

---

# **ROSCO**

***Release 2.9.0***

**Nikhar J. Abbas, Daniel S. Zalkind**

**Jan 19, 2024**



# CONTENTS

<b>1</b>	<b>Standard ROSCO Workflow</b>	<b>3</b>
<b>2</b>	<b>Installing ROSCO toolset</b>	<b>5</b>
2.1	Complete ROSCO Installation . . . . .	5
2.2	Installing only the ROSCO controller . . . . .	5
2.2.1	Direct Download . . . . .	6
2.2.2	Anaconda Download . . . . .	6
2.2.3	Compile using CMake . . . . .	7
<b>3</b>	<b>ROSCO Examples</b>	<b>9</b>
3.1	Tuning Controllers and Generating DISCON.IN . . . . .	9
3.2	Running OpenFAST Simulations . . . . .	10
3.3	Testing ROSCO . . . . .	10
<b>4</b>	<b>ROSCO Structure: Controller</b>	<b>13</b>
4.1	ROSCO File Structure . . . . .	13
4.2	The DISCON.IN file . . . . .	13
<b>5</b>	<b>ROSCO Structure: Toolbox</b>	<b>19</b>
5.1	ROSCO Toolbox File Structure . . . . .	19
5.1.1	ROSCO_toolbox . . . . .	19
5.1.2	Examples . . . . .	19
5.1.3	ROSCO_testing . . . . .	19
5.1.4	Examples/Test_Cases . . . . .	20
5.1.5	Examples/Tune_Cases . . . . .	20
5.2	The ROSCO Toolbox Tuning File . . . . .	20
5.2.1	Matlab_Toolbox . . . . .	20
<b>6</b>	<b>API changes between versions</b>	<b>21</b>
6.1	2.8.0 to 2.9.0 . . . . .	21
6.2	2.7.0 to 2.8.0 . . . . .	24
6.3	2.6.0 to 2.7.0 . . . . .	25
6.4	2.5.0 to develop . . . . .	26
6.5	ROSCO v2.4.1 to ROSCO v2.5.0 . . . . .	28
<b>7</b>	<b>ROSCO_Toolbox tuning .yaml</b>	<b>29</b>
7.1	path_params . . . . .	29
7.2	turbine_params . . . . .	29
7.3	controller_params . . . . .	30
7.3.1	filter_params . . . . .	36
7.3.2	open_loop . . . . .	37

7.3.3	DISCON . . . . .	38
7.4	linmodel_tuning . . . . .	46
<b>8</b>	<b>How to contribute code to ROSCO</b>	<b>47</b>
8.1	Issues . . . . .	47
8.2	Documentation . . . . .	47
8.3	Testing . . . . .	47
8.4	Pull requests . . . . .	48
<b>9</b>	<b>Running Bladed simulations with ROSCO controller</b>	<b>49</b>
9.1	Bladed versions 4.6 to current (4.12) . . . . .	49
9.2	Bladed 4.5 & earlier . . . . .	50
<b>10</b>	<b>License</b>	<b>53</b>

**Version**

2.9.0

**Date**

Jan 19, 2024

NREL’s “Reference Open Source Controller” (ROSCO) is a reference controller framework that facilitates design and implementation of wind turbine and wind farm controllers for fixed and floating offshore wind turbines.

ROSCO framework includes a large set of available controllers and advanced functionalities that can be combined in a modular fashion based on the intended application and can be easily adapted to a wide variety of wind turbines. For example, ROSCO can be used to design turbine yaw controller along with an individual blade pitch controller with floating platform feedback for an offshore turbine while simulating a pitch actuator fault and running a user-defined torque controller.

ROSCO provides a single framework for designing controllers for onshore and offshore turbines of varying sizes. It can be used to run representative dynamic simulations using OpenFAST. This helps researchers perform ‘apples-to-apples’ comparison of controller capabilities across turbines. Control engineers can also design their own controllers and compare them with reference controller design using ROSCO for existing and new turbines. ROSCO has been used to provide reference controllers for many recent reference turbines including the [IEA 3.4-MW](#) , [IEA 10-MW](#) , [IEA 15-MW](#) and the upcoming [IEA 22-MW](#) turbines.

The ROSCO framework also includes a python based toolbox that primarily enables tuning the controllers. The tuning process is extremely simple where only a few tuning parameters need to be provided. It is not necessary to run aeroelastic simulations or provide linearized state-space models to tune the controllers. The toolbox has other capabilities like simple 1-DOF turbine simulations for quick controller capability verifications, linear model analysis, and parsing of input and output files.

Source code for ROSCO toolset can be found in this [github repository](#) and it can be installed following the instructions provided in [Installing ROSCO toolset](#).

**Documentation Directory**



## STANDARD ROSCO WORKFLOW

Fig. 1.1: ROSCO toolchain general workflow

Fig. 1.1 shows the general workflow for the ROSCO tool-chain with OpenFAST. For the standard use case in OpenFAST (or similar), ROSCO controller needs to be compiled. The controller is a fortran based module that follows the bladed-style control interface. Compiling the controller outputs a dynamic-link library (or equivalent) called `libdiscon.dll` for windows, `libdiscon.so` for linux and, `libdiscon.dylib` for mac-os. Instructions for the compilation are provided in *Installing ROSCO toolset*. Once the controller is compiled the turbine simulation tool must point to the compiled library. In OpenFAST, this is ensured by changing the `DLL_FileName` parameter in the `ServoDyn` input file. This step enables communication between the ROSCO controller and OpenFAST.

The compiled ROSCO controller library requires an input file (generally called `DISCON.IN`). It stores several flags and parameters needed by the controller and is read by the compiled dynamic-link library. Several different `DISCON.IN` files, for various turbines and controller tunings, can use the same dynamic-link library. In OpenFAST, the `DLL_InFile` parameter in the `ServoDyn` input file determines the desired input file.

The ROSCO toolbox is used to tune the ROSCO controller and generate a `DISCON.IN` input file. To tune the controller, ROSCO toolbox needs the OpenFAST model of the turbine and some user inputs in the form of a `tuning.yaml` file. The functionality of ROSCO toolset can be best understood by following the set of included example scripts in *ROSCO Examples*. ROSCO toolset can be installed using the instructions provided in *Installing ROSCO toolset*.





## INSTALLING ROSCO TOOLSET

ROSCO toolsets can be utilized either to run an existing controller or to design and tune a controller from scratch. We recommend using the instructions provided in the *Complete ROSCO Installation* to install the full ROSCO toolset. This allows for full use of the provided functionalities including the controller and toolbox to facilitate controller tuning. However, if only the ROSCO binary is needed (to run an existing controller, for example), then users should follow the instructions provided in *Installing only the ROSCO controller*

### 2.1 Complete ROSCO Installation

Steps for the installation of the complete rosko toolset are:

1. Create a conda environment for ROSCO

```
conda config --add channels conda-forge # (Enable Conda-forge Channel For Conda Package Manager)
conda create -y --name rosko-env python=3.10 # (Create a new environment named "rosko-env" that contains Python 3.8)
conda activate rosko-env # (Activate your "rosko-env" environment)
```

2. Clone and Install the ROSCO toolbox with ROSCO controller

```
git clone https://github.com/NREL/ROSCO.git
cd ROSCO
pip install -e .
```

This step creates the rosko controller binary (`libdiscon.so` (Linux), `libdiscon.dylib` (Mac), or `libdiscon.dll` (Windows)) in the directory `ROSCO/rosko/lib` and installs the python toolbox in the conda environment in the develop mode.

### 2.2 Installing only the ROSCO controller

Table 2.1 provides an overview of the primary methods available for installing only the ROSCO controller binary.

Table 2.1: Methods for Installing the ROSCO Controller

Method	Use Case
<i>Direct Download</i>	Best for users who simply want to use a released version of the controller binary without working through the compilation procedures.
<i>Anaconda Download</i>	Best for users who just want to use the controller binary but prefer to download using the Anaconda package manager.
<i>Compile using CMake</i>	Best for users who need to re-compile the source code often, plan to use non-released versions of ROSCO (including modified source code), or who simply want to compile the controller themselves so they have the full code available locally.

Anaconda is a popular package manager used to distribute software packages of various types. Anaconda is used to download requisite packages and distribute pre-compiled versions of the ROSCO tools. CMake is a build configuration system that creates files as input to a build tool like GNU Make, Visual Studio, or Ninja. CMake does not compile code or run compilers directly, but rather creates the environment needed for another tool to run compilers and create binaries. CMake is used to ease the processes of compiling the ROSCO controller locally. For more information on CMake, please see [understanding CMake](#) in the OpenFAST documentation.

## 2.2.1 Direct Download

The most recent tagged version releases of the controller are [available for download](#). One can simply download these compiled binary files for their system and point to them in their simulation tools (e.g. through `DLL_FileName` in the ServoDyn input file of OpenFAST).

## 2.2.2 Anaconda Download

Using the popular package manager, [Anaconda](#), the tagged 64-bit versions of ROSCO are available through the conda-forge channel. In order to download the most recently compiled version release, from an anaconda powershell (Windows) or terminal (Mac/Linux) window, create a new anaconda virtual environment:

```
conda config --add channels conda-forge
conda create -y --name rosco-env python=3.10
conda activate rosco-env
```

navigate to your desired folder to save the compiled binary using:

```
cd <desired_folder>
```

and download the controller:

```
conda install -y ROSCO
```

This will download a compiled ROSCO binary file into the default filepath for any dynamic libraries downloaded via anaconda while in the ROSCO-env. The ROSCO binary file can be copied to your desired folder using:

```
cp $CONDA_PREFIX/lib/libdiscon.* <desired_folder>
```

on linux or:

```
copy %CONDA_PREFIX%\lib\libdiscon.* <desired_folder>
```

on Windows.

### 2.2.3 Compile using CMake

CMake eases the compiling process significantly. We recommend that users use CMake if at all possible, as we cannot guarantee support for the use of other tools to aid with compiling ROSCO.

On Mac/Linux, standard compilers are generally available without any additional downloads. On 32-bit windows, we recommend that you [install MinGW](#) (Section 2). On 64-bit Windows, you can simply install the MSYS2 toolchain through Anaconda:

```
conda install m2w64-toolchain libpython
conda install cmake make # if Windows users would like to install these in anaconda_
↪environment
```

Once the CMake and the required compilers are downloaded, the following code can be used to compile ROSCO.

```
# Clone ROSCO
git clone https://github.com/NREL/ROSCO.git

# Compile ROSCO
cd ROSCO/rosco/controller
mkdir build
cd build
cmake .. # Mac/linux only
cmake .. -G "MinGW Makefiles" # Windows only
make install
```

This will generate a file called libdiscon.so (Linux), libdiscon.dylib (Mac), or libdiscon.dll (Windows).



