Overview of FAST

NREL Wind Turbine Modeling Workshop

September 11-12, 2014
Bergen, Norway

Jason Jonkman, Ph.D.
Senior Engineer, NREL
Outline

• Introduction & Background:
  – FAST – What Is It?
  – History

• FAST Modularization Framework:
  – Why was a New Framework Needed?
  – What is the FAST Modularization Framework?
  – Design Features of the Framework
  – Module Form
  – Functions of the FAST Driver
  – Input-Output Transformations
  – Coupling Dilemma
  – Loose-Coupling Algorithm
  – Independent Spatial Discretizations
  – Units & Coordinate Systems
  – Benefits of the Framework
  – Status

• Sample Models Provided with the FAST Archive

• FAST Input & Output Files

• Modeling Guidance

• Recent Work

• Current & Planned Work

• Future Opportunities
Introduction & Background

FAST – What Is It?

- Aero-hydro-servo-elastic model for wind turbines:
  - Used to stand for Fatigue, Aerodynamics, Structures, & Turbulence
  - Now just “FAST”
  - Couples individual modules (AeroDyn, HydroDyn, ServoDyn, ElastoDyn, SubDyn, MAP, IceFloe) for aero-hydro-servo-elastic simulation
  - Evaluated by Germanischer Lloyd WindEnergie

- Latest version:
  - v8.08.00c-bjj (July 2014)
  - Newer in progress

- User’s Guide:
  - ReadMe for v8 (2014)

- Theory Manual:
  - Framework: Jonkman (AIAA 2013)
  - Coupling: Sprague & Jonkman (AIAA 2014)
**Introduction & Background**

**History**

**FAST2, FAST3** (pre-1996)
- Developer: B. Wilson, OSU
- Different code for 2- & 3-blades
- Built-in aerodynamics

**FAST_AD2, FAST_AD3** (1996)
- Developer: A. Wright, NREL
- Different code for 2- & 3-blades
- AeroDyn aerodynamics

**FAST v4 – v7** (2002-2013)
- Developer: J. Jonkman, NREL
- Single code for 2- & 3-blades
- Rederived & implemented EoM
- New DOFs (furling, platform)
- AeroDyn aerodynamics
- HydroDyn hydrodynamics
- DLL, MATLAB/Simulink, & LabVIEW controller interfaces
- Linearization
- FAST-to-ADAMS preprocessor

**FAST_AD v1 – v3** (1997-2002)
- Developers: N. Weaver, M. Buhl, et al., NREL
- Single code for 2- & 3-blades
- AeroDyn aerodynamics
Introduction & Background

History – FAST v8 (2013-Present)

• Conversion of **FAST** & its various modules (including **AeroDyn** & **HydroDyn**) to new modularization framework

• Splitting of **FAST** into:
  – **FAST** driver (glue) code
  – **ElastoDyn** module for structural dynamics
  – **ServoDyn** module for controller & electrical drive

• Introduction of:
  – **SubDyn** module for multi-member substructure structural dynamics
  – **MAP** module for multi-segmented mooring quasi-statics
  – **IceFloe** for quasi-steady sea ice
## FAST Modularization Framework

### Why was a New Framework Needed?

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
</table>
| • Limited range of modeling fidelity                                   | • Framework allowing modules to be exchanged  
| • Solution driven by structural solver                                 | • Development of new modules of higher fidelity                                             |
| • Inability to isolate a given model                                   | • Separate module interface & coupler                                                        |
| • Inability to be driven by other codes                                | • Modules that can be called by separate driver programs or interfaced together to form a coupled solution |
| • Dependent spatial discretizations & time steps across modules        | • Library of spatial elements & mesh-to-mesh mapping                                         |
| • Inability to linearize all system equations                          | • Data transfer with interpolation/extrapolation in time                                     |
| • Focus on single turbine                                             | • Tight coupling with options for operating-point determination & linearization             |
| • “Spaghetti code” due to unclear data transfer & global data          | • Dynamic allocation of modules for wind-plant simulation                                    |
| • Limited number of developers due to code size & complexity           | • Modularization with data encapsulation                                                    |
| • Potentially poor numerical accuracy & stability                       | • Modularization of code into separate components                                            |
|                                                                         | • Programmer’s handbook explaining code development requirements & best practices           |
|                                                                         | • Multiple coupling schemes & integration/solver options                                    |
FAST Modularization Framework

What is the FAST Modularization Framework?

