Handbook for FAST-WInDS v8.10 (Draft)

Shujian Liu
shujian@umass.edu

Department of Mechanical and Industrial Engineering
University of Massachusetts Amherst

Last edited August 13, 2015

Contents

Part I

Aerodynamic Theory

Chapter 1

Introduction

The Wake Induced Dynamics Simulator (WInDS) is developed by Dr. Thomas Sebastian and Prof. Matthew Lackner at University of Massachusetts Amherst, and updated by Prof. Matthew Lackner, Nathaniel deVelder, Evan Gaertner, Shujian Liu, Andrew Sciotti and researchers in University of Stuttgart, including Friedemann Beyer and David Lenz.

WInDS is a free vortex wake method (FVM) code and based lifting-line theory (LLT). An overall description of vortex method for wind turbine from review paper [4]:

> *In vortex models the rotor blades, trailing and shed vorticity in the wake are represented by lifting lines or surfaces. On the blades the vortex strength is determined from the bound circulation that stems from the amount of lift created locally by the flow past the blades. The trailing wake is generated by the spanwise variation of the bound circulation while the shed wake is generated by a temporal variation and ensures that the total circulation over each section along the blade remains constant in time. Knowing the strength and position of the vortices the induced velocity can be found in any point using the BiotSavart law. In some models (namely the lifting-line models) the bound circulation is found from airfoil data table-look up just as in the BEM method. The inflow is determined as the sum of the induced velocity, the blade velocity and the undisturbed wind velocity... The relationship between the bound circulation and the lift is denoted as the Kutta-Joukowski theorem...*

Chapter 2

Theory behind WInDS

This document is not intended to present the detailed aerodynamic theory of WInDS or free vortex method. All the details can be found in the following references:

- Basic theory behind WInDS can be found in Chapter 5 of Dr. Thomas Sebastian's dissertation[1] [13].
- Wind turbine tower interference is described in [8].
- Ground effects and acceleration of WInDS via graphics processing unit (GPU) are studied by Nathaniel deVelder in his thesis[2] [2].
- Dynamic stall for wind turbine is presented in Evan Gaertner's thesis[3] [3].
- Parallelized Treecode algorithm is modified from Prof. Hans Johnston[4].
- Modification to freeze and cutoff the far wake by researchers in University of Stuttgart and Andrew Sciotti.

More detailed aerodynamic theory can be found in related research books [1, 7, 9].

[1]Download link: http://scholarworks.umass.edu/open_access_dissertations/516
[2]Download link: http://scholarworks.umass.edu/theses/1176/
[3]Download link: http://scholarworks.umass.edu/masters_theses_2/85/
[4]Download link: http://people.math.umass.edu/ johnston/newtreecode.html

Part II

Programming Guide

Chapter 1

General

1.1 New modularization framework for the FAST

In order to enable aero-hydro-servo-elastic simulation for wind turbine, WInDS is integrated with FAST(v8.10.00a-bjj)/AeroDyn (v14.03.01a-bjj) in present research.

The WInDS is rewritten in Fortran 2003 on the new modularization framework for the FAST [5].

WInDS is made as an alternative(others are BEM and GDW) in the current aerodynamic module of FAST: AeroDyn [12].

1.2 Data Structure

The data structure in WInDS is usually 5D arrays:

1st rank: dimension index (1-3),

2nd rank: time index,

3rd rank: stored previous time-steps,

4th rank: radial index,

5th rank: blade index.

Chapter 2

"Big picture"

The original algorithmic structure of AeroDyn is shown in Algorithm 1.

Algorithm 1: Original Algorithm Structure of AeroDyn

```
  Data: Turbine position, velocity and orientation
  Result: Aerodynamic loads
1 begin
2     Subroutine AD_Init - to initialize module;
3     for all timesteps do
4         Subroutine AD_CalcOutput - to calculate the aerodynamic loads;
5     Subroutine AD_End - to terminate the module;
```

The algorithmic structure of AeroDyn with WInDS is shown in Algorithm 2.

Algorithm 2: Algorithm Structure of AeroDyn with WInDS

```
1  begin AD_Init
2      Original code in AD_Init;
3      Subroutine WInDS_ReadInput;
4      Subroutine WINDS_SetParameters ;
5      Subroutine WINDS_Allocate ;
6      (optional) Subroutine WINDS_Shear_Model ;
7      (optional) Subroutine WINDS_Ground_Model ;
8      (optional) Subroutine LB_Initialize_AirfoilData (optional) Subroutine
       Initialize_Paraview_Files ;
9      (optional) Subroutine Write_Treecode ;
10     (optional) Subroutine Write_KJ ;
11 for all timesteps do
12     begin AD_CalcOutput
13         Original code in AD_CalcOutput;
14         if use WInDS then
15             if 1st global timestep then
16                 Subroutine WINDS_check_cutoff;
17                 Subroutine WINDS_Kinematics;
18                 Subroutine WINDS_Velocity;
19                 Subroutine WINDS_FVMInitial (see Algorithm 3) ;
20             else
21                 if global timestep used by WInDS then
22                     Subroutine WINDS_Kinematics;
23                     Subroutine WINDS_Velocity;
24                     Subroutine WINDS_FVM (see Algorithm 4);
25                 else
26                     Copy the loads from the previous global timestep;
```

The algorithmic structure of WINDS_FVMInitial is shown in Algorithm 3

Algorithm 3: Algorithm Structure of WINDS_FVMInitial

```
1 begin WINDS_FVMInitial
2     Make inflow to be steady;
3     Subroutine BEM ;
4     Define initial vortex strength;
5     (Optional) Output First Timestep to Paraview file;
```

The algorithmic structure of WINDS_FVM is shown in algorithm 4.

Algorithm 4: Algorithm Structure of WINDS_FVM

```
begin WINDS_FVM
    Subroutine CUR_2_PREV;
    Subroutine SHEAR ;
    if not last timesetp then
        Subroutine NUM_ADVECT ;
    Subroutine UPDATE_WAKE ;
    Subroutine VCORE ;
    Subroutine KuttaJoukowski ;
    Subroutine Aero_Loads ;
    (Optional) Subroutine CreateVTUembedded ;
```

Chapter 3

Subroutines

If the subroutine name starts with "WINDS_", it is directly called from AeroDyn subroutine. Others are called insides WInDS.

Note: some of the explanations come from the original Matlab WInDS source code or *Wake Induced Dynamics Simulator (WInDS) version 0.9 Theory & User Manual*.

3.1 Main Subroutines for Calculation (WINDS.f90)

Main subroutines for calculation are written in WINDS.f90.

3.1.1 WINDS_SetParameters

This subroutine is called by SUBROUTINE AD_Init. It sets the constants of Ramasamy-Leishman vortex model and some other parameters such as air density and gravity.

3.1.2 WINDS_Allocate

This subroutine is called by SUBROUTINE AD_Init. It allocates the variable arrays.

3.1.3 WINDS_check_cutoff

This subroutine handles the variable to freeze and cutoff the far wake.

This feature was originally added to Matlab code by researchers in University of Stuttgart.

3.1.4 WINDS_Ground_Model

This subroutine allocates variable arrays for ground effects and calculates variables for ground model.

Theory is in Chapter 3 of Nathaniel deVelder's thesis [2].

3.1.5 WINDS_Kinematics

This subroutine computes the station locations of blades in the inertial coordinate system.

An illustration of blade element is shown in Figure 3.1.

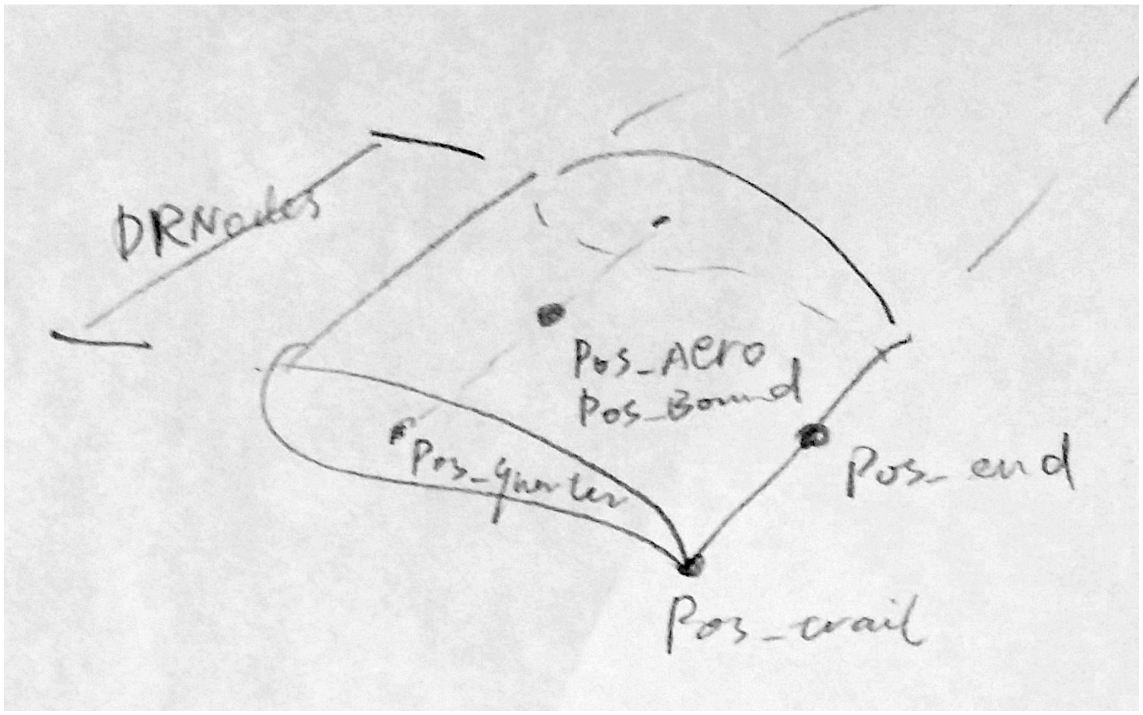


Figure 3.1: An illustration of blade element

3.1.6 WINDS_Velocity

This subroutine computes the velocity contributions due to turbine and platform motions and freestream flow in the inertial and blade coordinate systems.

Theory is in Section 5.2 of Thomas Sebastian's dissertation [13].

3.1.7 WINDS_FVMInitial

This subroutine computes initial results for the first timestep via blade element and momentum method.

3.1.8 WINDS_FVM

This subroutine uses free vortex wake method to calculate the aerodynamic loads after first timestep.

3.1.9 WINDS_Shear_Model

This subroutine defines parameters used in shear model.

Theory is in Section 3.1 of Nathaniel deVelder's thesis [2].

3.1.10 Aero_Loads

This subroutine is called in WINDS_FVM and it calculates the aerodynamic loads.

3.1.11 BEM

This subroutine is called in WINDS_FVMInitial and it calculates coefficient of lifting and aerodynamic loads in first time-step via blade element and momentum method. It is modified from the original AeroDyn code.

Theory is in Section 4.2 of Thomas Sebastian's dissertation [13].

3.1.12 BiotSavart

This subroutine is called in InducedVelocity, Induced_Velocity_Ground_Mirror and Induced_Velocity_Ground_Panels. It computes the induced velocity at a point in space because of the influence of defined vortex filaments.