## ROSCO EXAMPLES

Methods for reading turbine models, generating the control parameters of a `DISCON.IN` file, and running aeroelastic simulations to test controllers. Reading Turbine Models ————— Control parameters depend on the turbine model. The `rosco.toolbox` uses OpenFAST inputs and an additional `.yaml` formatted file to set up a turbine object in python. Several OpenFAST inputs are located in `Test_Cases/`. The controller tuning `.yaml` are located in `Tune_Cases/`. A detailed description of the ROSCO control inputs and tuning `.yaml` are provided in *The DISCON.IN file* and *ROSCO\_Toolbox tuning .yaml*, respectively.

- `01_turbine_model.py` loads an OpenFAST turbine model and displays a summary of its information

ROSCO requires the power and thrust coefficients for tuning control inputs and running the extended Kalman filter wind speed estimator.

- `02_ccblade.py` runs `cc-blade`, a blade element momentum solver from WISDEM, to generate a  $C_p$  surface.

The `Cp_Cq_Ct.txt` (or similar) file contains the rotor performance tables that are necessary to run the ROSCO controller. This file can be located wherever you desire, just be sure to point to it properly with the `PerfFileName` parameter in `DISCON.IN`.

### 3.1 Tuning Controllers and Generating DISCON.IN

The ROSCO turbine object, which contains turbine information required for controller tuning, along with control parameters in the tuning `yaml` and the  $C_p$  surface are used to generate control parameters and `DISCON.IN` files. To tune the PI gains of the torque control, set `omega_vs` and `zeta_vs` in the `yaml`. Similarly, set `omega_pc` and `zeta_pc` to tune the PI pitch controller; gain scheduling is automatically handled using turbine information. Generally `omega_*` increases the responsiveness of the controller, reducing generator speed variations, but also increases loading on the turbine. `zeta_*` changes the damping of the controller and is generally less important of a tuning parameter, but could also help with loading. The default parameters in `Tune_Cases/` are known to work well with the turbines in this repository.

- `03_tune_controller.py` loads a turbine and tunes the PI control gains
- `04_simple_sim.py` tunes a controller and runs a simple simulation (not using OpenFAST)
- `05_openfast_sim.py` loads a turbine, tunes a controller, and runs an OpenFAST simulation

Each of these examples generates a `DISCON.IN` file, which is an input to `libdiscon.*`. When running the controller in OpenFAST, `DISCON.IN` must be appropriately named using the `DLL_FileName` parameter in `ServoDyn`.

OpenFAST can be installed from [source](#) or in a conda environment using:

```
conda install -c conda-forge openfast
```

ROSCO can implement peak shaving (or thrust clipping) by changing the minimum pitch angle based on the estimated wind speed:

- `06_peak_shaving.py` loads a turbine and tunes a controller with peak shaving.

By setting the `ps_percent` value in the tuning yaml, the minimum pitch versus wind speed table changes and is updated in the `DISCON.IN` file.

ROSCO also contains a method for distributed aerodynamic control (e.g., via trailing edge flaps):

- `09_distributed_aero.py` tunes a controller for distributed aerodynamic control

The ROSCO toolbox also contains methods for working with OpenFAST linear models \* `10_linear_params.py` exports a file of the parameters used for the simplified linear models used to tune ROSCO \* `11_robust_tuning.py` shows how linear models generated using OpenFAST can be used to tune controllers with robust stability properties. \* `12_tune_ipc.py` shows the tuning procedure for IPC

## 3.2 Running OpenFAST Simulations

To run an aeroelastic simulation with ROSCO, the ROSCO input (`DISCON.IN`) must point to a properly formatted `Cp_Cq_Ct.txt` file using the `PerfFileName` parameter. If called from OpenFAST, the main OpenFAST input points to the `ServoDyn` input, which points to the `DISCON.IN` file and the `libdiscon.*` dynamic library.

For example in *Test\_Cases/NREL-5MW*:

- `NREL-5MW.fst` has "`NRELOffshrbaseline5MW_Onshore_ServoDyn.dat`" as the `ServoFile` input
- `NRELOffshrbaseline5MW_Onshore_ServoDyn.dat` has "`../../ROSCO/build/libdiscon.dylib`" as the `DLL_FileName` input and "`DISCON.IN`" as the `DLL_InFile` input. Note that these file paths are relative to the path of the main fast input (`NREL-5MW.fst`)
- `DISCON.IN` has "`Cp_Ct_Cq.NREL5MW.txt`" as the `PerfFileName` input

The `roscotoolbox` has methods for running OpenFAST (and other) binary executables using system calls, as well as post-processing tools in `ofTools/`.

Several example scripts are set up to quickly simulate ROSCO with OpenFAST:

- `05_openfast_sim.py` loads a turbine, tunes a controller, and runs an OpenFAST simulation
- `07_openfast_outputs.py` loads the OpenFAST output files and plots the results
- `08_run_turbsim.py` runs TurbSim, for generating turbulent wind inputs
- `14_open_loop_control.py` runs an OpenFAST simulation with ROSCO providing open loop control inputs

## 3.3 Testing ROSCO

The `roscotoolbox` also contains tools for testing ROSCO in IEC design load cases (DLCs), located in `ROSCO_testing/`. The script `run_Testing.py` allows the user to set up their own set of tests. By setting `testtype`, the user can run a variety of tests:

- `lite`, which runs DLC 1.1 simulations at 5 wind speed from cut-in to cut-out, in 330 second simulations
- `heavy`, which runs DLC 1.3 from cut-in to cut-out in 2 m/s steps and 2 seeds for each, in 630 seconds, as well as DLC 1.4 simulations
- `binary-comp`, where the user can compare `libdiscon.*` dynamic libraries (compiled ROSCO source code), with either a lite or heavy set of simulations
- `discon-comp`, where the user can compare `DISCON.IN` controller tunings (and the compiled ROSCO source is constant)

Setting the `turbine2test` allows the user to test either the IEA-15MW with the UMaine floating semisubmersible or the NREL-5MW reference onshore turbine.





## ROSCO STRUCTURE: CONTROLLER

Here, we give an overview of the structure of the ROSCO controller and how the code is implemented.

---

### 4.1 ROSCO File Structure

The primary functions of the ROSCO toolbox are separated into several files. They include the following:

- `DISCON.f90` is the primary driver function.
- `ReadSetParameters.f90` primarily handles file I/O and the Bladed Interface.
- `ROSCO_Types.f90` allocates variables in memory; it is procedurally generated from `rosco_registry`
- `Constants.f90` establishes some global constants.
- `Controllers.f90` contains the primary controller algorithms (e.g. blade pitch control)
- `ControllerBlocks.f90` contains additional control features that are not necessarily primary controllers (e.g. wind speed estimator)
- `Filters.f90` contains the various filter implementations.
- `Functions.f90` contains various functions used in the controller.
- `ExtControl.f90` contains subroutines for calling external dynamic libraries
- `ROSCO_Helpers.f90` contains subroutines for file I/O and other helpful routines, borrowed heavily from NWTC.IO in OpenFAST
- `ROSCO_IO.f90` is procedurally generated using the `rosco_registry` for writing debug and checkpoint files

### 4.2 The DISCON.IN file

A standard file structure is used as an input to the ROSCO controller. This is, generically, dubbed the DISCON.IN file, though it can be renamed (In [OpenFAST](#), this file is pointed to by `DLL_InFile` in the `ServoDyn` file. Examples of the DISCON.IN file are found in each of the Test Cases in the ROSCO toolbox, and in the `parameter_files` folder of ROSCO.

Table 4.1: DISCON.IN

Primary Section	Variable	Type	Description
DE- BUG CON- TROLLEI FLAGS	LoggingI	Int	0: write no debug files, 1: write standard output .dbg-file, 2: write standard output .dbg-file and complete avrSWAP-array .dbg2-file
	F_LPFTyp	Int	Filter type for generator speed feedback signal. 1: first-order low-pass filter, 2: second-order low-pass filter.
	F_NotchI	Int	Notch filter on the measured generator speed and/or tower fore-aft motion (used for floating). 0: disable, 1: generator speed, 2: tower-top fore-aft motion, 3: generator speed and tower-top fore-aft motion.
	IPC_Cont	Int	Individual Pitch Control (IPC) type for fatigue load reductions (pitch contribution). 0: off, 1: 1P reductions, 2: 1P+2P reductions.
	VS_Contr	Int	Generator torque control mode type. 0: $k\omega^2$ below rated, constant torque above rated, 1: $k\omega^2$ below rated, constant power above rated, 2: TSR tracking PI control below rated, constant torque above rated, 3: TSR tracking PI control below rated, constant torque above rated
	PC_Contr	Int	Blade pitch control mode. 0: No pitch, fix to fine pitch, 1: active PI blade pitch control.
	Y_Contrc	Int	Yaw control mode. 0: no yaw control, 1: yaw rate control, 2: yaw-by-IPC.
	SS_Mode	Int	Setpoint Smoother mode. 0: no set point smoothing, 1: use set point smoothing.
	WE_Mode	Int	Wind speed estimator mode. 0: One-second low pass filtered hub height wind speed, 1: Immersion and Invariance Estimator, 2: Extended Kalman Filter.
	PS_Mode	Int	Pitch saturation mode. 0: no pitch saturation, 1: implement pitch saturation
	SD_Mode	Int	Shutdown mode. 0: no shutdown procedure, 1: shutdown triggered by max blade pitch.
	F1_Mode	Int	Floating feedback mode. 0: no nacelle velocity feedback, 1: nacelle velocity feedback (parallel compensation).
	Flp_Mode	Int	Flap control mode. 0: no flap control, 1: steady state flap angle, 2: PI flap control.
	F_LPFCor	Float	Corner frequency (-3dB point) in the generator speed low-pass filter, [rad/s]
FIL- TERS	F_LPFDam	Float	Damping coefficient in the generator speed low-pass filter, [-]. Only used only when F_FilterType = 2
	F_NotchC	Float	Natural frequency of the notch filter, [rad/s]
	F_NotchB	Float	Notch damping values of numerator and denominator - determines the width and depth of the notch, [-]
	F_SSCor	Float	Corner frequency (-3dB point) in the first order low pass filter for the set point smoother, [rad/s].
	F_FlCor	Float	Corner frequency and damping ratio for the second order low pass filter of the tower-top fore-aft motion for floating feedback control [rad/s, -].
	F_WECor	Float	Corner frequency (-3dB point) in the first order low pass filter for the wind speed estimate [rad/s].
	F_FlHigh	Float	Natural frequency of first-order high-pass filter for nacelle fore-aft motion [rad/s].
	F_FlpCor	Float	Corner frequency and damping ratio in the second order low pass filter of the blade root bending moment for flap control [rad/s, -].
BLADE PITCH CON- TROL	PC_GS_n	Int	Number of gain-scheduling table entries

