ReadMe File for FAST v8.08.00c-bjj

Bonnie Jonkman, Jason Jonkman  
National Renewable Energy Laboratory  
June 30, 2014

# Introduction

This document is designed to guide you through some of the changes that the FAST wind turbine computer-aided engineering (CAE) tool is undergoing. FAST v8.08.00c-bjj is the second public release of FAST under the [new modularization framework](http://wind.nrel.gov/designcodes/simulators/developers/) developed at NREL. The architecture of FAST v8 is entirely different from FAST v7.02.00d-bjj. These differences are highlighted in Figure 1.

The modules of FAST (AeroDyn, HydroDyn, etc.) correspond to different physical domains of the coupled aero-hydro-servo-elastic solution, most of which are separated by spatial boundaries. Figure 2 shows the control volumes associated with each module for fixed-bottom offshore wind turbines. For land-based wind turbines, the HydroDyn hydrodynamics module would not be used and the SubDyn multi-member substructure structural-dynamics module is optional. Figure 3 shows the control volumes associated with each module for floating offshore wind turbines.

Many features of [FAST v7.02.00d-bjj](http://wind.nrel.gov/designcodes/simulators/fast) have not yet been added to FAST v8, so we will continue to support both versions of the software (FAST v7 and FAST v8) until FAST v8 is deemed a suitable replacement.

Table 1 summarizes the different features available in each version.