Theory is in Section 4.6.1 and 5.3.2 of Thomas Sebastian's dissertation [13].

3.1.13 CLCD_FVM

This subroutine looks up the coefficient of lifting and drag from airfoil data. It is modified from the original AeroDyn code.

3.1.14 CUR_2_PREV

This subroutine copies wake and vel from "current" timestep to "previous" .

3.1.15 InducedVelocity

This subroutine calculates the induced velocity via Biot-Savart Law.

3.1.16 Induced_Velocity_Ground_Mirror

This subroutine calculates the induced velocity of ground effects (method of images) via Biot-Savart Law.

3.1.17 Induced_Velocity_Ground_Panels

This subroutine calculates the induced velocity of ground effects (panels method) via Biot-Savart Law.

3.1.18 KuttaJoukowski

This subroutine computes the bound vortex filament strength via Kutta-Joukowski theorem, solving via fixed-point iteration.

Theory is in Section 4.6.2.1 and 5.4.3 of Thomas Sebastian's dissertation [13].

3.1.19 NUM_ADVECT

This subroutine numerically convects wake nodes to next timestep.

These are several methods can be used: 2nd-order Runge-Kutta and 4th-order Runge-Kutta.

Theory is in Section 5.4.2 of Thomas Sebastian's dissertation [13].

The stability of those methods are discussed in Section 5.2 of Nathaniel deVelder's thesis [2].

3.1.20 Shear

This subroutine calculates the inflow velocity of all points based on selected shear model

3.1.21 Shear_calc

This subroutine calculates the inflow velocity of one point based on selected shear model

3.1.22 TOWER_SHADOW

Not finished for Fortran version yet...

Theory is described in [8].

3.1.23 UPDATE_WAKE

This subroutine updates bound vortices and latest trailed and shed vortices

3.1.24 VCORE

This subroutine computes the effective vortex filament core size using the Ramasamy-Leishman model and filament stretching.

Theory is in Section 5.3.2 of Thomas Sebastian's dissertation [13].

3.2 Subroutines for dynamic stall (WINDS_DS.f90)

Dynamic stall for wind turbine is presented in Evan Gaertner's thesis [3].

3.2.1 LB_Initialize_Variables

This subroutine initializes the stall structure of recursive dynamic stall model variables

3.2.2 LB_Initialize_AirfoilData

This subroutine computes additional airfoil data needed for the dynamic stall model on initialization.

3.2.3 LB_AfData_spline_fit

This subroutine provides piecewise cubic spline fits to the 2D steady airfoil data and the Kirchhoff-Helmholtz flat plate trailing edge flow model.

3.2.4 LB_DynStall

This subroutine provides the aerodynamic forces at a bound analysis node based on the instantaneous local angle of attack and velocity.

3.3 Subroutines for I/O (WINDS_IO.f90)

Those subroutines are written in WINDS_IO.f90.

3.3.1 WInDS_ReadInput

This subroutine reads the input file of WInDS, checks possible error and sets parameters.

3.3.2 Initialize_paraview_files

This subroutine initializes Paraview files.

3.3.3 CreateVTUembedded

This subroutine writes Paraview file at each WInDS timestep

3.3.4 Close_paraview_files

This subroutine closes Paraview files.

3.3.5 WRITE_Treecode

When user set to record the behavior of Treecode for Biot-Savart Law compared with direct OpenMP acceleration.

It outputs information includes: the number of source vortex filaments, the number of receiver points, CPU time of direct OpenMP calculation, the CPU time of Treecode calculation, the speedup, and relative error.

3.3.6 WRITE_KJ

This subroutine outputs the timestep number, iteration used for calculation and max of D_Gamma at last iteration. It is original intended for debugging.

3.3.7 WInDS_WriteSum

This subroutine write running time, basic settings to output file.

3.4 Subroutines to Accelerate Biot-Savart Law

Several methods are used to accelerate Biot-Savart Law. .

3.4.1 BiotSavart_OpenMP

In WInDS_Acce.f90.

Direct OpenMP acceleration on multi-core machines of Biot-Savart Law.

3.4.2 BiotSavart_Treecode

In WINDS_Treecode.f90.

Treecode algorithm is intended to speedup the Biot-Savart Law. It does direct calculation at from near vortex filaments while Taylor approximation from far filaments. This will reduce from the original computational cost $O(n^2)$ to $O(n \log n)$, or even $O(n)$.

Original parallized Treecode algorithm in Fortran is developed by Prof. Hans Johnston[1].

There are two main modification to the original code:

[1]Download link: http://people.math.umass.edu/ johnston/newtreecode.html

1. Use Vatistas viscous model (with index of 2) for the near filaments.
2. Use Simpsons' rule for far filaments.

3.5 Subroutines in WInDS Library (WINDS_Lib.f90)

At development period, some subroutines are written for debug purpose, e.g. write number or matrix (1D or 2D) to TEXT files.

They are put together in the WInDS_Lib.f90.

Chapter 4

Details

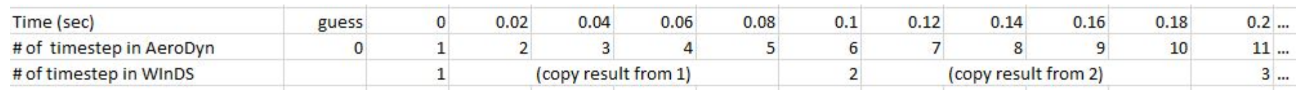
This part is for the readers who have interest to edit the source code.

Firstly and foremostly, please read *NWTC Programmer's Handbook.* [1] **Every detail matters.**

4.1 Timesteps

The timestep duration in AeroDyn is usually very small ($< 0.01sec$). However, WInDS (FVM) need much larger timesteps (usually 0.05 - 0.1 sec) for convergence, as studied in Nathaniel B. deVelder's thesis (section 5.2.2).

One solution is the call WInDS at selected timesteps and for the other timesteps, copy the output from previous selected timestep. In the code, the parameter DT_ratio is used. If DT_ratio is 5 and AeroDyn is 50 hz, WInDS will be 10 hz. The relationship with AeroDyn and WInDS is shown in Figure 4.1.



Time (sec)	guess	0	0.02	0.04	0.06	0.08	0.1	0.12	0.14	0.16	0.18	0.2 ...
# of timestep in AeroDyn	0	1	2	3	4	5	6	7	8	9	10	11 ...
# of timestep in WInDS		1	(copy result from 1)				2	(copy result from 2)				3 ...

Figure 4.1: Timestep in AeroDyn and WInDS

4.2 Other details

- For the unit of angles, Matlab WInDS uses degree, while Fortran code (FAST-WInDS) uses radius.
- Rotor thrust in Matlab WInDS is calculated simply from aerodynamic force. While in FAST, it is related to the hub mass, the tip brake, and aerodynamic force of blade element (according to ElastoDyn.f90 code).
- Aerodynamic center of blade element is set to be the one quarter because of the assumption of lifting-line theory [1].

[1]NWTC Programmer's Handbook (Draft Version for External Review, 2.1 MB, 26-March-2013) by Bonnie Jonkman, John Michalakes, Jason Jonkman, Marshall Buhl, Andy Platt, and Michael Sprague. http://wind.nrel.gov/designcodes/simulators/developers/docs/ProgrammingHandbook_Mod20130326.pdf

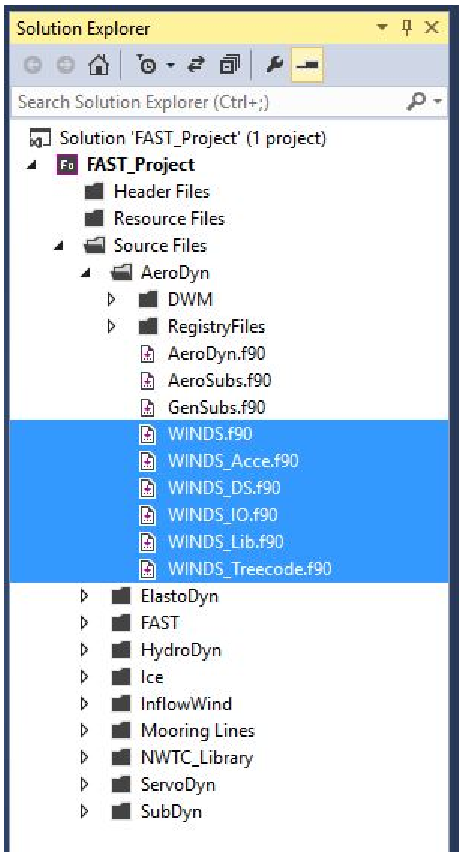
- Since Matlab WInDS uses interpolated twist angle and trail chord to calculate trailing nodes positions, while the present work interpolates element nodes positions to obtain trailing nodes positions, there is slight difference of the lifting coefficient between them. However, the differences are usually less than 1% in some simple comparisons. The largest differences often occurs at the tip elements, however, these elements have relatively small area and contribute small force to the total. Hence it is reasonable to deal in this way.
- WInDS needs to read in the stall angle while original AeroDyn doesn't.
- The AeroDyn defines types in a separate Text file and generate the Fortran code automatically upon compiling. In order to make future development convenient, the types for WInDS are usually grouped together.
- Currently, WInDS doesn't support multiple columns of airfoil table.

