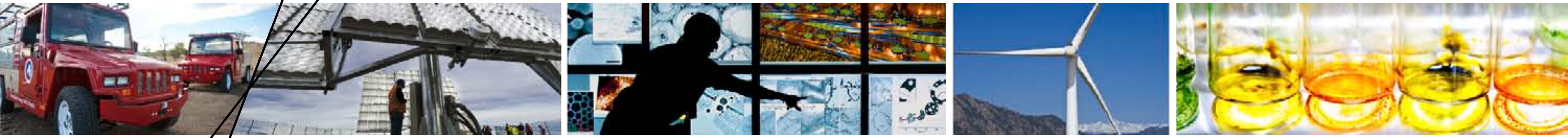
Overview of the Simulator for Offshore Wind Farm Application (SOWFA)



Matthew Churchfield
Sang Lee
Patrick Moriarty

May 3, 2012 (8:00 AM US MST)

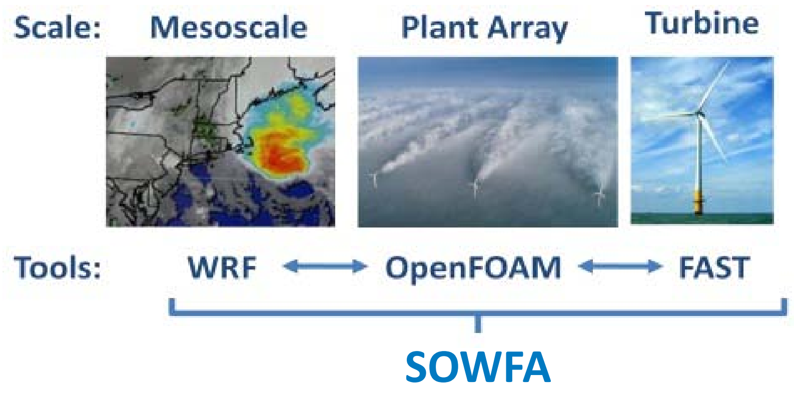
NREL is a national laboratory of the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, operated by the Alliance for Sustainable Energy, LLC.

Overview of SOWFA

- Simulator for Offshore Wind Farm Applications
- It is applicable to onshore too, but over next couple of years will contain offshore-specific tools
- Currently, it is composed of CFD tools based on OpenFOAM coupled with a NREL's FAST wind turbine structural/system dynamics model
- It is meant to be modular and open-source so that others can put in their own “modules”
- Open-source and freely available
- It can be downloaded at: http://wind.nrel.gov/designcodes/simulators/sowfa/

Overview of SOWFA

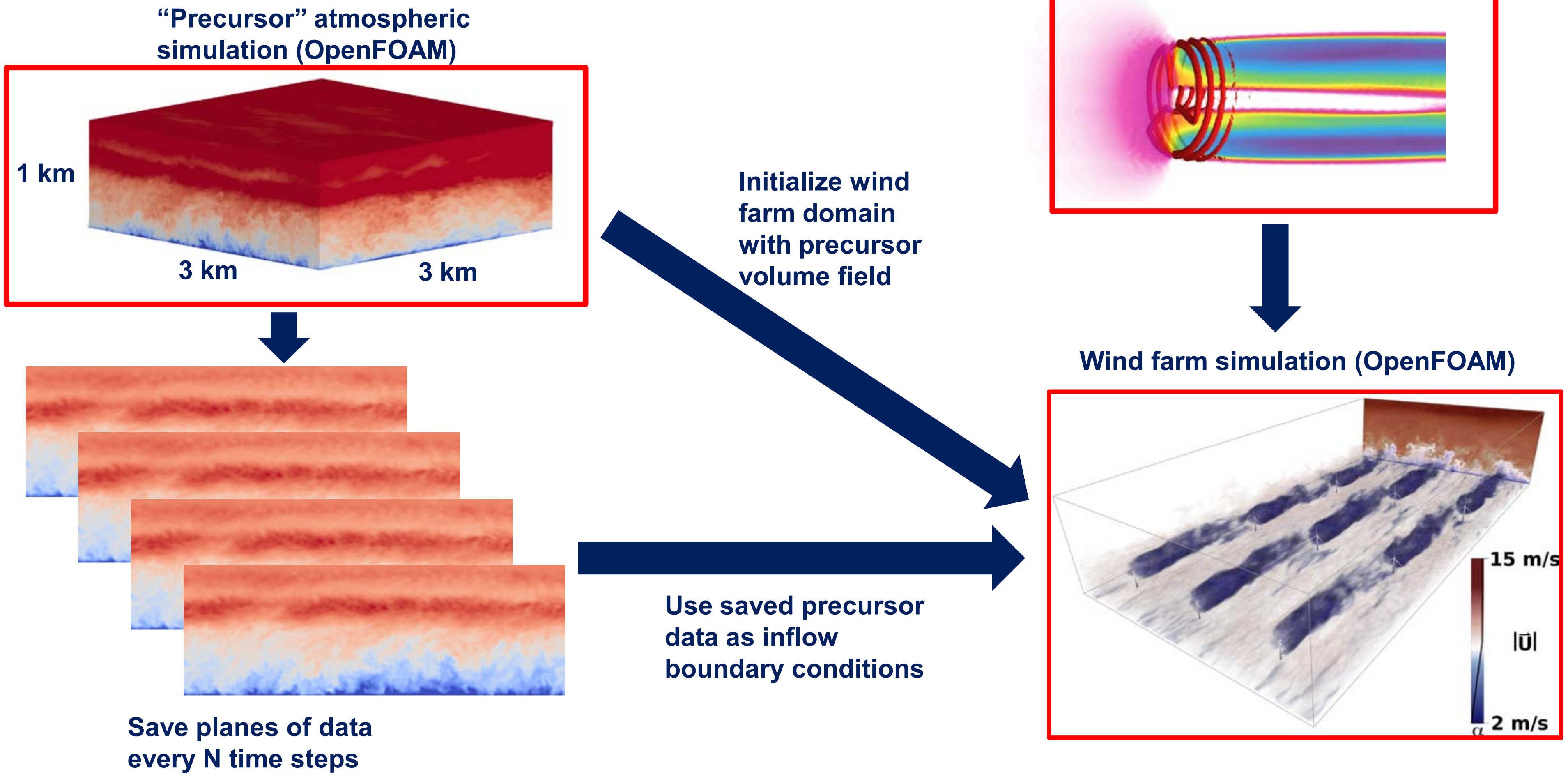
- **Simulator for Offshore Wind Farm Applications**
- **The overall vision is:**



- **It will have a range of fidelity levels**
  - Wakes computed from CFD or dynamic wake meandering model
  - Inflow turbulence computed from CFD or stochastic turbulence model

Overview of SOWFA

- **In its current form, SOWFA is:**



Actuator line turbine aerodynamics models (coupled with NREL's FAST turbine dynamics model)
"Precursor" atmospheric simulation (OpenFOAM)
1 km
3 km
3 km
Initialize wind farm domain with precursor volume field
Wind farm simulation (OpenFOAM)
15 m/s
|Ū|
2 m/s
Use saved precursor data as inflow boundary conditions
Save planes of data every N time steps

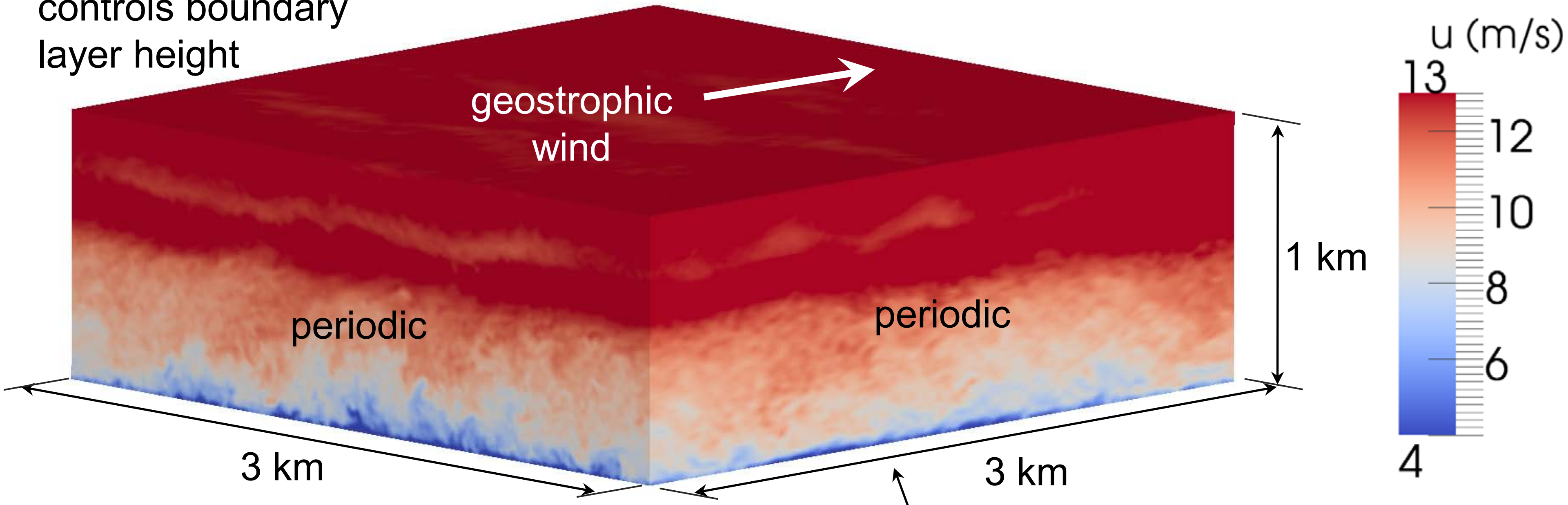
Atmospheric Boundary Layer Solver

Overview

ABLPisoSolver is a large-eddy simulation solver developed out of the buoyantBoussinesqPisoFoam that came with OpenFOAM-1.6. It cannot be run in RANS mode. It creates turbulent wind fields under a variety of atmospheric stability conditions.

capping inversion controls boundary layer height

simulation time 10000 –20000 s



Boussinesq approximation for buoyancy effects

Coriolis forces included

rough lower surface with temperature flux

Transport Equations

Momentum transport

$$\underbrace{\frac{\partial \overline{u}_i}{\partial t}}_{\text{I}} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{u}_j \overline{u}_i\right)}_{\text{II}} = \underbrace{-2\varepsilon_{i3k}\Omega_3 \overline{u}_k}_{\text{III}} \underbrace{- \frac{\partial \widetilde{p}}{\partial x_i}}_{\text{IV}} \underbrace{- \frac{1}{\rho_0}\frac{\partial}{\partial x_i}\overline{p}_0(x,y)}_{\text{V}} \underbrace{- \frac{\partial}{\partial x_j}\left(\tau_{ij}^D\right)}_{\text{VI}} \underbrace{- g\left(\frac{\overline{\theta} - \theta_0}{\theta_0}\right)\delta_{i3}}_{\text{VII}} \underbrace{+ \frac{1}{\rho_0} f_i^T}_{\text{VIII}}$$

- **I. time rate of change**
- **II. convection**
- **III. Coriolis force due to planetary rotation**
- **IV. density-normalized pressure gradient (deviation from hydrostatic and horizontal-mean gradient)**
- **V. horizontal-mean driving pressure gradient**
- **VI. SFS momentum fluxes (stresses)**
- **VII. buoyancy**
- **VIII. other density-normalized forces (from turbine actuator line model)**

Notice there is no viscous term. This is high Reynolds number flow. Viscous effects only significant very near planetary surface. Wall model will handle this (more later)

Transport Equations

Potential temperature transport

$$\underbrace{\frac{\partial \overline{\theta}}{\partial t}}_{\textbf{I}} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{u}_j \overline{\theta}\right)}_{\textbf{II}} = \underbrace{-\frac{\partial}{\partial x_j}\left(q_j\right)}_{\textbf{III}}$$

I. **time rate of change**

II. **convection**

III. **SFS temperature fluxes**

Notice there is no molecular temperature conduction term. This is high Reynolds number flow. Molecular effects only significant very near planetary surface. Wall model will handle this (more later)

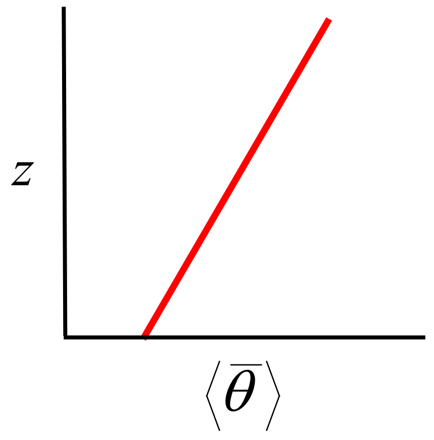
[1] provides a good explanation of atmospheric boundary layer physics.

[2] is a good outline of atmospheric boundary layer LES.

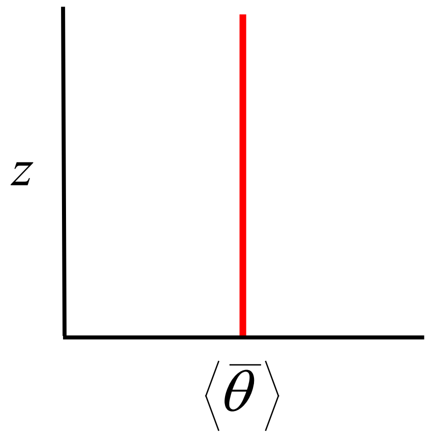
[1] R. B. Stull. *An Introduction to Boundary Layer Meteorology*. Springer Science + Business Media B. V., 2009.
[2] C.-H. Moeng. A Large-Eddy Simulation Model for the Study of Planetary Boundary Layer Turbulence. *Journal of the Atmospheric Sciences*, Vol. 41, No. 13,

Potential Temperature

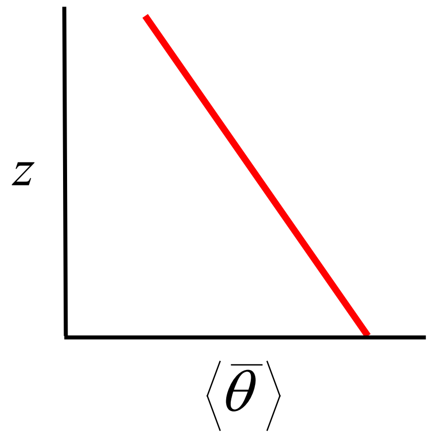
- **Temperature that a parcel of dry air would have if adiabatically brought from some pressure level to a reference pressure, usually 100kPA**
- **Simplifies the study of atmospheric stability**



stable



neutral



unstable

Buoyancy Force

- **This is an incompressible formulation, with constant density, so we need a way to account for buoyancy effects caused by variable density**
- **Use the Boussinesq approximation**
- **Buoyancy term is**

$$-g\left(\frac{\overline{\theta}-\theta_0}{\theta_0}\right)\delta_{i3} \qquad \theta_0 = 300\mathrm{K}$$

$$-(0,0,-9.81)\frac{\mathrm{m}}{\mathrm{s}^2}\left(\frac{299\mathrm{K}-300\mathrm{K}}{300\mathrm{K}}\right)\cdot(0,0,1) = -0.0327\frac{\mathrm{m}}{\mathrm{s}^2}$$

locally stable (cool air pushed **up** into warm air): **negative force**

$$-(0,0,-9.81)\frac{\mathrm{m}}{\mathrm{s}^2}\left(\frac{300\mathrm{K}-300\mathrm{K}}{300\mathrm{K}}\right)\cdot(0,0,1) = 0.0000\frac{\mathrm{m}}{\mathrm{s}^2}$$

locally neutral (air pushed into air of equal temperature): **zero force**

$$-(0,0,-9.81)\frac{\mathrm{m}}{\mathrm{s}^2}\left(\frac{301\mathrm{K}-300\mathrm{K}}{300\mathrm{K}}\right)\cdot(0,0,1) = +0.0327\frac{\mathrm{m}}{\mathrm{s}^2}$$

locally unstable (**warm** air pushed **up** into **cool** air): **positive force**

Coriolis Force

- Due to planetary rotation, there is an apparent force called Coriolis force
- If +x is east, +y is north, and +z is up, then

$$-2\varepsilon_{ijk}\Omega_j\overline{u}_k$$

$$\Omega_j = \omega\begin{bmatrix} 0 \\ \cos\phi \\ \sin\phi \end{bmatrix}$$

- Ω_j is the rotation rate vector at a location on the planetary surface, ω is the planetary rotation rate (rad/s), and ϕ is the lattitude

Sub-Filter Scale Model

Gradient-diffusion hypothesis

$$\tau_{ij}^{D} = -\upsilon^{SFS}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)$$

$$q_j = -\kappa^{SFS}\frac{\partial \overline{\theta}}{\partial x_j}$$

Smagorinsky model[1]

$$\upsilon^{SFS} = \left(C_s\Delta\right)^2\left[2\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)\right]^{1/2}$$

$$\kappa^{SFS} = \frac{\upsilon^{SFS}}{\mathrm{Pr}_t}$$

1 J. Smagorinsky. General Circulation Experiments with the Primitive Equations, *Monthly Weather Review*, Vol. 91, 1963, pp. 99–164.

Sub-Filter Scale Model

$C_s = 0.13 - 0.17$ (we use closer to 0.13) — Smagorinsky constant

$\Delta = V^{1/3}$ — SFS filter width (V is grid cell volume)

$$\Pr_t = \frac{1}{\left(1 + 2\frac{l}{\Delta}\right)}$$ Turbulent Prandtl number

$$l = \begin{cases} \min\left(7.6\frac{\upsilon^{SFS}}{\Delta}\left(\sqrt{\frac{1}{s}}\right), \Delta\right) & \text{if} \quad s > 0 \\ \Delta & \text{if} \quad s \le 0 \end{cases}$$ Length-scale for Prt

$$s = \frac{|g_i|}{\theta_0}\frac{\partial \overline{\theta}}{\partial z}$$ Measure of stability

If locally stable or neutral ($s \le 0$):	$\Pr_t = 1/3$	$\kappa^{SFS} = 3\upsilon^{SFS}$
If locally stable ($s > 0$):	$\Pr_t$ approaches 1	$\kappa^{SFS} \to \upsilon^{SFS}$

Sub-Filter Scale Model

- **Model implementation in ABLPisoSolver is done completely at cell-faces**
- **Avoids interpolation to faces for taking divergence of stress**
- **Provides a less dissipative effect near planetary surface**

Wall Model

- **The cost of high-Re fully-resolved LES of wall-bounded flow scales strongly with Re.**

  "The only economical way to perform LES of high Reynolds-number attached flow, therefore, is by computing the outer layer only." "Because the grid is too coarse to resolve the inner-layer structures, the effect of the wall layer must be modeled. In particular, the momentum flux at the wall (i.e., the wall stress) cannot be evaluated by discrete differentiation because the grid cannot resolve either the sharp velocity gradients in the inner layer or the quasi-streamwise and hairpin vortices that transfer momentum in this region of the flow. Therefore, some phenomelogical relation must be found to relate the wall stress to the outer-layer flow."[1]

- **The planetary surface is covered with roughness elements (dirt, rocks, vegetation) that would be extremely expensive to resolved with the grid.**
- **It is inappropriate to apply no-slip at the surface**
- **Instead apply a model for surface stress**

[1] U. Piomelli and E. Balaras, "Wall-Layer Models for Large-Eddy Simulations," Annual Review of Fluid Mechanics, Vol. 34, pp. 349–374, 2002.