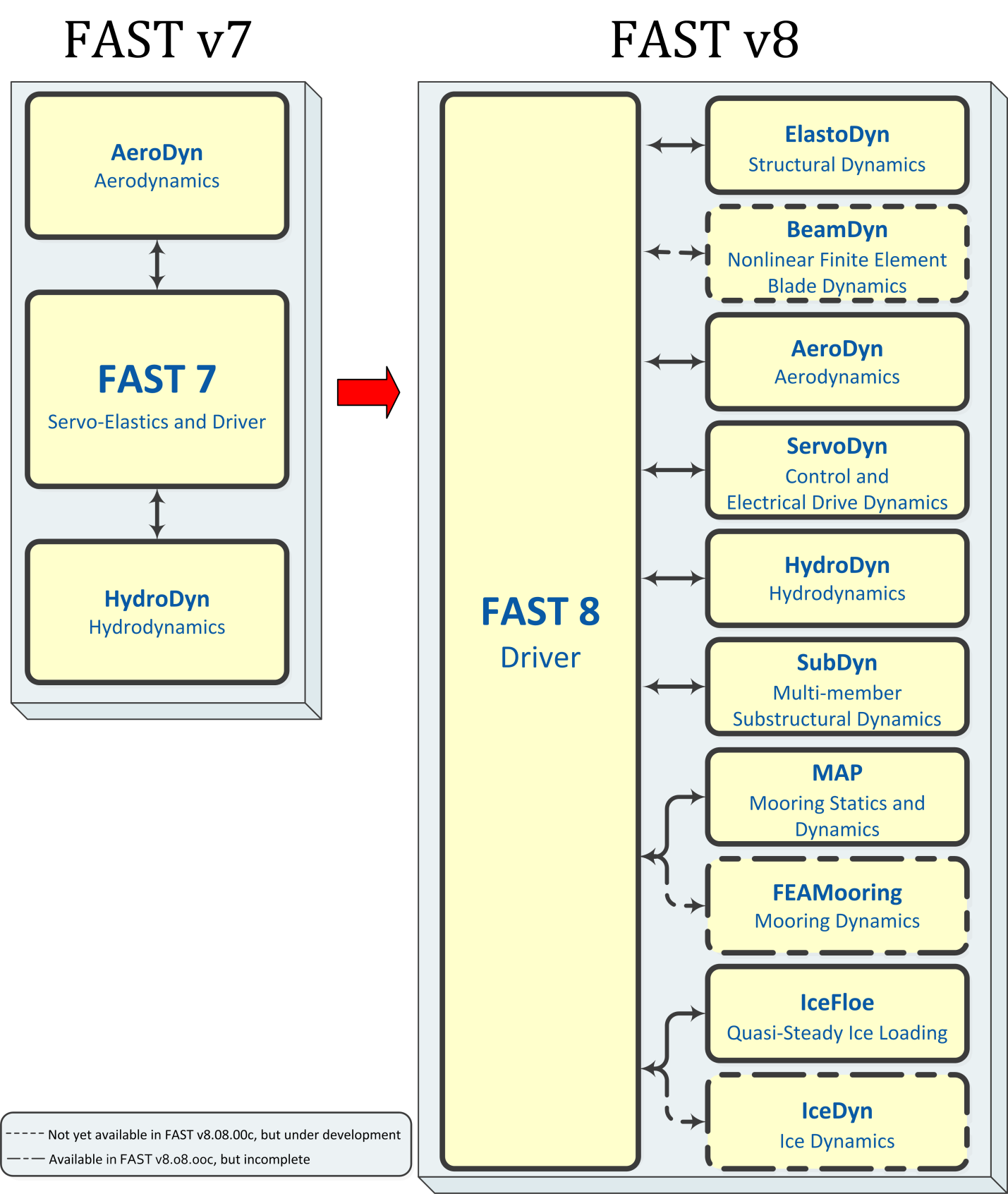


Figure : Architectural comparison of FAST 7 and FAST 8



Figure : FAST control volumes for fixed-bottom systems



Figure : FAST control volumes for floating systems

Table : Comparison of features between FAST v7 and v8

|  |  |  |
| --- | --- | --- |
| **Aerodynamics (AeroDyn and InflowWind)** |  |  |
| **FAST Features** | **v7.02** | **v8.08** |
| • Quasi-steady or dynamic wake |  |  |
| • Steady or unsteady airfoil aerodynamics |  |  |
| • Tower shadow for downwind rotors |  |  |
| • Tower influence for upwind rotors |  |  |
| • Tower drag loading |  |  |
| • Tail-fin aerodynamic loading |  |  |
| • "Hub-height", TurbSim, and GH Bladed wind file formats |  |  |
| • Other wind formats |  |  |
| • Aeroacoustics (noise) |  |  |
| **Hydrodynamics (HydroDyn)** |  |  |
| **FAST Features** | **v7.02** | **v8.08** |
| • Linear regular or irregular waves |  |  |
| • White-noise waves |  |  |
| • Wave directional spreading |  |  |
| • Wave stretching |  |  |
| • Externally generated wave data |  |  |
| • Sea current |  |  |
| • Morison's equation for central member |  |  |
| • Morison's equation for multiple intersecting members |  |  |
| • Static buoyancy and dynamic pressure on members |  |  |
| • Support for inclined and tapered members |  |  |
| • Support for flooded and ballasted members |  |  |
| • Support for marine growth |  |  |
| • First-order potential flow (from WAMIT) |  |  |
| • Radiation "memory effect" captured through time-domain convolution |  |  |
| • Radiation "memory effect" captured through linear state-space form |  |  |
| **Control and Electrical System (Servo) Dynamics (ServoDyn)** |  |  |
| **FAST Features** | **v7.02** | **v8.08** |
| • Blade-pitch control |  |  |
| • Override pitch maneuvers |  |  |
| • Generator models |  |  |
| • Torque control |  |  |
| • High-speed shaft brake |  |  |
| • Nacelle-yaw control |  |  |
| • Override yaw maneuvers |  |  |
| • Blade-tip brakes |  |  |
| • GH Bladed DLL interface[[1]](#footnote-1) |  |  |
| • Simulink interface |  |  |
| • LabVIEW interface |  |  |
| **Structural Dynamics (ElastoDyn, SubDyn, and MAP)** |  |  |
| **FAST Features** | **v7.02** | **v8.08** |
| • Blade-bending DOFs |  |  |
| • Rotor teeter DOF |  |  |
| • Generator azimuth and drivetrain torsion DOFs |  |  |
| • Nacelle-yaw DOF |  |  |
| • Tower-bending DOFs |  |  |
| • Rigid-body platform DOFs |  |  |
| • Furling DOFs |  |  |
| • Fixed-bottom multi-member substructure DOFs: |  |  |
| − Solved with linear frame finite-element or Craig-Bampton reduction |  |  |
| • Gravitational loading |  |  |
| • Gearbox friction |  |  |
| • System of independent mooring lines solved quasi-statically |  |  |
| • System of multi-segmented mooring lines solved quasi-statically |  |  |
| • Earthquake excitation |  |  |
| **General** |  |  |
| **FAST Features** | **v7.02** | **v8.03** |
| • Time marching |  |  |
| • Operating-point determination |  |  |
| • Linearization |  |  |
| • FAST-to-ADAMS preprocessor |  |  |
| • Follows the new FAST modularization framework |  |  |
| • Structural and control routines separated from driver code |  |  |
| • Independent time steps between modules | [[2]](#footnote-2) | [[3]](#footnote-3) |
| • Independent spatial discretization between modules |  |  |
| • Multiple integration options |  |  |
| • Loose coupling with predictor-corrector across modules | [[4]](#footnote-4) |  |
| • Both 32-bit and 64-bit applications available |  | [[5]](#footnote-5) |
| • Supports both Windows and Linux operating systems |  |  |
| • Optimized for efficiency |  |  |
| • Supports mixed Fortran/C |  |  |
| • Compiles with gfortran[[6]](#footnote-6) |  |  |

# Major changes in FAST

## v8.08.00c-bjj

* Coupling between ElastoDyn, SubDyn, and HydroDyn was added, allowing FAST to model fixed-bottom offshore turbines.
* The input-output solves have been enhanced; rigorous input-output solve refer to 2014 AIAA;
* The mesh mapping algorithms have been enhanced.
* We now use LAPACK routines for solving linear systems, which has increased the speed of the simulations.
* The glue code allows the option for time-step subcycling. Modules can now choose to use *smaller* time steps than the glue code, as long as the module time step is an integer divisor of the glue-code time step. Note that we have found no cases where this option would be necessary. Future versions of FAST will also allow modules to use *larger* time steps.
* Added new modules for ice loading: IceFloe and IceDyn[[7]](#footnote-7).
* Added another module for mooring lines: FEAMooring‡‡.
* We standardized the names of output files generated by both FAST and its modules. Files generated by FAST will be named  
   <RootName>.<ext>  
  and files generated by FAST modules will be named   
   <RootName>.<ModName>.<ext>   
  where <RootName> is the root name of the primary FAST input file (the file name, including path, without the extension), <ModName> is an abbreviation for the module generating the file, and <ext> is the file extension. File extensions currently are

|  |  |
| --- | --- |
| Output file extension | **File type** |
| sum | Summary file |
| out | Time-marching tabular text output |
| outb | Time-marching tabular binary output |
| ech | Echo of input file(primarily debugging) |

* The “Time Ratio” displayed at the end of a simulation now includes only the CPU time *after* initialization. This ratio was changed to help users better predict the amount of time longer simulations will take (e.g., extrapolating the time a 1-hr simulation will take based on running a 1-min simulation).
* Information about the Jacobian and time steps was added to the FAST summary file.
* Bugs in handling errors were fixed. (In FAST v8.03.02b-bjj, InflowWind did not return all of its errors to the glue code, so it was using zero wind velocity when it went outside the turbulence grid.)
* FAST no longer allows the tower influence model, “NEWTOWER,” to be used in AeroDyn on floating offshore turbines. This tower influence model assumes the tower does not move, which is a poor assumption for floating turbines.
* FAST will now abort if ElastoDyn’s **PtfmRefzt** value is negative for floating offshore systems.
* We fixed a bug in ElastoDyn (also in FAST v7.02.00c) where the linear teeter-damper moment did not use **TeetDmpP.**
* We fixed a problem where the ElastoDyn Aximuth channel would be negative in rare cases.
* We fixed a problem with ElastoDyn’s implementation of AM4, which incorrectly initialized the state history if corrections steps were taken.
* We no longer allow extrapolation orders of 0 in FAST v8.08.00c-bjj.
* We fixed the DISCON\*.DLL files used in the 5MW model certification tests. Previously, they did not work if users did not have Intel Visual Fortran installed on the computers they ran the simulations on.
* We fixed some bugs in the AeroDyn input files of the 5MW Land-Based turbine.
* We set the time steps of the floating offshore certification tests to be the same as they were in FAST v7.02.00d
* We added certification tests for the OC3 Monopile, OC3 Tripod, OC4 Jacket, and OC4 DeepCwind Semi models.
* We have added some more utility files to the FAST archive, including:
  + PlotFASToutput.m, a MATLAB function that can plot some or all channels of one or more FAST output files.
  + ReadSubDynSummary.m, a MATLAB function that can read the SubDyn summary file and put the data into a Matlab data-structure.
* We have fixed the MAP\_win32.dll file distributed with FAST so that it no longer depends on python being installed on the computer running FAST.
* We have added a 64-bit FAST executable to the archive, as well as a 64-bit version of DISCON\_win64.DLL, and a “dummy” 64-bit version of MAP. This executable may be useful for running long simulations of large fixed-bottom offshore models (e.g., the OC4 Jacket); it *cannot* run any models that want to call the MAP module.