- A means by which various mathematical models are implemented in distinct modules & interconnected to solve for the global, coupled, dynamic response of a system.

Loose- (Left) & Tight- (Right) Coupling Schemes

Uncoupled Solution of a Module Intended for Loose (Top) & Tight (Bottom) Coupling
# FAST Modularization Framework

## Design Features of the Framework

- **Module-independent inputs, outputs, states, & parameters**
- **States in continuous-time, discrete-time, & in constraint form**
- **Loose & tight coupling**
- **Independent time & spatial discretizations**
- **Time marching, operating-point determination, & linearization**
- **Data encapsulation & dynamic allocation**
- **Save/retrieve capability**

## Features

<table>
<thead>
<tr>
<th>Features</th>
<th>Loose</th>
<th>Tight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module-Independent Variables</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Inputs</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Outputs</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Parameters</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Continuous states</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Discrete states</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Constraint states</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>System Formulation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Explicit continuous-time ODEs</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Explicit discrete-time updates</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Constraint equations of index 1</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Output equations with direct feedthrough</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Semi-explicit DAEs of index 1</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Systems of any form</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Independent Spatial Discretizations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Available</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Operating-Point Determination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Static equilibrium</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>• Steady state</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>• Periodic steady state</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>• With trim of inputs</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Linearization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• About given initial conditions</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>• About given time</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>• About operating point</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Time Marching</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• From given initial conditions</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• From operating point</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Independent time steps for continuous states between modules</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Independent time steps for discrete states between modules</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Solution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Solver implementation is up to the module developer</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Solver is selectable from those available in the glue</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Overall solvability, numerical stability, and convergence verifiable</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Data Encapsulation and No Global Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Required</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dynamic Allocation of Instances of Modules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Available</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Save/Retrieve Capability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Available</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
FAST Modularization Framework

Module Form

<table>
<thead>
<tr>
<th>Inputs</th>
<th>States</th>
<th>Parameters</th>
<th>Outputs</th>
</tr>
</thead>
</table>

- General (need-not-be linear) state-space formulation
- No constraints → ODEs
- With constraints → DAEs
- Tight coupling restricted to a hybrid semi-explicit index-1 DAE
- No restriction on formulation in loose coupling
- A module can be set up for both loose & tight coupling

Given: \( p, x(0), x^d[0], & u \)

\[
\dot{x} = X \left( x, x^d, z, u, t \right)
\]

\[
x^d[n + 1] = X^d \left( \left. x \right|_{t=n\Delta t}, \left. x^d[n], z \right|_{t=n\Delta t}, \left. u \right|_{t=n\Delta t}, t \right|_{t=n\Delta t} \right)
\]

\[
0 = Z \left( x, x^d, z, u, t \right)
\]

\[
y = Y \left( x, x^d, z, u, t \right)
\]

Note: Index -1 DAE implies \( \frac{\partial Z}{\partial z} \neq 0 \),

thus \( \left[ \frac{\partial Z}{\partial z} \right]^{-1} \) exists

Note: If \( x^d \) exists, \( x^d[n] \) is applied over \( n\Delta t \leq t < (n+1)\Delta t \) in all equations
### FAST Modularization Framework

**Module Form – Data Structures & Subroutines**

#### Data Structures
- Initialization input
- Initialization output
- System input \((u)\)
- System output \((y)\)
- System states:
  - Continuous \((x)\)
  - Discrete \((x^d)\)
  - Constraint \((z)\)
  - Other (added for code efficiency/flexibility)
- System parameters \((p)\)

#### Subroutines
- Initialization
- End
- Calculate output \((Y())\)
- For loose coupling only:
  - Update states
  - Input/output extrap./interpolation*
- For tight coupling only:
  - Calculate continuous-state derivatives \((X())\)
  - Update discrete states \((X^d())\)
  - Calculate constraint-state residual \((Z())\)
  - Jacobians
- Pack/unpack data*
- Copy/destroy data*

*Auto-generated by FAST Registry
FAST Modularization Framework
Functions of the FAST Driver (“Glue Code”)

FAST
Driver (“Glue Code”)

Loose coupling:
• Drives time-domain solution forward
• Calls individual modules
• Derives module inputs from outputs:
  – Including mesh-to-mesh mapping
  – Including interpolation/extrapolation in time

Tight coupling*:
• All of the above, plus
• Integrates coupled system equations using a common solver
• Determines operating points
• Performs model linearization

*A tight coupling is not yet available.
Module inputs \( (u) \) are algebraically derived from module outputs \( (y) \).