continues on next page

Table 4.1 – continued from previous page

Primary Section	Variable	Type	Description
	PC_GS_an	Float array, length = PC_GS_n	Gain-schedule table: pitch angles [rad].
	PC_GS_KP	Float array, length = PC_GS_n	Gain-schedule table: pitch controller proportional gains [s].
	PC_GS_KI	Float array, length = PC_GS_n	Gain-schedule table: pitch controller integral gains [-].
	PC_GS_KD	Float array, length = PC_GS_n	Gain-schedule table: pitch controller derivative gains [ $s^2$ ]. Currently unused!
	PC_GS_TF	Float array, length = PC_GS_n	Gain-schedule table: transfer function gains [ $s^2$ ]. Currently unused!
	PC_MaxPi	Float	Maximum physical pitch limit, [rad].
	PC_MinPi	Float	Minimum physical pitch limit, [rad].
	PC_MaxRa	Float	Maximum pitch rate (in absolute value) of pitch controller, [rad/s].
	PC_MinRa	Float	Minimum pitch rate (in absolute value) in pitch controller, [rad/s].
	PC_RefSp	Float	Desired (reference) HSS speed for pitch controller, [rad/s].
	PC_FineP	Float	Below-rated pitch angle set-point, [rad]
	PC_Switc	Float	Angle above lowest PC_MinPi to switch to above rated torque control, [rad]. Used for VS_ControlMode = 0,1.
INDIVIDUAL PITCH CONTROL	IPC_IntS	Float	Integrator saturation point (maximum signal amplitude contribution to pitch from IPC), [rad]
	IPC_KI	Float	Integral gain for the individual pitch controller: first parameter for 1P reductions, second for 2P reductions, [-, -].
	IPC_azic	Float	Phase offset added to the azimuth angle for the individual pitch controller: first parameter for 1P reductions, second for 2P reductions, [rad].
	IPC_Corn	Float	Corner frequency of the first-order actuators model, used to induce a phase lag in the IPC signal [rad/s]. 0: Disable.
VS TORQUE CONTROL	VS_GenEf	Float	Generator efficiency from mechanical power -> electrical power, [should match the efficiency defined in the generator properties!], [%]
	VS_ArSat	Float	Above rated generator torque PI control saturation limit, [Nm].
	VS_MaxRa	Float	Maximum generator torque rate (in absolute value) [Nm/s].
	VS_MaxTq	Float	Maximum generator torque (HSS), [Nm].
	VS_MinTq	Float	Minimum generator torque (HSS) [Nm].

continues on next page

Table 4.1 – continued from previous page

Primary Section	Variable	Type	Description
	VS_MinOM	Float	Cut-in speed towards optimal mode gain path, [rad/s]. Used if VS_ControlMode = 0,1.
	VS_Rgn2K	Float	Generator torque constant in Region 2 (HSS side), [N-m/(rad/s)^2]. Used if VS_ControlMode = 0,1.
	VS_RtPwr	Float	Rated power [W]
	VS_RtTq	Float	Rated torque, [Nm].
	VS_RefSp	Float	Rated generator speed used by torque controller [rad/s].
	VS_n	Int	Number of generator PI torque controller gains. Only 1 is currently supported.
	VS_KP	Float	Proportional gain for generator PI torque controller [1/(rad/s) Nm]. (Used in the transition 2.5 region if VS_ControlMode = 0,1. Always used if VS_ControlMode = 2,3)
	VS_KI	Float	Integral gain for generator PI torque controller [1/rad Nm]. (Only used in the transition 2.5 region if VS_ControlMode = 0,1. Always used if VS_ControlMode = 2,3)
	VS_TSRop	Float	Region 2 tip-speed-ratio [rad]. Generally, the power maximizing TSR. Can use non-optimal TSR for low axial induction rotors.
SET-POINT SMOOTHER	SS_VSGai	Float	Variable speed torque controller setpoint smoother gain, [-].
	SS_PCGai	Float	Collective pitch controller setpoint smoother gain, [-].
WIND SPEED ESTIMATOR	WE_Blade	Float	Blade length (distance from hub center to blade tip), [m]
	WE_CP_n	Int	Number of parameters in the Cp array
	WE_CP	Float Float Float Float	Parameters that define the parameterized CP(lambda) function
	WE_Gamma	Float	Adaption gain for the I&I wind speed estimator algorithm [m/rad]
	WE_Gearb	Float	Gearbox ratio [ $\geq 1$ ], [-]
	WE_Jtot	Float	Total drivetrain inertia, including blades, hub and casted generator inertia to LSS, [kg m^2]
	WE_RhoAi	Float	Air density, [kg m^-3]
	PerfFile	String	File containing rotor performance tables (Cp,Ct,Cq)
	PerfTabl	Int Int	Size of rotor performance tables in PerfFileName, first number refers to number of blade pitch angles (num columns), second number refers to number of tip-speed ratios (num rows)
	WE_FOPol	Int	Number of first-order system poles used in the Extended Kalman Filter
	WE_FOPol	Float array, length = WE_FOPol	Wind speeds for first-order system poles lookup table [m/s]
	WE_FOPol	Float array, length = WE_FOPol	First order system poles [1/s]

continues on next page

Table 4.1 – continued from previous page

Primary Section	Variable	Type	Description
YAW CONTROL	Y_ErrThr	Float	Yaw error threshold. Turbine begins to yaw when it passes this. [rad <sup>2</sup> s]
	Y_IPC_In	Float	Integrator saturation (maximum signal amplitude contribution to pitch from yaw-by-IPC), [rad]
	Y_IPC_n	Int	Number of controller gains for yaw-by-IPC
	Y_IPC_KP	Float array, length = Y_IPC_n	Yaw-by-IPC proportional controller gains Kp [s]
	Y_IPC_KI	Float array, length = Y_IPC_n	Yaw-by-IPC integral controller gain Ki [-]
	Y_IPC_or	Float	Low-pass filter corner frequency for the Yaw-by-IPC controller to filtering the yaw alignment error, [rad/s].
	Y_IPC_ze	Float	Low-pass filter damping factor for the Yaw-by-IPC controller to filtering the yaw alignment error, [-].
	Y_MErrSe	Float	Yaw alignment error set point, [rad].
	Y_omegaI	Float	Corner frequency fast low pass filter, [rad/s].
	Y_omegaL	Float	Corner frequency slow low pass filter, [rad/s].
	Y_Rate	Float	Yaw rate, [rad/s].
TOWER FORE-AFT DAMPING	FA_KI	Float	Integral gain for the fore-aft tower damper controller [rad*s/m]. -1 = off
	FA_HPF_C	Float	Corner frequency (-3dB point) in the high-pass filter on the fore-aft acceleration signal [rad/s]
	FA_IntSa	Float	Integrator saturation (maximum signal amplitude contribution to pitch from FA damper), [rad]
MINIMUM PITCH SATURATION	PS_BldPi	Int	Number of values in minimum blade pitch lookup table.
	PS_WindS	Float array, length = PS_BldPi	Wind speeds corresponding to minimum blade pitch angles [m/s]
	PS_BldPi	Float array, length = PS_BldPi	Minimum blade pitch angles [rad]
SHUT-DOWN	SD_MaxPi	Float	Maximum blade pitch angle to initiate shutdown, [rad]
	SD_Corne	Float	Cutoff Frequency for first order low-pass filter for blade pitch angle, [rad/s]

continues on next page

Table 4.1 – continued from previous page

Primary Section	Variable	Type	Description
FLOAT- ING FLAP ACTU- ATION	Fl_Kp	Float	Nacelle velocity proportional feedback gain [s]
	Flp_Angl	Float	Initial or steady state flap angle [rad]
	Flp_Kp	Float	Trailing edge flap control proportional gain [s]
	Flp_Ki	Float	Trailing edge flap control integral gain [s]
	Flp_MaxF	Float	Maximum (and minimum) flap angle [rad]

## ROSCO STRUCTURE: TOOLBOX

Here, we give an overview of the structure of the ROSCO toolbox and how the code is implemented.

---

### 5.1 ROSCO Toolbox File Structure

The primary tools of the ROSCO toolbox are separated into several folders. They include the following:

#### 5.1.1 ROSCO\_toolbox

The source code for the ROSCO toolbox generic tuning implementations lives here.

- `turbine.py` loads a wind turbine model from [OpenFAST](#) input files.
- `controller.py` contains the generic controller tuning scripts
- `utilities.py` has most of the input/output file management scripts
- `control_interface.py` enables a python interface to the ROSCO controller
- `sim.py` is a simple 1-DOF model simulator
- **ofTools** is a folder containing a large set of tools to handle [OpenFAST](#) input files - this is primarily used to run large simulation sets and to handle reading and processing of OpenFAST input and output files.

#### 5.1.2 Examples

A number of examples are included to showcase the numerous capabilities of the ROSCO toolbox; they are described in the [ROSCO Examples](#).

#### 5.1.3 ROSCO\_testing

Testing scripts for the ROSCO toolbox are held here and showcased with `run_testing.py`. These can be used to compare different controller tunings or different controllers all together.

### 5.1.4 Examples/Test\_Cases

Example OpenFAST models consistent with the latest release of OpenFAST are provided here for simple testing and simulation cases.

### 5.1.5 Examples/Tune\_Cases

Some example tuning scripts and tuning input files are provided here. The code found in `tune_ROSCO.py` can be modified by the user to easily enable tuning of their own wind turbine model.

## 5.2 The ROSCO Toolbox Tuning File

A `yaml` formatted input file is used for the standard ROSCO toolbox tuning process. This file contains the necessary inputs for the ROSCO toolbox to load an OpenFAST input file deck and tune the ROSCO controller. It can be found here: *ROSCO\_Toolbox tuning .yaml*.