Chapter 5

Compile FAST_v8.10 + WInDS

5.1 In Windows (using Intel Visual Fortran Composer XE 2013 SP1)

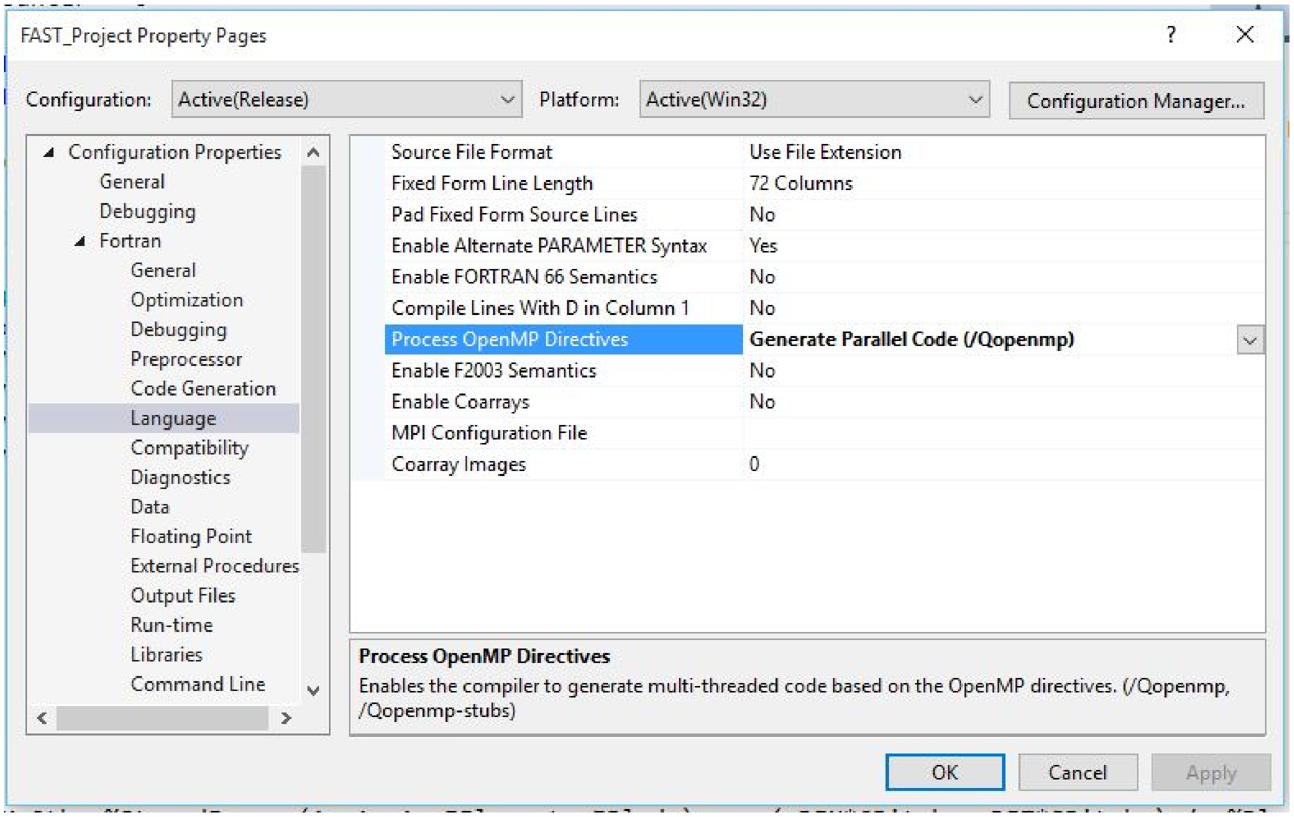
The source code of WInDS is added into "Solution Explorer" (Figure 5.1).



```
Solution 'FAST_Project' (1 project)
◢ FAST_Project
    Header Files
    Resource Files
  ◢ Source Files
    ◢ AeroDyn
      ▷ DWM
      ▷ RegistryFiles
        AeroDyn.f90
        AeroSubs.f90
        GenSubs.f90
        WINDS.f90
        WINDS_Acce.f90
        WINDS_DS.f90
        WINDS_IO.f90
        WINDS_Lib.f90
        WINDS_Treecode.f90
    ▷ ElastoDyn
    ▷ FAST
    ▷ HydroDyn
    ▷ Ice
    ▷ InflowWind
    ▷ Mooring Lines
    ▷ NWTC_Library
    ▷ ServoDyn
    ▷ SubDyn
```

Figure 5.1: Added files to FAST

In the Microsoft Visual Studio development environment, the project property has been changed to use OpenMP in both Debug and Release modes (Figure 5.2).



Configuration: Active(Release) Platform: Active(Win32) Configuration Manager...

Source File Format	Use File Extension
Fixed Form Line Length	72 Columns
Pad Fixed Form Source Lines	No
Enable Alternate PARAMETER Syntax	Yes
Enable FORTRAN 66 Semantics	No
Compile Lines With D in Column 1	No
Process OpenMP Directives	**Generate Parallel Code (/Qopenmp)**
Enable F2003 Semantics	No
Enable Coarrays	No
MPI Configuration File	
Coarray Images	0

Process OpenMP Directives
Enables the compiler to generate multi-threaded code based on the OpenMP directives. (/Qopenmp, /Qopenmp-stubs)

Figure 5.2: Change of FAST project property

Since the Registry-AD.txt has been modified, it is necessary to use "Rebuild Solution" instead of "Build Solution" to compile (Figure 5.3).

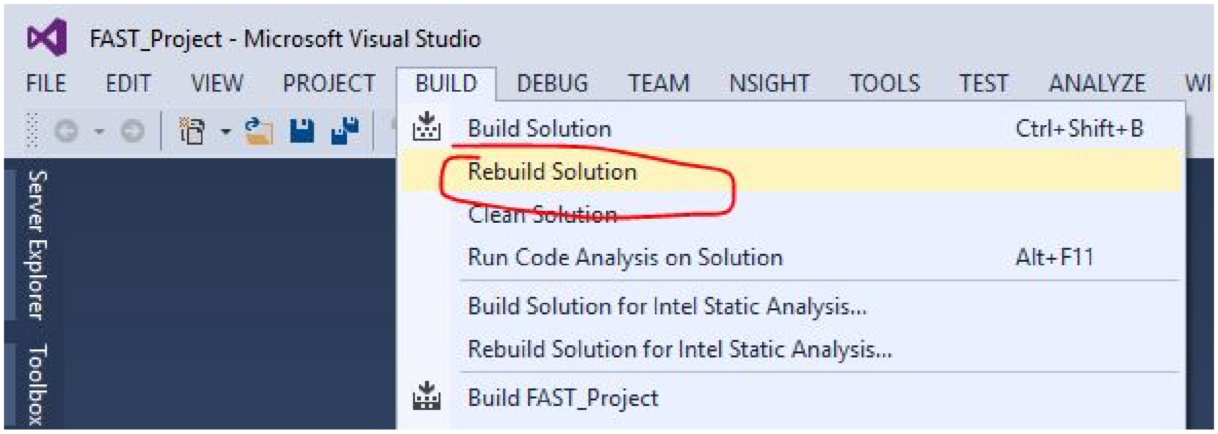

FAST_Project - Microsoft Visual Studio
FILE
EDIT
VIEW
PROJECT
BUILD
DEBUG
TEAM
NSIGHT
TOOLS
TEST
ANALYZE
WI
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Run Code Analysis on Solution Alt+F11
Build Solution for Intel Static Analysis...
Rebuild Solution for Intel Static Analysis...
Build FAST_Project
Server Explorer
Toolbox

Figure 5.3: Change of FAST project property

If 32-bit is used, "FAST_dev_Win32.exe" will be generated for release mode and "FAST_dev_Debug_Win32.exe" will be generated for debug mode in “bin” folder.

After adding WInDS into FAST, the code will need OpenMP dynamic library (libiomp5md.dll) to run. It is not a problem for developers whose computers have Intel compiler. However, for users without Intel compiler, they need to download the redistributable libraries [1].

[1]Intel link: https://software.intel.com/en-us/articles/redistributable-libraries-for-intel-c-and-visual-fortran-composer-xe-2013-sp1-for-windows

5.2 In Linux/Unix on MGHPCC

Compiling FAST on Linux/Unix may be different on different systems/compilers/settings. The following instructions may only work on Massachusetts Green High Performance Computing Center (MGHPCC).

5.2.1 Original FAST for Bladed-style DLL controller

There is an issue in SysGnuLinux.f90, this section will compile the FAST with Bladed-style DLL controller. This version of FAST is usually used for NREL 5 MW wind turbines.

Preparations:

1. Download: FAST_v8.10.00a-bjj.tar.gz [2] and MAP_v1.10.0rc.tar_0.gz [3] from NREL website.
2. After login MGPHCC, load the compilers:

```
module load intel/composer_xe_2015_sp1.0.080
module load intel/mkl_libraries_from_composer_xe_2013_sp1
```

3. Unzip FAST, and unzip MAP in /Source/dependencies/MAP in FAST directory.

```
mkdir FAST
tar -zxvf FAST_v8.10.00a-bjj.tar.gz -C FAST/
tar -zxvf MAP_v1.10.0rc.tar_0.gz -C ./FAST/Source/dependencies/MAP/
```

Notice: ifort 2013 has issues to compile FAST, and I can only compile 64-bit FAST on MGHPCC.

Step 1. Compile MAP with icc

```
$ cd FAST/Source/dependencies/MAP/src/
$ vi makefile
```

Modify the file to be:

```
BITS = -m64
PLATFORM = Linux #$(shell uname -s)
# CMINPACK_DIR = cminpack
# BSTRING_DIR = bstring
# SIMCLIST_DIR = simclist
# LAPACK_DIR = lapack

VPATH = cminpack:simclist:bstring:lapack


#ifeq ($(PLATFORM),Darwin)
```

[2]Download link: https://nwtc.nrel.gov/FAST8
[3]Download link: https://nwtc.nrel.gov/MAP

```
#  CC_TOOLS = clang
#  CFLAGS   = -g -O1 -fsanitize=address -fno-omit-frame-pointer -fPIC -D DEBUG -Icminpack
    -Isimclist
#  LDFLAGS  = -g -fsanitize=address -fno-omit-frame-pointer -dynamiclib
#else ifeq ($(PLATFORM),Linux)
  CC_TOOLS = icc
  #CFLAGS   = -g -O1 -fsanitize=address -fno-omit-frame-pointer -fPIC -D DEBUG
      -DGITVERSION=\"$(GIT_VERSION)\" -Icminpack -Isimclist
  CFLAGS   = -g -fPIC -std=c99 -Wuninitialized -Wall #-D DEBUG
      -DGITVERSION=\"$(GIT_VERSION)\"
  #CFLAGS   = -fPIC $(BITS) -g -std=c99
  #LDFLAGS  = $(BITS) -g -shared -llapacke
  LDFLAGS  = -fPIC $(BITS) -g -shared -mkl=squential -lmkl_lapack95_lp64
#endif

# GIT_VERSION := $(shell git describe --abbrev=4 --dirty --always)
```

Also:

```
all : $(OBJ)
   $(CC_TOOLS) $(LDFLAGS) -o libmap-1.10.00.so $(DEBUG) $(OBJ) #-lm -llapacke
```

Then compile and copy

```
$ make
$ cp libmap-1.10.00.so ../MAP.so
```

Step 2. Compile Registry with gcc

In /Source/dependencies/Registry of FAST directory, compile Registry.

```
$ cd ../../Registry/
$ make
```

Then “registry.exe” will be in /Source/dependencies/

Step 3. Compile FAST with ifort

```
$ cd ../NWTC_Library/
```

Replace SysGnuLinux.f90 with:

https://wind.nrel.gov/forum/wind/viewtopicphp?f=4&t=642&start=120# p5673

```
$ cd ../../../Compiling/
$ vi makefile
```

Change makefile to be:

```
    # 32-bit or 64-bit?
  #BITS = 32
  BITS = 64
```

And:

```
#ifeq ($(OS),Windows_NT)
#   Registry    = $(BIN_DIR)/Registry_win32.exe
#   MAP_lib     = $(MAP_DIR)/MAP_win32.lib
#   LAPACK_LINK = -llapack -lblas -LC:/LAPACK/win32
#else
  Registry    = registry.exe
  MAP_lib     = $(MAP_DIR)/MAP.so
  LAPACK_LINK = # -llapack -lblas
#endif


  # Name of compiler to use and flags to use.
#FC     = gfortran

#FFLAGS = -O2 -m$(BITS) -fbacktrace -ffree-line-length-none -x f95-cpp-input
#LDFLAGS = -O2 -m$(BITS) -fbacktrace -Wl,--stack=999999999,--large-address-aware
FC     = ifort
F90_FLAGS     =      -fpp -threads -O2 -mkl:sequential -finline-functions -xhost
F77_FLAGS     =      -fpp -threads -O2 -mkl:sequential -finline-functions -xhost
LDFLAGS       =      -fpp -threads -O2 -I$(MKLROOT)/include -mkl:sequential
    -finline-functions -xhost
```