Input-output transformations \( (U()) \) form additional constraint equations:
- Solved via Newton iterations

Transformations include mapping between module-independent spatial discretizations.

\[\begin{align*}
\left\{ u^{(1)} \right\}, \quad y^{(1)} \\
\left\{ u^{(2)} \right\}, \quad y^{(2)} \\
\vdots \\
\left\{ u^{(N)} \right\}, \quad y^{(N)}
\end{align*}\]

Given: \( N \) coupled systems

\[0 = U(u, y, t)\]

\[0 = U\left(u, Y(x, x^d, z, u, t), t\right)\]

\[0 = U\left(x, x^d, z, u, t\right)\]
Coupled System (Left) & Effective Tightly Coupled System (Right)
FAST Modularization Framework
Loose-Coupling Algorithm

- Predictor-Corrector (PC)-based with time-step subcycling of modules
- Set-up for future parallelization (steps 1, 2, & 3a)

For Each Time Step
(From $t_n$):
1) Extrapolate Inputs to $t_{n+1}$
2) Advance States to $t_{n+1}$
3) Solve for Outputs & Inputs @ $t_{n+1}$:
   3a) Calculate Outputs
   3b) Derive Inputs From Outputs
   3c) Iterate (Go Back to 3a) or Save
4) Correct (Go Back to 2) or Save

PC-Based Loose-Coupling Algorithm

Within an Individual Module

Coupling of Modules
Module inputs & outputs residing on a spatial boundary use a mesh.

A mesh consists of:
- Nodes
- Elements (nodal connectivity)
- Nodal reference locations (position & orientation)
- One or more nodal fields, including motion, load, &/or scalar quantities

Library of spatial meshes:
- Points (0D – rigid bodies / concentrated loads)
- Lines (1D – beams / loads per unit length)
- Surfaces* (2D – shells / traction loads, per unit area)
- Volumes* (3D – solids / body loads, per unit volume)

*2D & 3D elements not yet supported
FAST Modularization Framework
Independent Spatial Discretizations – Mapping

- Mesh-to-mesh mapping supports:
  - Extremely disparate meshes
  - Large motion/deformation
  - Relative motion or follower meshes
  - Point-to-Point, Line2-to-Line2, Point-to-Line2, & Line2-to-Point connections

- Mesh-to-mesh mappings involve:
  - Mapping search – Nearest neighbors are found
  - Mapping transfer – Nodal fields are transferred

- Mapping guiding principles:
  - Load transfer maintains force & moment balance
  - Motion transfer maintains rigid-body displacement, velocity, & acceleration
  - Load & motion mappings are conjugate
  - When meshes are identical, there is a one-to-one mapping of fields

A Multi-Member Hydrodynamics Model Coupled to a Single-Beam Model
FAST Modularization Framework
Independent Spatial Discretizations – Example

Module 1
- Input: Loads
- Output: Motions

Module 2:
- Input: Motions
- Output: Loads

Reference Locations

Module 1 Output: Translate 2 units along X & rotate 20° about X

Motion Mapping

Module 2 Output: Apply uniform force of 1 unit along X

Load Mapping
**FAST Modularization Framework**

**Units & Coordinate Systems**

- Module inputs & outputs (also, most input & output files) use SI base units:
  - Kilograms
  - Seconds
  - Newtons
  - Meters
  - Radians
  - Watts

- Module inputs & outputs are in **FAST**’s global coordinate system (an inertial frame):
  - **Origin:**
    - Land – Intersection of undeflected tower centerline & ground
    - Offshore – Intersection of undeflected support structure centerline & mean sea level (MSL)
  - **Orthogonal axes:**
    - $X_i$ – Directed nominally downwind (0° wind/wave direction)
    - $Y_i$ – Directed left when looking downwind
    - $Z_i$ – Directed vertically opposite gravity
  - Also, most other coordinate systems used for input & output files follow IEC convention (tower, nacelle, shaft, blade, etc.)
FAST Modularization Framework

Benefits of the Framework

• Improve ability to read, implement, & maintain source code
• Increase module sharing & shared code development across wind community
• Improve numerical performance & robustness
• Greatly enhance flexibility & expandability to enable further developments of functionality without need to recode established modules
FAST Modularization Framework
Status – Framework Features Developed to Date

- Core mathematical basis
- Module source code template (Fortran)
- Registry for automatic generation of general code
- Programmer’s Handbook
- Glue code supporting PC-based loose coupling with time-step subcycling of modules
- Mappings between module-unique spatial discretizations for point & line element meshes
- Tests of simple examples
- Conversion of FAST
FAST Modularization Framework