### 5.2.1 Matlab\_Toolbox

A simulink implementation of the ROSCO controller is included in the Matlab Toolbox. Some requisite MATLAB utility scripts are also included. These scripts are not maintained by NREL as of ROSCO v2.5.0. The Simulink controller there should not be used and referenced as “ROSCO” because it has never been validated against the official ROSCO dynamic library and has drifted away from the official implementation. We will leave the implementation in place to be used only as a learning tool.



## API CHANGES BETWEEN VERSIONS

This page lists the main changes in the ROSCO API (input file) between different versions.

The changes are tabulated according to the line number, and flag name. The line number corresponds to the resulting line number after all changes are implemented. Thus, be sure to implement each in order so that subsequent line numbers are correct.

### 6.1 2.8.0 to 2.9.0

#### Flag to use extended Bladed Interface

- Set *Ext\_Interface* to 1 to use the extended bladed interface with OpenFAST v3.5.0 and greater

#### Gain scheduling of floating feedback

- The floating feedback gain can be scheduled on the low pass filtered wind speed signal. Note that *Fl\_Kp* can now be an array.

#### Rotor position tracking

- Control the azimuth position of the rotor with *OL\_Mode* of 2 using a PID torque controller with gains defined by *RP\_Gains*.
- Control all three blade pitch inputs in open loop

#### New torque control mode settings

- *VS\_ControlMode* determines how the generator speed set point is determined: using the WSE (mode 2) or  $(P/K)^{1/3}$  (mode 3). The power signal in mode 3 is filtered using *VS\_PwrFiltF*.
- *VS\_ConstPower* determines whether constant power is used (0 is constant torque, 1 is constant power)

#### Multiple notch filters

- Users can list any number of notch filters and apply them to either the generator speed and/or tower top acceleration signal based on their index

#### Power reference control via generator speed set points

- With this feature, enabled with *PRC\_Mode*, a user can prescribe a set of generator speed set points (*PRC\_GenSpeeds*) vs. the estimated wind speed (*PRC\_WindSpeeds*), which can be used to avoid certain natural frequencies or implement a soft cut-out scheme.
- A low pass filter with frequency *PRC\_LPF\_Freq* is used to filter the wind speed estimate. A lower value increases the stability of the generator speed reference signal.

#### ZeroMQ Interface

- Each turbine is assigned a *ZMQ\_ID* by the controller, which is tracked by a farm-level controller

### Tower resonance avoidance

- When *TRA\_Mode* is 1, change the torque control generator speed setpoint to avoid *TRA\_ExclSpeed* +/- *TRA\_ExclBand*.
- The set point is changed at a slow rate *TRA\_RateLimit* to avoid generator power spikes. *VS\_RefSpd/100* is recommended.

Removed in ROSCO develop		
Line	Input Name	Example Value
11	F_NotchTy	2 ! F_NotchType - Notch on the measured generator speed and/or tower fore-aft motion (for floating) {0: disable, 1: generator speed, 2: tower-top fore-aft motion, 3: generator speed and tower-top fore-aft motion}
35	F_NotchCo	3.35500 ! F_NotchCornerFreq - Natural frequency of the notch filter, [rad/s]
36	F_NotchBe	0.000000 0.250000 ! F_NotchBetaNumDen - Two notch damping values (numerator and denominator, resp) - determines the width and depth of the notch, [-]

New in ROSCO develop		
Line	Input Name	Example Value
7	Ext_Interface	1 ! Ext_Interface - (0 - use standard bladed interface, 1 - Use the extened DLL interface introduced in OpenFAST 3.5.0.)
14	VS_ConstPow	0 ! VS_ConstPower - Do constant power torque control, where above rated torque varies, 0 for constant torque}
18	PRC_Mode	0 ! PRC_Mode - Power reference tracking mode{0: use standard rotor speed set points, 1: use PRC rotor speed setpoints}
38	F_NumNotch	1 ! F_NumNotchFilt - Number of notch filters placed on sensors
39	F_NotchFreqs	3.3550 ! F_NotchFreqs - Natural frequency of the notch filters. Array with length F_NumNotchFilt
40	F_NotchBetaNum	0.0000 ! F_NotchBetaNum - Damping value of numerator (determines the width of notch). Array with length F_NumNotchFilt, [-]
41	F_NotchBetaDen	0.2500 ! F_NotchBetaDen - Damping value of denominator (determines the depth of notch). Array with length F_NumNotchFilt, [-]
42	F_GenSpdNotch_N	0 ! F_GenSpdNotch_N - Number of notch filters on generator speed
43	F_GenSpdNotch_Ind	0 ! F_GenSpdNotch_Ind - Indices of notch filters on generator speed
44	F_TwrTopNotch_N	1 ! F_TwrTopNotch_N - Number of notch filters on tower top acceleration signal
45	F_TwrTopNotch_Ind	1 ! F_TwrTopNotch_Ind - Indices of notch filters on tower top acceleration signal
92	VS_PwrFiltF	0.3140 ! VS_PwrFiltF - Low pass filter on power used to determine generator speed set point. Only used in VS_ControlMode = 3.
98	PRC_Section	!----- POWER REFERENCE TRACKING -----
99	PRC_n	2 ! PRC_n - Number of elements in PRC_WindSpeeds and PRC_GenSpeeds array
100	PRC_LPF_Freq	0.07854 ! PRC_LPF_Freq - Frequency of the low pass filter on the wind speed estimate used to set PRC_GenSpeeds [rad/s]
101	PRC_WindSpeeds	3.0000 25.0000 ! PRC_WindSpeeds - Array of wind speeds used in rotor speed vs. wind speed lookup table [m/s]
102	PRC_GenSpeeds	0.7917 0.7917 ! PRC_GenSpeeds - Array of generator speeds corresponding to PRC_WindSpeeds [rad/s]
103	Empty Line	
128	TRA_ExclSpeed	0.00000 ! TRA_ExclSpeed - Rotor speed for exclusion [LSS, rad/s]
129	TRA_ExclBand	0.00000 ! TRA_ExclBand - Size of the rotor frequency exclusion band [LSS, rad/s]. Torque controller reference will be TRA_ExclSpeed +/- TRA_ExclBand/2
130	TRA_RateLimit	0.00000e+00 ! TRA_RateLimit - Rate limit of change in rotor speed reference [LSS, rad/s]. Suggested to be VS_RefSpd/100.
145	Fl_n	1 ! Fl_n - Number of Fl_Kp gains in gain scheduling, optional with default of 1
147	Fl_U	0.0000 ! Fl_U - Wind speeds for scheduling Fl_Kp, optional if Fl_Kp is single value [m/s]
161	Ind_Azimuth	0 ! Ind_Azimuth - The column in OL_Filename that contains the desired azimuth position in rad (used if OL_Mode = 2)
162	RP_Gains	0.0000 0.0000 0.0000 0.0000 ! RP_Gains - PID gains and Tf of derivative for rotor position control (used if OL_Mode = 2)
186	ZMQ_ID	0 ! ZMQ_ID - Integer identifier of turbine

Changed in ROSCO develop		
Line	In-put Name	Example Value
12	VS_Cc	2 ! VS_ControlMode - Generator torque control mode in above rated conditions (0- no torque control, 1- $k \cdot \omega^2$ with PI transitions, 2- WSE TSR Tracking, 3- Power-based TSR Tracking)}126
		OL_mode 0 ! OL_Mode - Open loop control mode {0: no open loop control, 1: open loop control vs. time, 2: rotor position control}
125	Twr_St	!----- TOWER CONTROL -----
141	Fl_Kp	0.0000 ! Fl_Kp - Nacelle velocity proportional feedback gain [s]
153	Ind_Bl	0 0 0 ! Ind_BldPitch - The columns in OL_Filename that contains the blade pitch (1,2,3) inputs in rad [array]

## 6.2 2.7.0 to 2.8.0

Optional Inputs - ROSCO now reads in the whole input file and searches for keywords to set the inputs. Blank spaces and specific ordering are no longer required. - Input requirements depend on control modes. E.g., open loop inputs are not required if  $OL\_Mode = 0$

- Cable Control - Can control OpenFAST cables (MoorDyn or SubDyn) using ROSCO
- Structural Control - Can control OpenFAST structural control elements (ServoDyn) using ROSCO
- Added Active Wake Control (AWC) implementation

New in ROSCO 2.8.0		
Line	Input Name	Example Value
6	Echo	0 ! Echo - (0 - no Echo, 1 - Echo input data to <RootName>.echo)
25	AWC_Mode	0 ! AWC_Mode - Active wake control mode [0 - not used, 1 - complex number method, 2 - Coleman transform method]
28	CC_Mode	0 ! CC_Mode - Cable control mode [0- unused, 1- User defined, 2- Open loop control]
29	StC_Mode	0 ! StC_Mode - Structural control mode [0- unused, 1- User defined, 2- Open loop control]
139	Ind_CableC	0 ! Ind_CableControl - The column(s) in OL_Filename that contains the cable control inputs in m [Used with CC_Mode = 2, must be the same size as CC_Group_N]
140	Ind_StructC	0 ! Ind_StructControl - The column(s) in OL_Filename that contains the structural control inputs [Used with StC_Mode = 2, must be the same size as StC_Group_N]
148	Empty Line	
149	AWC_Secti	!----- Active Wake Control -----
150	AWC_Num	1 ! AWC_NumModes - AWC- Number of modes to include [-]
151	AWC_n	1 ! AWC_n - AWC azimuthal mode [-] (only used in complex number method)
152	AWC_harm	1 ! AWC_harmonic - AWC Coleman transform harmonic [-] (only used in Coleman transform method)
153	AWC_freq	0.03 ! AWC_freq - AWC frequency [Hz]
154	AWC_amp	2.0 ! AWC_amp - AWC amplitude [deg]
155	AWC_clock	0.0 ! AWC_clockangle - AWC clock angle [deg]
165	Empty Line	
166	CC_Section	!----- Cable Control -----
167	CC_Group_	3 ! CC_Group_N - Number of cable control groups
168	CC_GroupI	2601 2603 2605 ! CC_GroupIndex - First index for cable control group, should correspond to deltaL
169	CC_ActTau	20.000000 ! CC_ActTau - Time constant for line actuator [s]
170	Empty Line	
171	StC_Sector	!----- Structural Controllers -----
172	StC_Group_	3 ! StC_Group_N - Number of cable control groups
173	StC_GroupI	2818 2838 2858 ! StC_GroupIndex - First index for structural control group, options specified in ServoDyn summary output