Then:

```
    # Destination and RootName for executable

  OUTPUT_NAME = FAST
  DEST_DIR   = ../bin
```

Then:

```
  # General rules for compliling the files.

#%.obj: %.f90
#      $(FC) -I $(INTER_DIR) $(FFLAGS) -c $< -o $(INTER_DIR)/$@ -J $(INTER_DIR) -B
    $(INTER_DIR)
#
#%.obj: %.F90
#      $(FC) -I $(INTER_DIR) $(FFLAGS) -c $< -o $(INTER_DIR)/$@ -J $(INTER_DIR) -B
    $(INTER_DIR)

#bjj: what is $(F77)????
#%.obj: %.f
#      $(F77) -I $(INTER_DIR) $(FFLAGS) -c $< -o $(INTER_DIR)/$@ -J $(INTER_DIR) -B
    $(INTER_DIR)
```

```
%.obj: %.f90
       $(FC) -I $(INTER_DIR) $(F90_FLAGS) -c $< -o $(INTER_DIR)/$@ -module $(INTER_DIR) -B
           $(INTER_DIR)

%.obj: %.F90
       $(FC) -I $(INTER_DIR) $(F90_FLAGS) -c $< -o $(INTER_DIR)/$@ -module $(INTER_DIR) -B
           $(INTER_DIR)

#bjj: what is $(F77)????
%.obj: %.f
       $(FC) -I $(INTER_DIR) $(F77_FLAGS) -c $< -o $(INTER_DIR)/$@ -module $(INTER_DIR) -B
           $(INTER_DIR)
```

Make the folder for FAST

```
 $ mkdir ../bin
```

And compile:

```
$ make
```

It will take a while. The FAST executable file will be in folder named “bin”.

Step 4. Compile controller

I can only use gfortran to compile the controllers On MGPHCC.

Open the compile file for Bladed-style DLL controller:

```
 $ vi makefile_DISCON_DLL
```

Modify it to be:

```
   # 32-bit or 64-bit?
#BITS = 32
BITS = 64


   # Location of source files for the DLL.
   # You may need to change these for your DLL.

DLL_DIR    = ../CertTest/5MW_Baseline/ServoData/Source/

SOURCE_FILE = DISCON.f90


   # Name of compiler to use and flags to use.
FC     = gfortran
#FFLAGS = -O2 -m$(BITS) -fbacktrace -ffree-line-length-none -x f95-cpp-input -C
#LDFLAGS = -shared -O2 -m$(BITS) -fbacktrace
FFLAGS = -fPIC -O2 -m$(BITS) -fbacktrace -ffree-line-length-none -x f95-cpp-input #-C
```

```
LDFLAGS = -fPIC -shared -O2 -m$(BITS) -fbacktrace
```

Compile Bladed-style DLL controller:

```
 $ make -f makefile_DISCON_DLL
```

The controller will be DISCON_glin64.so. Move it to:

```
 $ cp DISCON_glin64.so ../CertTest/5MW_Baseline/ServoData/Source/
```

To compile other two controllers:

```
 $ cp makefile_DISCON_DLL makefile_DISCON_ITIBarge_DLL
 $ vi makefile_DISCON_ITIBarge_DLL
```

Change one line:

```
SOURCE_FILE = DISCON_ITIBarge.f90
```

Then compile:

```
 $ make -f makefile_DISCON_ITIBarge_DLL
```

Also copy this controller to the input files and compile DISCON_OC3Hywind.f90 in the same method.

5.2.2 Original FAST for other controllers

The only difference is:

Uncomment these lines: 550,551,554,608,628,629,632,661 in SysGnuLinux.f90

5.2.3 FAST+WInDS for Bladed-style DLL controller

There are some minor differences compared to Section 5.2.1.

Add files WINDS$< * >$.f90 and Registry-AD.txt to /Source/dependencies/AeroDyn/.

Then modify makefile for FAST to include these files (full version is in Appendix A):

```
   # 32-bit or 64-bit?
BITS = 64
#BITS = 64
```

```
#ifeq ($(OS),Windows_NT)
#   Registry    = $(BIN_DIR)/Registry_win32.exe
#   MAP_lib     = $(MAP_DIR)/MAP_win32.lib
#   LAPACK_LINK = -llapack -lblas -LC:/LAPACK/win32
#else
```

```
  Registry    = registry.exe
  MAP_lib     = $(MAP_DIR)/MAP.so
  LAPACK_LINK = # -llapack -lblas
#endif


  # Name of compiler to use and flags to use.
FC     = ifort

#FFLAGS = -O2 -m$(BITS) -fbacktrace -ffree-line-length-none -x f95-cpp-input
#LDFLAGS = -O2 -m$(BITS) -fbacktrace -Wl,--stack=999999999,--large-address-aware

F90_FLAGS     =      -fpp -lpthread -O2 -mkl:parallel -openmp -finline-functions -xhost
F77_FLAGS     =      -fpp -lpthread -O2 -mkl:parallel -openmp -finline-functions -xhost
LDFLAGS       =      -fpp -lpthread -O2 -I$(MKLROOT)/include -mkl:parallel -openmp
    -finline-functions -xhost


#FFLAGS = -O0 -m$(BITS) -fbacktrace -ffree-line-length-none -x f95-cpp-input -g -pg
#LDFLAGS = -O0 -m$(BITS) -fbacktrace -Wl,--stack=999999999,--large-address-aware -g -pg
```

```
  # Destination and RootName for executable

OUTPUT_NAME = FAST
DEST_DIR   = ../bin
```

```
AD_SOURCES =          \
  AeroDyn_Types.f90   \
  WINDS_Treecode.f90      \
  WINDS_Lib.f90       \
  WINDS_IO.f90    \
  WINDS_DS.f90    \
  WINDS_Acce.f90      \
  WINDS.f90           \
  GenSubs.f90         \
  AeroSubs.f90        \
  AeroDyn.f90
```

```
  # General rules for compliling the files.

%.obj: %.f90
  $(FC) -I $(INTER_DIR) $(F90_FLAGS) -c $< -o $(INTER_DIR)/$@ -module $(INTER_DIR) -B
      $(INTER_DIR)

%.obj: %.F90
  $(FC) -I $(INTER_DIR) $(F90_FLAGS) -c $< -o $(INTER_DIR)/$@ -module $(INTER_DIR) -B
      $(INTER_DIR)

#bjj: what is $(F77)????
%.obj: %.f
  $(FC) -I $(INTER_DIR) $(F77_FLAGS) -c $< -o $(INTER_DIR)/$@ -module $(INTER_DIR) -B
```

```
      $(INTER_DIR)
```

```
#AeroDyn dependency rules:
AeroDyn_Types.obj:       NWTC_Library.obj InflowWind_Types.obj DWM_Types.obj Registry-AD.txt
WINDS_Treecode.obj:      NWTC_Library.obj AeroDyn_Types.obj
WINDS_Lib.obj:           NWTC_Library.obj AeroDyn_Types.obj
WINDS_IO.obj:            NWTC_Library.obj AeroDyn_Types.obj
WINDS_DS.obj:            NWTC_Library.obj AeroDyn_Types.obj WINDS_IO.obj
WINDS_Acce.obj:          NWTC_Library.obj AeroDyn_Types.obj
WINDS.obj:               NWTC_Library.obj AeroDyn_Types.obj WINDS_IO.obj WINDS_Acce.obj
    WINDS_DS.obj WINDS_Lib.obj WINDS_Treecode.obj
GenSubs.obj:             NWTC_Library.obj AeroDyn_Types.obj
AeroSubs.obj:            NWTC_Library.obj AeroDyn_Types.obj InflowWind.obj GenSubs.obj
AeroDyn.obj:             AeroDyn_Types.obj AeroSubs.obj GenSubs.obj InflowWind.obj DWM.obj
    DWM_Types.obj WINDS.obj WINDS_IO.obj WINDS_DS.obj WINDS_Lib.obj
```

Part III

User guide

It is highly recommended to read the FAST user guide [6] before use WInDS. This part will explain the input file and output files of WInDS.

Chapter 1

Input file

The input file of WInDS has similar style with FAST and AeroDyn. It is named “AeroDyn_WInDS.dat” and put in the same folder with AeroDyn input file. A sample input file is shown in Appendix B.

The explanation of AeroDyn_WInDS.dat.

- TMax: It is tricky to transfer the total simulation time from FAST to WInDS, so the user has to set this time same with in FAST input file. Program will terminate if this time is shorter than total time in < ... >.fst.
- WakeFLAG: This flag will control whether the simulation has full wake or simplified wake.
- AveSpeed: Unlike Matlab version of WInDS, the code cannot get the average speed at the very beginning, hence user needs to set this value equal or approximate to the wind speed from wind profile files.

Chapter 2

Settings besides AeroDyn_WInDS.dat

Take Test18 in CertTest for instance:

- In NRELOffshrBsline5MW_Onshore_AeroDyn.dat:

  use steady inflow without turbulence because free vortex method is based on irrotational flow.

- In NRELOffshrBsline5MW_Blade.dat:

  set all PitchAxis: 0.25, because WInDS assume the blades are thin and the aerodynamic center lies exactly one quarter of the chord behind the leading edge.

If user run code on Linux, make sure:

- to change all "\" to "/". This may happens in AeroDyn and ServoDyn inputs files.
- in HydroDyn input file, change "Barge.hst" to "barge.hst", and "Spar.hst" to "spar.hst", because Linux is case sensitive.
- in ServoDyn input file, change "DISCON_xxx.dll" to "DISCON_xxx.so".

Chapter 3

Run FAST_WInDS_v8.10

3.1 In Windows

3.1.1 Using Command Prompt window

Open Command Prompt window, drag "FAST_dev_Win32.exe" into it, type a space and drag the *.fst input file into it. Then run.

3.1.2 Using Matlab

If the user want to run multiple simulations, it is easier to use Matlab via a for loop.

Example Matlab code:

```
clc
clear all
close all

FAST_WInDS_dir = 'C:\Test18\bin\FAST_dev_Win32.exe';
fst_dir = strcat(' C:\Test18\Test18\Test18.fst'); % NOTICE: the space in the beginning

status = dos(strcat(FAST_WInDS_dir, fst_dir));

copyfile(strcat('C:\Test18\Test18\Test18.outb'), ...
        strcat('C:\Test18\Test18\Test18_1234.outb'));
```

3.2 In Linux on MGHPCC

Remember to load the compilers before run the code:

module load intel/composer_xe_2015_sp1.0.080

module load intel/mkl_libraries_from_composer_xe_2013_sp1

Submit job to long queue:

```
$ bsub -q long -n 20 -W 10:00 -R select[ib] -R "rusage[mem=500]" ./FAST_glin64
    Test/Test18.fst
```

FAST may only runs on nodes with infiniband, so use “-R select[ib]” to choose.

Highly recommended to run one job on single node by setting: -R “span[hosts=1]”.

On MGHPCC, user can check the average number of cores being used by:

```
 $ /share/bin/jobcpucheck
```

For more information about submit jobs, check MGHPCC wiki[1] or IBM LFS[2].