Status – Framework Features Still to be Developed

• Loose coupling with module-independent time steps
• Mesh-to-mesh mappings for surface & volume element meshes
• Tight coupling
• Operating-point determination
• Linearization
• General algorithmic improvement for code efficiency (sparse storage, etc.)
• Advanced modularization features (parallel processing, etc.)
This workshop will apply **FAST v8**

All new features are being added to the new framework

Until all features of v7 are included in v8, both will be supported

<table>
<thead>
<tr>
<th>FAST Features</th>
<th>v7.02</th>
<th>v8.03</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time marching</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Operating-point determination</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Linearization</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FAST-to-ADAMS preprocessor</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Follows the new FAST modularization framework</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Structural and control routines separated from driver code</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Independent time steps between modules</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Independent spatial discretization between modules</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Multiple integration options</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Loose coupling with predictor-corrector across modules</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Both 32-bit and 64-bit applications available</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Supports both Windows and Linux operating systems</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Optimized for efficiency</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Supports mixed Fortran/C</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Compiles with gfortran</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Sample Models Provided with the FAST Archive

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Turbine Name</th>
<th>No. Blades</th>
<th>Rotor Diameter (m)</th>
<th>Rated Power (kW)</th>
<th>Test Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test01</td>
<td>AWT-27CR2</td>
<td>2</td>
<td>27</td>
<td>175</td>
<td>Flexible, fixed yaw error, steady wind</td>
</tr>
<tr>
<td>Test02</td>
<td>AWT-27CR2</td>
<td>2</td>
<td>27</td>
<td>175</td>
<td>Flexible, steady wind</td>
</tr>
<tr>
<td>Test03</td>
<td>AWT-27CR2</td>
<td>2</td>
<td>27</td>
<td>175</td>
<td>Flexible, free yaw, steady wind</td>
</tr>
<tr>
<td>Test04</td>
<td>AWT-27CR2</td>
<td>2</td>
<td>27</td>
<td>175</td>
<td>Flexible, free yaw, turbulence</td>
</tr>
<tr>
<td>Test05</td>
<td>AWT-27CR2</td>
<td>2</td>
<td>27</td>
<td>175</td>
<td>Flexible, steady wind</td>
</tr>
<tr>
<td>Test06</td>
<td>AOC-15/50</td>
<td>3</td>
<td>15</td>
<td>50</td>
<td>Flexible, steady wind</td>
</tr>
<tr>
<td>Test07</td>
<td>AOC-15/50</td>
<td>3</td>
<td>15</td>
<td>50</td>
<td>Flexible, free yaw error, steady wind</td>
</tr>
<tr>
<td>Test08</td>
<td>AOC-15/50</td>
<td>3</td>
<td>15</td>
<td>50</td>
<td>Flexible, fixed yaw error, steady wind</td>
</tr>
<tr>
<td>Test09</td>
<td>UAE VI downwind</td>
<td>2</td>
<td>10</td>
<td>20</td>
<td>Flexible, yaw ramp, steady wind</td>
</tr>
<tr>
<td>Test10</td>
<td>UAE VI upwind</td>
<td>2</td>
<td>10</td>
<td>20</td>
<td>Rigid, power curve, ramp wind</td>
</tr>
<tr>
<td>Test11</td>
<td>WP 1.5 MW</td>
<td>3</td>
<td>70</td>
<td>1500</td>
<td>Flexible, variable speed &amp; pitch control, pitch failure, turbulence</td>
</tr>
<tr>
<td>Test12</td>
<td>WP 1.5 MW</td>
<td>3</td>
<td>70</td>
<td>1500</td>
<td>Flexible, variable speed &amp; pitch control, ECD event</td>
</tr>
<tr>
<td>Test13</td>
<td>WP 1.5 MW</td>
<td>3</td>
<td>70</td>
<td>1500</td>
<td>Flexible, variable speed &amp; pitch control, turbulence</td>
</tr>
<tr>
<td>Test14</td>
<td>Not currently available</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test15</td>
<td>SWRT</td>
<td>3</td>
<td>5.8</td>
<td>10</td>
<td>Flexible, variable speed control, free yaw, EOG01 event</td>
</tr>
<tr>
<td>Test16</td>
<td>SWRT</td>
<td>3</td>
<td>5.8</td>
<td>10</td>
<td>Flexible, variable speed control, free yaw, EDC01 event</td>
</tr>
<tr>
<td>Test17</td>
<td>SWRT</td>
<td>3</td>
<td>5.8</td>
<td>10</td>
<td>Flexible, variable speed control, free yaw, turbulence</td>
</tr>
<tr>
<td>Test18</td>
<td>NREL 5 MW - Land-based</td>
<td>3</td>
<td>126</td>
<td>5000</td>
<td>Flexible, DLL control, tower potential flow and drag, turbulence</td>
</tr>
<tr>
<td>Test19</td>
<td>NREL 5 MW - OC3-Monopile</td>
<td>3</td>
<td>126</td>
<td>5000</td>
<td>Flexible, DLL control, tower potential flow, turbulence, irregular waves</td>
</tr>
<tr>
<td>Test20</td>
<td>NREL 5 MW - OC3-Tripod</td>
<td>3</td>
<td>126</td>
<td>5000</td>
<td>Flexible, DLL control, tower potential flow, steady wind, regular waves with 0 phase</td>
</tr>
<tr>
<td>Test21</td>
<td>NREL 5 MW - OC4-Jacket</td>
<td>3</td>
<td>126</td>
<td>5000</td>
<td>Flexible, DLL control, tower potential flow, turbulence, irregular waves, marine growth</td>
</tr>
<tr>
<td>Test22</td>
<td>NREL 5 MW - ITI Barge</td>
<td>3</td>
<td>126</td>
<td>5000</td>
<td>Flexible, DLL control, irregular &amp; multidirectional waves, turbulence</td>
</tr>
<tr>
<td>Test23</td>
<td>NREL 5 MW - MIT/NREL TLP</td>
<td>3</td>
<td>126</td>
<td>5000</td>
<td>Flexible, DLL control, turbulence, irregular waves</td>
</tr>
<tr>
<td>Test24</td>
<td>NREL 5 MW - OC3-Hywind</td>
<td>3</td>
<td>126</td>
<td>5000</td>
<td>Flexible, DLL control, turbulence, irregular waves</td>
</tr>
<tr>
<td>Test25</td>
<td>NREL 5 MW - OC4-DeepCwind Semi-Submersible</td>
<td>3</td>
<td>126</td>
<td>5000</td>
<td>Shortened OC4 Load Case (LC) 3.7: steady wind, white noise waves</td>
</tr>
</tbody>
</table>
FAST Input & Output Files