## 6.3 2.6.0 to 2.7.0

Pitch Faults - Constant pitch actuator offsets (PF\_Mode = 1) IPC Saturation Modes - Added options for saturating the IPC command with the peak shaving limit

New in ROSCO 2.7.0		
Line	Input Name	Example Value
22	PA_Moc	0 ! PA_Mode - Pitch actuator mode {0 - not used, 1 - first order filter, 2 - second order filter}
23	PF_Moc	0 ! PF_Mode - Pitch fault mode {0 - not used, 1 - constant offset on one or more blades}
56	IPC_Sat	2 ! IPC_SatMode - IPC Saturation method (0 - no saturation (except by PC_MinPit), 1 - saturate by PS_BldPitchMin, 2 - saturate softly (full IPC cycle) by PC_MinPit, 3 - saturate softly by PS_BldPitchMin)
139	PF_Sect	! — Pitch Actuator Faults —
140	PF_Offs	0.00000000 0.00000000 0.00000000 ! PF_Offsets - Constant blade pitch offsets for blades 1-3 [rad]
141	Empty Line	

## 6.4 2.5.0 to develop

IPC - A wind speed based soft cut-in using a sigma interpolation is added for the IPC controller

Pitch Actuator - A first or second order filter can be used to model a pitch actuator

External Control Interface - Call another control library from ROSCO

ZeroMQ Interface - Communicate with an external routine via ZeroMQ. Only yaw control currently supported

Updated yaw control - Filter wind direction with deadband, and yaw until direction error changes signs (<https://iopscience.iop.org/article/10.1088/1742-6596/1037/3/032011>)

New in ROSCO 2.6.0		
Line	Input Name	Example Value
19	TD_Mode	0 ! TD_Mode - Tower damper mode {0: no tower damper, 1: feed back translational nacelle acceleration to pitch angle}
22	PA_Mode	0 ! PA_Mode - Pitch actuator mode {0 - not used, 1 - first order filter, 2 - second order filter}
23	Ext_Mode	0 ! Ext_Mode - External control mode {0 - not used, 1 - call external dynamic library}
24	ZMQ_Mode	0 ! ZMQ_Mode - Fuse ZeroMQ interface {0: unused, 1: Yaw Control}
33	F_YawErr	0.17952 ! F_YawErr - Low pass filter corner frequency for yaw controller [rad/s].
54	IPC_Vramp	9.120000 11.400000 ! IPC_Vramp - Start and end wind speeds for cut-in ramp function. First entry: IPC inactive, second entry: IPC fully active. [m/s]
96	Y_uSwitch	0.00000 ! Y_uSwitch - Wind speed to switch between Y_ErrThresh. If zero, only the first value of Y_ErrThresh is used [m/s]
133	Empty Line	N/A
134	PitchActSec	!----- Pitch Actuator Model -----
135	PA_CornerFreq	3.140000000000 ! PA_CornerFreq - Pitch actuator bandwidth/cut-off frequency [rad/s]
136	PA_Damping	0.707000000000 ! PA_Damping - Pitch actuator damping ratio [-, unused if PA_Mode = 1]
137	Empty Line	
138	ExtConSec	!----- External Controller Interface -----
139	DLL_FileName	"unused" ! DLL_FileName - Name/location of the dynamic library in the Bladed-DLL format
140	DLL_InFile	"unused" ! DLL_InFile - Name of input file sent to the DLL (-)
141	DLL_ProcName	"DISCON" ! DLL_ProcName - Name of procedure in DLL to be called (-)
142	Empty Line	
143	ZeroMQSec	!----- ZeroMQ Interface -----
144	ZMQ_CommAddress	"tcp://localhost:5555" ! ZMQ_CommAddress - Communication address for ZMQ server, (e.g. "tcp://localhost:5555")
145	ZMQ_UpdatePeriod	2 ! ZMQ_UpdatePeriod - Call ZeroMQ every [x] seconds, [s]

Modified in ROSCO 2.6.0		
Line	Input Name	Example Value
97	Y_ErrTh	4.000000 8.000000 ! Y_ErrThresh - Yaw error threshold/deadbands. Turbine begins to yaw when it passes this. If Y_uSwitch is zero, only the second value is used. [deg].
98	Y_Rate	0.00870 ! Y_Rate - Yaw rate [rad/s]
99	Y_MErrSet	0.00000 ! Y_MErrSet - Integrator saturation (maximum signal amplitude contribution to pitch from yaw-by-IPC), [rad]

Removed in ROSCO 2.6.0		
Line	Input Name	Example Value
96	Y_IPn	1 ! Y_IPC_n - Number of controller gains (yaw-by-IPC)
99	Y_IPC_omegaLP	0.20940 ! Y_IPC_omegaLP - Low-pass filter corner frequency for the Yaw-by-IPC controller to filtering the yaw alignment error, [rad/s].
100	Y_IPC_zetaLP	1.00000 ! Y_IPC_zetaLP - Low-pass filter damping factor for the Yaw-by-IPC controller to filtering the yaw alignment error, [-].
102	Y_omegaLPIFast	0.20940 ! Y_omegaLPIFast - Corner frequency fast low pass filter, 1.0 [rad/s]
103	Y_omegaLPSlow	0.10470 ! Y_omegaLPSlow - Corner frequency slow low pass filter, 1/60 [rad/s]

## 6.5 ROSCO v2.4.1 to ROSCO v2.5.0

Two filter parameters were added to - change the high pass filter in the floating feedback module - change the low pass filter of the wind speed estimator signal that is used in torque control

Open loop control inputs, users must specify: - The open loop input filename, an example can be found in Examples/Example\_OL\_Input.dat - Indices (columns) of values specified in OL\_Filename

IPC - Proportional Control capabilities were added, 1P and 2P gains should be specified

Line	Input Name	Example Value
20	OL_Mode	0 ! OL_Mode - Open loop control mode {0: no open loop control, 1: open loop control vs. time, 2: open loop control vs. wind speed}
27	F_WECorner	0.20944 ! F_WECornerFreq - Corner frequency (-3dB point) in the first order low pass filter for the wind speed estimate [rad/s].
29	F_FlHighPas	0.01000 ! F_FlHighPassFreq - Natural frequency of first-order high-pass filter for nacelle fore-aft motion [rad/s].
50	IPC_KP	0.000000 0.000000 ! IPC_KP - Proportional gain for the individual pitch controller: first parameter for 1P reductions, second for 2P reductions, [-]
125	OL_Filename	"14_OL_Input.dat" ! OL_Filename - Input file with open loop timeseries (absolute path or relative to this file)
126	Ind_Breakpoi	1 ! Ind_Breakpoint - The column in OL_Filename that contains the breakpoint (time if OL_Mode = 1)
127	Ind_BldPitch	2 ! Ind_BldPitch - The column in OL_Filename that contains the blade pitch input in rad
128	Ind_GenTq	3 ! Ind_GenTq - The column in OL_Filename that contains the generator torque in Nm
129	Ind_YawRate	4 ! Ind_YawRate - The column in OL_Filename that contains the generator torque in Nm



## ROSCO\_TOOLBOX TUNING .YAML

Definition of inputs for ROSCO tuning procedure

toolbox\_schema

### 7.1 path\_params

**FAST\_InputFile**

[String] Name of \*.fst file

**FAST\_directory**

[String] Main OpenFAST model directory, where the \*.fst lives, relative to directory of this yaml (if applicable)

**rotor\_performance\_filename**

[String] Filename for rotor performance text file (if it has been generated by ccblade already), relative to directory of this yaml

### 7.2 turbine\_params

**rotor\_inertia**

[Float, kg m<sup>2</sup>] Rotor inertia [kg m<sup>2</sup>], { Available in Elastodyn .sum file }

**rated\_rotor\_speed**

[Float, rad/s] Rated rotor speed [rad/s]

*Minimum* = 0

**v\_min**

[Float, m/s] Cut-in wind speed of the wind turbine.

*Minimum* = 0

**v\_max**

[Float, m/s] Cut-out wind speed of the wind turbine.

*Minimum* = 0

**max\_pitch\_rate**

[Float, rad/s] Maximum blade pitch rate [rad/s]

*Minimum* = 0

**max\_torque\_rate**

[Float, Nm/s] Maximum torque rate [Nm/s], {~1/4 VS\_RtTq/s}

*Minimum* = 0

**rated\_power**

[Float, W] Rated Power [W]

*Minimum* = 0

**bld\_edgewise\_freq**

[Float, rad/s] Blade edgewise first natural frequency [rad/s]

*Default* = 4.0

*Minimum* = 0

**bld\_flapwise\_freq**

[Float, rad/s] Blade flapwise first natural frequency [rad/s]

*Default* = 0

*Minimum* = 0

**TSR\_operational**

[Float] Optimal tip speed ratio, if 0 the optimal TSR will be determined by the Cp surface

*Default* = 0

*Minimum* = 0

**reynolds\_ref**

[Float] Reynolds number near rated speeds, used to interpolate airfoils, if provided

*Default* = 0

*Minimum* = 0

## 7.3 controller\_params

**LoggingLevel**

[Float] 0- write no debug files, 1- write standard output .dbg-file, 2- write standard output .dbg-file and complete avrSWAP-array .dbg2-file

*Default* = 1

*Minimum* = 0 *Maximum* = 3

**F\_LPFTType**

[Float] 1- first-order low-pass filter, 2- second-order low-pass filter, [rad/s] (currently filters generator speed and pitch control signals)

*Default* = 1

*Minimum* = 1 *Maximum* = 2

**F\_NotchType**

[Float] Notch on the measured generator speed and/or tower fore-aft motion (for floating) {0- disable, 1- generator speed, 2- tower-top fore- aft motion, 3- generator speed and tower-top fore-aft motion}

*Default* = 0

*Minimum* = 0 *Maximum* = 3

**IPC\_ControlMode**

[Float] Turn Individual Pitch Control (IPC) for fatigue load reductions (pitch contribution) (0- off, 1- 1P reductions, 2- 1P+2P reduction)

*Default = 0*

*Minimum = 0 Maximum = 2*

**VS\_ControlMode**

[Float] Generator torque control mode in above rated conditions (0- no torque control, 1-  $k \cdot \omega^2$  with PI transitions, 2- WSE TSR Tracking, 3- Power-based TSR Tracking)

*Default = 2*

*Minimum = 0 Maximum = 3*

**VS\_ConstPower**

[Float] Do constant power torque control, where above rated torque varies, 0 for constant torque

*Default = 0*

*Minimum = 0 Maximum = 1*

**PC\_ControlMode**

[Float] Blade pitch control mode (0- No pitch, fix to fine pitch, 1- active PI blade pitch control)

*Default = 1*

*Minimum = 0 Maximum = 1*

**Y\_ControlMode**

[Float] Yaw control mode (0- no yaw control, 1- yaw rate control, 2- yaw- by-IPC)

*Default = 0*

*Minimum = 0 Maximum = 2*

**SS\_Mode**

[Float] Setpoint Smoother mode (0- no setpoint smoothing, 1- introduce setpoint smoothing)

*Default = 1*

*Minimum = 0 Maximum = 2*

**WE\_Mode**

[Float] Wind speed estimator mode (0- One-second low pass filtered hub height wind speed, 1- Immersion and Invariance Estimator (Ortega et al.))