Wall Model

- Surface stress model predicts total (viscous + SFS) stress at surface
- Assumes that first cell centers away from surface lie within surface layer of the atmospheric boundary layer
- So at the surface

$$\tau_{ij}^{D} = \begin{bmatrix} 0 & 0 & \tau_{13}^{tot} \\ 0 & 0 & \tau_{23}^{tot} \\ \tau_{13}^{tot} & \tau_{23}^{tot} & 0 \end{bmatrix}$$

- The wall model models τ_{13}^{tot} and τ_{23}^{tot}

Wall Model

- **ABLPisoSolver contains the wall models of**
  - Schumann[1]
  - Moeng[2]
- **Moeng's model**

$$\tau_{13}^{tot} = -u_*^2 \frac{S_{1/2}\left\langle \overline{u}_{1/2} \right\rangle + \left\langle S_{1/2} \right\rangle \left(\overline{u}_{1/2} - \left\langle \overline{u}_{1/2} \right\rangle \right)}{\left\langle S_{1/2} \right\rangle \left(\left\langle \overline{u}_{1/2} \right\rangle^2 + \left\langle \overline{v}_{1/2} \right\rangle^2 \right)^{1/2}}$$

$$\tau_{23}^{tot} = -u_*^2 \frac{S_{1/2}\left\langle \overline{v}_{1/2} \right\rangle + \left\langle S_{1/2} \right\rangle \left(\overline{v}_{1/2} - \left\langle \overline{v}_{1/2} \right\rangle \right)}{\left\langle S_{1/2} \right\rangle \left(\left\langle \overline{u}_{1/2} \right\rangle^2 + \left\langle \overline{v}_{1/2} \right\rangle^2 \right)^{1/2}}$$

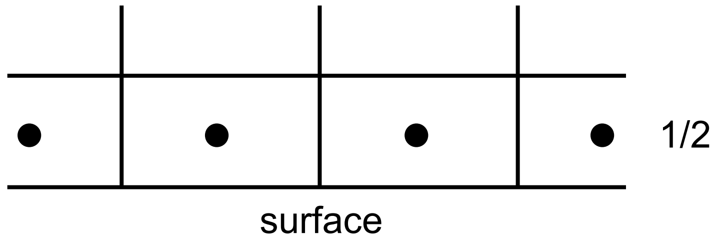
[1] U. Schumann. Subgrid-Scale Model for Finite-Difference Simulations of Turbulent Flow in Plane Channels and Annuli. Journal of Computational Physics, Vol. 18, 1975, pp. 76–404.
[2] C.-H. Moeng. A Large-Eddy Simulation Model for the Study of Planetary Boundary Layer Turbulence. Journal of the Atmospheric Sciences, Vol. 41, No. 13, 1984, pp. 2052–2062.

Wall Model

$$\tau_{13}^{tot} = -u_*^2 \frac{S_{1/2}\left\langle \overline{u}_{1/2} \right\rangle + \left\langle S_{1/2} \right\rangle \left(\overline{u}_{1/2} - \left\langle \overline{u}_{1/2} \right\rangle \right)}{\left\langle S_{1/2} \right\rangle \left(\left\langle \overline{u}_{1/2} \right\rangle^2 + \left\langle \overline{v}_{1/2} \right\rangle^2 \right)^{1/2}}$$

$$\tau_{23}^{tot} = -u_*^2 \frac{S_{1/2}\left\langle \overline{v}_{1/2} \right\rangle + \left\langle S_{1/2} \right\rangle \left(\overline{v}_{1/2} - \left\langle \overline{v}_{1/2} \right\rangle \right)}{\left\langle S_{1/2} \right\rangle \left(\left\langle \overline{u}_{1/2} \right\rangle^2 + \left\langle \overline{v}_{1/2} \right\rangle^2 \right)^{1/2}}$$

- **1/2 denotes values at first cell centers away from surface**



- **Angle brackets denote a horizontal average at a certain height**
- S **is the resolved velocity magnitude**

Wall Model

$$\tau_{13}^{tot} = -u_*^2 \frac{S_{1/2}\langle \overline{u}_{1/2}\rangle + \langle S_{1/2}\rangle\left(\overline{u}_{1/2} - \langle \overline{u}_{1/2}\rangle\right)}{\langle S_{1/2}\rangle\left(\langle \overline{u}_{1/2}\rangle^2 + \langle \overline{v}_{1/2}\rangle^2\right)^{1/2}}$$

$$\tau_{23}^{tot} = -u_*^2 \frac{S_{1/2}\langle \overline{v}_{1/2}\rangle + \langle S_{1/2}\rangle\left(\overline{v}_{1/2} - \langle \overline{v}_{1/2}\rangle\right)}{\langle S_{1/2}\rangle\left(\langle \overline{u}_{1/2}\rangle^2 + \langle \overline{v}_{1/2}\rangle^2\right)^{1/2}}$$

- **Friction velocity is defined as**

$$u_*^2 = \left(\left\langle \tau_{13}^{tot}\right\rangle^2 + \left\langle \tau_{23}^{tot}\right\rangle^2\right)^{1/2}$$

- **It needs to be approximated. Use rough wall log law**

$$\frac{\left(\langle \overline{u}_{1/2}\rangle + \langle \overline{v}_{1/2}\rangle\right)^{1/2}}{u_*} = \frac{1}{\kappa}\ln\left(\frac{z}{z_0} + f(L)\right)$$

Wall Model

$$\frac{\left(\left\langle \overline{u}_{1/2} \right\rangle + \left\langle \overline{v}_{1/2} \right\rangle\right)^{1/2}}{u_*} = \frac{1}{\kappa} \ln\left(\frac{z}{z_0} + f(L)\right)$$

- ***f* (*L*)** is an atmospheric stability-related function that is zero for neutral stability. See Etling[1] for more information
- ***L*** is the Obuhkov length
- **z_0** is the aerodynamic roughness height. It depends on height, distribution, and shape of roughness elements on planetary surface. See Stull[2] for more information

z_0 (m)	Terrain
1×10^{-1} – 5×10^{-1}	Many trees, hedges, few buildings
3×10^{-3} – 2×10^{-2}	Level grass plains
1×10^{-4} – 1×10^{-3}	Large expanses of water

[1] D. Etling. Modelling the Vertical ABL Structure, in *Modelling of Atmospheric Flow Fields*, D. P. Lalas and C. F. Ratto, editors, World Scientific, 1996, pp. 56–57.
[2] R. B. Stull. *An Introduction to Boundary Layer Meteorology*. Springer Science

Wall Model

- **A similar approach is taken to model the total temperature flux at the surface[1]**

$$q_j = \begin{bmatrix} 0 \\ 0 \\ q_3^{tot} \end{bmatrix}$$

- **Total average temperature flux, Q_s, is specified, and the wall model creates the fluctuating temperature flux** q_3^{tot}

1 C.-H. Moeng. A Large-Eddy Simulation Model for the Study of Planetary Boundary Layer Turbulence. Journal of the Atmospheric Sciences, Vol. 41, No. 13, 1984, pp. 2052–2062.

SGS Flux Formulation

- **ABLPisoSolver does not use “standard” OpenFOAM SGS flux divergence form**
- **Standard:**
  - Compute SGS viscosity at cell centers, interpolate to faces
  - Calls the function divDevReff()

```
return
(
  - fvm::laplacian(nuEff(), U)
  - fvc::div(nuEff()*dev(T(fvc::grad(U))))
);
```

$$\frac{\partial}{\partial x_j}\tau_{ij}^{D} = \underbrace{\frac{\partial}{\partial x_j}\left[-2\left(\upsilon+\upsilon^{SGS}\right)\overline{S}_{ij}\right]}_{\textbf{eddy viscosity form}} = \underbrace{-\frac{\partial}{\partial x_j}\left[\left(\upsilon+\upsilon^{SGS}\right)\frac{\partial \overline{u}_i}{\partial x_j}\right]}_{\textbf{implicit}} \underbrace{-\frac{\partial}{\partial x_j}\left[\left(\upsilon+\upsilon^{SGS}\right)\frac{\partial \overline{u}_j}{\partial x_i}\right]}_{\textbf{explicit}}$$

SGS Flux Formulation

- **ABLPisoSolver does not use "standard" OpenFOAM SGS flux divergence form**
- **Our way:**
  - Computes SGS viscosity on cell faces, not cell centers. *Less dissipative near wall solution*
  - Explicitly computes momentum flux on boundaries with wall stress model
  - Explicitly compute SGS stress at interior

$$\tau_{ij}^{D} = -2\left(\upsilon + \upsilon^{SGS}\right)\overline{S}_{ij}$$

Interior cell faces: explicit

$$\tau_{ij}^{D} = \tau_{ij}^{tot}$$

Boundary cell faces: explicit

$$\frac{\partial}{\partial x_j}\tau_{ij}^{D}$$

Once momentum fluxes formed, then take divergence

SGS Flux Formulation

- **ABLPisoSolver does not use “standard” OpenFOAM SGS flux divergence form**
- **Our way:**
  - Limitations
    - It’s explicit, but we have not been limited by that
    - Can not use SGS models that come with OpenFOAM
- **We desire to better understand why our way is less dissipative (favorable) near wall**
- **Find a way to go to “standard” formulation, but maintain the low dissipation**

Numerical Scheme

- **Like most other OpenFOAM solvers, ABLPisoSolver uses the PISO[1] (Pressure Implicit Splitting Operation) to "implicitly" solve the momentum and pressure equation**
  - I say "implicitly" because the SFS stress, temperature, and buoyancy are not solved implicitly. They are based on previous time step and solved sequentially
  - Predictor-Corrector approach

[1] R. I. Issa. Solution of the Implicitly Discretized Fluid Flow Equations by Operator-Splitting. *Journal of Computational Physics*, Vol. 62, 1985, pp. 40–65.

Numerical Scheme

- **Finite-volume formulation**
  - Linear interpolate of cell-center values to cell faces when needed
  - Equivalent to second-order central differencing
  - Rhie-Chow[1]-like flux interpolation is used to avoid pressure-velocity decoupling

[1] C. M. Rhie and W. L. Chow. Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation. *AIAA Journal*, Vol. 21, No. 11, 1983, pp. 1552–1532.

Linear System Solvers

- **Velocity and Temperature**
  - Biconjugate Gradient
  - Diagonal incomplete LU matrix preconditioner
- **Pressure**
  - Preconditioned Conjugate Gradient
  - Diagonal incomplete Cholesky matrix preconditioner

    OR

  - Geometric agglomerated algebraic multigrid solver
  - Diagonal incomplete Cholesky smoother

Solver Inputs

- **“0” directory**
  - U
  - T
  - pd
- **“system” directory**
  - controlDict
  - fvSchemes
  - fvSolution
  - decomposeParDict
- **“constant” directory**
  - “polyMesh” directory
  - ABLProperties
  - transportProperties
  - g
  - Omega

Solver Inputs

constant/ABLProperties

```
// Is average wind at a specified height driven to a specified velocity?
driveWindOn              true;

// Desired horizontally-averaged wind speed at a certain height (m/s)
UWindSpeed               UWindSpeed [0 1 -1 0 0 0 0] 9.0;

// Desired horizontally-averaged wind direction at a height (degrees)
UWindDir                 225.0;

// Height at which horizontally-averaged wind vector is specified (m)
hWind                    hWind [0 1 0 0 0 0 0] 90.0;

// Relaxation factor on the pressure gradient control
alpha                    0.9;

// Name of the lower boundary
lowerBoundaryName        "bottom";

// Name of the upper boundary
upperBoundaryName        "top";

// Are statistics to be gathered?
statisticsOn             true;

// At which frequency are statistics to be taken and written?
statisticsFrequency      5;

// ************************************************************
```

drive wind to specified velocity at specified height

specified wind speed

specified wind direction (direction blowing from)

specified wind height

relaxation factor on driving pressure gradient update

boundary patch name corresponding to lower surface

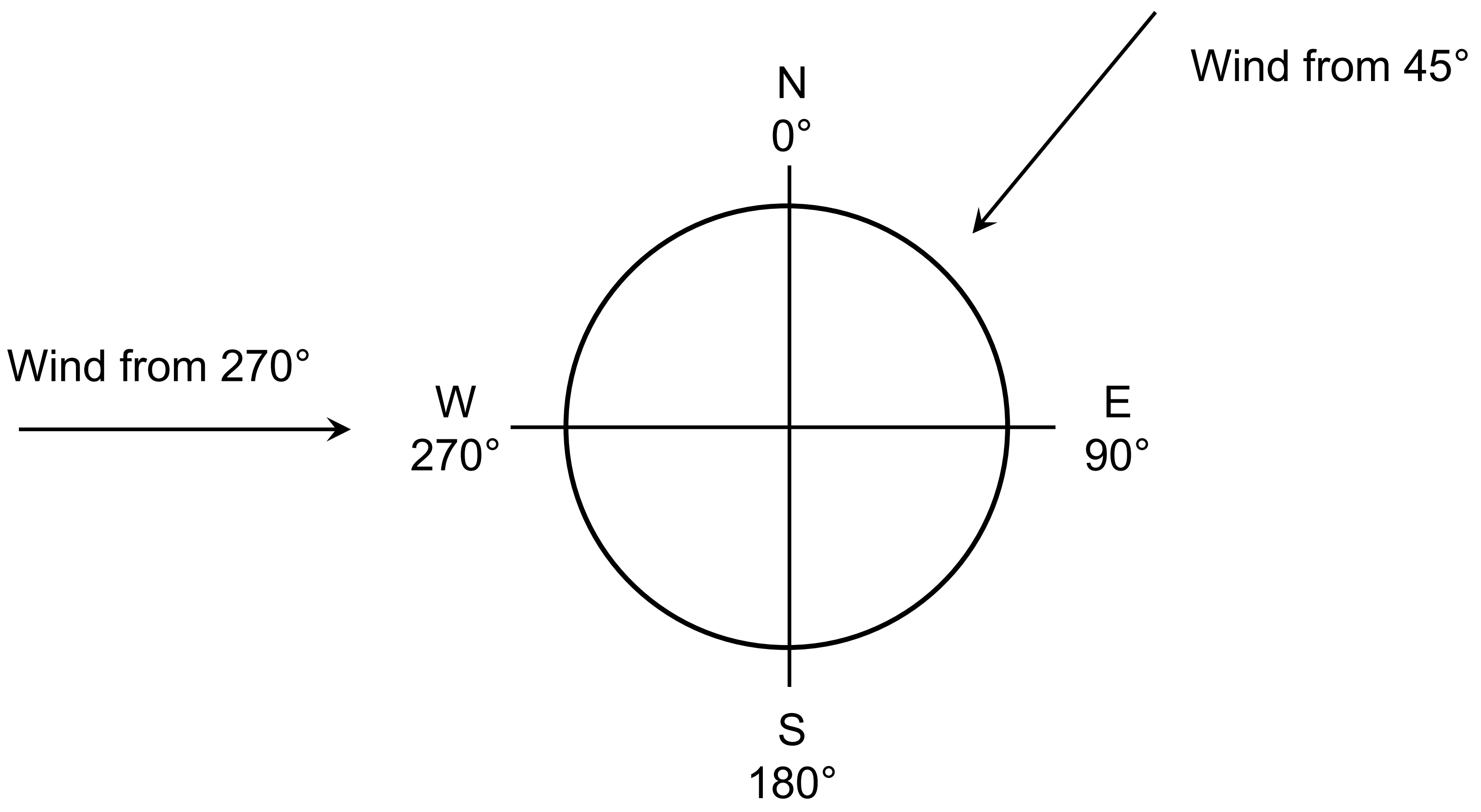
boundary patch name corresponding to upper surface

gather statistics about boundary layer?

statistics gathering frequency (every n time steps)

Solver Inputs

constant/ABLProperties



Solver Inputs

constant/transportProperties

```
transportModel Newtonian;

// Molecular viscosity (m^2/s^2)
nu                  nu [0 2 -1 0 0 0 0] 0.0;

// Reference temperature (K)
TRef                TRef [0 0 0 1 0 0 0] 300;

// LES SGS model (options are "standardSmagorinsky")
LESModel           "standardSmagorinsky";

// Smagorinsky Constant
Cs                  0.135;

// LES filter width scalar
deltaLESCoeff       1.0;

// von Karman constant
kappa               0.40;

// Constants for Monin-Obuhkov universal constants
betaM               16.0;
gammM               5.0;

// Roughness height (m)
z0                  z0 [0 1 0 0 0 0 0] 0.016;

// Surface temperature flux (K-m/s)
q0                  q0 [0 1 -1 1 0 0 0] 0.0;

// Surface stress model (options are "Schummann“ or "Moeng")
surfaceStressModel "Moeng";


// ************************************************************************* //
```

solver reads this molecular viscosity, but does not use it (need to fix this in the future)

reference temperature (inverse should correspond to fluid expansion ratio)

SFS model (currently limited to standard Smagorinsky)

Smagorinsky model constant

LES filter width is cube root of cell volume times this coefficient

von Karman constant

used for calculating friction velocity in non-neutral flow

aerodynamic roughness height

mean surface temperature flux

surface stress model (wall model)

Solver Inputs

constant/g

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           ( 0.0 0.0 -9.81 );


// ************************************************************************* //
```

value of acceleration due to gravity

constant/Omega

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 -1 0 0 0 0];
value           (0.0 5.1422E-5 5.1422E-5);


// ************************************************************************* //
```

rotation rate vector at a location on planet for Coriolis force

Remember, this rotation rate is:

$$\Omega_j = \omega \begin{bmatrix} 0 \\ \cos\phi \\ \sin\phi \end{bmatrix}$$

Earth's rotation speed is 1 rev / 24 hours, or 7.2722 × 10-5 rad / second.