FAST v8.08.00c-bjj is compiled with the components listed in Table 2.

Table 2: Components in FAST v8.08.00c-bjj

|  |  |  |
| --- | --- | --- |
| Component | Version |  |
| Modules |  | ModName (for output file) |
| ElastoDyn | v1.01.02b-bjj | ED |
| AeroDyn | v14.02.00c-mlb | AD |
| InflowWind | v2.00.00b-adp | IfW |
| ServoDyn | v1.01.01b-bjj | SrvD |
| HydroDyn | v2.00.10a-gjh | HD |
| SubDyn | v0.04.00a-rrd | SD |
| MAP | v0.87.06a-mdm | MAP |
| FEAMooring |  | FEAM |
| IceFloe |  | IceF |
| IceDyn |  | IceD |
| Other Components |  |  |
| NWTC Subroutine Library | v2.03.00b-bjj | |
| FAST Registry[[8]](#footnote-8) | v2.03.01 | |
| Third Party Content |  |  |
| LAPACK | v3.3 as part of Intel® Math Kernel Library;  (v3.5.0 compiled with gfortran) | |
| ScaLAPACK | 2.0.2 | |
| FFTPACK | v4.1 | |

## v8.03.02b-bjj

Tasks completed to develop FAST v8 included:

* Converted FAST and its various modules (including AeroDyn and HydroDyn) into the [new modularization framework](http://wind.nrel.gov/designcodes/simulators/developers/) (splitting out the controls and electrical drive dynamics into a new ServoDyn module and structural dynamics into a new ElastoDyn module),
* Implemented a new driver program (glue code) supporting loose coupling of the modules,
* Developed mesh-to-mesh mapping schemes between module-independent discretizations of the spatial boundaries between modules,
* Coupled in the recently developed SubDyn module for multi-member substructure structural dynamics and MAP module for multi-segmented mooring quasi-statics, and
* Included a series of models using the NREL 5-MW Baseline wind turbine in the CertTest, including offshore configurations.

The driver program (glue code) couples the modules together; it controls the overall simulation progress and maps module outputs to inputs. We use the name “FAST” both for the driver program (glue code) and overall coupled code.

FAST and each of its modules, except InflowWind, have their own input files; see Figure 4.

# Certification Tests

Several new models have been added to the FAST v8.08.00c-bjj archive. Table 3 lists the tests and models available in the FAST CertTest folder:

Table 3: Certification Tests Distributed with FAST v8.08.00c-bjj

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Name | Turbine Name | No. Blades  (-) | Rotor Diameter (m) | Rated  Power  (kW) | Test Description |
| Test01 | AWT-27CR2 | 2 | 27 | 175 | Flexible, fixed yaw error, steady wind |
| Test02 | AWT-27CR2 | 2 | 27 | 175 | Flexible, start-up, HSS brake shut-down, steady wind |
| Test03 | AWT-27CR2 | 2 | 27 | 175 | Flexible, free yaw, steady wind |
| Test04 | AWT-27CR2 | 2 | 27 | 175 | Flexible, free yaw, turbulence |
| Test05 | AWT-27CR2 | 2 | 27 | 175 | Flexible, generator start-up, tip-brake shutdown, steady wind |
| Test06 | AOC-15/50 | 3 | 15 | 50 | Flexible, tip-brake shutdown, steady wind |
| Test07 | AOC-15/50 | 3 | 15 | 50 | Flexible, free yaw, turbulence |
| Test08 | AOC-15/50 | 3 | 15 | 50 | Flexible, fixed yaw error, steady wind |
| Test09 | UAE VI downwind | 2 | 10 | 20 | Flexible, yaw ramp, steady wind |
| Test10 | UAE VI upwind | 2 | 10 | 20 | Rigid, power curve, ramp wind |
| Test11 | WP 1.5 MW | 3 | 70 | 1500 | Flexible, variable speed & pitch control, pitch failure, turbulence |
| Test12 | WP 1.5 MW | 3 | 70 | 1500 | Flexible, variable speed & pitch control, ECD event |
| Test13 | WP 1.5 MW | 3 | 70 | 1500 | Flexible, variable speed & pitch control, turbulence |
|  |  |  |  |  |  |
| Test15 | SWRT | 3 | 5.8 | 10 | Flexible, variable speed control, free yaw, tail-furl, EOG01 event |
| Test16 | SWRT | 3 | 5.8 | 10 | Flexible, variable speed control, free yaw, tail-furl, EDC01 event |
| Test17 | SWRT | 3 | 5.8 | 10 | Flexible, variable speed control, free yaw, tail-furl, turbulence |
| Test18 | NREL 5 MW - Land-based | 3 | 126 | 5000 | Onshore (land-based) baseline turbine |
| Test19 | NREL 5 MW - OC3 Monopile | 3 | 126 | 5000 | OC3 Monopile |
| Test20 | NREL 5 MW - OC3 Tripod | 3 | 126 | 5000 | OC3 Tripod |
| Test21 | NREL 5 MW - OC4 Jacket | 3 | 126 | 5000 | OC4 Jacket |
| Test 22 | NREL 5 MW - ITI Barge | 3 | 126 | 5000 | ITI Barge |
| Test 23 | NREL 5 MW - MIT-NREL TLP | 3 | 126 | 5000 | MIT-NREL TLP |
| Test 24 | NREL 5 MW - OC3 Hywind | 3 | 126 | 5000 | OC3 Hywind |
| Test 25 | NREL 5 MW - OC4 DeepCwind Semi | 3 | 126 | 5000 | OC4 DeepCwind Semi |

# FAST Primary Input File

## Simulation Control

This section of the input file contains options for controlling the simulation.

### Echo: Echo input data to <RootName>.ech [T/F]

Setting this flag to “true” will result in the primary FAST file being echoed to a file named “<RootName>.ech” where <RootName> is the name of the primary FAST input file, excluding its file extension. This flag is useful for debugging.

### AbortLevel: Error level when simulation should abort [“warning”, “severe”, or “fatal”]

This string tells FAST what error level should cause an abort. Typically we set this to abort on fatal errors, but there may be instances when a user wishes to abort on severe errors or warnings.

Fatal errors are those from which the program cannot recover. For example:

* Running out of memory when trying to allocate space for variables.
* Trying to read a number from a line of an input file that does not contain numeric values.
* Reaching the end of an input file before reading all the necessary information.
* Trying to open a file for writing, but the file is locked from another process.