*Default = 2*

*Minimum = 0 Maximum = 2*

**PS\_Mode**

[Float] Pitch saturation mode (0- no pitch saturation, 1- peak shaving, 2- Cp-maximizing pitch saturation, 3- peak shaving and Cp-maximizing pitch saturation)

*Default = 3*

*Minimum = 0 Maximum = 3*

**SD\_Mode**

[Float] Shutdown mode (0- no shutdown procedure, 1- pitch to max pitch at shutdown)

*Default = 0*

*Minimum = 0 Maximum = 1*

**TD\_Mode**

[Float] Tower damper mode (0- no tower damper, 1- feed back translational nacelle acceleration to pitch angle)

*Default = 0*

*Minimum = 0 Maximum = 1*

**TRA\_Mode**

[Float] Tower resonance avoidance mode (0- no tower resonance avoidance, 1- use torque control setpoints to avoid a specific frequency)

*Default = 0*

*Minimum = 0 Maximum = 1*

**Fl\_Mode**

[Float] Floating specific feedback mode (0- no nacelle velocity feedback, 1 - nacelle velocity feedback, 2 - nacelle pitching acceleration feedback)

*Default = 0*

*Minimum = 0 Maximum = 2*

**Flp\_Mode**

[Float] Flap control mode (0- no flap control, 1- steady state flap angle, 2- Proportional flap control)

*Default = 0*

*Minimum = 0 Maximum = 2*

**PwC\_Mode**

[Float] Active Power Control Mode (0- no active power control 1- constant active power control, 2- open loop power vs time, 3- open loop power vs. wind speed)

*Default = 0*

*Minimum = 0 Maximum = 2*

**ZMQ\_Mode**

[Float] ZMQ Mode (0 - ZMQ Interface, 1 - ZMQ for yaw control)

*Default = 0*

*Minimum = 0 Maximum = 1*

**ZMQ\_UpdatePeriod**

[Float] Call ZeroMQ every [x] seconds, [s]

*Default = 2*

*Minimum = 0*

**PA\_Mode**

[Float] Pitch actuator mode {0 - not used, 1 - first order filter, 2 - second order filter}

*Default = 0*

*Minimum = 0 Maximum = 2*

**PF\_Mode**

[Float] Pitch fault mode {0 - not used, 1 - constant offset on one or more blades}

*Default = 0*

*Minimum = 0 Maximum = 1*

**OL\_Mode**

[Float] Open loop control mode {0- no open loop control, 1- open loop control}

*Default* = 0

*Minimum* = 0 *Maximum* = 2

**AWC\_Mode**

[Float] Active wake control mode {0 - not used, 1 - SNL method, 2 - NREL method}

*Default* = 0

*Minimum* = 0 *Maximum* = 2

**Ext\_Mode**

[Float] External control mode [0 - not used, 1 - call external dynamic library]

*Default* = 0

*Minimum* = 0 *Maximum* = 1

**CC\_Mode**

[Float] Cable control mode [0- unused, 1- User defined, 2- Open loop control]

*Default* = 0

*Minimum* = 0 *Maximum* = 2

**StC\_Mode**

[Float] Structural control mode [0- unused, 1- User defined, 2- Open loop control]

*Default* = 0

*Minimum* = 0 *Maximum* = 2

**U\_pc**

[Array of Floats] List of wind speeds to schedule pitch control zeta and omega

*Default* = [12]

*Minimum* = 0

**zeta\_pc**

[Array of Floats or Float] List of pitch controller desired damping ratio at U\_pc [-]

*Default* = [1.0]

**omega\_pc**

[Array of Floats or Float, rad/s] List of pitch controller desired natural frequency at U\_pc [rad/s]

*Default* = [0.2]

**interp\_type**

[String from, ['sigma', 'linear', 'quadratic', 'cubic']] Type of interpolation between above rated tuning values (only used for multiple pitch controller tuning values)

*Default* = sigma

**zeta\_vs**

[Float] Torque controller desired damping ratio [-]

*Default* = 1.0

*Minimum* = 0

**omega\_vs**

[Float, rad/s] Torque controller desired natural frequency [rad/s]

*Default* = 0.2

*Minimum* = 0

**max\_pitch**

[Float, rad] Maximum pitch angle [rad], {default = 90 degrees }

*Default* = 1.57

**min\_pitch**

[Float, rad] Minimum pitch angle [rad], {default = 0 degrees }

*Default* = 0

**vs\_minspd**

[Float, rad/s] Minimum rotor speed [rad/s], {default = 0 rad/s }

*Default* = 0

**ss\_vsgain**

[Float] Torque controller setpoint smoother gain bias percentage [%, <= 1 ], {default = 100% }

*Default* = 1.0

**ss\_pcgain**

[Float, rad] Pitch controller setpoint smoother gain bias percentage [%, <= 1 ], {default = 0.1% }

*Default* = 0.001

**ps\_percent**

[Float, rad] Percent peak shaving [%, <= 1 ], {default = 80% }

*Default* = 0.8 *Maximum* = 1

**sd\_maxpit**

[Float, rad] Maximum blade pitch angle to initiate shutdown [rad], {default = 40 deg. }

*Default* = 0.6981

**flp\_maxpit**

[Float, rad] Maximum (and minimum) flap pitch angle [rad]

*Default* = 0.1745

**twr\_freq**

[Float, rad/s] Tower natural frequency, for floating only

*Minimum* = 0

**ptfm\_freq**

[Float, rad/s] Platform natural frequency, for floating only

*Minimum* = 0

**WS\_GS\_n**

[Float] Number of wind speed breakpoints

*Default* = 60

*Minimum* = 0

**PC\_GS\_n**

[Float] Number of pitch angle gain scheduling breakpoints

*Default* = 30

*Minimum* = 0

### **Kp\_float**

[Float, s or Array of Floats] Gain(s) of floating feedback control

### **tune\_Fl**

[Boolean] Whether to automatically tune Kp\_float

*Default* = True

### **U\_Fl**

[Array of Floats or String or Float] List of wind speeds for tuning floating feedback, or “all” for all above-rated wind speeds

*Default* = []

### **zeta\_flp**

[Float] Flap controller desired damping ratio [-]

*Minimum* = 0

### **omega\_flp**

[Float, rad/s] Flap controller desired natural frequency [rad/s]

*Minimum* = 0

### **flp\_kp\_norm**

[Float] Flap controller normalization term for DC gain (kappa)

*Minimum* = 0

### **flp\_tau**

[Float, s] Flap controller time constant for integral gain

*Minimum* = 0

### **max\_torque\_factor**

[Float] Maximum torque = rated torque \* max\_torque\_factor

*Default* = 1.1

*Minimum* = 0

### **IPC\_Kp1p**

[Float, s] Proportional gain for IPC, 1P [s]

*Default* = 0.0

*Minimum* = 0

### **IPC\_Kp2p**

[Float] Proportional gain for IPC, 2P [-]

*Default* = 0.0

*Minimum* = 0

### **IPC\_Ki1p**

[Float, s] Integral gain for IPC, 1P [s]

*Default* = 0.0

*Minimum* = 0

**IPC\_Ki2p**

[Float] integral gain for IPC, 2P [-]

*Default* = 0.0

*Minimum* = 0

**IPC\_Vramp**

[Array of Floats] wind speeds for IPC cut-in sigma function [m/s]

*Default* = [0.0, 0.0]

*Minimum* = 0.0

**rgn2k\_factor**

[Float] Factor on VS\_Rgn2K to increase/decrease optimal torque control gain, default is 1. Sometimes environmental conditions or differences in BEM solvers necessitate this change.

*Default* = 1

*Minimum* = 0

### 7.3.1 filter\_params

**f\_lpf\_cornerfreq**

[Float, rad/s] Corner frequency (-3dB point) in the first order low pass filter of the generator speed [rad/s]

*Minimum* = 0

**f\_lpf\_damping**

[Float, rad/s] Damping ratio in the first order low pass filter of the generator speed [-]

*Minimum* = 0

**f\_we\_cornerfreq**

[Float, rad/s] Corner frequency (-3dB point) in the first order low pass filter for the wind speed estimate [rad/s]

*Default* = 0.20944

*Minimum* = 0

**f\_fl\_highpassfreq**

[Float, rad/s] Natural frequency of first-order high-pass filter for nacelle fore-aft motion [rad/s]

*Default* = 0.01042

*Minimum* = 0

**f\_ss\_cornerfreq**

[Float, rad/s] First order low-pass filter cornering frequency for setpoint smoother [rad/s]

*Default* = 0.6283

*Minimum* = 0

**f\_yawerr**

[Float, rad/s] Low pass filter corner frequency for yaw controller [rad/

*Default* = 0.17952

*Minimum* = 0

**f\_sd\_cornerfreq**

[Float, rad] Cutoff Frequency for first order low-pass filter for blade pitch angle [rad/s], {default = 0.41888 ~ time constant of 15s}



*Default* = 0.41888

### 7.3.2 open\_loop

#### **flag**

[Boolean] Flag to use open loop control

*Default* = False

#### **filename**

[String] Filename of open loop input that ROSCO reads

*Default* = unused

#### **Ind\_Breakpoint**

[Float] Index (column, 1-indexed) of breakpoint (time) in open loop index

*Default* = 1

*Minimum* = 0

#### **Ind\_BldPitch**

[Array of Floats] Indices (columns, 1-indexed) of pitch (1,2,3) inputs in open loop input

*Default* = [0, 0, 0]

*Minimum* = 0

#### **Ind\_GenTq**

[Float] Index (column, 1-indexed) of generator torque in open loop input

*Default* = 0

*Minimum* = 0

#### **Ind\_YawRate**

[Float] Index (column, 1-indexed) of nacelle yaw in open loop input

*Default* = 0

*Minimum* = 0

#### **Ind\_Azimuth**

[Float] The column in OL\_Filename that contains the desired azimuth position in rad (used if OL\_Mode = 2)

*Default* = 0

#### **Ind\_CableControl**

[Array of Floats] The column in OL\_Filename that contains the cable control inputs in m

#### **Ind\_StructControl**

[Array of Floats] The column in OL\_Filename that contains the structural control inputs in various units

#### **PA\_CornerFreq**

[Float, rad/s] Pitch actuator natural frequency [rad/s]

*Default* = 3.14

*Minimum* = 0

#### **PA\_Damping**

[Float] Pitch actuator damping ratio [-]

*Default* = 0.707

*Minimum = 0*

### 7.3.3 DISCON

These are pass-through parameters for the DISCON.IN file. Use with caution. Do not set defaults in schema.