At a latitude of 45° north, we have:

$$\Omega_j = 7.2722\times10^{-5} \begin{bmatrix} 0 \\ \cos 45° \\ \sin 45° \end{bmatrix} = \begin{bmatrix} 0 \\ 5.14522\times10^{-5} \\ 5.14522\times10^{-5} \end{bmatrix} \text{rad/s}$$

Solver Inputs

system/controlDict

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application       ABLPisoSolver;
libs              ("libuserfiniteVolume.so");
startFrom         startTime;
startTime         0.0;
stopAt            endTime;
endTime           10000.0;
deltaT            0.1;
writeControl      adjustableRunTime;
writeInterval     2000.0;
purgeWrite        0;
writeFormat       binary;
writePrecision    12;
writeCompression  uncompressed;
timeFormat        general;
timePrecision     6;
runTimeModifiable yes;
adjustTimeStep    yes;
maxCo             0.75;
maxDeltaT         25.0;
// ************************************************************************* //
```

Need to use the library to use custom buoyantBoussinesqMod boundary condition for pressure

run at a constant Courant number (adjust time step)

Solver Inputs

system/fvSchemes

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default                     CrankNicholson 1.0;
}

gradSchemes
{
    default                     Gauss linear;
}

divSchemes
{
    default                     Gauss linear;
}

laplacianSchemes
{
    default                     Gauss linear uncorrected;
}

interpolationSchemes
{
    default                     linear;
}

snGradSchemes
{
    default                     uncorrected;
}

fluxRequired
{
    default                     no;
    pd                            ;
}
// ************************************************************************* //
```

we use Crank Nicolson time marching

all interpolation to faces is linear (second-order central) because when doing LES, we do not want dissipation associated with upwind schemes

Typical canonical ABL meshes are completely orthogonal, so no non-orthogonal correction is needed

d(pd)/dx at faces is needed to update velocity fluxes

Solver Inputs

system/fvSolution

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    pd
    {
        solver          GAMG;
        tolerance       1e-6;
        relTol          0.01;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 100;
        agglomerator    faceAreaPair;
        mergeLevels     2;
    }

    pdFinal
    {
        solver          GAMG;
        tolerance       1e-8;
        relTol          0.0;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 100;
        agglomerator    faceAreaPair;
        mergeLevels     2;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-6;
        relTol          0;
    }

    T
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-6;
        relTol          0;
    }
```

Typical solver settings

GAMG is generally used for pressure solve

PCG is generally fine for U and T on all sizes of grids

Solver Inputs

system/fvSolution

```
options
{
    nCorrectors                3;
    nNonOrthogonalCorrectors 0;


    pdRefOn                    true;
    pdRefCell                  55;
    pdRefValue                 0;


    tempEqnOn                  true;
}
// ************************************************************************* //
```

typical solver settings (continued)

1 PISO predictor followed by 3 correctors. No non-orthogonal correction on typical orthogonal grids

gradient boundary conditions are used on pressure, so pressure level needs to be set at some cell to "tack" down pressure level

turn temperature equation on or off

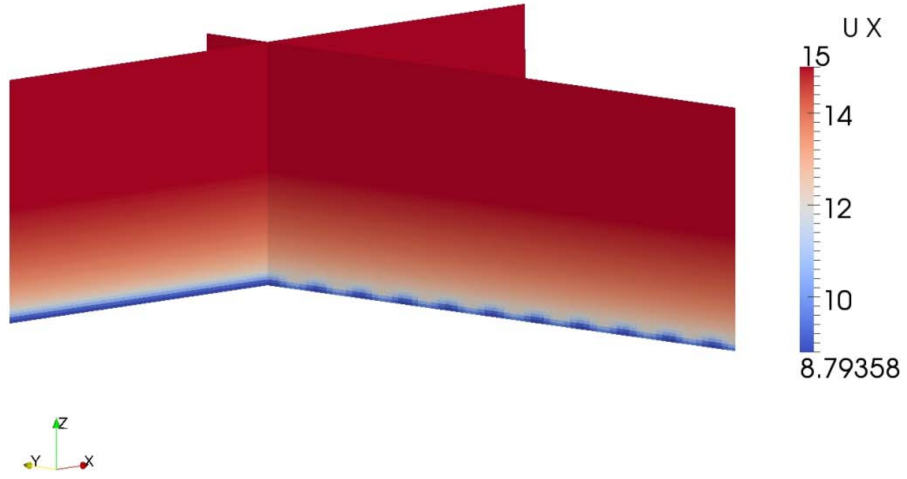
Solver Inputs

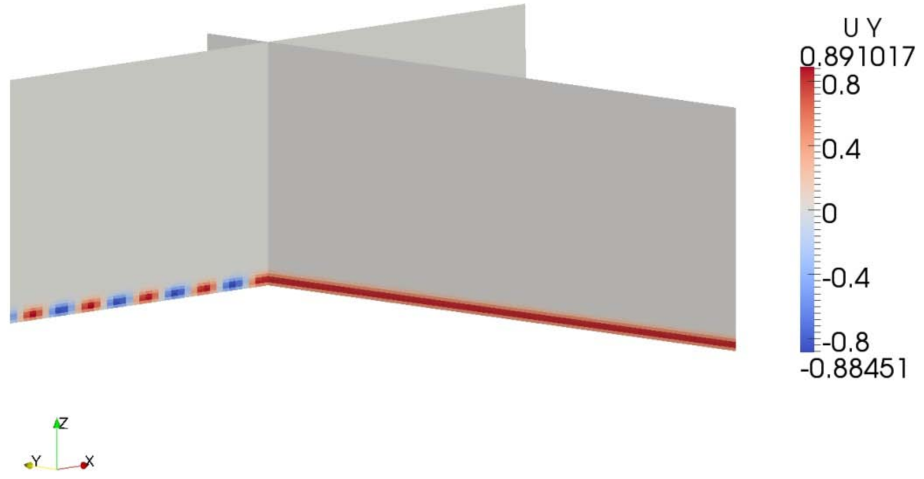
Initial conditions

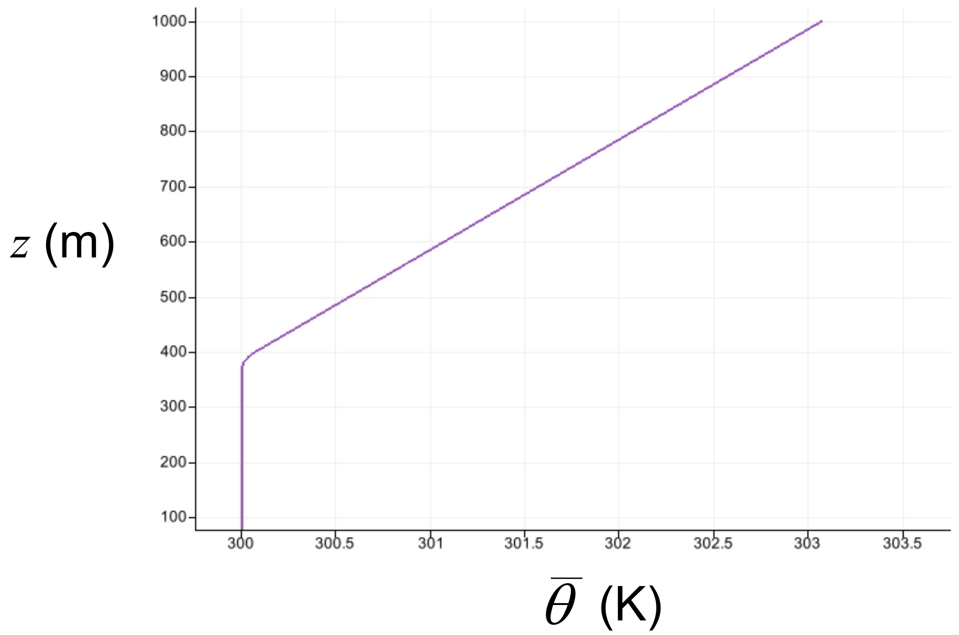
- **Velocity**
  - Given a logarithmic base profile
  - Non-random, divergence-free perturbations added near surface to cause turbulence to quickly happen (similar to method used by DeVillier's in channel flow[1]).
- **Temperature**
  - Constant temperature (300K) up to some height, then temperature increases
  - This creates a capping inversion that caps the boundary layer and slows boundary layer vertical growth
- **Pressure variable**
  - Initialized to zero
- **Initial conditions set using "setABLFields" utility (find in precursorABL tutorial). Could use something like "funkySetFields"**

[1] De Villiers, E., "The Potential of Large Eddy Simulation for the Modeling of Wall Bounded Flows", PhD Thesis, Imperial College, London, 2006.

Solver Inputs







Solver Outputs

- **Solution files (inside time directories)**
  - U, pd, T, Uprime, Tprime, nuLES*, kappaLES*
  - * means defined on cell faces instead of cell centers
- **“averaging” directory**
  - Horizontally-averaged profiles of quantities like velocity, temperature, velocity variances, velocity fluxes, temperature fluxes, third-order moments
  - Histories of friction velocity, boundary layer depth, and more

Solver Outputs

- **“averaging” file structure**
  - Within averaging directory are time directories corresponding to run start times. If you start a run at 0, there will be a “0” directory. If you restart a run at 1000, there will also be a “1000” directory.
  - Most files are structured as follows where each line represents a different time step, and starting at the third column, each column represents a horizontally-averaged value at a progressively greater height on the grid

$$
\begin{array}{llllll}
\texttt{time}^0 & \texttt{dt}^0 & \texttt{value}_0 & \texttt{value}_1 & \texttt{value}_2 \ldots & \texttt{value}_J \\
\texttt{time}^1 & \texttt{dt}^1 & \texttt{value}_0 & \texttt{value}_1 & \texttt{value}_2 \ldots & \texttt{value}_J \\
 & & & & \ldots & \\
\texttt{time}^N & \texttt{dt}^N & \texttt{value}_0 & \texttt{value}_1 & \texttt{value}_2 \ldots & \texttt{value}_J
\end{array}
$$

  - Heights corresponding the value_0 through value_J are in either the hLevelsCell or hLevelsFace file
    - hLevelsCell are cell-centered heights
    - hLevelsFace are heights of horizontally-situated faces

Solver Outputs

- **“averaging” file structure**

Cell-center quantities	Description
T_mean	$\langle \overline{\theta} \rangle$
U_mean, V_mean, W_mean	$\langle \overline{u} \rangle$ $\langle \overline{v} \rangle$ $\langle \overline{w} \rangle$
uu_mean, vv_mean, ww_mean	$\langle u'u' \rangle$ $\langle v'v' \rangle$ $\langle w'w' \rangle$
uv_mean, uw_mean, vw_mean	$\langle u'v' \rangle$ $\langle u'w' \rangle$ $\langle v'w' \rangle$
wuu_mean, wvv_mean, www_mean	$\langle w'u'u' \rangle$ $\langle w'v'v' \rangle$ $\langle w'w'w' \rangle$
wuv_mean, wuw_mean, wvw_mean	$\langle w'u'v' \rangle$ $\langle w'u'w' \rangle$ $\langle w'v'w' \rangle$
Tu_mean, Tv_mean, Tw_mean	$\langle \theta'u' \rangle$ $\langle \theta'v' \rangle$ $\langle \theta'w' \rangle$

Solver Outputs

- **“averaging” file structure**

Cell-face quantities	Description
R11_mean, R22_mean, R33_mean	$\langle \tau_{11}^D \rangle$ $\langle \tau_{22}^D \rangle$ $\langle \tau_{33}^D \rangle$
R12_mean, R13_mean, R23_mean	$\langle \tau_{12}^D \rangle$ $\langle \tau_{13}^D \rangle$ $\langle \tau_{23}^D \rangle$
q1_mean, q2_mean, q3_mean	$\langle q_1 \rangle$ $\langle q_2 \rangle$ $\langle q_3 \rangle$
phiM	ϕ_m Non-dimensional velocity shear

Global quantities	Description
ReLES	Re_{LES} LES Reynolds number[1]
scriptR	$\Re$ Near surface ratio of resolved to subgrid scale stress[1]
uStar	u_* Friction velocity
zi	z_i Boundary layer depth

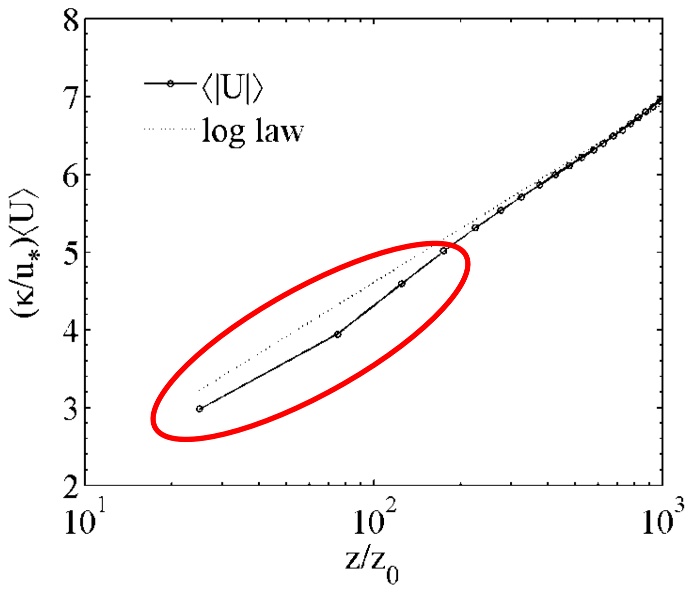
[1] J. Brasseur and T. Wei. Designing Large-Eddy Simulation of the Turbulent Boundary Layer to Capture Law-of-the-Wall Scaling, *Physics of Fluids*, Vol. 22, No. 2, 2010.

Guidelines for Use

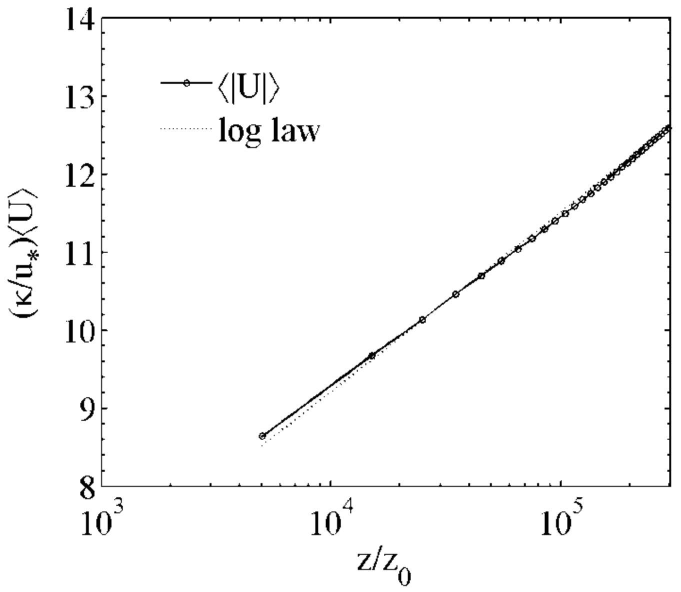
- **$+x$ must be east, $+y$ must be north, $+z$ must be up**
- **Domain must be large enough to resolve large structures**
  - At least 3km in horizontal and 1km in vertical for neutral and lightly unstable cases
  - At least 5km in horizontal and 2km in vertical for moderately to strongly convective cases
    - The cases will resolve large convection cells or rolls
- **Must use adequate vertical grid resolution, small enough cell aspect ratio, and proper Smagorinsky constant to recover law-of-the-wall scaling**

Guidelines for Use

- **Law-of-the-wall scaling**
  - This follows the work of Brasseur and Wei[1]
  - The problem:



Log-law mismatch



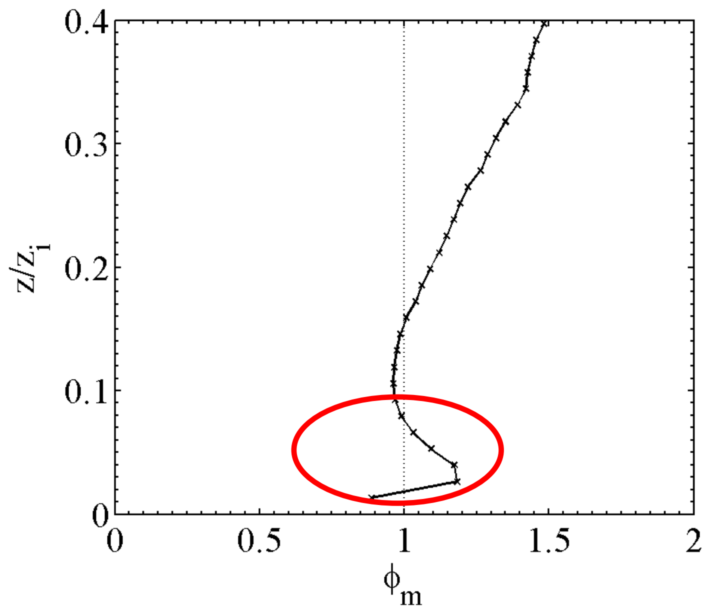
Improved log-law agreement

[1] J. Brasseur and T. Wei. Designing Large-Eddy Simulation of the Turbulent Boundary Layer to Capture Law-of-the-Wall Scaling, *Physics of Fluids*, Vol. 22, No. 2, 2010.

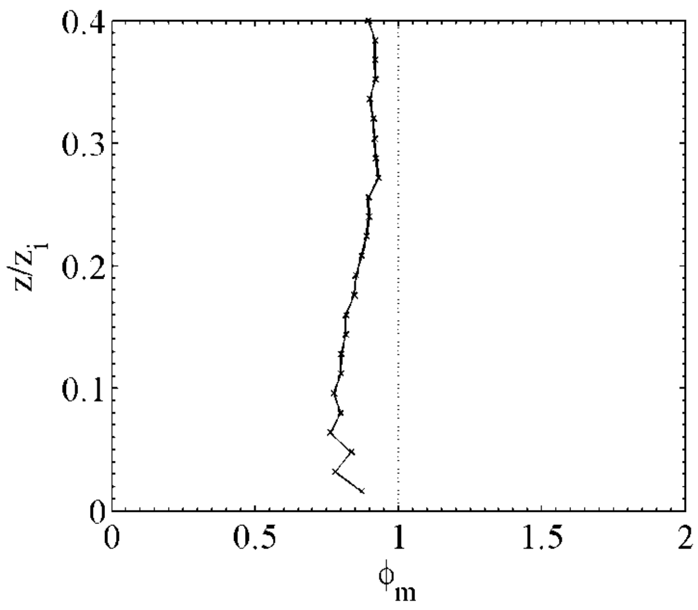
Guidelines for Use

- **Law-of-the-wall scaling**
  - This follows the work of Brasseur and Wei[1]
  - The problem:

$$\phi_m = \frac{\kappa z}{u_*}\frac{\partial \langle U \rangle}{\partial z}$$



overshoot



Improved log-law agreement

[1] J. Brasseur and T. Wei. Designing Large-Eddy Simulation of the Turbulent Boundary Layer to Capture Law-of-the-Wall Scaling, *Physics of Fluids*, Vol. 22, No. 2, 2010.

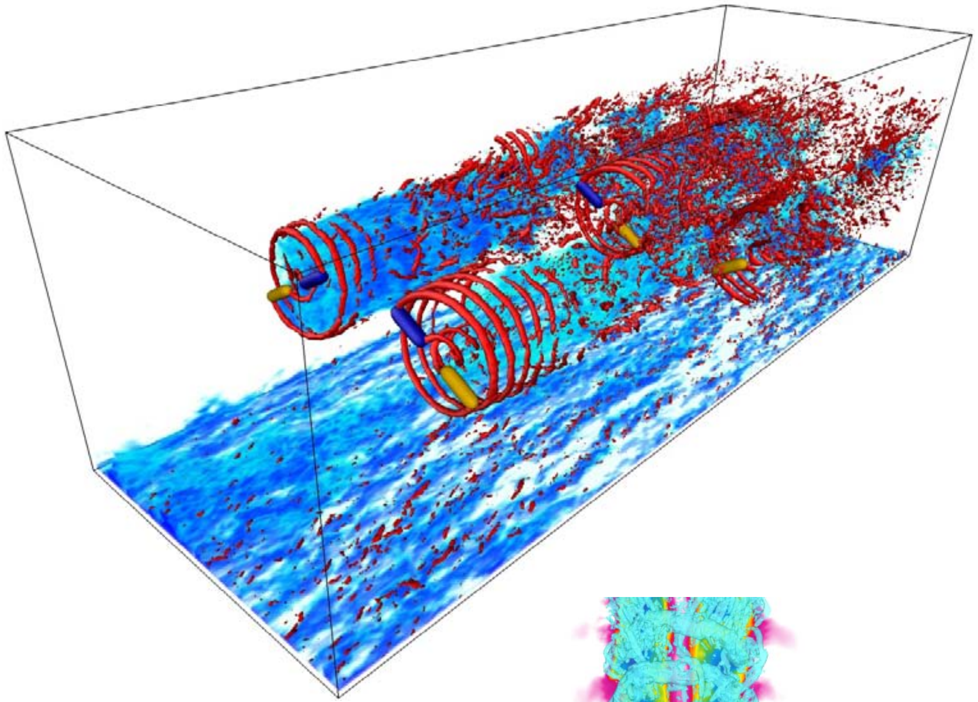
Limitations

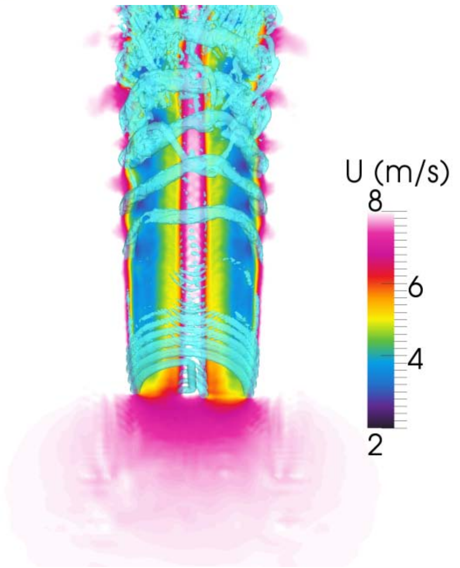
- **Think of this as a beta version**
  - We need to perform more validation, you can help
- **Neutral or unstable flow only**
  - Needs a more sophisticated SFS model to compute stable flow
  - We are working on implementing a dynamic Smagorinsky model and/or finding way back to OpenFOAM standard stress formulation to use SGS models that come with OpenFOAM
- **Only works on flat terrain**
  - Will add in irregular terrain capability later this year
- **Currently set up for homogeneous surface roughness and heating**
  - We are thinking about how to locally apply wall model
- **Not tested on truly unstructured meshes**
  - We have designed the solver with hexahedral cells of uniform height at the surface in mind

Actuator Line Turbine Model

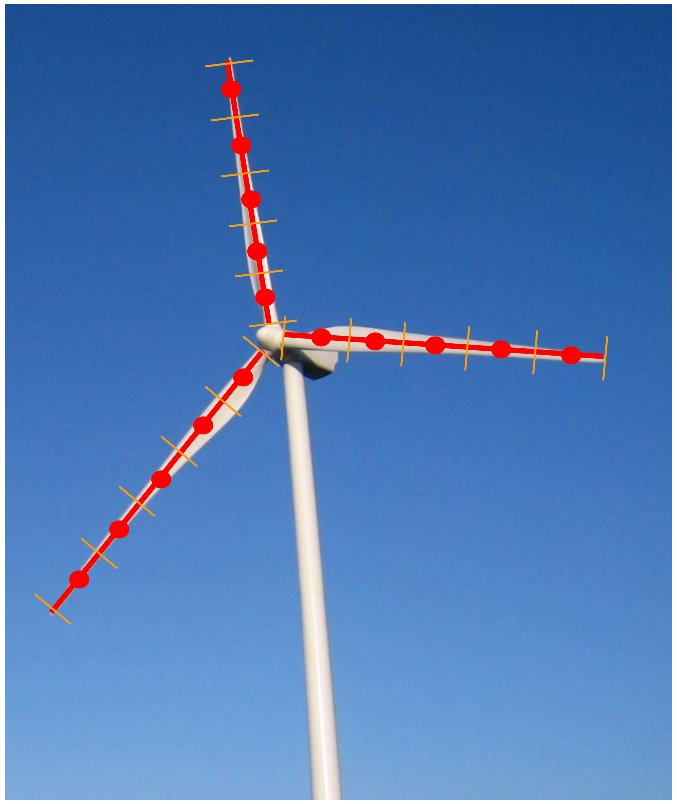
Overview

- Resolving turbine blade geometry with high-Re LES is infeasible
- An actuator approach does not require a very fine grid around turbine blades
- Creates wake, tip, root, and bound vortices
- Does not create blade boundary layer turbulence
- Depends upon airfoil look-up tables





Theory



- Method of Sørensen and Shen[1]
- Blades discretized into spanwise sections of constant airfoil, chord, twist, oncoming wind
- Airfoil lookup tables used to calculate lift and drag at each actuator section
- Force on flow is equal and opposite to blade force
- Force is normalized and projected back to flow

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{u}_j \overline{u}_i\right) = -2\varepsilon_{i3k}\Omega_3 \overline{u}_k - \frac{\partial \tilde{p}}{\partial x_i} - \frac{1}{\rho_0}\frac{\partial}{\partial x_i}\overline{p}_0(x,y) - \frac{\partial}{\partial x_j}\left(\tau_{ij}^D\right) - g\left(\frac{\overline{\theta} - \theta_0}{\theta_0}\right)\delta_{i3} + \frac{1}{\rho_0} f_i^T$$

[1] Sørensen, J. N. and Shen, W. Z., "Numerical Modeling of Wind Turbine Wakes", *Journal of Fluids Engineering* 124, 2002, pp. 393-399.

Theory

- **Force Projection**
  - How do you take force calculated at actuator line points and project it onto the CFD grid as a body force?
  - How do you smooth the force to avoid numerical oscillation?
  - Sørensen and Shen use a Gaussian projection

$$f_i^T(r) = \frac{F_i^A}{\varepsilon^3 \pi^{3/2}} \exp\left[-\left(\frac{r}{\varepsilon} \right)^2 \right]$$

  - F_i^A is the actuator element force
  - f_i^T is the force field projected as a body force onto CFD grid
  - r is distance between CFD cell center and actuator point
  - ε controls Gaussian width.

Theory

- **Projection Width**
  - Troldborg[1] recommends $\varepsilon / \Delta x = 2$ where Δx is the grid cell length near actuator line
  - We found this to be the minimum in order to maintain an oscillation-free solution using central differences
  - We think ε should be tied to some physical blade length, like chord, but have not come up with a definitive guideline.
  - See the AIAA paper by Martínez et al.[2]

[1] Troldborg, N., "Actuator Line Modeling of Wind Turbine Wakes", PhD Thesis, Technical University of Denmark, Lyngby, Denmark, 2008.
[2] Martinez, L. A., Leonardi, S., Churchfield, M. J., Moriarty, P. J., "A Comparison of Actuator Disk and Actuator Line Wind Turbine Models and Best Practices for Their Use", AIAA Paper 2012-900, Jan. 2012.

Actuator Line Model Inputs

constant/turbineArrayProperties