Some examples of severe errors include the following:

* A format specifier for real numbers is to narrow to print –1.0, so output files will almost certainly contain “\*\*\*” instead of actual numbers.
* When trying to read a numeric value, logical “True” or “False” values were found instead. Fortran interprets them as 0 or 1, but that may not be what the user intended.
* A routine is using math based on the assumption that the angles are small, but the angles it found were larger than what it considers “small.”

### TMax: Total run time [s]

This is the total length of the simulation to be run, in seconds. The first output is calculated at t=0; the last output is calculated at t=TMax seconds.

### DT: Recommended module time step [s]

This is the global, or glue-code, time step, and is the value it will suggest modules use.

Our rule of thumb is to make DT = 1/(10 \* highest natural frequency in the model).

### InterpOrder: Interpolation/Extrapolation order for input/output time history [1 or 2]

This is the order of the interpolation or extrapolation used for inputs in the FAST glue code. Valid entries are “1” for linear interpolation/extrapolation or “2” for quadratic interpolation/extrapolation. Previous inputs are extrapolated at the beginning of each step in the time-advancement loop to provide a guess for the actual inputs at that step. Inputs are typically interpolated in the UpdateStates routines.

We have found that quadratic extrapolation typically works well. However, there are times when linear extrapolation provides a stable solution while quadratic does not. We have found this to be true for cases where the model has poor initial values or cases where the simulation may have errors building up.

### NumCrctn: Number of correction iterations [-]

This is the number of corrections to be taken on each step of the predictor-corrector scheme implemented in FAST. The value of **NumCrctn** must not be negative. Most models can achieve stable solutions by using explicit calculations (i.e., no corrections: **NumCrctn** = 0), particularly if using **InterpOrder** = 2 and the **DT** set to 1/(10 times the highest natural frequency of the model). However, corrections may be needed if you wish to achieve a given accuracy (e.g., if you are using a 4th-order accurate integration scheme, you may get a 2nd-order accurate solution with no corrections. If you want a 4th-order accurate solution, you may need one or more corrections).

### DT\_UJac: Time between calls to get Jacobians [s]

We use a Jacobian matrix to solve the input-output relationship between accelerations and loads in the ElastoDyn-HydroDyn-SubDyn coupling. This Jacobian is computed with computed with finite differences and can be time consuming. However, it rarely needs to be calculated each time step.

**DT\_UJac** determines how often the Jacobian needs to be updated. If the platform reference point doesn't rotate much, **DT\_UJac** can be set to a value larger than **TMax**. **DT\_UJac** is not currently used for land-based systems.

### UJacSclFact: Scaling factor used in Jacobians [-]

This factor is used to divide the magnitude of the load terms in the Jacobian (see **DT\_UJac)** so that they are approximately the same order of magnitude as the acceleration terms. We set it to 1E+06 and have not found any cases where that value did not work.

## Feature Switches and Flags

This section of the input file contains switches and flags that tell FAST which modules should be used in the simulation.

### CompElast: Compute structural dynamics [1 or 2]

1: Use ElastoDyn for all structural dynamics  
2: Use BeamDyn for the structural dynamics on the blades and ElastoDyn for the rest. *This feature is currently disabled. It will be implemented when BeamDyn is integrated into FAST.*

### CompAero: Compute aerodynamic loads [0 or 1]

0: Do not calculate aerodynamic loads  
1: Use AeroDyn

### CompServo: Compute control and electrical-drive dynamics [0 or 1]

0: Do not calculate control and electrical-drive dynamics  
1: Use ServoDyn

### CompHydro: Compute hydrodynamic loads [0 or 1]

0: Do not calculate hydrodynamic loads  
1: Use HydroDyn

### CompSub: Compute sub-structural dynamics [0 or 1]

0: Do not calculate sub-structural dynamics  
1: Use SubDyn

### CompMooring: Compute mooring system [0, 1, or 2]

0: Do not calculate mooring system  
1: Use MAP  
2: Use FEAMooring

Note that FEAMooring is not complete.

### CompIce: Compute ice loads [0, 1, or 2]

0: Do not calculate mooring system  
1: Use IceFloe  
2: Use IceDyn

Note that IceDyn is not complete.

### CompUserPtfmLd: Compute additional platform loading [T/F]

This feature is currently disabled.

### CompUserTwrLd: Compute additional tower loading [T/F]

This feature is currently disabled.

## Input Files

The input files specified in this section can be specified relative to the location of the primary FAST input file or specified with an absolute path. We recommend that you use quotes around the path/file names.

### EDFile: Name of file containing ElastoDyn input parameters [-]

This is the name of the ElastoDyn input file.

### BDBldFile(1): Name of file containing BeamDyn input parameters for blade 1 [-]

This feature is currently disabled.

### BDBldFile(2): Name of file containing BeamDyn input parameters for blade 2 [-]

This feature is currently disabled.

### BDBldFile(3): Name of file containing BeamDyn input parameters for blade 3 [-]

This feature is currently disabled.

### AeroFile: Name of file containing aerodynamic input parameters [-]

This is the name of the AeroDyn input file. This file must be specified *even if* ***CompAero = 0***. This is because ElastoDyn currently reads the AeroDyn input file for its blade discretization. We plan to fix this problem in a future release.

### ServoFile: Name of file containing control and electrical-drive input parameters [-]

This is the name of the ServoDyn input file. It is not used if **CompServo** = 0.

### HydroFile: Name of file containing hydrodynamic input parameters [-]

This is the name of the HydroDyn input file. It is not used if **CompHydro** = 0.

### SubFile: Name of file containing sub-structural input parameters [-]

This is the name of the SubDyn input file. It is not used if **CompSub** = 0.

### MooringFile: Name of file containing mooring system input parameters [-]

This is the name of the MAP (**CompMooring** = 1) or FEAMooring (**CompMooring** = 2) input file. It is not used if **CompMooring =** 0.

### IceFile: Name of file containing ice input parameters [-]

This is the name of the IceFloe (**CompIce** = 1) or IceDyn (**CompIce** = 2) input file. It is not used if **CompIce** **=** 0.

## Output

This section deals with what can be output from a FAST simulation.

### SumPrint: Print summary data to “<RootName>.sum” [T/F]

When set to “true”, FAST will generate a file named “<RootName>.sum”. This summary file contains the version number of all modules being used, the time steps for each module, and information about the channels being written to the time-marching output file(s). If **SumPrint** is “false”, no summary file will be generated.