It can be convenient to write a bash script with loops to run multiple simulations [3].

[1]MGHPCC wiki: http://wiki.umassrc.org/wiki/index.php/Submitting_Cluster_Jobs

[2]IBM LFS: http://ghpcc06.umassrc.org/doc/lsf/

[3]Slides of Introduction to Linux shell scripting from the 6/11/2015: http://wiki.umassrc.org/wiki/index.php/Main_Page

Chapter 4

Optional output files

All the output files will be grouped into one folder with the name of it starting time.

4.1 Summary file

In the input file, 'SumPrint' is the flag. The summary file is named <data>_<time>_WInDS_sum.dat. It records the running time and some basic parameters.

4.2 Output files for Treecode algorithm

The Biot-Savart Law is the most time-consuming part in WInDS. Several methods are applied to accelerate it.

In the input file, 'Speedup' is the flag for writing output file of parallelized Treecode performance compared with direct OpenMP acceleration.

The output file is named <data>_<time>_Speedup.dat.

In the output file, the information includes: the number of source vortex filaments, the number of receiver points, CPU time of direct OpenMP calculation, the CPU time of Treecode calculation, the speedup, and relative error.

4.3 Output files for Kutta Joukowski

In the input file, 'KJCoverg' is the flag for output the convergence of Kutta Joukowski subroutine at every timestep. This will output the timestep number, iteration used for calculation and max of D_Gamma at last iteration. It is original intended for debugging.

4.4 Output files for Element Information

In the input file, ‘Element’ is the flag for output Cl, Cd, AOA and V_tot of each blade element. This flag is recommended to be off to reduce running time. It is original intended for debugging.

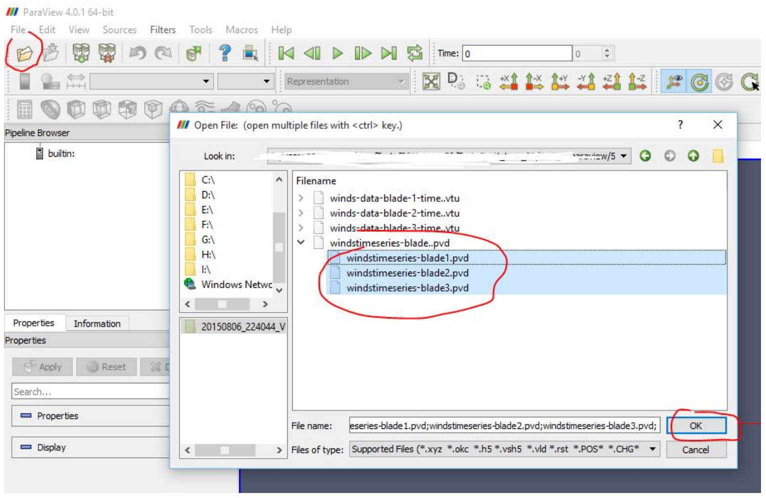
4.5 Output files for Paraview

In the input file, ‘AnimFLAG’ is the flag for writing output files for Paraview.

The output files are written in the folder named <data>_<time>_VISUAL.

To view the wind turbine wake in Paraview:

1. Download and install Paraview (http://www.paraview.org/download/).
2. Open Paraview. Click File-Open. Select the location of output files of FAST-WInDS. Select: windstimeseries-blade1.pvd, windstimeseries-blade2.pvd, windstimeseries-blade3.pvd, and click OK (Figure 4.1).
3. Click Properties-Apply (Figure 4.2).
4. Select one blade and change the ‘Solid Color’ to ‘velocity’. And apply this to all blades (Figure 4.3).
5. Click the play button (Figure 4.4).



Look in: [redacted]araview/5

Filename
winds-data-blade-1-time..vtu
winds-data-blade-2-time..vtu
winds-data-blade-3-time..vtu
windstimeseries-blade..pvd
windstimeseries-blade1.pvd
windstimeseries-blade2.pvd
windstimeseries-blade3.pvd

File name: eseries-blade1.pvd;windstimeseries-blade2.pvd;windstimeseries-blade3.pvd;
Files of type: Supported Files (*.xyz *.okc *.h5 *.vsh5 *.vld *.rst *.POS* *.CHG*

Figure 4.1: Paraview guide: step 2

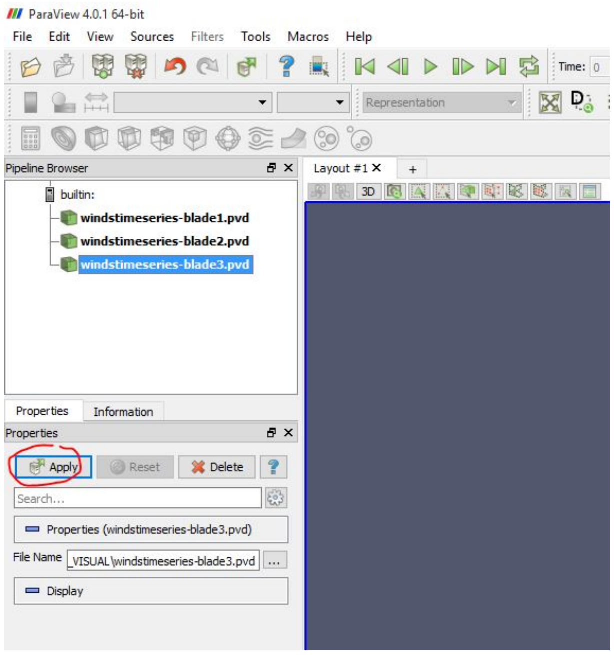

ParaView 4.0.1 64-bit
File Edit View Sources Filters Tools Macros Help
Time: 0
Representation
Pipeline Browser
builtin:
windstimeseries-blade1.pvd
windstimeseries-blade2.pvd
windstimeseries-blade3.pvd
Layout #1
Properties Information
Apply Reset Delete
Search...
Properties (windstimeseries-blade3.pvd)
File Name _VISUAL\windstimeseries-blade3.pvd
Display

Figure 4.2: Paraview guide: step 3

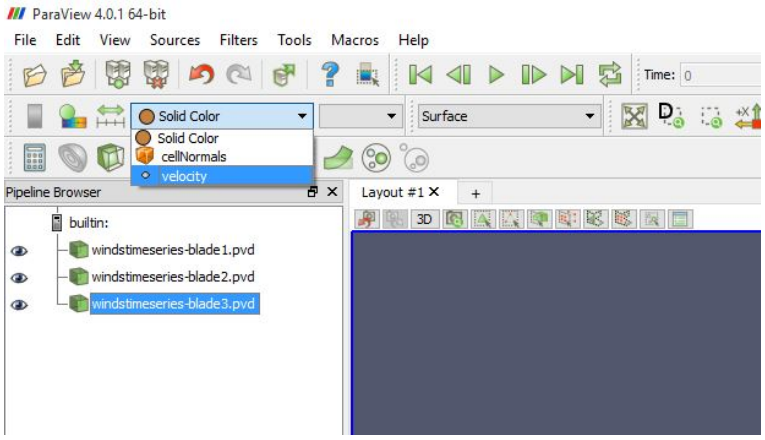

ParaView 4.0.1 64-bit
File Edit View Sources Filters Tools Macros Help
Time: 0
Solid Color
Solid Color
cellNormals
velocity
Surface
Pipeline Browser
builtin:
windstimeseries-blade1.pvd
windstimeseries-blade2.pvd
windstimeseries-blade3.pvd
Layout #1

Figure 4.3: Paraview guide: step 4

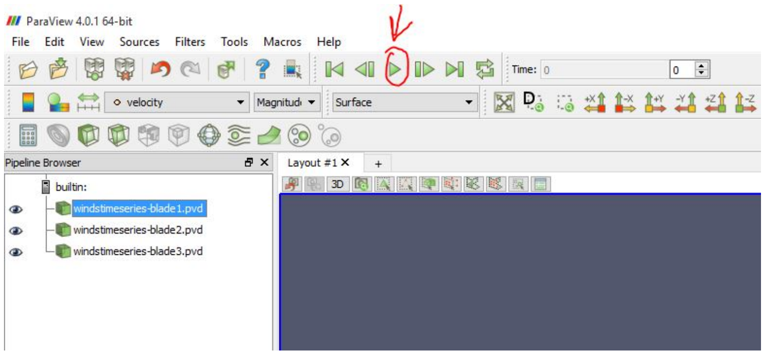

ParaView 4.0.1 64-bit
File Edit View Sources Filters Tools Macros Help
Time: 0
velocity
Magnitude
Surface
Pipeline Browser
builtin:
windstimeseries-blade1.pvd
windstimeseries-blade2.pvd
windstimeseries-blade3.pvd
Layout #1