```
globalProperties
{
    outputControl       "timeStep";             Either “timeStep” or “runTime”
    outputInterval       1;                     Output interval in timesteps or seconds
}

turbine1                                        List turbines in the following blocks
{
    turbineType         "NREL5MWRef";           Type of turbine
    baseLocation        (1500.0 1500.0 0.0);    Location of the center of the tower base
    numBladePoints      40;                     Number of actuator elements along a blade
    pointDistType       "uniform";              Currently, elements are uniformly distributed
    pointInterpType     "linear";               Type of interpolation of velocity to actuator point, “linear” or “cellCenter”
    bladeUpdateType     "oldPosition";          Use velocity at old or new blade position
    epsilon              5.0;                   Gaussian projection width
    tipRootLossCorrType "none";                 Options are “none” or “Glauert”
    rotationDir         "cw";                   Rotor rotation sense as viewed from upstream
    Azimuth              232.0105;              Initial rotor azimuth angle
    RotSpeed             9.0;                   Initial rotor speed in RPM
    Pitch                0;                     Initial blade collective pitch
    NacYaw              225.0;                  Initial nacelle yaw position
    fluidDensity        1.23;                   Density use to compute forces, torque, and power
}

turbine2                                        List as many other turbines of any kind as you desire
{
    turbineType         “GE1.5SLE";
    …
```

Actuator Line Model Inputs

constant/turbineProperties/"turbineName"

A file is needed for each type turbine in the array

```
NumBl                    3;
TipRad                  63.0;
…
TorqueControllerType    “fiveRegion";
PitchControllerType     “none”
YawControllerType       "none";
```

Closely follows NREL FAST input file, so see FAST manual[1]

Either uses fixed speed ("none") or like NREL 5MW[2] ("fiveRegion")

```
TorqueControllerParams
{
    CutInGenSpeed            670.0;
    RatedGenSpeed           1173.7;
    Region2StartGenSpeed     871.0;
    Region2EndGenSpeed      1161.963;
    CutInGenTorque             0.0;
    RatedGenTorque            43.09355E3;
    RateLimitGenTorque        15.0E3;
    KGen                       2.55764E-2;
    TorqueControllerRelax      1.0;
}
```

Torque control parameters (controls rotor speed below Region 3)

Generator speed at cut-in wind speed (RPM)
Generator speed at rated wind speed (RPM)
Generator speed at the start of Region 2 (the end of Region 1-1/2) (RPM)
Generator speed at the end of Region 2 (the beginning of Region 2-1/2) (RPM)
Generator control torque at cut-in wind speed (N-m)
Generator control torque at rated wind speed (N-m)
Maximum allowable rate of generator control torque change (N-m/s)
Region 2 generator control constant (N-m/RPM) - torque = K*Omega^2
Relaxation factor on generator control torque update each time step.

```
PitchControllerParams
{
    PitchControlStartPitch     0.0;
    PitchControlEndPitch       7.6;
    PitchControlStartSpeed    15.77;
    PitchControlEndSpeed      16.0;
    RateLimitPitch             4.5;
}
```

Simple linear pitch control based on rotor speed (not realistic, though!) Just linearly varies pitch between two specified rotor speeds with a maximum rate of change limit.

[1] Jonkman, J. and Buhl, M., FAST User's Guide, NREL/EL-500-38230, NREL technical report, 2005. Accessible at: http://wind.nrel.gov/designcodes/simulators/fast/FAST.pdf

[2] Jonkman, J., Butterfield, S., Musial, W., and Scott, G., Definition of a 5-MW Reference Wind Turbine for Offshore System Development, NREL/TP-500-38060, Feb. 2009.

Actuator Line Model Inputs

constant/turbineProperties/”turbineName”

```
Airfoils
(
    "Cylinder1"
    "Cylinder2"
    …
    "NACA64_A17"
);

BladeData
(
//  radius(m)   c(m)     twist(deg) airfoil
    (2.8667     3.542    13.308     0)
    (5.6        3.854    13.308     0)
    …
    (58.9       2.086    0.37       7)
    (61.6333    1.419    0.106      7)
);
```

List of airfoils used to define blade

Blade properties vs. radius. Note that airfoil 0 corresponds to first airfoil in “Airfoils” list, and so on

Actuator Line Model Inputs

constant/airfoilProperties/”airfoilName”

```
airfoildata
(
//   alpha    C_l     C_d
    (-180     0       0.0185)
    (-175     0.394   0.0332)
    (-170     0.788   0.0945)
    (-160     0.67    0.2809)
    (-155     0.749   0.3932)
    (-150     0.797   0.5112)
    (-145     0.818   0.6309)
    …
    (-0.5     0.458   0.0057)
    ( 0       0.521   0.0057)
    ( 0.5     0.583   0.0057)
    ( 1       0.645   0.0058)
    ( 1.5     0.706   0.0058)
    ( 2       0.768   0.0059)
    …
    ( 170    -0.788   0.0969)
    ( 175    -0.394   0.0334)
    ( 180     0       0.0185)
);
```

An airfoil file is needed for every different airfoil used by each distinct turbine in the array

This is simply a list of coefficient of lift and drag versus angle of attack

Actuator Line Model Outputs

- **Solution files (inside time directories)**
  - bodyForce: body force projected onto flow field
- **“turbineOutput” directory**
  - Outputs various turbine information such as power, torque, rotor speed, etc.
  - Outputs information at each blade point such as angle of attack, velocity magnitude, lift, drag, etc.

Actuator Line Model Outputs

- **“turbineOutput” file structure**
  - Within turbineOutput directory are time directories corresponding to run start times. If you start a run at 0, there will be a “0” directory. If you restart a run at 1000, there will also be a “1000” directory.
    - Within the specific time directories are a files for global turbine data files for quantities like power, torque, rotor speed, etc.
    - Also there are files for blade local quantities like lift, drag, angle of attack, etc. vs. span.

Actuator Line Model Outputs

- **Global quantity file structure**

turbine^0 time^0 dt^0 value
turbine^1 time^0 dt^0 value
…
turbine^M time^1 dt^0 value

turbine^0 time^1 dt^1 value
turbine^1 time^1 dt^1 value
…
turbine^M time^1 dt^1 value

…

turbine^0 time^N dt^N value
turbine^1 time^N dt^N value
…
turbine^M time^N dt^N value

Actuator Line Model Outputs

- **Blade radius dependent file structure**

turbine^0 blade^0 time^0 dt^0 value$_0$ value$_1$ value$_2$ … value$_J$
turbine^0 blade^1 time^0 dt^0 value$_0$ value$_1$ value$_2$ … value$_J$
turbine^0 blade^2 time^0 dt^0 value$_0$ value$_1$ value$_2$ … value$_J$

turbine^1 blade^0 time^0 dt^0 value$_0$ value$_1$ value$_2$ … value$_J$
turbine^1 blade^1 time^0 dt^0 value$_0$ value$_1$ value$_2$ … value$_J$
turbine^1 blade^2 time^0 dt^0 value$_0$ value$_1$ value$_2$ … value$_J$
…

turbine^M blade^0 time^0 dt^0 value$_0$ value$_1$ value$_2$ … value$_J$
turbine^M blade^1 time^0 dt^0 value$_0$ value$_1$ value$_2$ … value$_J$
turbine^M blade^2 time^0 dt^0 value$_0$ value$_1$ value$_2$ … value$_J$

…

turbine^0 blade^0 time^N dt^N value$_0$ value$_1$ value$_2$ … value$_J$
turbine^0 blade^1 time^N dt^N value$_0$ value$_1$ value$_2$ … value$_J$
turbine^0 blade^2 time^N dt^N value$_0$ value$_1$ value$_2$ … value$_J$

turbine^1 blade^0 time^N dt^N value$_0$ value$_1$ value$_2$ … value$_J$
turbine^1 blade^1 time^N dt^N value$_0$ value$_1$ value$_2$ … value$_J$
turbine^1 blade^2 time^N dt^N value$_0$ value$_1$ value$_2$ … value$_J$
…

turbine^M blade^0 time^N dt^N value$_0$ value$_1$ value$_2$ … value$_J$
turbine^M blade^1 time^N dt^N value$_0$ value$_1$ value$_2$ … value$_J$
turbine^M blade^2 time^N dt^N value$_0$ value$_1$ value$_2$ … value$_J$

Actuator Line Model Outputs

Global turbine quantities	Description
powerRotor	Rotor power/density (W)
rotSpeed	Rotor speed (rpm)
thrust	Thrust (N)
torqueRotor	Rotor torque (N-m)
torqueGen	Generator torque (N-m)
azimuth	Rotor azimuth angle (degrees)
nacYaw	Nacelle yaw angle (degrees)
pitch	Blade collective pitch (degrees)

Actuator Line Model Outputs

Blade Local quantities	Description
alpha	Angle of attack (degrees)
axialForce	Force along rotor shaft axis (N)
Cd	Coefficient of drag
Cl	Coefficient of lift
drag	Drag force (N)
lift	Lift force (N)
tangentialForce	Force in rotor rotation tangential direction (N)
Vaxial	Component of velocity along rotor shaft axis (m/s)
Vradial	Component of velocity along blade radius (m/s)
Vtangential	Component of velocity in rotation tangential direction (m/s)
x, y, z	Actuator point position in space (m)

Guidelines for Use

- **$+x$ must be east, $+y$ must be north, $+z$ must be up**
- **Use at least 20 CFD grid cells across the rotor diameter**
- **Use at least 40 CFD grid cells across the rotor if you want to well resolve tip/root vortices**
- **We are currently performing a study to better understand power production dependence on surrounding grid resolution, epsilon, number of actuator points, and use of the tip loss correction**
- **Set epsilon parameter to at least twice the local grid cell length, but somewhere around the mean blade chord length**

Implementation

- **Turbine model implemented as a class**
  - "horizontalAxisWindTurbinesALM"
  - See src/turbineModels/horizontalAxisWindTuribinesALM
- **Any solver can be modified to contain an object of the class**
- **That object is the entire turbine array**

Implementation

- **Modifying pisoFoam to include turbine class**
  - Add this to createFields.H to declare object of turbine class

```
// Create an object of the horizontalWindTurbineArray class if there
// is to be a turbine array
//
turbineModels::horizontalAxisWindTurbinesALM turbines(U);
```

  - Add this to the includes part of the solver code

```
#include “horizontalAxisWindTurbinesALM.H”
```

  - Add this line to solver code momentum equation to apply forces

```
fvVectorMatrix UEqn
(
       fvm::ddt(U)
     + fvm::div(phi, U)
     + turbulence->divDevReff(U)
     - turbines.force()
```

  - Add this line at the beginning or end of the time loop to advance the turbine one time step

```
turbines.update();
```

Implementation

- **Make/options file needs to be modified**

```
 EXE_INC = \
    -I$(LIB_SRC)/turbulenceModels/incompressible/turbulenceModel \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(WM_PROJECT_USER_DIR)/src/turbineModels/lnInclude

EXE_LIBS
    -L$(FOAM_USER_LIBBIN) \
    -lincompressibleTurbulenceModel \
    -lincompressibleRASModels \
    -lincompressibleLESModels \
    -lincompressibleTransportModels \
    -lfiniteVolume \
    -lmeshTools \
    -llduSolvers \
    -luserTurbineModels
```

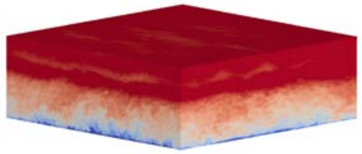
FAST Coupling to OpenFOAM

Coupling FAST to OpenFOAM

- **NREL's FAST[1] (Fatigue, Aerodynamics, Stress, and Turbulence) tool is a model for wind turbine structural, aero, and system dynamics**
- **Its aerodynamics part is through blade element momentum theory (BEM)**
- **Here, we coupled FAST to the actuator line model**
- **The "momentum" part of BEM is replaced by CFD**
  - CFD feeds FAST inflow information at blade elements
  - Aerodynamic forces computed by look-up table ("blade element" theory--just like normal actuator line)
  - Turbine structural and system response computed
  - Aerodynamic forces fed back to CFD

[1] Jonkman, J. and Buhl, M., FAST User's Guide, NREL/EL-500-38230, NREL technical report, 2005. Accessible at: http://wind.nrel.gov/designcodes/simulators/fast/FAST.pdf

Coupling FAST to OpenFOAM



OpenFOAM

```
Do while (t < tmax)
    call FLOW_Solver
    call openFOAM2FAST
    call FAST
    call Fast2OpenFOAM
End do
```

Turbulence is different than a TurbSim result!

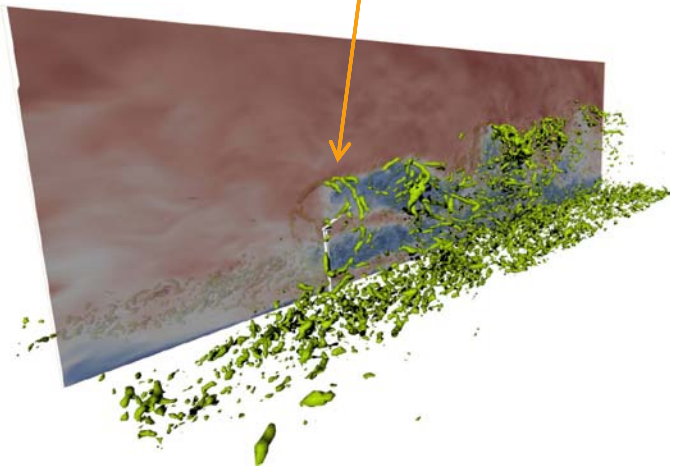
Multiple-Turbine capability

(NREL aero-elastic code)

velocity →

Compute structural response and blade rotation

← aeroforces w/ blade coord. in actuator line representation



Implementation

- **Similar to standard actuator line**
- **Turbine model implemented as a class**
  - “horizontalAxisWindTurbinesFAST”
  - See src/fastturb/horizontalAxisWindTuribinesFAST
- **Any solver can be modified to contain an object of the class**
- **That object is the entire turbine array**

Implementation - fastPisoSolver

```
…

label pRefCell = 0;
scalar pRefValue = 0.0;
setRefCell(p, mesh.solutionDict().subDict("PISO"), pRefCell, pRefValue);

singlePhaseTransportModel laminarTransport(U, phi);

autoPtr<incompressible::turbulenceModel> turbulence
(
    incompressible::turbulenceModel::New(U, phi, laminarTransport)
);

turbineModels::horizontalAxisWindTurbinesFAST turbfast(U);
```

createFields.H

-Create an object of the horizontalWindTurbinesFAST class if there is to be a turbine array

•Add “createFields.H” file to the includes part of the solver code (pisoFoam.C)

```
int main(int argc, char *argv[])
{
   #include "setRootCase.H"
   #include "createTime.H"
   #include "createMesh.H"
   #include "createFields.H"
   #include "initContinuityErrs.H”
```

Implementation - fastPisoSolver

pisoFoam.C