### SttsTime: Amount of time between screen status messages [s]

During a FAST simulation, the program prints a message like this:

**SttsTime** sets how frequently this message is updated. For example, if **SttsTime** is 2 seconds, you will see this message updated every 2 seconds of *simulation* time.

### DT\_Out: Time step for tabular output [s]

This is time step of the data in the tabular (time-marching) output files. **DT\_Out** must be an integer multiple of **DT.**

### TStart: Time to begin tabular output [s]

This is time step that must be reached before FAST will begin writing data in the tabular (time-marching) output files. Note that the output files may not actually start at **TStart** secondsif **TStart** is not an integer multiple of **DT\_Out.**

### OutFileFmt: Time to begin tabular output [1, 2, or 3]

This indicates which type of tabular (time-marching) output files will be generated. If **OutFileFmt** is 1, only a text file will be written. If **OutFileFmt** is 2, only a binary file will be written. If **OutFileFmt** is 3, both text and binary files will be written.

Text files write a line to the file each time step. This can make the simulation run slower, but it can be useful for debugging, particularly if a simulation doesn’t run to completion or if you want to look at some results before the entire simulation finishes.

Binary files write the entire file at the end of the simulation. If a lot of output channels are requested for a long simulation, this can take up a moderate amount of memory. However, they tend to run faster and the resulting files take up much less space. Binary files keep more precise values than text files (which are limited by the chosen output format specifier).

We recommend you use text files for debugging and binary files for production work.

### TabDelim: Use tab delimiters in text tabular output file? [T/F]

When **OutFileFmt** = 1 or 3, setting **TabDelim** to “true” will put tabs between columns in the text tabular output file. Otherwise, spaces will separate columns in text tabular output file. If **OutFileFmt** = 2, **TabDelim** has no effect.

### OutFmt: Format used for text tabular output, excluding the time channel [-]

When **OutFileFmt** = 1 or 3, FAST will use **OutFmt** to formatthe channels printed in the text tabular output file. **OutFmt** should result in a field that is 10 characters long (channel headers are 10 characters long, and NWTC post-processing software sometimes assume 10 characters.) The time channel is printed using the “F10.4” format. We commonly specify **OutFmt** to be “ES10.3E2”.

If **OutFileFmt** = 2, **OutFmt** has no effect.

# Modeling Tips

If a model is unstable, you can try these steps

* Add a correction step (**NumCrctn**).
* Make **DT** smaller*.*
* Change **InterpOrder**.
* Set better initial conditions in the module input files (particularly ElastoDyn).

# Limitations

We are working to add the features of FAST v7.02.00d-bjj into v8.

Table 1 shows a comparison of features, including the limitations of FAST v8.08.00c-bjj.

Please note that FAST v8.08. 00c-bjj runs slower than FAST v7.02.00d-bjj (though the offshore cases run significantly faster than FAST v8.03.02b-bjj). We have put our effort into getting the framework to work and will address computational efficiency later. We expect great improvements in efficiency as development continues.

# Future Work

All future developments of FAST will follow the framework.

* Items from the “FAST Primary Input File

## Simulation Control

This section of the input file contains options for controlling the simulation.

### Echo: Echo input data to <RootName>.ech [T/F]

Setting this flag to “true” will result in the primary FAST file being echoed to a file named “<RootName>.ech” where <RootName> is the name of the primary FAST input file, excluding its file extension. This flag is useful for debugging.

### AbortLevel: Error level when simulation should abort [“warning”, “severe”, or “fatal”]

This string tells FAST what error level should cause an abort. Typically we set this to abort on fatal errors, but there may be instances when a user wishes to abort on severe errors or warnings.

Fatal errors are those from which the program cannot recover. For example:

* Running out of memory when trying to allocate space for variables.
* Trying to read a number from a line of an input file that does not contain numeric values.
* Reaching the end of an input file before reading all the necessary information.
* Trying to open a file for writing, but the file is locked from another process.

Some examples of severe errors include the following:

* A format specifier for real numbers is to narrow to print –1.0, so output files will almost certainly contain “\*\*\*” instead of actual numbers.
* When trying to read a numeric value, logical “True” or “False” values were found instead. Fortran interprets them as 0 or 1, but that may not be what the user intended.
* A routine is using math based on the assumption that the angles are small, but the angles it found were larger than what it considers “small.”

### TMax: Total run time [s]

This is the total length of the simulation to be run, in seconds. The first output is calculated at t=0; the last output is calculated at t=TMax seconds.

### DT: Recommended module time step [s]

This is the global, or glue-code, time step, and is the value it will suggest modules use.

Our rule of thumb is to make DT = 1/(10 \* highest natural frequency in the model).

### InterpOrder: Interpolation/Extrapolation order for input/output time history [1 or 2]

This is the order of the interpolation or extrapolation used for inputs in the FAST glue code. Valid entries are “1” for linear interpolation/extrapolation or “2” for quadratic interpolation/extrapolation. Previous inputs are extrapolated at the beginning of each step in the time-advancement loop to provide a guess for the actual inputs at that step. Inputs are typically interpolated in the UpdateStates routines.

We have found that quadratic extrapolation typically works well. However, there are times when linear extrapolation provides a stable solution while quadratic does not. We have found this to be true for cases where the model has poor initial values or cases where the simulation may have errors building up.

### NumCrctn: Number of correction iterations [-]

This is the number of corrections to be taken on each step of the predictor-corrector scheme implemented in FAST. The value of **NumCrctn** must not be negative. Most models can achieve stable solutions by using explicit calculations (i.e., no corrections: **NumCrctn** = 0), particularly if using **InterpOrder** = 2 and the **DT** set to 1/(10 times the highest natural frequency of the model). However, corrections may be needed if you wish to achieve a given accuracy (e.g., if you are using a 4th-order accurate integration scheme, you may get a 2nd-order accurate solution with no corrections. If you want a 4th-order accurate solution, you may need one or more corrections).

### DT\_UJac: Time between calls to get Jacobians [s]

We use a Jacobian matrix to solve the input-output relationship between accelerations and loads in the ElastoDyn-HydroDyn-SubDyn coupling. This Jacobian is computed with computed with finite differences and can be time consuming. However, it rarely needs to be calculated each time step.

**DT\_UJac** determines how often the Jacobian needs to be updated. If the platform reference point doesn't rotate much, **DT\_UJac** can be set to a value larger than **TMax**. **DT\_UJac** is not currently used for land-based systems.

### UJacSclFact: Scaling factor used in Jacobians [-]

This factor is used to divide the magnitude of the load terms in the Jacobian (see **DT\_UJac)** so that they are approximately the same order of magnitude as the acceleration terms. We set it to 1E+06 and have not found any cases where that value did not work.