Figure 4.4: Paraview guide: step 5

Chapter 5

Sample result

*** To be added later ***

Part IV

Appendix

Appendix A

Full makefile for FAST_WInDS v8.10

```
#================================================================================#
# This makefile created by B. Jonkman on 2-Apr-2013,                            #
# adapted from Crunch (M. Buhl on 25-Jan-2013).                                  #
# (c) 2013-2014 National Renewable Energy Laboratory                            #
#                                                                                #
# This makefile has been tested on Windows 7 with gfortran.                     #
# This makefile works with mingw32-make.exe.                                     #
#                                                                                #
# It was designed to be used with:                                               #
#    FAST                 (v8.09.00a-bjj, 30-Sept-2014)                          #
#                                                                                #
# Older versions of the source code may not work with this makefile.             #
#================================================================================#

   # 32-bit or 64-bit?
BITS = 64
#BITS = 64


   # Location of source files for FAST, and its modules.

FAST_DIR    = ../Source

NWTC_Lib_DIR = $(FAST_DIR)/dependencies/NWTC_Library
NETLIB_DIR  = $(FAST_DIR)/dependencies/NetLib
ED_DIR      = $(FAST_DIR)/dependencies/ElastoDyn
SrvD_DIR    = $(FAST_DIR)/dependencies/ServoDyn
AD_DIR      = $(FAST_DIR)/dependencies/AeroDyn
IfW_DIR     = $(FAST_DIR)/dependencies/InflowWind
HD_DIR      = $(FAST_DIR)/dependencies/HydroDyn
SD_DIR      = $(FAST_DIR)/dependencies/SubDyn
MAP_DIR     = $(FAST_DIR)/dependencies/MAP
FEAM_DIR    = $(FAST_DIR)/dependencies/FEAMooring
MD_DIR      = $(FAST_DIR)/dependencies/MoorDyn
IceF_DIR    = $(FAST_DIR)/dependencies/IceFloe
IceD_DIR    = $(FAST_DIR)/dependencies/IceDyn
```

```
TMD_DIR     = $(SrvD_DIR)
DWM_DIR     = $(AD_DIR)
HD_DIR_Reg  = $(HD_DIR)
IfW_DIR_Reg = $(IfW_DIR)
BIN_DIR     = ../bin

   # Names and locations of the Registry, MAP libraries, and instructions for linking with
       LAPACK.
   # You will probably need to change these for your system.

#ifeq ($(OS),Windows_NT)
#   Registry    = $(BIN_DIR)/Registry_win32.exe
#   MAP_lib     = $(MAP_DIR)/MAP_win32.lib
#   LAPACK_LINK = -llapack -lblas -LC:/LAPACK/win32
#else
   Registry    = registry.exe
   MAP_lib     = $(MAP_DIR)/MAP.so
   LAPACK_LINK = # -llapack -lblas
#endif


   # Name of compiler to use and flags to use.
FC     = ifort

#FFLAGS = -O2 -m$(BITS) -fbacktrace -ffree-line-length-none -x f95-cpp-input
#LDFLAGS = -O2 -m$(BITS) -fbacktrace -Wl,--stack=999999999,--large-address-aware

F90_FLAGS     =      -fpp -lpthread -O2 -mkl:parallel -openmp -finline-functions -xhost
F77_FLAGS     =      -fpp -lpthread -O2 -mkl:parallel -openmp -finline-functions -xhost
LDFLAGS       =      -fpp -lpthread -O2 -I$(MKLROOT)/include -mkl:parallel -openmp
    -finline-functions -xhost


#FFLAGS = -O0 -m$(BITS) -fbacktrace -ffree-line-length-none -x f95-cpp-input -g -pg
#LDFLAGS = -O0 -m$(BITS) -fbacktrace -Wl,--stack=999999999,--large-address-aware -g -pg

#-DDOUBLE_PRECISION

# -mthreads
# some useful gfortran options:
#  -DFPE_TRAP_ENABLED
#  -Wl,--large-address-aware # not necessary when $(BITS) is 64 (64-bit target)
#  -Wl,--stack=999999999    # not necessary when $(BITS) is 64
#  -Wconversion-extra -Wconversion
#  -fdefault-real-8 -fcheck=bounds,do,mem,pointer -std=f2003 -O0 -v -Wall
#  -pg                        # generate debugging info for debugger
# http://gcc.gnu.org/onlinedocs/gfortran/Option-Index.html#Option-Index
#
# makefile:
# --warn-undefined-variables



   # Destination and RootName for executable
```

```
OUTPUT_NAME = FAST
DEST_DIR   = ../bin

   #===========================================================#
   # You should not need to change anything beyond this point #
   #===========================================================#

   # System-specific settings.

ifeq ($(OS),Windows_NT)
      # Windows
   DEL_CMD  = del
   EXE_EXT  = _gwin$(BITS).exe
   INTER_DIR = Obj_win$(BITS)
   MD_CMD   = @mkdir
   OBJ_EXT  = .obj
   PATH_SEP = \\
   SYS_FILE = SysGnuWin
else
      # Linux
   DEL_CMD  = rm -f
   EXE_EXT  = _glin$(BITS)

   INTER_DIR = Obj_lin$(BITS)
   MD_CMD   = @mkdir -p
   OBJ_EXT  = .o
   PATH_SEP = /
   SYS_FILE = SysGnuLinux
endif


   # Source files (by module)

LIB_SOURCES =         \
   SingPrec.f90        \
   NWTC_Base.f90      \
   $(SYS_FILE).f90    \
   NWTC_Library_Types.f90 \
   NWTC_IO.f90        \
   NWTC_Num.f90       \
   ModMesh_Types.f90  \
   ModMesh.f90        \
   ModMesh_Mapping.f90 \
   NWTC_Library.f90

NETLIB_SOURCES=          \
   NWTC_ScaLAPACK.f90    \
   NWTC_FFTPACK.f90      \
   NWTC_LAPACK.f90       \
   fftpack4.1.f          \
   dlasrt2.f             \
   slasrt2.f
```

```
IfW_SOURCES =          \
  IfW_FFWind_Types.f90 \
  IfW_FFWind.f90       \
  IfW_HHWind_Types.f90 \
  IfW_HHWind.f90       \
  InflowWind_Subs.f90 \
  InflowWind_Types.f90 \
  Lidar_Types.f90      \
  Lidar.f90            \
  InflowWind.f90

AD_SOURCES =           \
  AeroDyn_Types.f90    \
  WINDS_Treecode.f90      \
  WINDS_Lib.f90        \
  WINDS_IO.f90     \
  WINDS_DS.f90     \
  WINDS_Acce.f90       \
  WINDS.f90            \
  GenSubs.f90          \
  AeroSubs.f90         \
  AeroDyn.f90

DWM_SOURCES =          \
  DWM_Types.f90        \
  DWM.f90              \
  DWM_Wake_Sub_ver2.f90


HD_SOURCES =               \
  SS_Radiation_Types.f90  \
  SS_Radiation.f90        \
  Waves2_Types.f90        \
  Waves2_Output.f90       \
  Waves2.f90              \
  Waves_Types.f90         \
  Waves.f90               \
  Current_Types.f90       \
  Current.f90             \
  Morison_Types.f90       \
  Morison_Output.f90      \
  Morison.f90             \
  Conv_Radiation_Types.f90 \
  Conv_Radiation.f90      \
  WAMIT2_Types.f90        \
  WAMIT2_Output.f90       \
  WAMIT2.f90              \
  WAMIT_Types.f90         \
  WAMIT_Interp.f90        \
  WAMIT_Output.f90        \
  WAMIT.f90               \
  HydroDyn_Output.f90     \
  HydroDyn_Types.f90      \
   HydroDyn_Input.f90      \
   HydroDyn.f90

MAP_SOURCES =            \
   MAP_Types.f90         \
   MAP.f90

FEAM_SOURCES =            \
   FEAMooring_Types.f90   \
   FEAM.f90

MD_SOURCES =            \
   MoorDyn_Types.f90    \
   MoorDyn_IO.f90       \
   MoorDyn.f90


SD_SOURCES =             \
   qsort_c_module.f90    \
   SD_FEM.f90            \
   SubDyn_Types.f90      \
   SubDyn_Output.f90     \
   SubDyn.f90


ED_SOURCES =          \
   ElastoDyn_Types.f90 \
   ElastoDyn.f90

SrvD_SOURCES =        \
   TMD_Types.f90      \
   TMD.f90            \
   ServoDyn_Types.f90 \
   ServoDyn.f90       \
   PitchCntrl_ACH.f90 \
   BladedInterface.f90 \
   UserSubs.f90       \
   UserVSCont_KP.f90


IceF_SOURCES =            \
   IceFloe_Types.f90      \
   iceLog.F90             \
   coupledCrushing.F90    \
   crushingIEC.F90        \
   crushingISO.F90        \
   IceFlexBase.F90        \
   IceFlexIEC.f90         \
   IceFlexISO.f90         \
   IceFloeBase.F90        \
   iceInput.f90           \
   intermittentCrushing.F90 \
   lockInISO.F90          \
   randomCrushing.F90     \
   RANLUX.f90             \
   IceFloe.f90

IceD_SOURCES =            \
   IceDyn_Types.f90       \
   IceDyn.f90


FAST_SOURCES =         \
   FAST_Types.f90      \
   FAST_Mods.f90       \
   FAST_Subs.f90       \
   FAST_Prog.f90

vpath %.f90 $(NWTC_Lib_DIR) $(AD_DIR) $(IfW_DIR) $(HD_DIR_Reg) $(HD_DIR) $(ED_DIR)
    $(SrvD_DIR) $(TMD_DIR) $(SD_DIR) $(MAP_DIR) \
           $(FAST_DIR) $(NETLIB_DIR) $(FEAM_DIR) $(MD_DIR) $(IceF_DIR) $(IceD_DIR) $(DWM_DIR)
vpath %.f  $(NETLIB_DIR)
vpath %.mod $(INTER_DIR)
vpath %.obj $(INTER_DIR)
vpath %.txt $(FAST_DIR) $(AD_DIR) $(IfW_DIR_Reg) $(HD_DIR_Reg) $(SD_DIR) $(ED_DIR)
    $(SrvD_DIR) $(TMD_DIR) $(FEAM_DIR) $(IceD_DIR) $(DWM_DIR) $(MD_DIR)
vpath %.inp $(IceF_DIR)
#apparently vpath doesn't work as expected if I use %.F90 (is it not case sensitive???)
#vpath %.F90   $(IceF_DIR)
vpath %    $(IceF_DIR)


ALL_SOURCES = $(FAST_SOURCES) $(LIB_SOURCES) $(NETLIB_SOURCES) $(IfW_SOURCES) $(AD_SOURCES)
    $(SrvD_SOURCES) $(ED_SOURCES) \
             $(HD_SOURCES) $(SD_SOURCES) $(MAP_SOURCES) $(FEAM_SOURCES) $(IceF_SOURCES)
                 $(IceD_SOURCES) $(DWM_SOURCES) \
             $(TMD_SOURCES) $(MD_SOURCES)
tmp_objs1  = $(ALL_SOURCES:.f90=.obj)
tmp_objs2  = $(tmp_objs1:.F90=.obj)    #note the upper case here (from IceFloe)
ALL_OBJS   = $(tmp_objs2:.f=.obj)


   # Rule to do everything.

all:    default
# use this for "all" for debugging: print variables:
#all:    ; $(info $$IceF_SOURCES is [${IceF_SOURCES}]) echo debugging
#
default: $(INTER_DIR) $(DEST_DIR)/$(OUTPUT_NAME)$(EXE_EXT)


   # General rules for compliling the files.

%.obj: %.f90
   $(FC) -I $(INTER_DIR) $(F90_FLAGS) -c $< -o $(INTER_DIR)/$@ -module $(INTER_DIR) -B
       $(INTER_DIR)

%.obj: %.F90
```

```
   $(FC) -I $(INTER_DIR) $(F90_FLAGS) -c $< -o $(INTER_DIR)/$@ -module $(INTER_DIR) -B
       $(INTER_DIR)

#bjj: what is $(F77)????
%.obj: %.f
   $(FC) -I $(INTER_DIR) $(F77_FLAGS) -c $< -o $(INTER_DIR)/$@ -module $(INTER_DIR) -B
       $(INTER_DIR)



   # General rules for creating _Types.f90 files from Registry files.

FAST_Types.f90:
   $(Registry) FAST_Registry.txt -I $(NWTC_Lib_DIR) -I $(ED_DIR) -I $(SrvD_DIR) -I $(SD_DIR)
       -I $(TMD_DIR) -I $(AD_DIR) \
   -I $(HD_DIR_Reg) -I $(IceF_DIR) -I $(IceD_DIR) -I $(MAP_DIR) -I $(FEAM_DIR) -I $(MD_DIR)
       -I $(IfW_DIR_Reg) -I $(DWM_DIR) -noextrap

TMD_Types.f90:
   $(Registry) TMD_Registry.txt -I $(NWTC_Lib_DIR)

ServoDyn_Types.f90:
   $(Registry) ServoDyn_Registry.txt -I $(NWTC_Lib_DIR) -I $(TMD_DIR)


ElastoDyn_Types.f90:
   $(Registry) ElastoDyn_Registry.txt -I $(NWTC_Lib_DIR)