```
…
#include "horizontalAxisWindTurbinesFAST.H“
…

extern "C"
{
 void fastinit_( float& , int& );
 void fastread_( float*, float*, float*);
 void fastrun_( );
 void fastgetbldpos_( float*, float*, float*);
 void fastgetbldforce_(float*, float*, float*);
 void fastend_( );
}
```

Declare wrapper functions written Fortran90

-Initialize FAST
-Read wind information from OpenFOAM
-Run FAST
-transfer updated blade element positions to OpenFOAM
-transfer updated aerodynamic forces from blade elements to OpenFOAM
-Terminate FAST

```
int main(int argc, char *argv[])
{
    … #include "createFields.H“ …

   // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
   // initialize FAST
   Info << "Number of Turbs: " << turbfast.turbNum << endl;
   float tstep = runTime.deltaT().value();
   for(int turbNo=0; turbNo<turbfast.turbNum; turbNo++)
   {
    if(Pstream::myProcNo() == turbNo)
    {
      fastinit_(tstep, turbNo);
      fastgetbldpos_(turbfast.bldptx[turbNo], turbfast.bldpty[turbNo], turbfast.bldptz[turbNo]);
    }
    turbfast.getBldPos(turbNo);
   }
…
```

FAST initialization
-Get number of blades
-Get time-step from OpenFOAM => FAST time step
-Loop through each turbines
-Turbine ID = MPI_RANK (CPU #)

-For given CPU #, initialize FAST

-Get current blade elem. pos.

-Transfer blade elem. Pos. to OpenFOAM

Implementation - fastPisoSolver.C

pisoFoam.C

Continued from last slide…

```
// Pressure-velocity PISO corrector
{

 for(int turbNo=0; turbNo<turbfast.turbNum; turbNo++)
 {
  turbfast.getWndVec(turbNo);
  if(Pstream::myProcNo() == turbNo)
  {
   fastread_(turbfast.uin[turbNo], turbfast.vin[turbNo], turbfast.win[turbNo]);
   fastrun_();
   fastgetbldpos_(turbfast.bldptx[turbNo], turbfast.bldpty[turbNo], turbfast.bldptz[turbNo]);
   fastgetbldforce_(turbfast.bldfx[turbNo], turbfast.bldfy[turbNo], turbfast.bldfz[turbNo]);
  }
  turbfast.computeBodyForce(turbNo);
 }

 // Momentum predictor
 fvVectorMatrix UEqn
 (
  fvm::ddt(U)
    + fvm::div(phi, U)
    + turbulence->divDevReff(U) - turbfast.force()
 );

…
 fastend_();
…
```

-Loop through turbines

-get wind data for specified turbine

-transfer OpenFOAM wind data to FAST
-run FAST
-pass updated blade elem. pos. to OpenFOAM
-pass updated aerodynmic force to OpenFOAM

-project the aerodynamic force into the OpenFOAM computational domain

-added the aerodynamic force from FAST as a bodyforce term in momentum eq.

-terminate FAST (loops through all the turbines)

Implementation – Make file

- **Make/options file needs to be modified**

```
 EXE_INC = \
    -I$(LIB_SRC)/turbulenceModels/incompressible/turbulenceModel \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(WM_PROJECT_USER_DIR)/src/fastturb/lnInclude

EXE_LIBS
    -Lfast/bin \
    -L$(FOAM_USER_LIBBIN) \
    -lincompressibleRASModels \
    -lincompressibleLESModels \
    -lincompressibleTransportModels \
    -lfiniteVolume \
    -lmeshTools \
    -luserfastturb \
    -lgfortran \
    -lfast
```

```
../fastPisoSolver/
   /fast/bin/
        libfast.a
   /Make/
        files
        options
  createFields.H
  pisoFoam.C
```

FAST compiled into static library

FAST Input files: NREL 5MW Turbine

/caseStudyDir/

Required files are:

Primary.fst
specifies configurations for initial conditions, controls, turbine geometry and mass, drive train, output file formats, etc…

USERWIND.wnd
file used to invoke reading in external flow data

NRELOffshrBsline5MW_AeroDyn.ipt
AeroDyn input for air specification, blade geometry, airfoil data (coefficients for lift/drag table are included in /caseStudyDir/AeroData/)

NRELOffshrBsline5MW_Blade.ipt
Specifies blade properties: stiffness, mode shapes etc..

NRELOffshrBsline5MW_Tower_Onshore.ipt
ditto for Tower properties

FAST Actuator Line Model Inputs

constant/**turbineArrayPropertiesFAST**

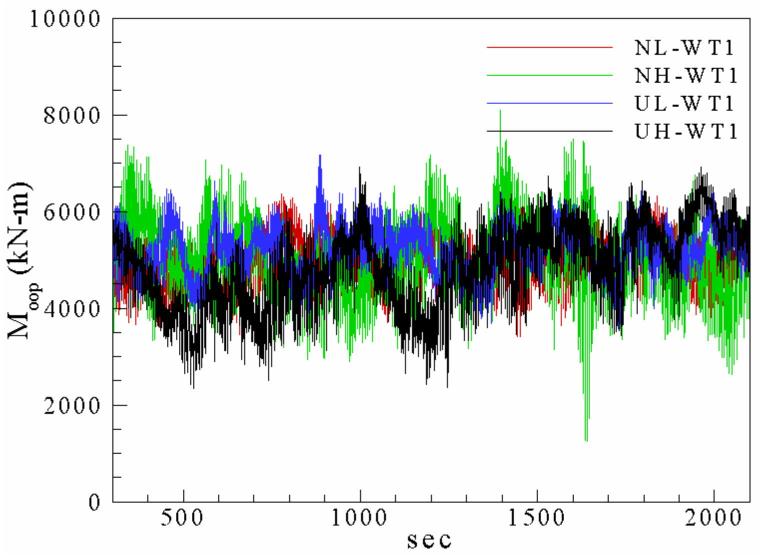
```
turbine0
{
    refx                200.0;        - x location of tower base
    refy                  0.0;        - y location of tower base
    refz                  0.0;        - z location of tower base
    hubz                100.0;        - hub height
}

turbine1
{
    refx                400.0;
    …


general
{
    yawAngle                 0.0;     - turbine yaw angle
    numberofBld              3        - # of blades
    numberofBldPts          62;       - # of actuator elements per blade
    rotorDiameter          126.3992;  - rotor diameter
    epsilon                  5.0;     - Gaussian width parameter
    smearRadius             13.15;    - radius beyond which Gaussian has no effect
    effectiveRadiusFactor    1.21;    - scale factor for rotor diameter
    pointInterpType          1;       - option for linear interpolation of velocities
}
```

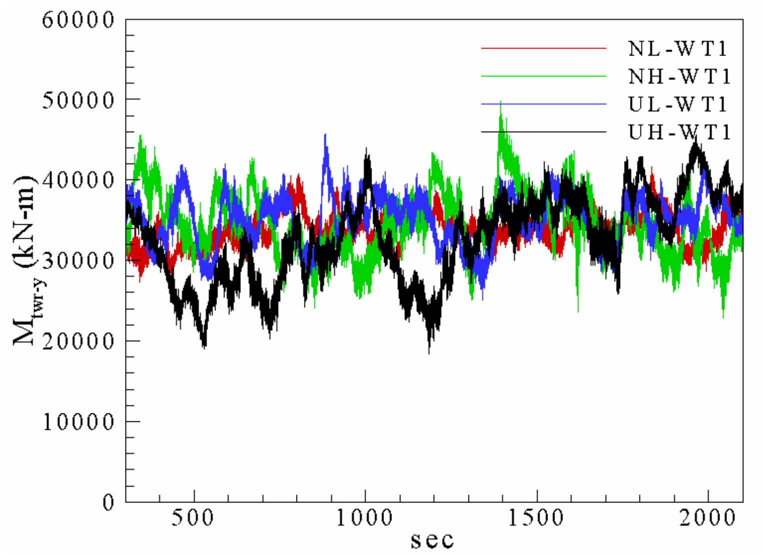
FAST Actuator Line Model Outputs

- **Load files : primary0.out, primary1.out, …**
- **These include time histories of load parameters specified in primary.fst**

  e.g. out-of-plane blade root bending moments, torque, yaw bearing moments, power, rotor speed …

- **Can be imported into Excel / MatLab for figures**

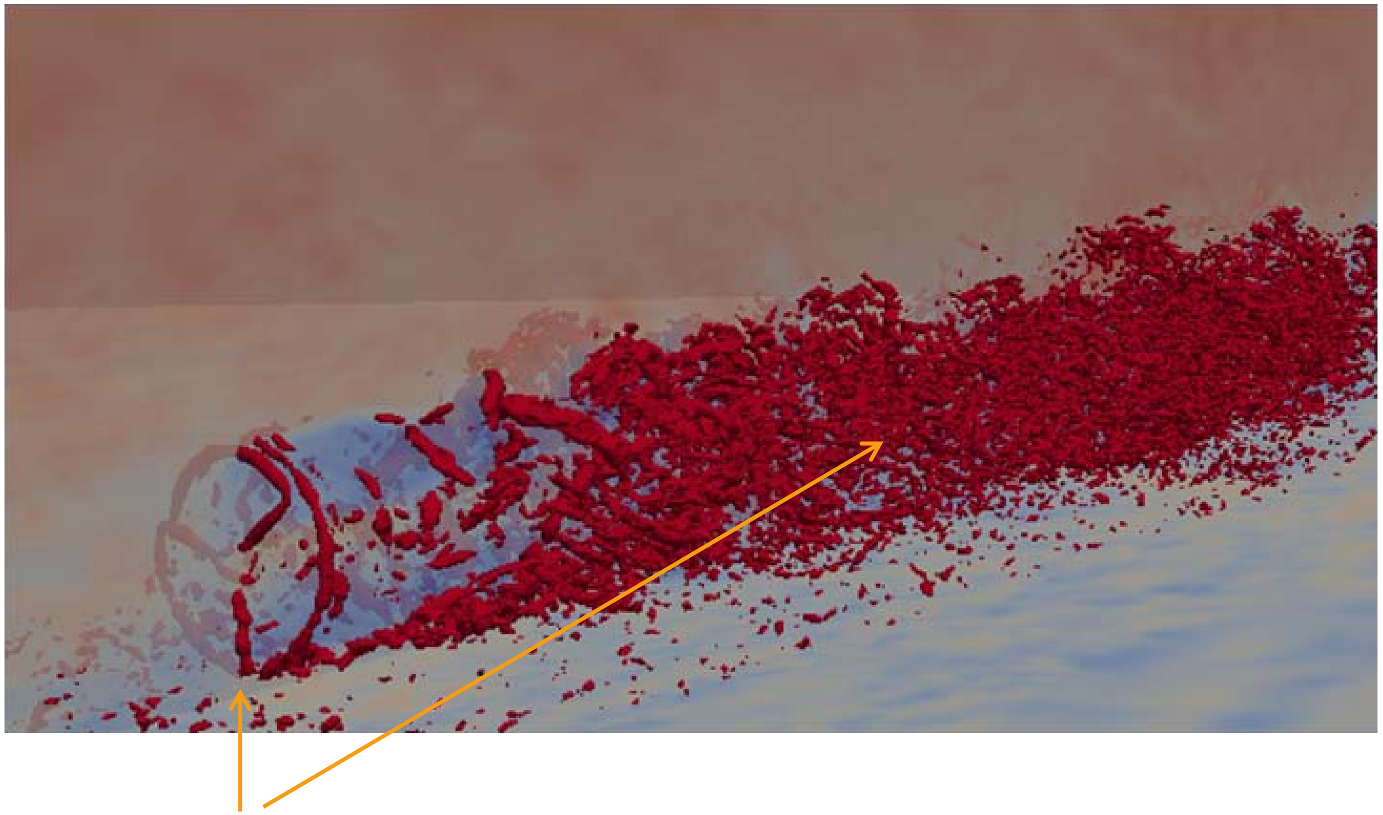




Guidelines for Use

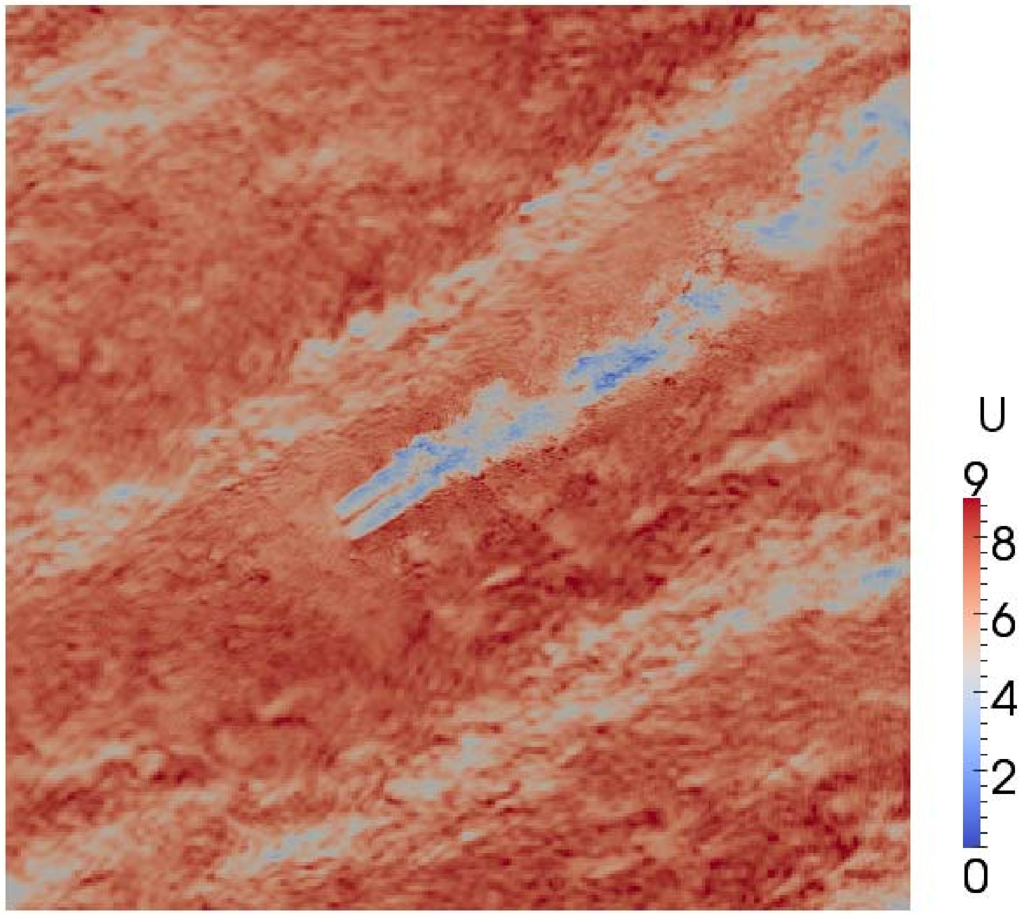
- **See actuator line guidelines**

Sample Output



Two NREL 5-MW turbines subjected to neutrally stable low-roughness atmospheric conditions showing the instantaneous streamwise velocity contours with iso-surface of Q invariant fixed at 0.0275 1/s

Sample Output



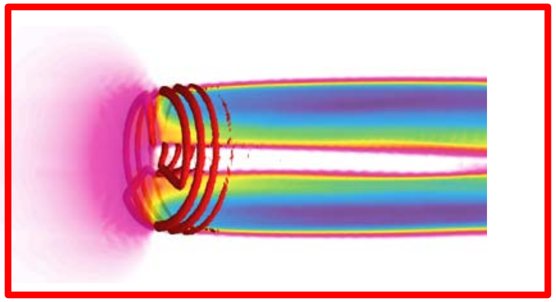
NREL 5MW turbine in unstable high-roughness atmospheric flow with mean speed at 8 m/s @ hub height

Wind Plant Simulation

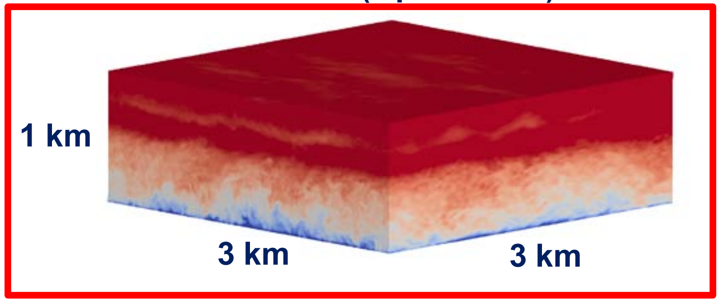
Wind Plant Simulation

- Combination of the elements discussed above

Actuator line turbine aerodynamics models (coupled with NREL’s FAST turbine dynamics model)

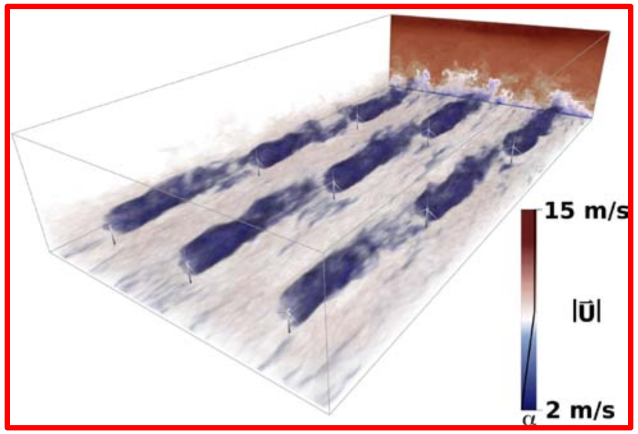


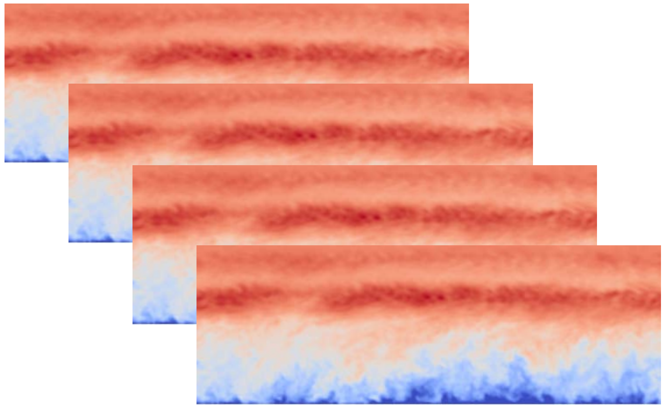
“Precursor” atmospheric simulation (OpenFOAM)



Initialize wind farm domain with precursor volume field

Wind farm simulation (OpenFOAM)





Use saved precursor data as inflow boundary conditions

Save planes of data every N time steps

Input

- **Nearly the same as atmospheric boundary layer solver**
- **Difference in constant/ABLProperties**

```
// Is the turbine array active?
turbineArrayOn             true;

…
```

Activate the actuator turbine models

```
// Mean field averaging start time.
meanAvgStartTime          12100.0;