## Feature Switches and Flags

This section of the input file contains switches and flags that tell FAST which modules should be used in the simulation.

### CompElast: Compute structural dynamics [1 or 2]

1: Use ElastoDyn for all structural dynamics  
2: Use BeamDyn for the structural dynamics on the blades and ElastoDyn for the rest. *This feature is currently disabled. It will be implemented when BeamDyn is integrated into FAST.*

### CompAero: Compute aerodynamic loads [0 or 1]

0: Do not calculate aerodynamic loads  
1: Use AeroDyn

### CompServo: Compute control and electrical-drive dynamics [0 or 1]

0: Do not calculate control and electrical-drive dynamics  
1: Use ServoDyn

### CompHydro: Compute hydrodynamic loads [0 or 1]

0: Do not calculate hydrodynamic loads  
1: Use HydroDyn

### CompSub: Compute sub-structural dynamics [0 or 1]

0: Do not calculate sub-structural dynamics  
1: Use SubDyn

### CompMooring: Compute mooring system [0, 1, or 2]

0: Do not calculate mooring system  
1: Use MAP  
2: Use FEAMooring

Note that FEAMooring is not complete.

### CompIce: Compute ice loads [0, 1, or 2]

0: Do not calculate mooring system  
1: Use IceFloe  
2: Use IceDyn

Note that IceDyn is not complete.

### CompUserPtfmLd: Compute additional platform loading [T/F]

This feature is currently disabled.

### CompUserTwrLd: Compute additional tower loading [T/F]

This feature is currently disabled.

## Input Files

The input files specified in this section can be specified relative to the location of the primary FAST input file or specified with an absolute path. We recommend that you use quotes around the path/file names.

### EDFile: Name of file containing ElastoDyn input parameters [-]

This is the name of the ElastoDyn input file.

### BDBldFile(1): Name of file containing BeamDyn input parameters for blade 1 [-]

This feature is currently disabled.

### BDBldFile(2): Name of file containing BeamDyn input parameters for blade 2 [-]

This feature is currently disabled.

### BDBldFile(3): Name of file containing BeamDyn input parameters for blade 3 [-]

This feature is currently disabled.

### AeroFile: Name of file containing aerodynamic input parameters [-]

This is the name of the AeroDyn input file. This file must be specified *even if* ***CompAero = 0***. This is because ElastoDyn currently reads the AeroDyn input file for its blade discretization. We plan to fix this problem in a future release.

### ServoFile: Name of file containing control and electrical-drive input parameters [-]

This is the name of the ServoDyn input file. It is not used if **CompServo** = 0.

### HydroFile: Name of file containing hydrodynamic input parameters [-]

This is the name of the HydroDyn input file. It is not used if **CompHydro** = 0.

### SubFile: Name of file containing sub-structural input parameters [-]

This is the name of the SubDyn input file. It is not used if **CompSub** = 0.

### MooringFile: Name of file containing mooring system input parameters [-]

This is the name of the MAP (**CompMooring** = 1) or FEAMooring (**CompMooring** = 2) input file. It is not used if **CompMooring =** 0.

### IceFile: Name of file containing ice input parameters [-]

This is the name of the IceFloe (**CompIce** = 1) or IceDyn (**CompIce** = 2) input file. It is not used if **CompIce** **=** 0.

## Output

This section deals with what can be output from a FAST simulation.

### SumPrint: Print summary data to “<RootName>.sum” [T/F]

When set to “true”, FAST will generate a file named “<RootName>.sum”. This summary file contains the version number of all modules being used, the time steps for each module, and information about the channels being written to the time-marching output file(s). If **SumPrint** is “false”, no summary file will be generated.

### SttsTime: Amount of time between screen status messages [s]

During a FAST simulation, the program prints a message like this:

**SttsTime** sets how frequently this message is updated. For example, if **SttsTime** is 2 seconds, you will see this message updated every 2 seconds of *simulation* time.

### DT\_Out: Time step for tabular output [s]

This is time step of the data in the tabular (time-marching) output files. **DT\_Out** must be an integer multiple of **DT.**

### TStart: Time to begin tabular output [s]

This is time step that must be reached before FAST will begin writing data in the tabular (time-marching) output files. Note that the output files may not actually start at **TStart** secondsif **TStart** is not an integer multiple of **DT\_Out.**

### OutFileFmt: Time to begin tabular output [1, 2, or 3]

This indicates which type of tabular (time-marching) output files will be generated. If **OutFileFmt** is 1, only a text file will be written. If **OutFileFmt** is 2, only a binary file will be written. If **OutFileFmt** is 3, both text and binary files will be written.

Text files write a line to the file each time step. This can make the simulation run slower, but it can be useful for debugging, particularly if a simulation doesn’t run to completion or if you want to look at some results before the entire simulation finishes.

Binary files write the entire file at the end of the simulation. If a lot of output channels are requested for a long simulation, this can take up a moderate amount of memory. However, they tend to run faster and the resulting files take up much less space. Binary files keep more precise values than text files (which are limited by the chosen output format specifier).

We recommend you use text files for debugging and binary files for production work.

### TabDelim: Use tab delimiters in text tabular output file? [T/F]

When **OutFileFmt** = 1 or 3, setting **TabDelim** to “true” will put tabs between columns in the text tabular output file. Otherwise, spaces will separate columns in text tabular output file. If **OutFileFmt** = 2, **TabDelim** has no effect.

### OutFmt: Format used for text tabular output, excluding the time channel [-]

When **OutFileFmt** = 1 or 3, FAST will use **OutFmt** to formatthe channels printed in the text tabular output file. **OutFmt** should result in a field that is 10 characters long (channel headers are 10 characters long, and NWTC post-processing software sometimes assume 10 characters.) The time channel is printed using the “F10.4” format. We commonly specify **OutFmt** to be “ES10.3E2”.

If **OutFileFmt** = 2, **OutFmt** has no effect.