Current_Types.f90:
   $(Registry) Current.txt -I $(NWTC_Lib_DIR)

Waves_Types.f90:
   $(Registry) Waves.txt -I $(NWTC_Lib_DIR)

Waves2_Types.f90:
   $(Registry) Waves2.txt -I $(NWTC_Lib_DIR)

SS_Radiation_Types.f90:
   $(Registry) SS_Radiation.txt -I $(NWTC_Lib_DIR)

Conv_Radiation_Types.f90:
   $(Registry) Conv_Radiation.txt -I $(NWTC_Lib_DIR)

WAMIT_Types.f90:
   $(Registry) WAMIT.txt -I $(NWTC_Lib_DIR)

WAMIT2_Types.f90:
   $(Registry) WAMIT2.txt -I $(NWTC_Lib_DIR)

Morison_Types.f90:
   $(Registry) Morison.txt -I $(NWTC_Lib_DIR)

HydroDyn_Types.f90:
   $(Registry) HydroDyn.txt -I $(NWTC_Lib_DIR) -I $(HD_DIR_Reg)


AeroDyn_Types.f90:
   $(Registry) Registry-AD.txt -I $(NWTC_Lib_DIR) -I $(IfW_DIR_Reg) -I $(DWM_DIR)

DWM_Types.f90:
   $(Registry) Registry-DWM.txt -I $(NWTC_Lib_DIR) -I $(IfW_DIR_Reg)


IfW_HHWind_Types.f90:
   $(Registry) IfW_HHWind.txt -I $(NWTC_Lib_DIR) -I $(IfW_DIR_Reg)

IfW_FFWind_Types.f90:
   $(Registry) IfW_FFWind.txt -I $(NWTC_Lib_DIR) -I $(IfW_DIR_Reg)

Lidar_Types.f90:
   $(Registry) Lidar.txt -I $(NWTC_Lib_DIR) -I $(IfW_DIR_Reg)

InflowWind_Types.f90:
   $(Registry) InflowWind.txt -I $(NWTC_Lib_DIR) -I $(IfW_DIR_Reg)


SubDyn_Types.f90:
   $(Registry) SubDyn_Registry.txt -I $(NWTC_Lib_DIR)


FEAMooring_Types.f90:
   $(Registry) FEAM_Registry.txt -I $(NWTC_Lib_DIR)

MoorDyn_Types.f90:
   $(Registry) MoorDyn_Registry.txt -I $(NWTC_Lib_DIR)


IceFloe_Types.f90:
   $(Registry) IceFloe_FASTRegistry.inp -I $(NWTC_Lib_DIR)


IceDyn_Types.f90:
   $(Registry) Registry_IceDyn.txt -I $(NWTC_Lib_DIR)


   #  Dependency rules.
#NWTC Library dependency rules:
NWTC_Base.obj:            SingPrec.obj
$(SYS_FILE).obj:          NWTC_Base.obj
NWTC_Library_Types.obj:   $(SYS_FILE).obj
NWTC_IO.obj:              NWTC_Library_Types.obj
NWTC_Num.obj:             NWTC_IO.obj
ModMesh_Types.obj:        NWTC_Num.obj
ModMesh.obj:              ModMesh_Types.obj
ModMesh_Mapping.obj:      ModMesh.obj NWTC_LAPACK.obj
NWTC_Library.obj:         ModMesh.obj ModMesh_Mapping.obj
```

```
NWTC_LAPACK.obj:         NWTC_Base.obj
NWTC_ScaLAPACK.obj:      NWTC_Base.obj dlasrt2.obj slasrt2.obj
NWTC_FFTPACK.obj:        NWTC_Library.obj fftpack4.1.obj
fftpack4.1.obj:          SingPrec.obj


#InflowWind dependency rules:
IfW_HHWind_Types.obj:    NWTC_Library.obj IfW_HHWind.txt
IfW_HHWind.obj:          IfW_HHWind_Types.obj
IfW_FFWind_Types.obj:    NWTC_Library.obj IfW_FFWind.txt
IfW_FFWind.obj:          IfW_FFWind_Types.obj
InflowWind_Types.obj:    NWTC_Library.obj IfW_FFWind_Types.obj IfW_HHWind_Types.obj
   Lidar_Types.obj InflowWind.txt
InflowWind_Subs.obj:     InflowWind_Types.obj IfW_FFWind.obj IfW_HHWind.obj
Lidar_Types.obj:         NWTC_Library.obj Lidar.txt
Lidar.obj:               Lidar_Types.obj InflowWind_Subs.obj
InflowWind.obj:          InflowWind_Subs.obj Lidar.obj


#AeroDyn dependency rules:
AeroDyn_Types.obj:       NWTC_Library.obj InflowWind_Types.obj DWM_Types.obj Registry-AD.txt
WINDS_Treecode.obj:      NWTC_Library.obj AeroDyn_Types.obj
WINDS_Lib.obj:           NWTC_Library.obj AeroDyn_Types.obj
WINDS_IO.obj:            NWTC_Library.obj AeroDyn_Types.obj
WINDS_DS.obj:            NWTC_Library.obj AeroDyn_Types.obj WINDS_IO.obj
WINDS_Acce.obj:          NWTC_Library.obj AeroDyn_Types.obj
WINDS.obj:               NWTC_Library.obj AeroDyn_Types.obj WINDS_IO.obj WINDS_Acce.obj
   WINDS_DS.obj WINDS_Lib.obj WINDS_Treecode.obj
GenSubs.obj:             NWTC_Library.obj AeroDyn_Types.obj
AeroSubs.obj:            NWTC_Library.obj AeroDyn_Types.obj InflowWind.obj GenSubs.obj
AeroDyn.obj:             AeroDyn_Types.obj AeroSubs.obj GenSubs.obj InflowWind.obj DWM.obj
   DWM_Types.obj WINDS.obj WINDS_IO.obj WINDS_DS.obj WINDS_Lib.obj


#DWM dependency rules:
DWM_Types.obj:           NWTC_Library.obj Registry-DWM.txt
DWM_Wake_Sub_ver2.obj:   NWTC_Library.obj DWM_Types.obj
DWM.obj:                 NWTC_Library.obj DWM_Types.obj DWM_Wake_Sub_ver2.obj


#HydroDyn dependency rules:
SS_Radiation_Types.obj:  NWTC_Library.obj SS_Radiation.txt
SS_Radiation.obj:        SS_Radiation_Types.obj

Waves2_Types.obj:        NWTC_Library.obj Waves2.txt
Waves2_Output.obj:       Waves2_Types.obj
Waves2.obj:              NWTC_Library.obj Waves2_Types.obj NWTC_FFTPACK.obj
   Waves2_Output.obj Waves.obj

Waves_Types.obj:         NWTC_Library.obj Waves.txt
Waves.obj:               Waves_Types.obj NWTC_FFTPACK.obj

Current_Types.obj:       NWTC_Library.obj Current.txt
Current.obj:             Current_Types.obj
```

```
Morison_Types.obj:       NWTC_Library.obj Morison.txt
Morison_Output.obj:      Morison_Types.obj Waves.obj
Morison.obj:             Morison_Types.obj Morison_Output.obj

Conv_Radiation_Types.obj: NWTC_Library.obj Conv_Radiation.txt
Conv_Radiation.obj:      Conv_Radiation_Types.obj

WAMIT2_Types.obj:        NWTC_Library.obj WAMIT2.txt
WAMIT2_Output.obj:       NWTC_Library.obj WAMIT2_Types.obj
WAMIT2.obj:              NWTC_Library.obj WAMIT_Interp.obj WAMIT2_Output.obj
    NWTC_FFTPACK.obj Waves.obj

WAMIT_Types.obj:         NWTC_Library.obj Conv_Radiation_Types.obj SS_Radiation_Types.obj
    Waves_Types.obj WAMIT.txt
WAMIT.obj:               WAMIT_Types.obj WAMIT_Output.obj Waves_Types.obj Conv_Radiation.obj
    SS_Radiation.obj NWTC_FFTPACK.obj WAMIT_Interp.obj

WAMIT_Interp.obj:        NWTC_Library.obj
WAMIT_Output.obj:        WAMIT_Types.obj Waves.obj
HydroDyn_Output.obj:     HydroDyn_Types.obj Waves.obj

HydroDyn_Types.obj:      NWTC_Library.obj Current_Types.obj Waves_Types.obj Waves2_Types.obj
    Conv_Radiation_Types.obj \
                         SS_Radiation_Types.obj WAMIT_Types.obj WAMIT2_Types.obj
                             Morison_Types.obj HydroDyn.txt
HydroDyn_Input.obj:      HydroDyn_Types.obj Waves.obj Morison.obj HydroDyn_Output.obj
    WAMIT_Output.obj Waves2_Output.obj WAMIT2_Output.obj
HydroDyn.obj:            HydroDyn_Types.obj HydroDyn_Input.obj HydroDyn_Output.obj WAMIT.obj
    Current.obj Morison.obj Waves2.obj WAMIT2.obj


# SubDyn dependency rules:
SubDyn_Types.obj:        NWTC_Library.obj  SubDyn_Registry.txt
SD_FEM.obj:              NWTC_Library.obj SubDyn_Types.obj qsort_c_module.obj
SubDyn_Output.obj:       SD_FEM.obj
SubDyn.obj:              SubDyn_Output.obj SD_FEM.obj qsort_c_module.obj NWTC_ScaLAPACK.obj
    NWTC_LAPACK.obj


# MAP dependency rules:
MAP_Types.obj:           NWTC_Library.obj
MAP.obj:                 NWTC_Library.obj MAP_Types.obj


# FEAMooring dependency rules:
FEAMooring_Types.obj:    NWTC_Library.obj FEAM_Registry.txt
FEAM.obj:                NWTC_Library.obj FEAMooring_Types.obj

# MoorDyn dependency rules:
MoorDyn_Types.obj:       NWTC_Library.obj MoorDyn_Registry.txt
MoorDyn_IO.obj:          NWTC_Library.obj MoorDyn_Types.obj
MoorDyn.obj:             NWTC_Library.obj MoorDyn_IO.obj
```

```
# ElastoDyn dependency rules:
ElastoDyn_Types.obj:     NWTC_Library.obj ElastoDyn_Registry.txt
ElastoDyn.obj:           NWTC_Library.obj ElastoDyn_Types.obj NWTC_LAPACK.obj


# ServoDyn dependency rules:
TMD_Types.obj:           NWTC_Library.obj TMD_Registry.txt
TMD.obj:                 NWTC_Library.obj TMD_Types.obj
ServoDyn_Types.obj:      NWTC_Library.obj TMD_Types.obj ServoDyn_Registry.txt
BladedInterface.obj:     NWTC_Library.obj ServoDyn_Types.obj
ServoDyn.obj:            NWTC_Library.obj ServoDyn_Types.obj PitchCntrl_ACH.obj UserSubs.obj
    UserVSCont_KP.obj BladedInterface.obj TMD.obj
PitchCntrl_ACH.obj:      NWTC_Library.obj
UserSubs.obj:            NWTC_Library.obj
UserVSCont_KP.obj:       NWTC_Library.obj


# IceFloe dependency rules:
#RANLUX.obj:
iceLog.obj:              NWTC_Library.obj
iceInput.obj:            iceLog.F90 NWTC_Library.obj
IceFloeBase.obj:         NWTC_Library.obj iceInput.obj RANLUX.obj IceFloe_Types.obj
coupledCrushing.obj:     IceFloeBase.obj
crushingIEC.obj:         IceFloeBase.obj
crushingISO.obj:         IceFloeBase.obj
intermittentCrushing.obj: crushingISO.obj
lockInISO.obj:           crushingISO.obj
randomCrushing.obj:      crushingISO.obj
IceFlexBase.obj:         IceFloeBase.obj
IceFlexIEC.obj:          IceFlexBase.obj
IceFlexISO.obj:          IceFlexBase.obj
IceFloe_Types.obj:       NWTC_Library.obj IceFloe_FASTRegistry.inp
IceFloe.obj:             IceFloe_Types.obj iceLog.obj coupledCrushing.obj crushingIEC.obj
    crushingISO.obj IceFlexBase.obj \
                         IceFlexIEC.obj IceFlexISO.obj intermittentCrushing.obj
                             lockInISO.obj randomCrushing.obj


# IceDyn dependency rules:
IceDyn_Types.obj:        NWTC_Library.obj Registry_IceDyn.txt
IceDyn.obj:              NWTC_Library.obj IceDyn_Types.obj


# FAST dependency rules:
FAST_Types.obj:          NWTC_Library.obj ElastoDyn_Types.obj ServoDyn_Types.obj
    AeroDyn_Types.obj SubDyn_Types.obj HydroDyn_Types.obj \
                         MAP_Types.obj FEAMooring_Types.obj MoorDyn_Types.obj
                             IceFloe_Types.obj IceDyn_Types.obj FAST_Registry.txt
FAST_Mods.obj:           NWTC_Library.obj FAST_Types.obj
FAST_Subs.obj:           NWTC_Library.obj FAST_Mods.obj NWTC_LAPACK.obj AeroDyn.obj
    InflowWind.obj ServoDyn.obj ElastoDyn.obj \
                         SubDyn.obj HydroDyn.obj MAP.obj FEAM.obj MoorDyn.obj IceFloe.obj
                             IceDyn.obj
```

```
FAST_Prog.obj:           NWTC_Library.obj FAST_Subs.obj



#$(OUTPUT_NAME)$(EXE_EXT): Fast_Prog.obj

   # Make sure the destination directory for the intermediate files exist.

$(INTER_DIR):
   $(MD_CMD) $(INTER_DIR)


   # For linking FAST.

$(DEST_DIR)/$(OUTPUT_NAME)$(EXE_EXT): $(ALL_OBJS) | $(INTER_DIR)
   $(FC) $(LDFLAGS) -I $(INTER_DIR) -o $(DEST_DIR)/$(OUTPUT_NAME)$(EXE_EXT) \
   $(foreach src, $(ALL_OBJS), $(addprefix $(INTER_DIR)/,$(src))) $(MAP_lib) $(LAPACK_LINK)

   # Cleanup afterwards.

clean:
   $(DEL_CMD) $(INTER_DIR)$(PATH_SEP)*.mod $(INTER_DIR)$(PATH_SEP)*.obj

#superclean:
#  $(DEL_CMD) $(INTER_DIR)$(PATH_SEP)*.mod $(INTER_DIR)$(PATH_SEP)*.obj
# also delete all the registry-generated ModuleName_Types.f90 files
```