- Input files
- Output files
- *echo files not shown
- Not yet available in FAST v8.08.00c, but under development
- Available in FAST v8.08.00c, but incomplete
Modeling Guidance

- **Time step:**
  - $DT = 1/(10 \times \text{highest frequency in Hz of coupling between modules})$

- **Interpolation/extrapolation of module inputs:**
  - $\text{InterpOrder} = 2$ – Quadratic

- **Number of correction iterations:**
  - $\text{NumCrctn} = 0$ – No corrections for speed
  - $\text{NumCrctn} > 0$ – may be needed to achieve a given convergence rate of an underlying integrator

- **Time between calls to get Jacobian:**
  - $DT_{\text{UJac}} > T_{\text{Max}}$ – If the **ElastoDyn** platform doesn’t rotate much, or
  - $DT_{\text{UJac}} = 1/(10 \times \text{highest frequency in Hz of platform rotation mode})$

- **Scaling factor used in Jacobian:**
  - $U_{\text{JacSclFact}} = 1E+06$ – Ratio of load values to acceleration values
Modeling Guidance (cont)

• Simplify models (e.g. disable modules & eliminate DOFs) to debug problems
• The ElastoDyn module will always be used
• Land-based versus offshore – Determined by setting of CompHydro
• Offshore fixed-bottom versus floating – Determined by setting of CompSub
• Spatial discretization – No more than 10:1 between modules
• MATLAB scripts available for converting input files from FAST v7 to v8
Recent Work, Current & Planned Work, & Future Opportunities

• Recent work:
  – Established the modularization framework
  – Converted FAST to the framework (v8)

• Current & planned work:
  – Address current limitations of FAST v8 relative to v7
  – Further development of the framework
  – Add animation capability
  – Documentation of FAST v8

• Future opportunities:
  – Develop limited-functionality version (FAST_EZ) for ease of use by students in class
  – Apply framework to the modeling of VAWTs
Questions?

Jason Jonkman, Ph.D.
+1 (303) 384 – 7026
jason.jonkman@nrel.gov