# Modeling Tips

If a model is unstable, you can try these steps

* Add a correction step (**NumCrctn**).
* Make **DT** smaller*.*
* Change **InterpOrder**.
* Set better initial conditions in the module input files (particularly ElastoDyn).
* Limitations” section will be addressed
* Introduce the new BeamDyn module for nonlinear finite-element modeling of blade dynamics
* Upgrade the loose coupling algorithm of the glue code to allow each module to have its own time step
* ElastoDyn will have a separate discretization scheme for the blades and tower (currently ElastoDyn uses AeroDyn’s blade discretization and AeroDyn uses ElastoDyn’s tower discretization)
* Optimize the code, particularly ElastoDyn, so that it runs faster
* Introduce tight coupling
* Introduce operating-point determination and linearization across the coupled aero-hydro-servo-elastic solution
* And much, much, more…

# Converting to FAST v8.08.x

We have created template input files for FAST v8.08.00c-bjj, ElastoDyn v1.00.02b-bjj, and ServoDyn v1.01.01b-bjj. These template files can be found in the Matlab Simulation Toolbox that is now included in the FAST archive: Utilities\SimulationToolbox\ConvertFASTVersions\TemplateFiles

See the “Matlab Conversion Scripts” section below for help in automatically converting input files to the latest version.

## Summary of Changes to Inputs

This section summarizes changes to the primary FAST input file between major releases.

### Changes in FAST v8.08.00c-bjj

The following list describes the differences in the primary FAST input file of FAST v8.08.00c-bjj relative to FAST v8.03.02b-bjj.

* Many variables in the primary FAST input file have been renamed:

|  |  |
| --- | --- |
| FAST v8.03.x Name | **FAST v8.08.x Name** |
| ADFile | AeroFile |
| SrvDFile | ServoFile |
| HDFile | HydroFile |
| SDFile | SubFile |
| MAPFile | MooringFile |
| CompMAP | CompMooring |

* Most of the feature *flags* changed to feature *switches* in the FAST primary input file. Instead of true/false inputs, CompAero, CompServo, CompHydro, CompSub, and CompMooring now require integer inputs.
* Inputs for coupling with modules IceFloe and IceDyn have been added: **CompIce** and **IceFile**
* Several new inputs for future coupling BeamDyn into FAST have been added. These inputs are **CompElast**, **BDBldFile(1)**, **BDBldFile(2)**,and **BDBldFile(3)**.
* The modules ElastoDyn, ServoDyn, AeroDyn, HydroDyn, and SubDyn allow users to input the string “default” for their respective time steps, which will then use the time step from the primary FAST input file.

### Changes in FAST v8.03.02b-bjj

The following list describes the differences in the FAST, ElastoDyn, and ServoDyn input files of FAST v8.03.02b-bjj relative to the input files of FAST v7.02.00d-bjj.

* The primary FAST input file has been converted to primary input files for FAST, ElastoDyn, and ServoDyn and some of the inputs have been reordered.
* The FAST Platform file has been eliminated, with some of the inputs now part of the ElastoDyn primary input file and some of the inputs now part of HydroDyn’s and MAP’s input files.
* All of the inputs formerly labeled “[CURRENTLY IGNORED]” have been removed.
* Switches for ADAMS preprocessing and linearization have been removed.
* Noise has been removed.
* PtfmLdMod has been converted to CompUsrPtfmLd.
* TwrLdMod has been converted to CompUserTwrLd.
* The tip-brake inputs have been removed.
* PtfmCM is now PtfmCMzt, with PtfmCMzt = -PtfmCM.
* Corresponding inputs PtfmCMxt and PtfmCMyt have been added.
* PtfmRef is now PtfmRefzt, with PtfmRefzt = -PtfmRef.
* TwrRBHt and TwrDraft have been replaced with TowerBsHt, with   
  TowerBsHt = TwrRBHt – TwrDraft.
* The output decimation factor (DecFact) has been converted to DT\_out (DT\_out = DT\*DecFact).
* The yaw and pitch maneuvers no longer specify end times for the maneuvers. Instead they specify a rate for the maneuver.
* The GBRevers variable has been removed; input GBRatio must now be specified as a negative number if GBRevers was previously set to TRUE.
* ElastoDyn’s blade input properties table no longer specifies AeroCent. Instead, it specifies the location of the pitch axis, PitchAxis, which is calculated as PitchAxis = 0.5 – AeroCent by the MATLAB conversion script; the aerodynamic center will become part of AeroDyn in a future release.
* The OutList variables have been divided among the various FAST modules, and several outputs are no longer valid.
* The GH Bladed Interface is now a standard option in ServoDyn, without requiring a recompile.
* Tower drag loading has been added to AeroDyn v14.02.00c-mlb with a new corresponding flag in the AeroDyn input file.
* The glue code allows options for AbortErrLevel, number of corrections in the predictor-corrector algorithm, and extrapolation/interpolation order of module inputs to be used for time advancement.
* For the ElastoDyn coupling to HydroDyn or SubDyn, FAST also has two inputs controlling the implicit solve (via Jacobian computed with finite differences). UJacSclFact is used to scale the loads from HydroDyn or SubDyn so that they are roughly the same of magnitude as the accelerations from ElastoDyn; DT\_UJac determines how often the Jacobian needs to be updated (if the platform reference point doesn't rotate much, DT\_UJac can be set to a value larger than TMax).

## Matlab Conversion Scripts

Because the changes to the input files are significant, we have created Matlab scripts to automatically convert FAST v7.x input files or FAST 8.03.02b-bjj input files to FAST v8.08.00c-bjj. The files you will need are included in the Simulation Toolbox, located in this directory of the FAST archive: Utilities\SimulationToolbox\ConvertFASTVersions

We recommend that you add the Simulation Toolbox to your Matlab path so that you can access all of the routines defined in it. For example:

FASTSimulationToolbox = 'C:\Users\bjonkman\FAST\UtilityCodes\SimulationToolbox';

addpath( genpath( FASTSimulationToolbox ) );

An example showing how we converted the NREL CertTest input files for use with FAST v8.08.00c-bjj is included in the FAST archive: CertTest\ConvertFiles.m. You can use this script as a basis for helping to convert your own input files; however, we *strongly* recommend that you make copies of all your input files before running any scripts to convert them.

### Converting from FAST v8.03.02b-bjj

added Matlab script to convert FAST/HydroDyn input files from FAST v8.3 to versions in this release

### Converting from FAST v7.x

You will need to provide the conversion routine (ConvertFAST7to8) with the name of the old FAST (v7.x) primary input file and the directory where the new input files should be placed. *The new directory should not be the directory where the old files are located!*

If your input file has pitch or yaw maneuvers, you may also provide the routine with the new rates (instead of the end times previously used). We have also provided a Matlab routine (CalculateYawAndPitchRates) that will calculate these rates, but you must provide the routine the name of the FAST output file that contains the previous results of the Pitch and/or Yaw channels.

If your input file was used with the custom interface to GH Bladed DLL controllers, you should also set the optional input parameter, usedBladedDLL, so that your input switches that previously called the DLL are now set to 5, the new switch for User-Defined Control from Bladed DLL.