Appendix B

Sample input file for WInDS

```
*************************** Input file for WInDS in AeroDyn *************************
NREL 5.0 MW offshore baseline. (Please do not delete or add any line)
---------------------- GENERAL SETTING ------------------------
      60    TMax        - Total run time (s). Same as in <...>.fst
       4    NTP         - Number of time steps stored in the arrays. To keep memory reasonable, usually ~3-4.
       8    DtRatio     - Timestep Duration(WINDS) / Timestep Duration(AERODYN)
    True    SteadyFlag  - Whether steady inflow wind, if steady, the mean speed of first timestep is used for whole
          simulation (flag)
---------------------- BIOT-SAVART LAW ------------------------
   False    WakeFLAG    - Whether simplify the far wake (flag).
    True    RollFLAG    - Whether to apply induction to all wake nodes (flag). (True is recommanded)
     126    RollDist    - Distance in meters, beyond which the wake is have on induction. Only works when WakeFLAG is
           true.
     252    WakeLength  - Distance in meters, beyond which the wake is cut off. -1 disables cutoff. Only works when
           WakeFLAG is true.
    12.0    AveSpeed    - Average wind speed. Only used to calculate wake length. Only works when WakeFLAG is true.
   False    UindPast    - If true, the "frozen" filaments will retain the induced velocity from the most recent
         timestep. Only works when WakeFLAG is true.
    True    ViscFLAG    - Whether Vatistas viscous model for filaments (flag)
       2    ViscModel   - Index of Vatistas viscous model (Do no change)
  0.0001    DELTA       - Smooth parameter (not used...but do no remove this line)
------------------NUMERICAL METHODS AND SOLUTION----------------
     RK4    INTEG       - Numerical integration scheme: RK4 (or RK2, PCC, AB4) (unquoted string)
   1e-10    Tolerance   - Tolerance value for convergence of numerical methods
  1000000     CO          - Distance from wake nodes beyond which influence is negligible
     0.20     RELAX       - Relaxation value for fixed-point iteration
      500     MaxIter     - Maximum number of iterations for Kutta-Joukowski theorem
     True     ExtrapWake  - Use quadratic extrapolation to guess next bound vorticity value, otherwise use previous value
        as initial guess
-------------------BEM (First timestep in WInDS) ---------------
     1E-6     BEMTOL      - Convergence tolerance
      200     MAX_ITER    - Maximum number of allowable iterations
      0.1     WEIGHT      - Weighting factor on corrections to balance speed with stability (faster as you approach 1, but
          less stable)
-------------------------- WIND SHEAR --------------------------
    False     ShearFLAG   - Wind shear (flag)
        1     ShearType   - Power law = 1, log law = 2, log law with stability = 3
     90.0     ZRef        - Hub height
------------------------- TOWER SHADOW -------------------------
    False     TWRFLAG     - Include induced velocity from tower effects (flag) - Not finished yet -
        1     TWRUPDOWN   - Upwind rotor=1, Downwind rotor=2
        1     TWRMETHOD   - Basic potential flow=1, Aerodyn=2
------------------------ GROUND EFFECTS ------------------------
    False     GroundFLAG  - Include vortex panels to model ground effects (flag)
    PANEL     GroundMethod - Method for calculating ground effects: PANEL or IMAGE (IMAGE is not ready yet) (unquoted
        string)
       10     SqrtPanels  - Square of panel quantity
     -100     GroundXmin  - Boundary of panels
      800     GroundXmax  - Boundary of panels
     -300     GroundYmin  - Boundary of panels
      300     GroundYmax  - Boundary of panels
------------------------ DYNAMIC STALL ------------------------
    False     DS_Flag     - Whether to use L-B dynamic stall model for calculating lift and drag
    False     RelaxTune   - Whether to tune the relaxation factor to find a stable integrator operating position
        0     StartTime   - Time in seconds to start the DS model
    False     LoadData    - Whether to load pre-calculated dynamic stall data (DO NOT CHANGE)
5MW_DS.dat    LoadFile    - File name for pre-calculated dynamic stall data (no '/' or '\' is allowed) (unquoted string)
      0.3     Indicial1   - Indicial coefs, default: A1 = 0.3
      0.7     Indicial2   - Indicial coefs, default: A2 = 0.7
     0.14     Indicial3   - Indicial coefs, default: b1 = 0.14
     0.53     Indicial4   - Indicial coefs, default: b2 = 0.53
      1.7     TimeConst1  - Time constants, default: Tp = 1.7
      3.0     TimeConst2  - Time constants, default: Tf = 3
      6.0     TimeConst3  - Time constants, default: Tv = 6
     11.0     TimeConst4  - Time constants, default: Tvl = 11
     True     WriteData   - Whether to write dynamic stall data to file
------------------------- ANIMATION --------------------------
    False     AnimFLAG    - Whether generate animation of wake evolution for Paraview (flag)
---------------------- PARALLEL COMPUTATION -------------------
     True     Accelerate  - Whether use Treecode/OpenMP.... (flag)
       20     OpenMPCores - Number of OpenMP cores
---------------------- TREECODE ALGORITHM ---------------------
    False     TreeFlag    - Whether use Treecode (flag). Only works when Accelerate is true
    False     Speedup     - Whether test and record the speedup (flag)
        0     Parallel    - Number of cores to use. '0' is all cores. '1'is no parallel.
      0.5     OpenAngle   - Opening angle theta
        5     TaylorOrder - Taylor expansion order
      500     Maxparnode  - Max particle quantity in one leaf
    10000     Dist_tol    - Ignored beyond this distance
    0.001     Delta       - Smoothing parameter
-------------------------- OUTPUT ----------------------------
    False     Element     - Print Cl, Cd, AOA and V_tot of each blade element (flag)
    False     KJCoverg    - Print the convergence of Kutta Joukowski subroutine at every timestep (flag)
     True     SumPrint    - Print summary data to "WInDS.sum" (flag)
```

Bibliography

[1] J. D. Anderson Jr. *Fundamentals of aerodynamics.* Tata McGraw-Hill Education, 2010.

[2] N. B. deVelder. Free wake potential flow vortex wind turbine modeling: Advances in parallel processing and integration of ground e ffects. 2014.

[3] E. M. Gaertner. Modeling dynamic stall for a free vortex wake model of a floating offshore wind turbine. 2014.

[4] M. O. L. Hansen, J. N. Sørensen, S. Voutsinas, N. Sørensen, and H. A. Madsen. State of the art in wind turbine aerodynamics and aeroelasticity. *Progress in aerospace sciences*, 42(4):285–330, 2006.

[5] J. M. Jonkman. The new modularization framework for the fast wind turbine cae tool. In *51st AIAA Aerospace Sciences Meeting and 31st ASME Wind Energy Symposium, Grapevine, Texas*, 2013.

[6] J. M. Jonkman and M. L. Buhl Jr. Fast users guide. *Golden, CO: National Renewable Energy Laboratory*, 2005.

[7] J. Katz and A. Plotkin. *Low-speed aerodynamics*, volume 13. Cambridge University Press, 2001.

[8] M. A. Lackner, N. deVelder, and T. Sebastian. On 2d and 3d potential flow models of upwind wind turbine tower interference. *Computers & Fluids*, 71:375–379, 2013.

[9] J. G. Leishman. *Principles of Helicopter Aerodynamics with CD Extra.* Cambridge university press, 2006.

[10] P. Li, H. Johnston, and R. Krasny. A cartesian treecode for screened coulomb interactions. *Journal of Computational Physics*, 228(10):3858–3868, 2009.

[11] K. Lindsay and R. Krasny. A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow. *Journal of Computational Physics*, 172(2):879–907, 2001.

[12] P. J. Moriarty and A. C. Hansen. *AeroDyn theory manual.* National Renewable Energy Laboratory Golden, Colorado, USA, 2005.

[13] T. Sebastian. The aerodynamics and near wake of an offshore floating horizontal axis wind turbine. 2012.

[14] T. Sebastian and M. Lackner. A comparison of first-order aerodynamic analysis methods for floating wind turbines. In *48th AIAA Aerospace Sciences Meeting, Orlando, Florida*, 2010.

[15] G. H. Vatistas, V. Kozel, and W. Mih. A simpler model for concentrated vortices. *Experiments in Fluids*, 11(1):73–76, 1991.