// Correlation field averaging start time.
corrAvgStartTime          12200.0;
```

We no longer take horizontal averages resulting in a mean profile. We take time averages. The mean is built up on the fly starting at meanAvgStartTime, and fluctuations away from that mean are taken to build correlations starting at corrAvgStartTime. Allow enough time for transients to pass before starting to build up mean, and allow enough time for mean to be built before building up correlations.

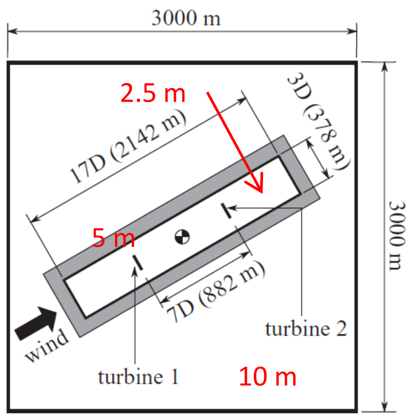
Output

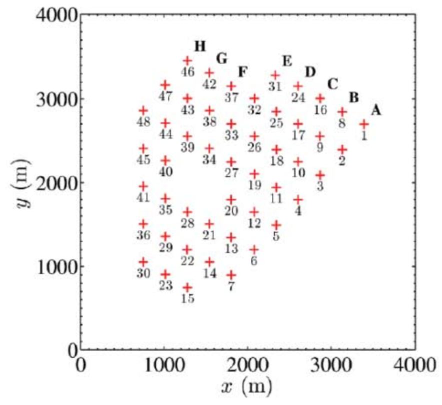
- **All the turbine information**
- **Instantaneous Fields**
  - U, T, p, u', T'
- **Mean Fields**
  - Umean, Tmean
- **Correlation Fields**
  - $\langle u_i' u_j' \rangle$, $\langle T' u_j' \rangle$

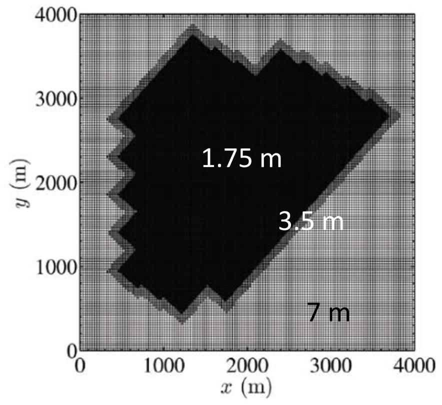
Guidelines for Use

- **Make sure domain boundaries have either predominant inflow or outflow**
  - Remember that with Coriolis, wind changes directions with altitude
  - Possible to have wind flowing in near ground and flowing out above
  - We do not have a good boundary condition for that case

- **Use local mesh refinement around the turbines**
  - but do it gently (i.e. give the turbulence time to cascade down before going to the next local refinement region)





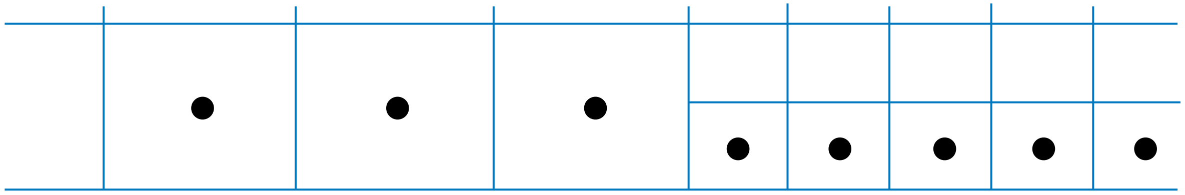


Guidelines for Use

- We generally use a time step such that the actuator line tip does not travel through more than one cell per time step
- Can use larger time steps with actuator disk (which will be part of SOWFA soon).

Limitations/Known Issues

- **The wall shear stress model in the atmospheric solver is based on horizontal averages in the first layer of cells away from the surface**
  - Horizontal averages do not make sense in the wind farm
  - Local refinement means that the first layer of grid cells are not all at the same height



- **What are the proper pressure boundary conditions with inflow/outflow?**
  - We use Neumann and retain the driving pressure gradient term in the equations, setting driving pressure to the average value from the precursor

Compiling The Codes

Compiling the codes

- Make sure you have OpenFOAM 2.0 or higher installed
- Download the SOWFA codes at: http://wind.nrel.gov/designcodes/simulators/sowfa/
- In your user OpenFOAM directory, put “user-2.0.x.tar.gz” and do “tar -xvzf user-2.0.x.tar.gz”. Rename the “user” part to your username, and rename the “2.0.x” to the version of OpenFOAM you have installed.
- Run ./Allwclean
- Run ./Allwmake
- See the README files

Example Cases:

Precursor Atmospheric Boundary Layer Simulation

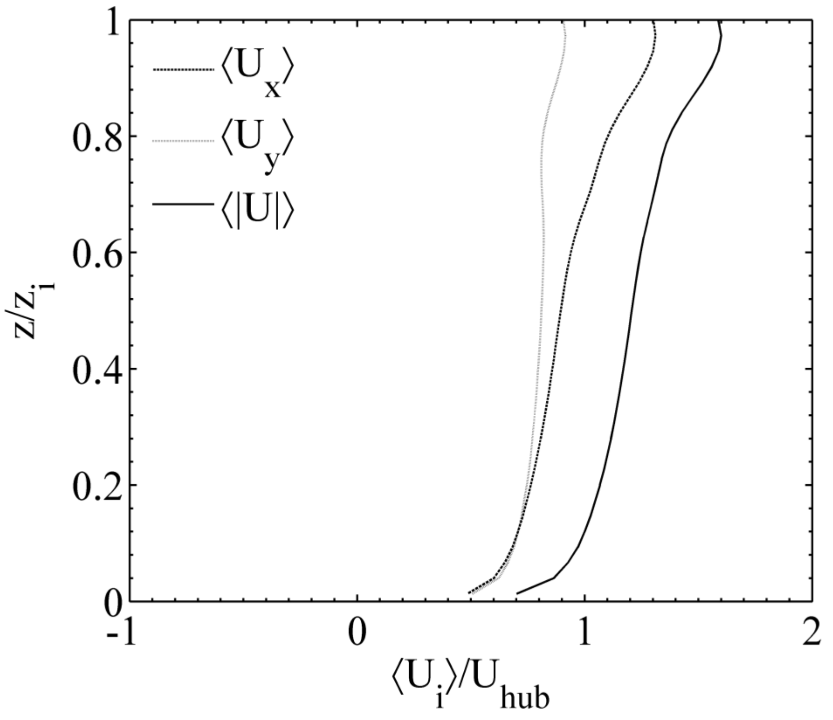
Atmospheric Boundary Layer

- **See “tutorials/precursorABL”**
- **Uses the solver ABLPisoSolver**
- **2 cases: Neutral and unstable ($-z_i/L \approx 4$)**
- **Wind: 9 m/s from 225 deg at 90 m**
- **Domain size: 3km × 3km × 1 km (x × y × z)**
  - Periodic in the horizontal
- **Grid size: 150 × 150 × 50**
  - 20 m resolution throughout
  - Coarser than we would normally run a simulation
- **Run on 32 processors**
  - Took about 27 min of wall clock time per 1000 s of simulation
  - Ran to 14,000 s of simulation time

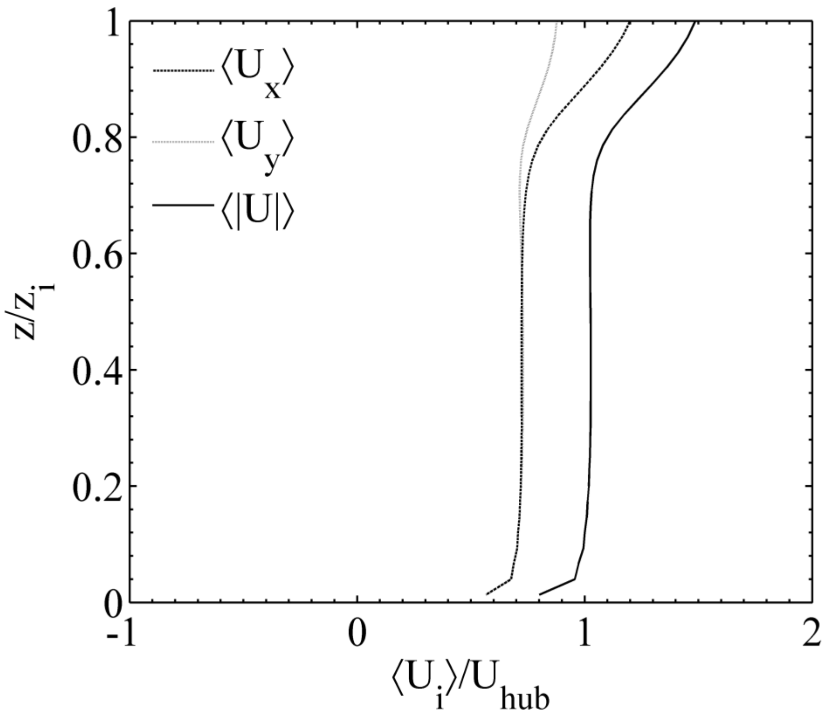
The Process (see the “Allrun” script)

- **Build a coarse mesh with blockMesh (serial)**
  - Builds a hexahedral mesh
- **Decompose the domain with decomposePar (serial)**
- **Use refineHexMesh (parallel) to globally refine mesh to desired resolution**
  - Splits hexahedral cells in half in each direction
- **Initialize the solution with setFieldsABL (parallel)**
- **Run the solver from time 0 to quasi-equilibrium**
- **Run the solver from quasi-equilibrium to +2000 s**
  - Run with sampling of contour planes and boundary data (boundary data to be used later in wind plant simulation as turbulent inflow)

Results

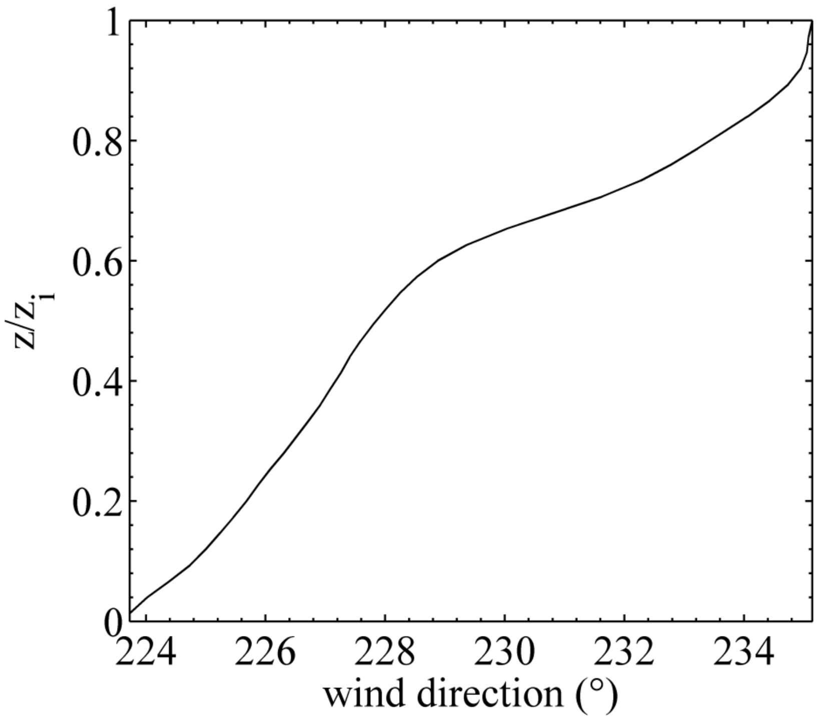


neutral

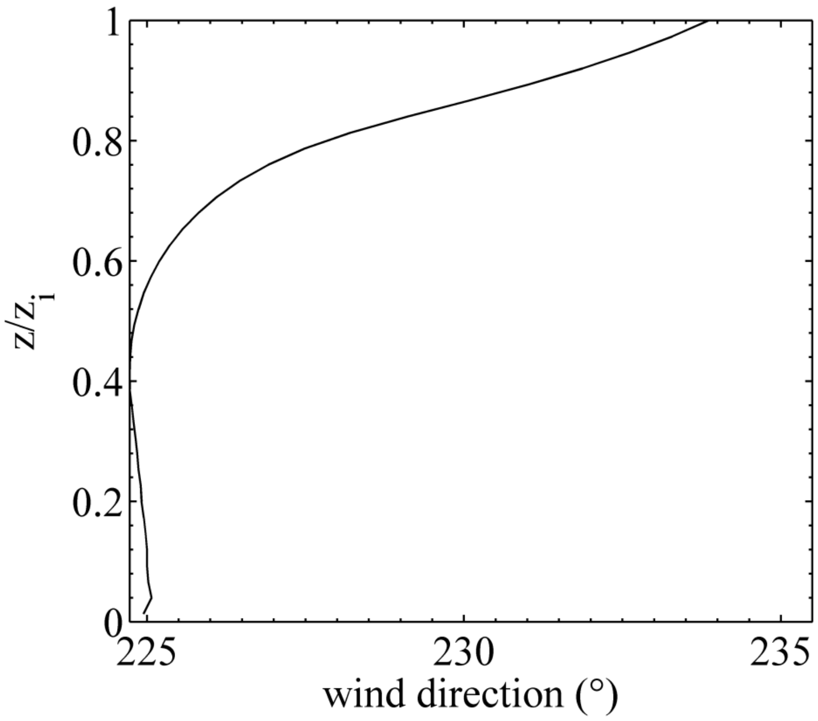


unstable

Results

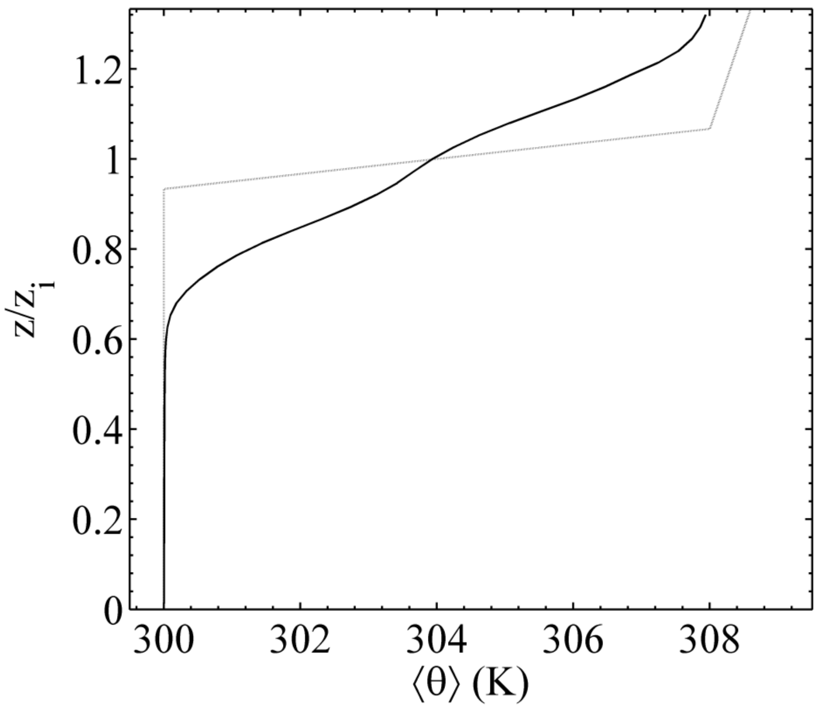


neutral

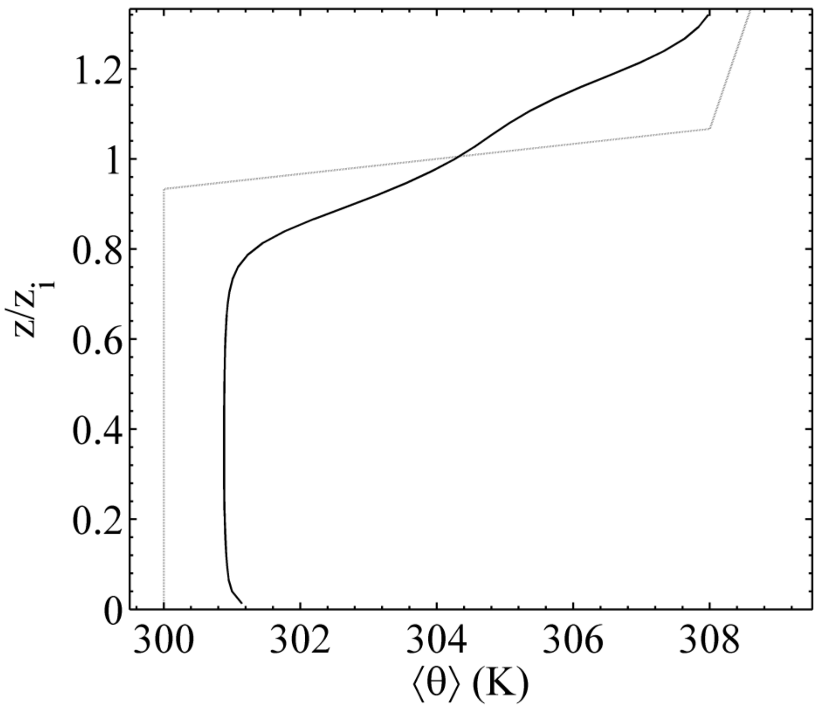


unstable

Results

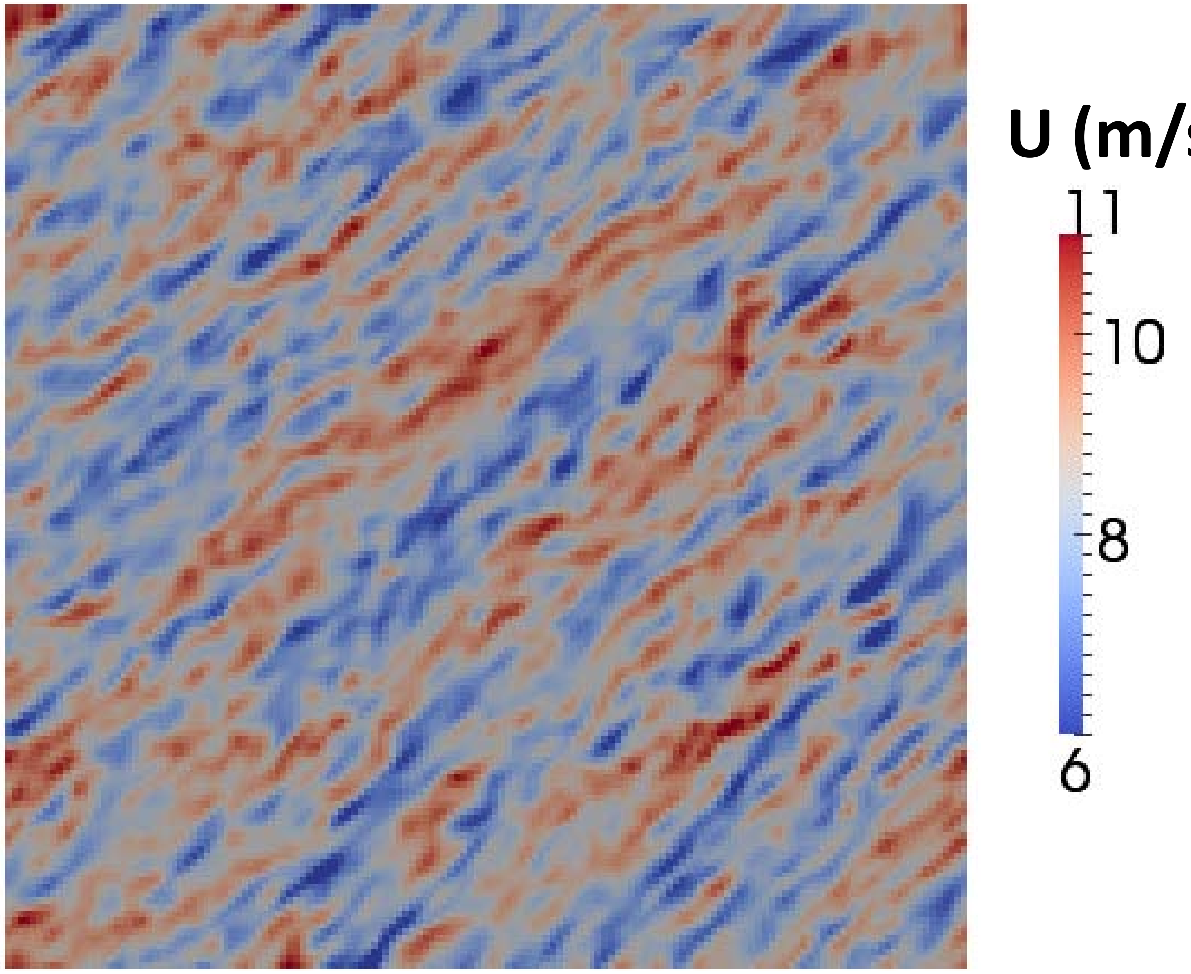


neutral



unstable

Results

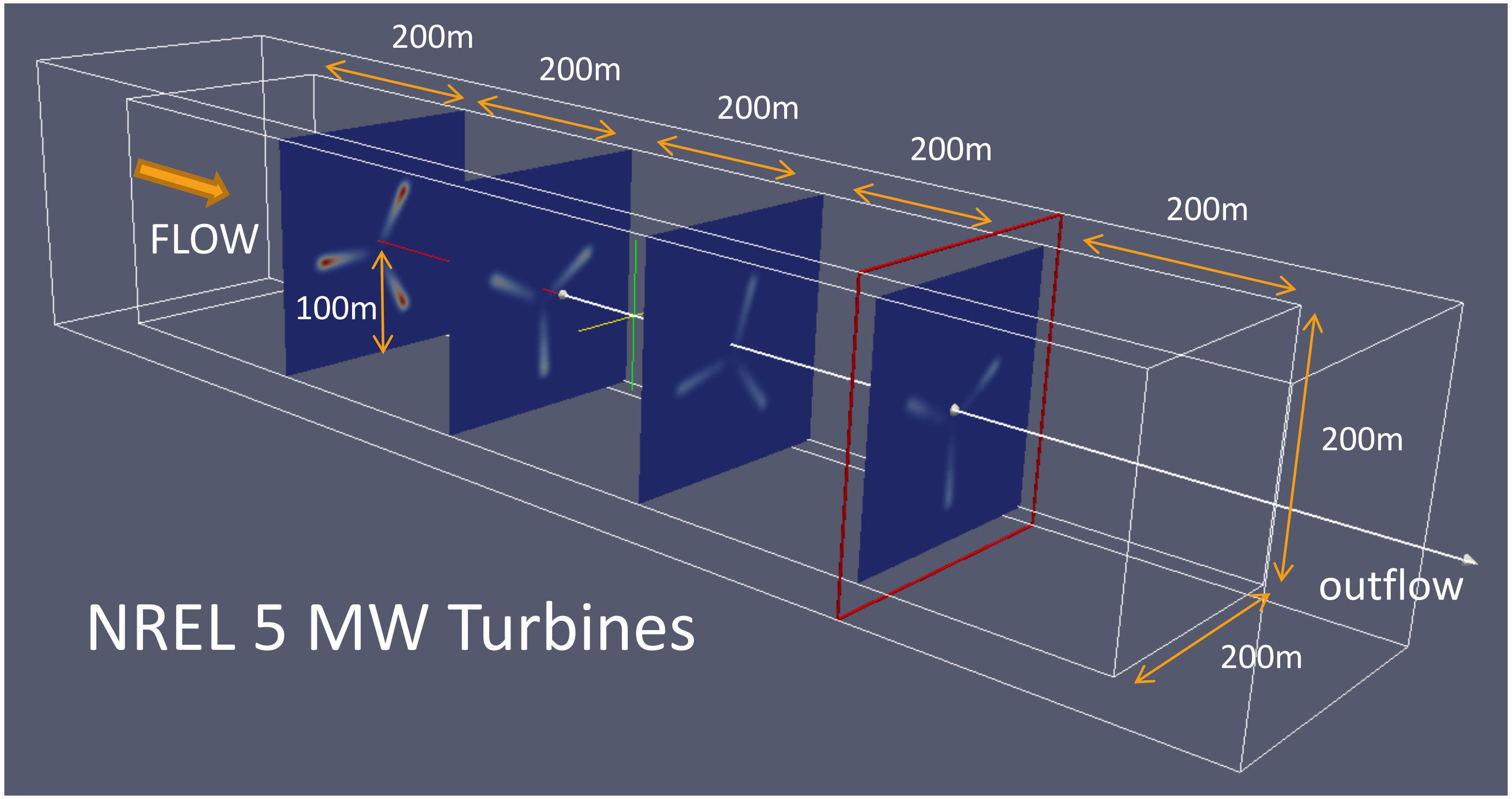


Example Cases:
FAST-Couple Actuator Lines in Duct Flow

Case Study: fastDuct

../tutorials/fastDuct/

Computational Domain



Uniform inflow condition at U_∞ = 8 m/s
Periodic BCs laterally (y and z directions)

Sample Run

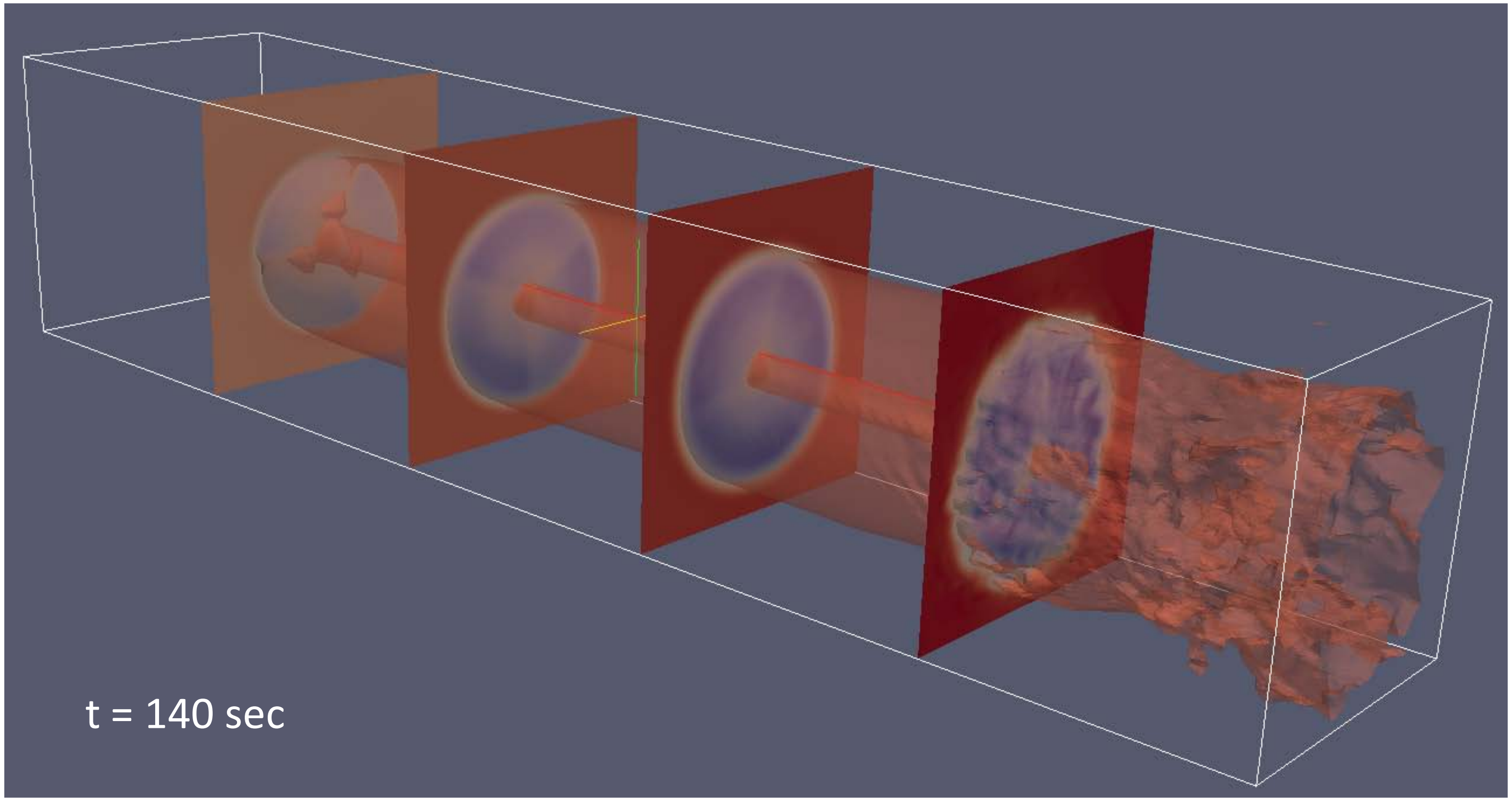
- In ../fastDuct/   execute “Allrun” script
  - currently set to run on a single node with 8 CPU cores
  - generates uniform mesh
  - decomposes the domain into nodes x cores
  - runs fastPisoSolver in parallel

- Once finished running:
  - execute “reconstructPar –time 140 (any desired saved time)
  - execute “foamToVTK –time 140
  - use ParaView for visualization
  - examine loads data from primary*.out using Excel/MatLab

- Run “AllClean” to remove saved flow data, loads, and the grid

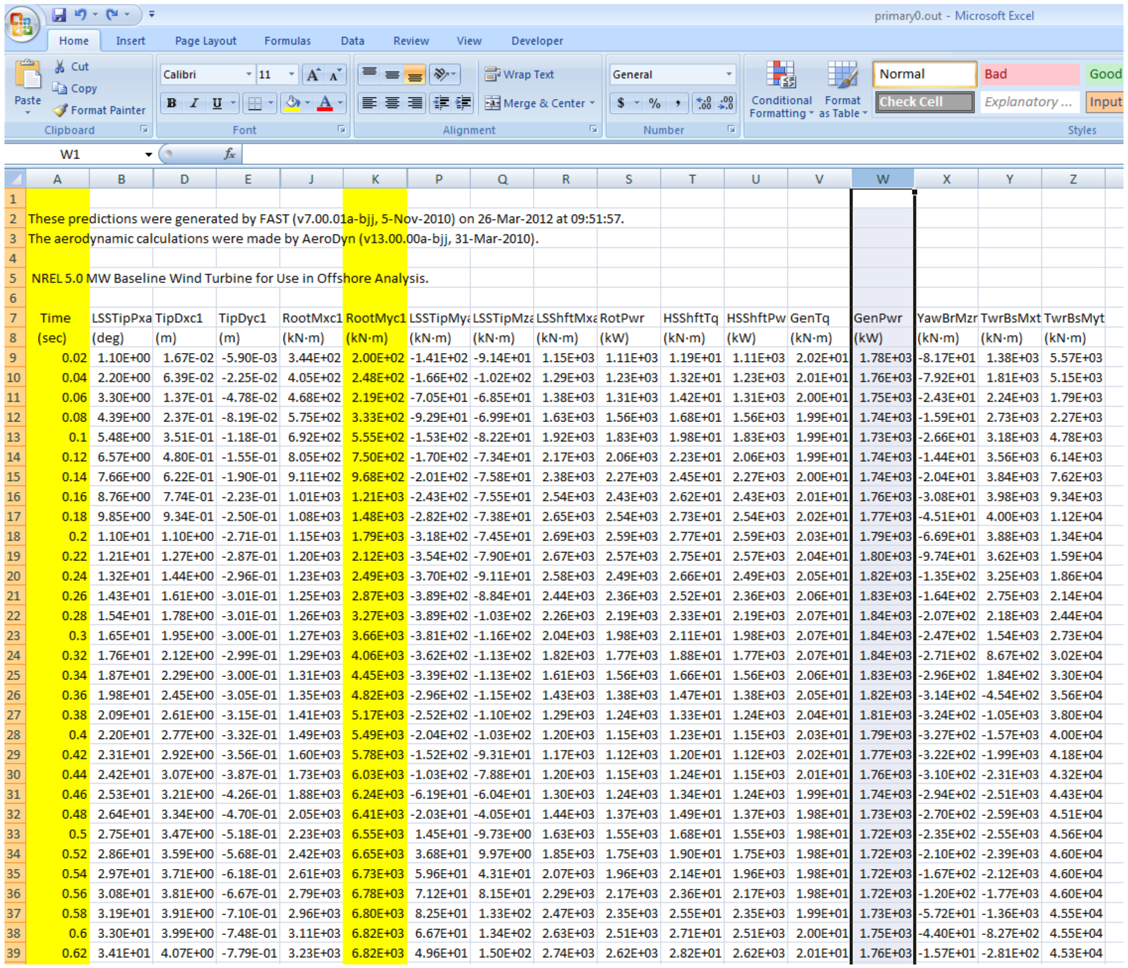
Sample Run

Streamwise Velocity Contours and iso-surface



Downstream turbine is being approached with wake structures

Out-of-plane Blade Loadings and Power Output from FAST


primary0.out - Microsoft Excel

These predictions were generated by FAST (v7.00.01a-bjj, 5-Nov-2010) on 26-Mar-2012 at 09:51:57.
The aerodynamic calculations were made by AeroDyn (v13.00.00a-bjj, 31-Mar-2010).

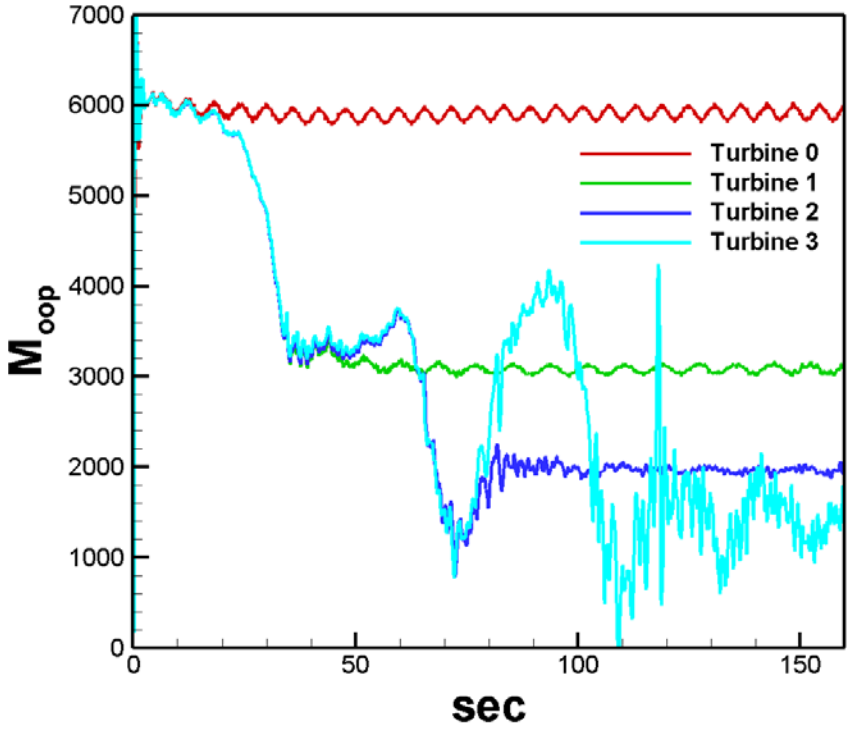
NREL 5.0 MW Baseline Wind Turbine for Use in Offshore Analysis.

Time	LSSTipPxa	TipDxc1	TipDyc1	RootMxc1	RootMyc1	LSSTipMya	LSSTipMza	LSShftMxa	RotPwr	HSShftTq	HSShftPw	GenTq	GenPwr	YawBrMzr	TwrBsMxt	TwrBsMyt
(sec)	(deg)	(m)	(m)	(kN·m)	(kN·m)	(kN·m)	(kN·m)	(kN·m)	(kW)	(kN·m)	(kW)	(kN·m)	(kW)	(kN·m)	(kN·m)	(kN·m)
0.02	1.10E+00	1.67E-02	-5.90E-03	3.44E+02	2.00E+02	-1.41E+02	-9.14E+01	1.15E+03	1.11E+03	1.19E+01	1.11E+03	2.02E+01	1.78E+03	-8.17E+01	1.38E+03	5.57E+03
0.04	2.20E+00	6.39E-02	-2.25E-02	4.05E+02	2.48E+02	-1.66E+02	-1.02E+02	1.29E+03	1.23E+03	1.32E+01	1.23E+03	2.01E+01	1.76E+03	-7.92E+01	1.81E+03	5.15E+03
0.06	3.30E+00	1.37E-01	-4.78E-02	4.68E+02	2.19E+02	-7.05E+01	-6.85E+01	1.38E+03	1.31E+03	1.42E+01	1.31E+03	2.00E+01	1.75E+03	-2.43E+01	2.24E+03	1.79E+03
0.08	4.39E+00	2.37E-01	-8.19E-02	5.75E+02	3.33E+02	-9.29E+01	-6.99E+01	1.63E+03	1.56E+03	1.68E+01	1.56E+03	1.99E+01	1.74E+03	-1.59E+01	2.73E+03	2.27E+03
0.1	5.48E+00	3.51E-01	-1.18E-01	6.92E+02	5.55E+02	-1.53E+02	-8.22E+01	1.92E+03	1.83E+03	1.98E+01	1.83E+03	1.99E+01	1.73E+03	-2.66E+01	3.18E+03	4.78E+03
0.12	6.57E+00	4.80E-01	-1.55E-01	8.05E+02	7.50E+02	-1.70E+02	-7.34E+01	2.17E+03	2.06E+03	2.23E+01	2.06E+03	1.99E+01	1.74E+03	-1.44E+01	3.56E+03	6.14E+03
0.14	7.66E+00	6.22E-01	-1.90E-01	9.11E+02	9.68E+02	-2.01E+02	-7.58E+01	2.38E+03	2.27E+03	2.45E+01	2.27E+03	2.00E+01	1.74E+03	-2.04E+01	3.84E+03	7.62E+03
0.16	8.76E+00	7.74E-01	-2.23E-01	1.01E+03	1.21E+03	-2.43E+02	-7.55E+01	2.54E+03	2.43E+03	2.62E+01	2.43E+03	2.01E+01	1.76E+03	-3.08E+01	3.98E+03	9.34E+03
0.18	9.85E+00	9.34E-01	-2.50E-01	1.08E+03	1.48E+03	-2.82E+02	-7.38E+01	2.65E+03	2.54E+03	2.73E+01	2.54E+03	2.02E+01	1.77E+03	-4.51E+01	4.00E+03	1.12E+04
0.2	1.10E+01	1.10E+00	-2.71E-01	1.15E+03	1.79E+03	-3.18E+02	-7.45E+01	2.69E+03	2.59E+03	2.77E+01	2.59E+03	2.03E+01	1.79E+03	-6.69E+01	3.88E+03	1.34E+04
0.22	1.21E+01	1.27E+00	-2.87E-01	1.20E+03	2.12E+03	-3.54E+02	-7.90E+01	2.67E+03	2.57E+03	2.75E+01	2.57E+03	2.04E+01	1.80E+03	-9.74E+01	3.62E+03	1.59E+04
0.24	1.32E+01	1.44E+00	-2.96E-01	1.23E+03	2.49E+03	-3.70E+02	-9.11E+01	2.58E+03	2.49E+03	2.66E+01	2.49E+03	2.05E+01	1.82E+03	-1.35E+02	3.25E+03	1.86E+04
0.26	1.43E+01	1.61E+00	-3.01E-01	1.25E+03	2.87E+03	-3.89E+02	-8.84E+01	2.44E+03	2.36E+03	2.52E+01	2.36E+03	2.06E+01	1.83E+03	-1.64E+02	2.75E+03	2.14E+04
0.28	1.54E+01	1.78E+00	-3.01E-01	1.26E+03	3.27E+03	-3.89E+02	-1.03E+02	2.26E+03	2.19E+03	2.33E+01	2.19E+03	2.07E+01	1.84E+03	-2.07E+02	2.18E+03	2.44E+04
0.3	1.65E+01	1.95E+00	-3.00E-01	1.27E+03	3.66E+03	-3.81E+02	-1.16E+02	2.04E+03	1.98E+03	2.11E+01	1.98E+03	2.07E+01	1.84E+03	-2.47E+02	1.54E+03	2.73E+04
0.32	1.76E+01	2.12E+00	-2.99E-01	1.29E+03	4.06E+03	-3.62E+02	-1.13E+02	1.82E+03	1.77E+03	1.88E+01	1.77E+03	2.07E+01	1.84E+03	-2.71E+02	8.67E+02	3.02E+04
0.34	1.87E+01	2.29E+00	-3.00E-01	1.31E+03	4.45E+03	-3.39E+02	-1.13E+02	1.61E+03	1.56E+03	1.66E+01	1.56E+03	2.06E+01	1.83E+03	-2.96E+02	1.84E+02	3.30E+04
0.36	1.98E+01	2.45E+00	-3.05E-01	1.35E+03	4.82E+03	-2.96E+02	-1.15E+02	1.43E+03	1.38E+03	1.47E+01	1.38E+03	2.05E+01	1.82E+03	-3.14E+02	-4.54E+02	3.56E+04
0.38	2.09E+01	2.61E+00	-3.15E-01	1.41E+03	5.17E+03	-2.52E+02	-1.10E+02	1.29E+03	1.24E+03	1.33E+01	1.24E+03	2.04E+01	1.81E+03	-3.24E+02	-1.05E+03	3.80E+04
0.4	2.20E+01	2.77E+00	-3.32E-01	1.49E+03	5.49E+03	-2.04E+02	-1.03E+02	1.20E+03	1.15E+03	1.23E+01	1.15E+03	2.03E+01	1.79E+03	-3.27E+02	-1.57E+03	4.00E+04
0.42	2.31E+01	2.92E+00	-3.56E-01	1.60E+03	5.78E+03	-1.52E+02	-9.31E+01	1.17E+03	1.12E+03	1.20E+01	1.12E+03	2.02E+01	1.77E+03	-3.22E+02	-1.99E+03	4.18E+04
0.44	2.42E+01	3.07E+00	-3.87E-01	1.73E+03	6.03E+03	-1.03E+02	-7.88E+01	1.20E+03	1.15E+03	1.24E+01	1.15E+03	2.01E+01	1.76E+03	-3.10E+02	-2.31E+03	4.32E+04
0.46	2.53E+01	3.21E+00	-4.26E-01	1.88E+03	6.24E+03	-6.19E+01	-6.04E+01	1.30E+03	1.24E+03	1.34E+01	1.24E+03	1.99E+01	1.74E+03	-2.94E+02	-2.51E+03	4.43E+04
0.48	2.64E+01	3.34E+00	-4.70E-01	2.05E+03	6.41E+03	-2.03E+01	-4.05E+01	1.44E+03	1.37E+03	1.49E+01	1.37E+03	1.98E+01	1.73E+03	-2.70E+02	-2.59E+03	4.51E+04