#### LoggingLevel

[Float] (0- write no debug files, 1- write standard output .dbg-file, 2- write standard output .dbg-file and complete avrSWAP-array .dbg2-file)

#### Echo

[Float] 0 - no Echo, 1 - Echo input data to <RootName>.echo

*Default = 0*

#### DT\_Out

[Float] Time step to output .dbg\* files, or 0 to match sampling period of OpenFAST

*Default = 0*

#### F\_LPFType

[Float] 1- first-order low-pass filter, 2- second-order low-pass filter (currently filters generator speed and pitch control signals)

#### VS\_ControlMode

[Float] Generator torque control mode in above rated conditions (0- no torque control, 1-  $k \cdot \omega^2$  with PI transitions, 2- WSE TSR Tracking, 3- Power-based TSR Tracking)

*Minimum = 0 Maximum = 3*

#### VS\_ConstPower

[Float] Do constant power torque control, where above rated torque varies

*Minimum = 0 Maximum = 1*

#### F\_NotchType

[Float] Notch on the measured generator speed and/or tower fore-aft motion (for floating) (0- disable, 1- generator speed, 2- tower-top fore- aft motion, 3- generator speed and tower-top fore-aft motion)

#### IPC\_ControlMode

[Float] Turn Individual Pitch Control (IPC) for fatigue load reductions (pitch contribution) (0- off, 1- 1P reductions, 2- 1P+2P reductions)

#### PC\_ControlMode

[Float] Blade pitch control mode (0- No pitch, fix to fine pitch, 1- active PI blade pitch control)

#### Y\_ControlMode

[Float] Yaw control mode (0- no yaw control, 1- yaw rate control, 2- yaw- by-IPC)

#### SS\_Mode

[Float] Setpoint Smoother mode (0- no setpoint smoothing, 1- introduce setpoint smoothing)

#### WE\_Mode

[Float] Wind speed estimator mode (0- One-second low pass filtered hub height wind speed, 1- Immersion and Invariance Estimator, 2- Extended Kalman Filter)

#### PS\_Mode

[Float] Pitch saturation mode (0- no pitch saturation, 1- implement pitch saturation)

#### SD\_Mode

[Float] Shutdown mode (0- no shutdown procedure, 1- pitch to max pitch at shutdown)

**Fl\_Mode**

[Float] Floating specific feedback mode (0- no nacelle velocity feedback, 1- feed back translational velocity, 2- feed back rotational velocity)

**Flp\_Mode**

[Float] Flap control mode (0- no flap control, 1- steady state flap angle, 2- Proportional flap control)

**OL\_Mode**

[Float] Open loop control mode (0 - no open-loop control, 1 - direct open loop control, 2 - rotor position control)

**F\_LPFCornerFreq**

[Float, rad/s] Corner frequency (-3dB point) in the low-pass filters,

**F\_LPFDamping**

[Float] Damping coefficient (used only when F\_FilterType = 2 [-])

**F\_NumNotchFilt**

[Float] Number of notch filters placed on sensors

**F\_NotchFreqs**

[Array of Floats or Float, rad/s] Natural frequency of the notch filters. Array with length F\_NumNotchFilt

**F\_NotchBetaNum**

[Array of Floats or Float] Damping value of numerator (determines the width of notch). Array with length F\_NumNotchFilt, [-]

**F\_NotchBetaDen**

[Array of Floats or Float] Damping value of denominator (determines the depth of notch). Array with length F\_NumNotchFilt, [-]

**F\_GenSpdNotch\_N**

[Float] Number of notch filters on generator speed

**F\_TwrTopNotch\_N**

[Float] Number of notch filters on tower top acceleration signal

**F\_GenSpdNotch\_Ind**

[Array of Floats or Float] Indices of notch filters on generator speed

**F\_TwrTopNotch\_Ind**

[Array of Floats or Float] Indices of notch filters on tower top acceleration signal

**F\_SSCornerFreq**

[Float, rad/s.] Corner frequency (-3dB point) in the first order low pass filter for the setpoint smoother,

**F\_WECornerFreq**

[Float, rad/s.] Corner frequency (-3dB point) in the first order low pass filter for the wind speed estimate

**F\_FlCornerFreq**

[Array of Floats] Natural frequency and damping in the second order low pass filter of the tower-top fore-aft motion for floating feedback control

**F\_FlHighPassFreq**

[Float, rad/s] Natural frequency of first-order high-pass filter for nacelle fore-aft motion

**F\_FlpCornerFreq**

[Array of Floats] Corner frequency and damping in the second order low pass filter of the blade root bending moment for flap control

**PC\_GS\_n**

[Float] Amount of gain-scheduling table entries

**PC\_GS\_angles**

[Array of Floats] Gain-schedule table- pitch angles

**PC\_GS\_KP**

[Array of Floats] Gain-schedule table- pitch controller kp gains

**PC\_GS\_KI**

[Array of Floats] Gain-schedule table- pitch controller ki gains

**PC\_GS\_KD**

[Array of Floats] Gain-schedule table- pitch controller kd gains

**PC\_GS\_TF**

[Array of Floats] Gain-schedule table- pitch controller tf gains (derivative filter)

**PC\_MaxPit**

[Float, rad] Maximum physical pitch limit,

**PC\_MinPit**

[Float, rad] Minimum physical pitch limit,

**PC\_MaxRat**

[Float, rad/s.] Maximum pitch rate (in absolute value) in pitch controller

**PC\_MinRat**

[Float, rad/s.] Minimum pitch rate (in absolute value) in pitch controller

**PC\_RefSpd**

[Float, rad/s.] Desired (reference) HSS speed for pitch controller

**PC\_FinePit**

[Float, rad] Record 5- Below-rated pitch angle set-point

**PC\_Switch**

[Float, rad] Angle above lowest minimum pitch angle for switch

**IPC\_IntSat**

[Float, rad] Integrator saturation (maximum signal amplitude contribution to pitch from IPC)

**IPC\_SatMode**

[Integer] IPC Saturation method (0 - no saturation, 1 - saturate by PC\_MinPit, 2 - saturate by PS\_BldPitchMin)

**IPC\_KP**

[Array of Floats] Proportional gain for the individual pitch controller- first parameter for 1P reductions, second for 2P reductions, [-]

**IPC\_KI**

[Array of Floats] Integral gain for the individual pitch controller- first parameter for 1P reductions, second for 2P reductions, [-]

**IPC\_aziOffset**

[Array of Floats] Phase offset added to the azimuth angle for the individual pitch controller

**IPC\_CornerFreqAct**

[Float, rad/s] Corner frequency of the first-order actuators model, to induce a phase lag in the IPC signal (0-Disable)

**VS\_GenEff**

[Float, percent] Generator efficiency mechanical power -> electrical power, should match the efficiency defined in the generator properties

**VS\_ArSatTq**

[Float, Nm] Above rated generator torque PI control saturation

**VS\_MaxRat**  
[Float, Nm/s] Maximum torque rate (in absolute value) in torque controller

**VS\_MaxTq**  
[Float, Nm] Maximum generator torque in Region 3 (HSS side)

**VS\_MinTq**  
[Float, Nm] Minimum generator torque (HSS side)

**VS\_MinOMSpd**  
[Float, rad/s] Minimum generator speed

**VS\_Rgn2K**  
[Float, Nm/(rad/s)^2] Generator torque constant in Region 2 (HSS side). Only used in VS\_ControlMode = 1,3

**VS\_RtPwr**  
[Float, W] Wind turbine rated power

**VS\_RtTq**  
[Float, Nm] Rated torque

**VS\_RefSpd**  
[Float, rad/s] Rated generator speed

**VS\_n**  
[Float] Number of generator PI torque controller gains

**VS\_KP**  
[Float] Proportional gain for generator PI torque controller. (Only used in the transitional 2.5 region if VS\_ControlMode  $\neq$  2)

**VS\_KI**  
[Float, s] Integral gain for generator PI torque controller (Only used in the transitional 2.5 region if VS\_ControlMode  $\neq$  2)

**VS\_TSRopt**  
[Float, rad] Power-maximizing region 2 tip-speed-ratio. Only used in VS\_ControlMode = 2.

**VS\_PwrFiltF**  
[Float, rad] Low pass filter on power used to determine generator speed set point. Only used in VS\_ControlMode = 3.  
*Default* = 0.314

**SS\_VSGain**  
[Float] Variable speed torque controller setpoint smoother gain

**SS\_PCGain**  
[Float] Collective pitch controller setpoint smoother gain

**PRC\_Mode**  
[Float] Power reference tracking mode, 0- use standard rotor speed set points, 1- use PRC rotor speed setpoints

**PRC\_WindSpeeds**  
[Array of Floats] Array of wind speeds used in rotor speed vs. wind speed lookup table [m/s]

**PRC\_GenSpeeds**  
[Array of Floats] Array of generator speeds corresponding to PRC\_WindSpeeds [rad/s]

**PRC\_LPF\_Freq**  
[Float] Frequency of the low pass filter on the wind speed estimate used to set PRC\_GenSpeeds [rad/s]  
*Default* = 0.078539

**PRC\_n**

[Float] Number of elements in PRC\_WindSpeeds and PRC\_GenSpeeds array

**TRA\_ExclSpeed**

[Float] Rotor speed for exclusion [LSS, rad/s]

*Default* = 0.0

*Minimum* = 0

**TRA\_ExclBand**

[Float] Size of the rotor frequency exclusion band [LSS, rad/s]. Torque controller reference will be TRA\_ExclSpeed +/- TRA\_ExclBand/2

*Default* = 0.0

*Minimum* = 0

**TRA\_RateLimit**

[Float] Rate limit of change in rotor speed reference [LSS, rad/s]. Suggested to be VS\_RefSpd/400.