Please note that the hydrodynamic- and mooring system-related inputs are not automatically converted.

## Model Time Steps

Please note that due to some of the changes in the coupling scheme of FAST v8, you may need to change the time steps in your existing models.

(bjj: is this still true?)

# Compiling

If you want to compile the code, please see the compiling folder in the FAST archive.

Unlike FAST v7 distributions, all of the source code you need to compile the project is contained in the archive’s Source directory, allowing us to distribute a Microsoft Visual Studio project file that contains links to all of the files needed to compile FAST v8.08.00c-bjj.

The compiling folder also contains a Windows® batch script that can be run from your Intel Fortran Command Prompt Window, with very little (if any) modification. Both the Visual Studio project and batch file run the FAST Registry and are set up to compile and link with all of the appropriate settings. (Note that there are several specialized compiling/linking options and that MAP is distributed as a dynamic-link library.)

The text files (.txt) in the source folders are input files for the FAST Registry. These files are used to generate the \*\_Types.f90 files for the component modules. The Compiling\Compile\_FAST.bat file and Compiling\VisualStudio\RunRegistry.bat files will run the Registry if necessary.

Compiling options:

+ added Registry source files and gcc makefile so Registry can be compiled on non-Windows systems

+ added makefile and source files to create an empty MAP dll, so that FAST can be compiled and run on other systems (except for floating models that use MAP)

--------------------------------------------

Changes in compiling scripts:

--------------------------------------------

+ added makefile, which uses LAPACK and BLAS libraries

+ Added compiling options:

-- Large Address Aware switch

-- increased stack reserve size

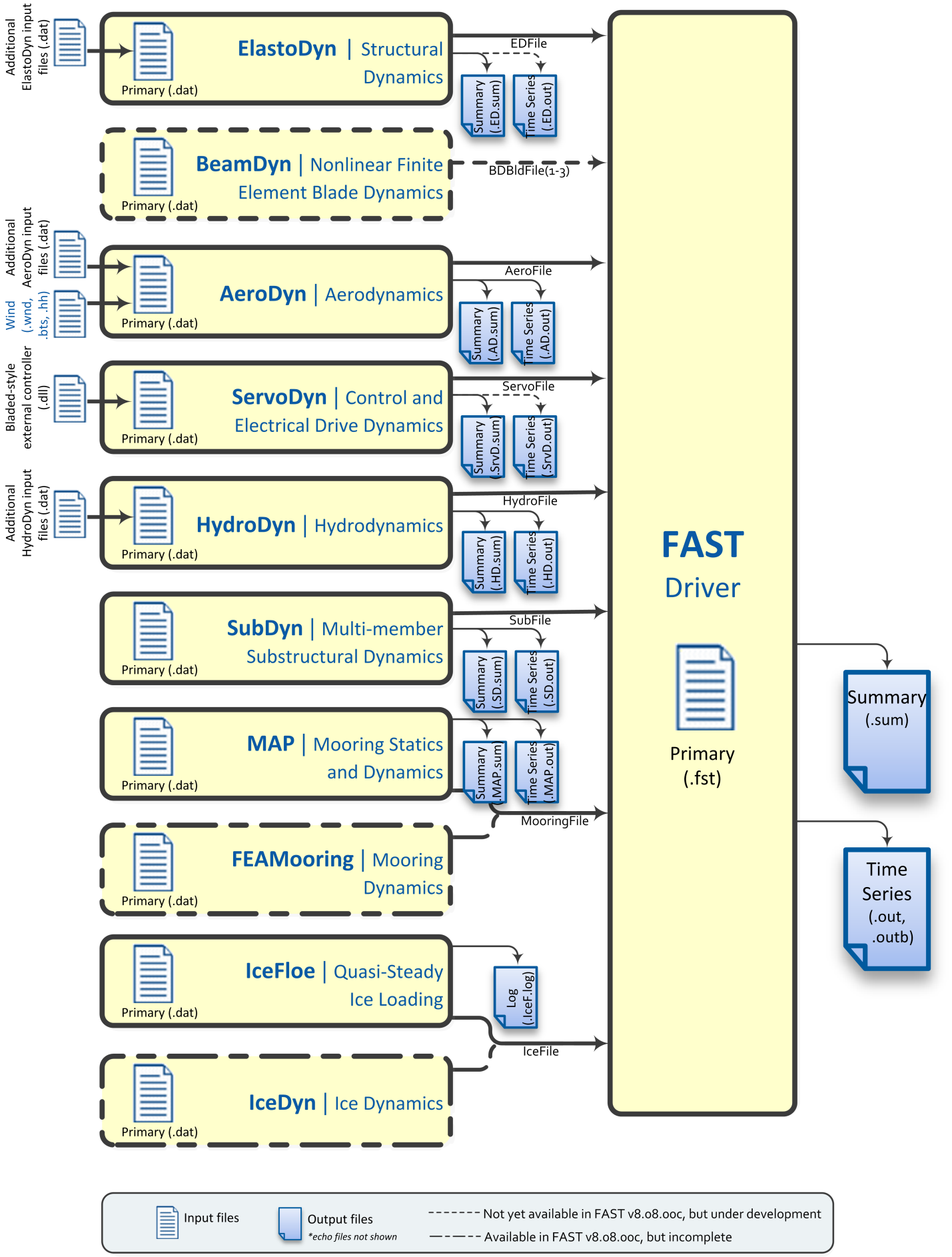


Figure : Summary of Input and Output Files for FAST v8.08.00c-bjj

1. This option is a custom feature in FAST v7.02.x, requiring a separate executable. In FAST v8.08, it is part of the standard distribution. [↑](#footnote-ref-1)
2. These steps must be integer multiples of the structural time step. [↑](#footnote-ref-2)
3. These steps must be integer divisors of the glue-code time step. Future versions will allow integer multiples of the glue-code time step as well. [↑](#footnote-ref-3)
4. FAST v7.02.x is limited to one correction step and this correction step only applies to some modules. [↑](#footnote-ref-4)
5. The 64-bit version of FAST v8.08 does not contain the ability to use the MAP module. [↑](#footnote-ref-5)
6. The FAST v8.08 source code can be compiled using gfortran, however the offshore cases do not run with this compiled executable. We are working to find the problem and fix it. [↑](#footnote-ref-6)
7. IceDyn and FEAMooring have been added to FAST v8.08.00c-bjj, but they are not complete and have not been tested well. [↑](#footnote-ref-7)
8. The FAST Registry reads input files from each module to auto-generate the \*\_Types.f90 files. [↑](#footnote-ref-8)