0.5	2.75E+01	3.47E+00	-5.18E-01	2.23E+03	6.55E+03	1.45E+01	-9.73E+00	1.63E+03	1.55E+03	1.68E+01	1.55E+03	1.98E+01	1.72E+03	-2.35E+02	-2.55E+03	4.56E+04
0.52	2.86E+01	3.59E+00	-5.68E-01	2.42E+03	6.65E+03	3.68E+01	9.97E+00	1.85E+03	1.75E+03	1.90E+01	1.75E+03	1.98E+01	1.72E+03	-2.10E+02	-2.39E+03	4.60E+04
0.54	2.97E+01	3.71E+00	-6.18E-01	2.61E+03	6.73E+03	5.96E+01	4.31E+01	2.07E+03	1.96E+03	2.14E+01	1.96E+03	1.98E+01	1.72E+03	-1.67E+02	-2.12E+03	4.60E+04
0.56	3.08E+01	3.81E+00	-6.67E-01	2.79E+03	6.78E+03	7.12E+01	8.15E+01	2.29E+03	2.17E+03	2.36E+01	2.17E+03	1.98E+01	1.72E+03	-1.20E+02	-1.77E+03	4.60E+04
0.58	3.19E+01	3.91E+00	-7.10E-01	2.96E+03	6.80E+03	8.25E+01	1.33E+02	2.47E+03	2.35E+03	2.55E+01	2.35E+03	1.99E+01	1.73E+03	-5.72E+01	-1.36E+03	4.55E+04
0.6	3.30E+01	3.99E+00	-7.48E-01	3.11E+03	6.82E+03	6.67E+01	1.34E+02	2.63E+03	2.51E+03	2.71E+01	2.51E+03	2.00E+01	1.75E+03	-4.40E+01	-8.27E+02	4.55E+04
0.62	3.41E+01	4.07E+00	-7.79E-01	3.23E+03	6.82E+03	4.96E+01	1.50E+02	2.74E+03	2.62E+03	2.82E+01	2.62E+03	2.01E+01	1.76E+03	-1.57E+01	-2.81E+02	4.53E+04

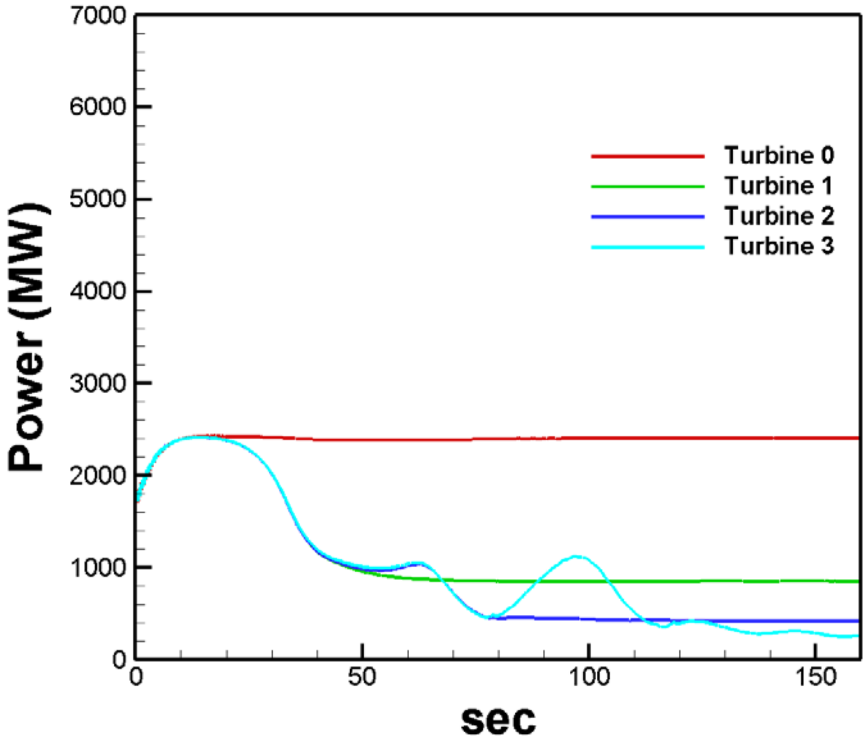
Example: primary0.out
- loads data primary*out can be opened using Excel with "tab delimited" options

- columns of data can be selected to generate figures

Out-of-plane Blade Loadings and Power Output from FAST



Blade root out-of-plane bending moment



Power generation

Example Cases:
Wind Farm Simulation

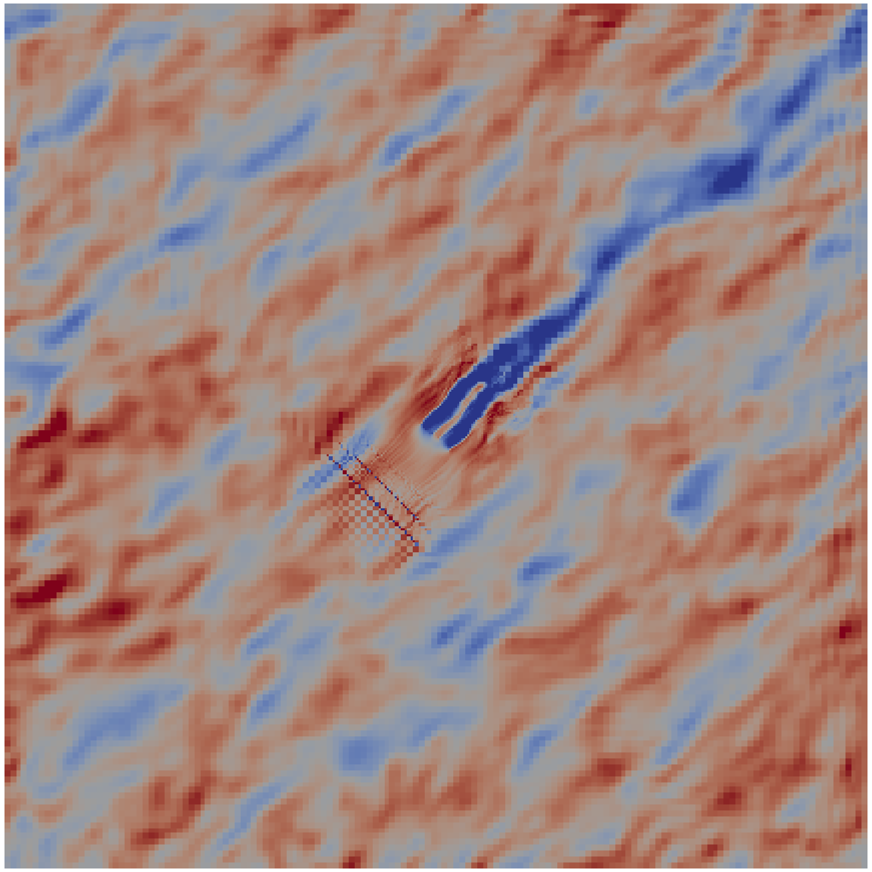
Wind Farm Simulation

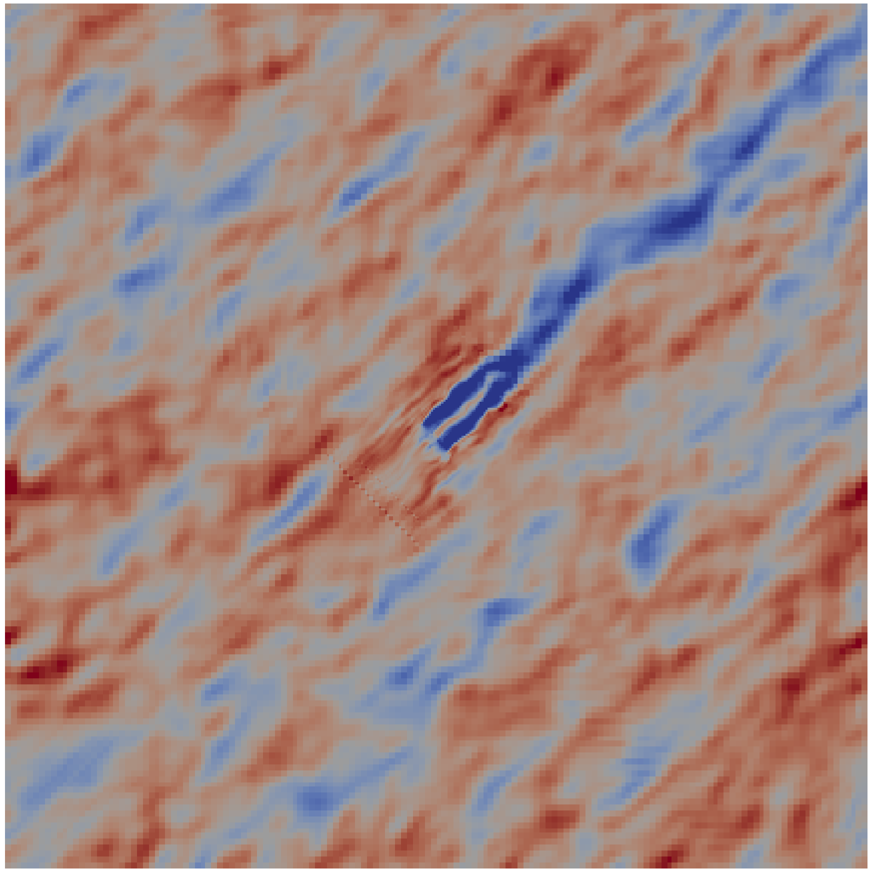
- **See "tutorials/windPlant"**
- **Uses the solver windPlantPisoSolver**
- **2 cases: Neutral and unstable ($-z_i/L \approx 4$)**
- **Wind: 9 m/s from 225 deg at 90 m**
- **Domain size: 3km × 3km × 1 km (x × y × z)**
- **Grid size:**
  - Background grid is same as ABL precursor
  - Locally refined down to 2.5 m around single 5MW turbine in horizontal center of domain with 90 m hub height
- **Run on 64 processors**
  - Took 21 hrs for 750 s of simulation time
  - Much smaller time step than precursor (dt = 0.015s)

The Process (see the “Allrun” script)

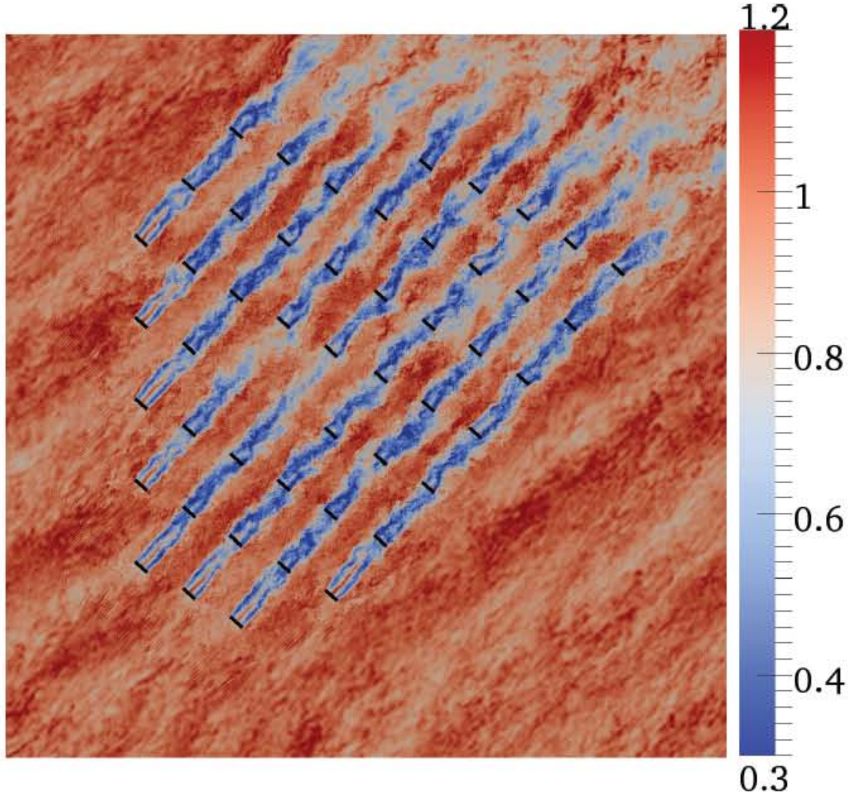
- **Build a coarse mesh with blockMesh (serial)**
  - Builds a hexahedral mesh
- **Locally refine with topoSet (serial) and refineMesh (serial)**
- **Use refineMesh (serial) to globally refine mesh to desired resolution**
  - Splits hexahedral cells in half in each direction
- **Use initial field files from precursor simulation, but change the periodic boundaries to inflow/outflow (timeVaryingFixedMapped) to use saved boundary data from precursor using changeDictionary (serial)**
- **Renumber the cells to get better matrix banding with renumberMesh (serial)**
- **Decompose the domain with decomposePar (serial)**
- **Initialize solution with precursor field using mapFields (serial)**
- **Run the solver**

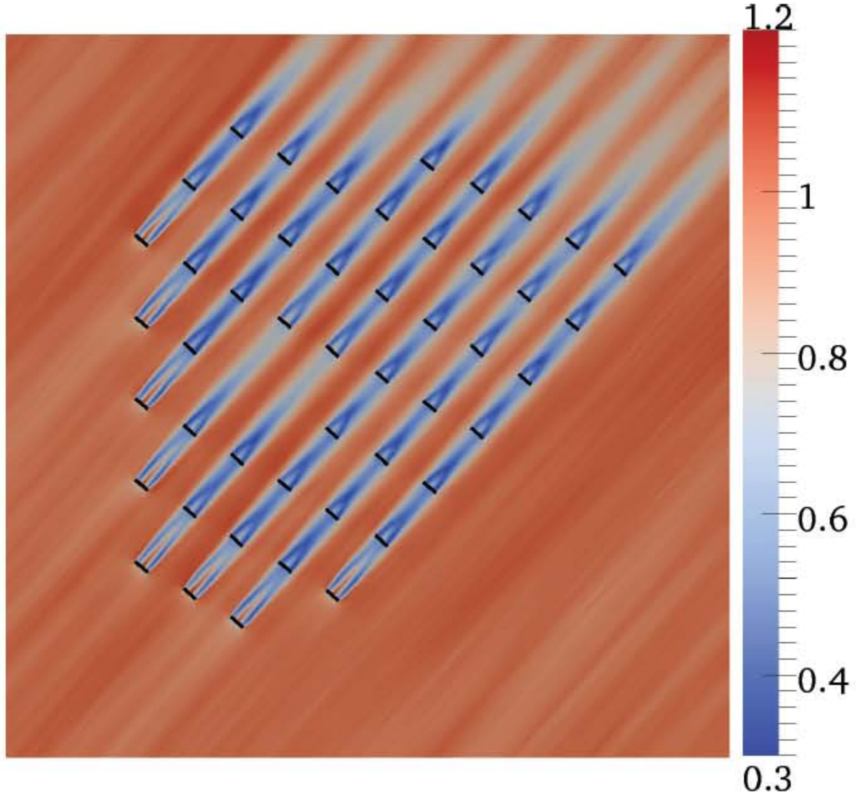
Results


The effect of too rapid a transition in grid resolution


Increasing the filter width helped, but not the best fix

Results





Results from a 48 turbine simulation[1] of the Lillgrund offshore wind farm

[1] Churchfield, M. J., Lee, S., Michalakes, J., and Moriarty, P. J., "A Numerical Study of the Effects of Atmospheric and Wake Turbulence on Wind Turbine Dynamics," Journal of Turbulence, Vol. 13, No. 14, pp. 1-32, 2012.

Some References

Churchfield, M. J., Lee, S., Michalakes, J., and Moriarty, P. J., “A Numerical Study of the Effects of Atmospheric and Wake Turbulence on Wind Turbine Dynamics,” Journal of Turbulence, Vol. 13, No. 14, pp. 1-32, 2012.

Churchfield, M. J., Lee, S., Moriarty, P. J., Martinez, L. A., Leonardi, S., Vijayakumar, G., and Brasseur, J. G., “A Large-Eddy Simulation of Wind-Plant Aerodynamics,” AIAA Paper AIAA-2012-537, 2012.

Lee, S., Churchfield, M. J., Moriarty, P. J., Jonkman, J., “Atmospheric and Wake Turbulence Impacts on Wind Turbine Fatigue Loading,” AIAA Paper AIAA-2012-540, 2012.

Martinez, L. A, Leonardi, S., Churchfield, M. J., Moriarty, P. J., “A Comparison of Actuator Disk and Actuator Line Wind Turbine Models and Best Practices for Their Use,” AIAA Paper AIAA-2012-900, 2012.

Acknowledgements

- **Atmospheric Boundary Layer and OpenFOAM-related**
  - Jim Brasseur, Eric Patterson, Ganesh Vijayakumar, Adam Lavely, Mike Kinzel
- **Actuator Line Model**
  - Tony Martínez, Stefano Leonardi
- **NREL collaborators**
  - Pat Moriarty, Mike Sprague, Julie Lundquist, John Michalakes, Avi Purkayastha

Help

- **First check the NWTC Codes forum at: https://wind.nrel.gov/forum/wind/**

- **Then contact**
    - Matt Churchfield (matt.churchfield@nrel.gov)
    - Sang Lee (sang.lee@nrel.gov)