*Default* = 0.0

*Minimum* = 0

**WE\_BladeRadius**

[Float, m] Blade length (distance from hub center to blade tip)

**WE\_CP\_n**

[Float] Amount of parameters in the Cp array

**WE\_CP**

[Array of Floats] Parameters that define the parameterized CP(lambda) function

**WE\_Gamma**

[Float, m/rad] Adaption gain of the wind speed estimator algorithm

**WE\_GearboxRatio**

[Float] Gearbox ratio, >=1

**WE\_Jtot**

[Float, kg m<sup>2</sup>] Total drivetrain inertia, including blades, hub and casted generator inertia to LSS

**WE\_RhoAir**

[Float, kg m<sup>-3</sup>] Air density

**PerffFileName**

[String] File containing rotor performance tables (Cp,Ct,Cq) (absolute path or relative to this file)

**PerfTableSize**

[Float] Size of rotor performance tables, first number refers to number of blade pitch angles, second number referse to number of tip-speed ratios

**WE\_FOPoles\_N**

[Float] Number of first-order system poles used in EKF

**WE\_FOPoles\_v**

[Array of Floats] Wind speeds corresponding to first-order system poles

**WE\_FOPoles**

[Array of Floats] First order system poles

**Y\_ErrThresh**

[Float, rad<sup>2</sup> s] Yaw error threshold. Turbine begins to yaw when it passes this

**Y\_IPC\_IntSat**  
[Float, rad] Integrator saturation (maximum signal amplitude contribution to pitch from yaw-by-IPC)

**Y\_IPC\_n**  
[Float] Number of controller gains (yaw-by-IPC)

**Y\_IPC\_KP**  
[Float] Yaw-by-IPC proportional controller gain  $K_p$

**Y\_IPC\_KI**  
[Float] Yaw-by-IPC integral controller gain  $K_i$

**Y\_IPC\_omegaLP**  
[Float, rad/s.] Low-pass filter corner frequency for the Yaw-by-IPC controller to filtering the yaw alignment error

**Y\_IPC\_zetaLP**  
[Float] Low-pass filter damping factor for the Yaw-by-IPC controller to filtering the yaw alignment error.

**Y\_MErrSet**  
[Float, rad] Yaw alignment error, set point

**Y\_omegaLPFast**  
[Float, rad/s] Corner frequency fast low pass filter, 1.0

**Y\_omegaLPSlow**  
[Float, rad/s] Corner frequency slow low pass filter, 1/60

**Y\_Rate**  
[Float, rad/s] Yaw rate

**FA\_KI**  
[Float, rad s/m] Integral gain for the fore-aft tower damper controller, -1 = off / >0 = on

**FA\_HPFCornerFreq**  
[Float, rad/s] Corner frequency (-3dB point) in the high-pass filter on the fore- aft acceleration signal

**FA\_IntSat**  
[Float, rad] Integrator saturation (maximum signal amplitude contribution to pitch from FA damper)

**PS\_BldPitchMin\_N**  
[Float] Number of values in minimum blade pitch lookup table (should equal number of values in PS\_WindSpeeds and PS\_BldPitchMin)

**PS\_WindSpeeds**  
[Array of Floats] Wind speeds corresponding to minimum blade pitch angles

**PS\_BldPitchMin**  
[Array of Floats] Minimum blade pitch angles

**SD\_MaxPit**  
[Float, rad] Maximum blade pitch angle to initiate shutdown

**SD\_CornerFreq**  
[Float, rad/s] Cutoff Frequency for first order low-pass filter for blade pitch angle

**Fl\_n**  
[Float, s] Number of Fl\_Kp gains in gain scheduling, optional with default of 1

**Fl\_Kp**  
[Array of Floats] Nacelle velocity proportional feedback gain

**Fl\_U**  
[Array of Floats] Wind speeds for scheduling Fl\_Kp, optional if Fl\_Kp is single value [m/s]

- Flp\_Angle**  
[Float, rad] Initial or steady state flap angle
- Flp\_Kp**  
[Float, s] Blade root bending moment proportional gain for flap control
- Flp\_Ki**  
[Float] Flap displacement integral gain for flap control
- Flp\_MaxPit**  
[Float, rad] Maximum (and minimum) flap pitch angle
- OL\_Filename**  
[String] Input file with open loop timeseries (absolute path or relative to this file)
- Ind\_Breakpoint**  
[Float] The column in OL\_Filename that contains the breakpoint (time if OL\_Mode > 0)
- Ind\_BldPitch**  
[Float] The column in OL\_Filename that contains the blade pitch input in rad
- Ind\_GenTq**  
[Float] The column in OL\_Filename that contains the generator torque in Nm
- Ind\_YawRate**  
[Float] The column in OL\_Filename that contains the generator torque in Nm
- Ind\_Azimuth**  
[Float] The column in OL\_Filename that contains the desired azimuth position in rad (used if OL\_Mode = 2)
- RP\_Gains**  
[Array of Floats] PID gains and Tf of derivative for rotor position control (used if OL\_Mode = 2)  
*Default* = [0, 0, 0, 0]
- Ind\_CableControl**  
[Array of Floats] The column in OL\_Filename that contains the cable control inputs in m
- Ind\_StructControl**  
[Array of Floats] The column in OL\_Filename that contains the structural control inputs in various units
- DLL\_FileName**  
[String] Name/location of the dynamic library { .dll [Windows] or .so [Linux] } in the Bladed-DLL format  
*Default* = unused
- DLL\_InFile**  
[String] Name of input file sent to the DLL  
*Default* = unused
- DLL\_ProcName**  
[String] Name of procedure in DLL to be called  
*Default* = DISCON
- PF\_Offsets**  
[Array of Floats] Pitch angle offsets for each blade (array with length of 3)  
*Default* = [0, 0, 0]
- CC\_Group\_N**  
[Float] Number of cable control groups  
*Default* = 0



**CC\_GroupIndex**

[Array of Floats] First index for cable control group, should correspond to deltaL

*Default* = [0]

**CC\_ActTau**

[Float] Time constant for line actuator [s]

*Default* = 20

**StC\_Group\_N**

[Float] Number of cable control groups

*Default* = 0

**StC\_GroupIndex**

[Array of Floats] First index for structural control group, options specified in ServoDyn summary output

*Default* = [0]

**AWC\_Mode**

[Float] Active wake control mode {0 - not used, 1 - complex number method, 2 - Coleman transformation method}

*Default* = 0

*Minimum* = 0 *Maximum* = 2

**AWC\_NumModes**

[Float, rad] Number of AWC modes

*Default* = 1

**AWC\_n**

[Array of Floats] AWC azimuthal number (only used in complex number method)

*Default* = [1]

**AWC\_harmonic**

[Array of Integers] AWC Coleman transform harmonic (only used in Coleman transform method)

*Default* = [1]

**AWC\_freq**

[Array of Floats] AWC frequency [Hz]

*Default* = [0.05]

**AWC\_amp**

[Array of Floats] AWC amplitude [deg]

*Default* = [1.0]

**AWC\_clockangle**

[Array of Floats] AWC clock angle [deg]

*Default* = [0]

**ZMQ\_CommAddress**

[String] Communication address for ZMQ server, (e.g. "tcp://localhost:5555")

*Default* = tcp://localhost:5555

**ZMQ\_UpdatePeriod**

[Float] Update period at zmq interface to send measurements and wait for setpoint [sec.]

*Default* = 1.0

### **ZMQ\_ID**

[Float] Integer identifier of turbine

*Default = 0*

## **7.4 linmodel\_tuning**

Inputs used for tuning ROSCO using linear (level 2) models

### **type**

[String from, ['none', 'robust', 'simulation']] Type of level 2 based tuning - robust gain scheduling (robust) or simulation based optimization (simulation)

*Default = none*

### **linfile\_path**

[String] Path to OpenFAST linearization (.lin) files, if they exist

*Default = none*

### **lintune\_outpath**

[String] Path for outputs from linear model based tuning

*Default = lintune\_outfiles*

### **load\_parallel**

[Boolean] Load linearization files in parallel (True/False)

*Default = False*

### **stability\_margin**

[Float or Array of Floats] Desired maximum stability margin

*Default = 0.1*

## HOW TO CONTRIBUTE CODE TO ROSCO

ROSCO is an open-source tool, thus we welcome users to submit additions or fixes to the code to make it better for everybody. When adding new features to ROSCO, we strive to make them as generic as possible, so they can be applied to a variety of turbines. We also hope that new features are also representative of methods in most OEM turbines.

### 8.1 Issues

If you have an issue with ROSCO, a bug to report, or a feature to request, please submit an issue on the GitHub repository. This lets other users know about the issue. If you are comfortable fixing the issue, please do so and submit a pull request.

### 8.2 Documentation

When you add or modify code, make sure to provide relevant documentation that explains the new code. This should be done in code via comments and also in the Sphinx documentation if you add a new feature or capability. Look at the .rst files in the docs section of the repo or click on `view source` on any of the doc pages to see some examples.

The best place to add documentation for your new feature is in an example of the new feature. We are planning to incorporate docstrings from the examples into these docs.

### 8.3 Testing

ROSCO tests its various features through the Examples. Ideally, each new feature would have an associated example. Most examples are set up to simply run the example and check for simulation failures. Some good examples check that the functionality is occurring properly.

An automated testing procedure occurs through GitHub Actions; you can see the progress on the GitHub repo under the `Actions` tab, [located here](#). If any example fails, this information is passed on to GitHub and a red X will be shown next to the commit. Otherwise, if all tests pass, a green check mark appears to signify the code changes are valid.

The examples that are covered are shown in `ROSCO/roscos/test/test_examples.py`. If you add an example to ROSCO, make sure to add a call to it in the `run_examples.py` script as well.

## 8.4 Pull requests

Once you have added or modified code, submit a pull request via the GitHub interface. This will automatically go through all of the tests in the repo to make sure everything is functioning properly. The main developers of ROSCO will then merge in the request or provide feedback on how to improve the contribution.

## RUNNING BLADED SIMULATIONS WITH ROSCO CONTROLLER

ROSCO controller can be used with Bladed.

ROSCO dll must be built to 32bit windows version. Most pre-built discon.dlls in ROSCO github are 64bit, so you may need to build from source.

Configuration in Bladed is as follows:

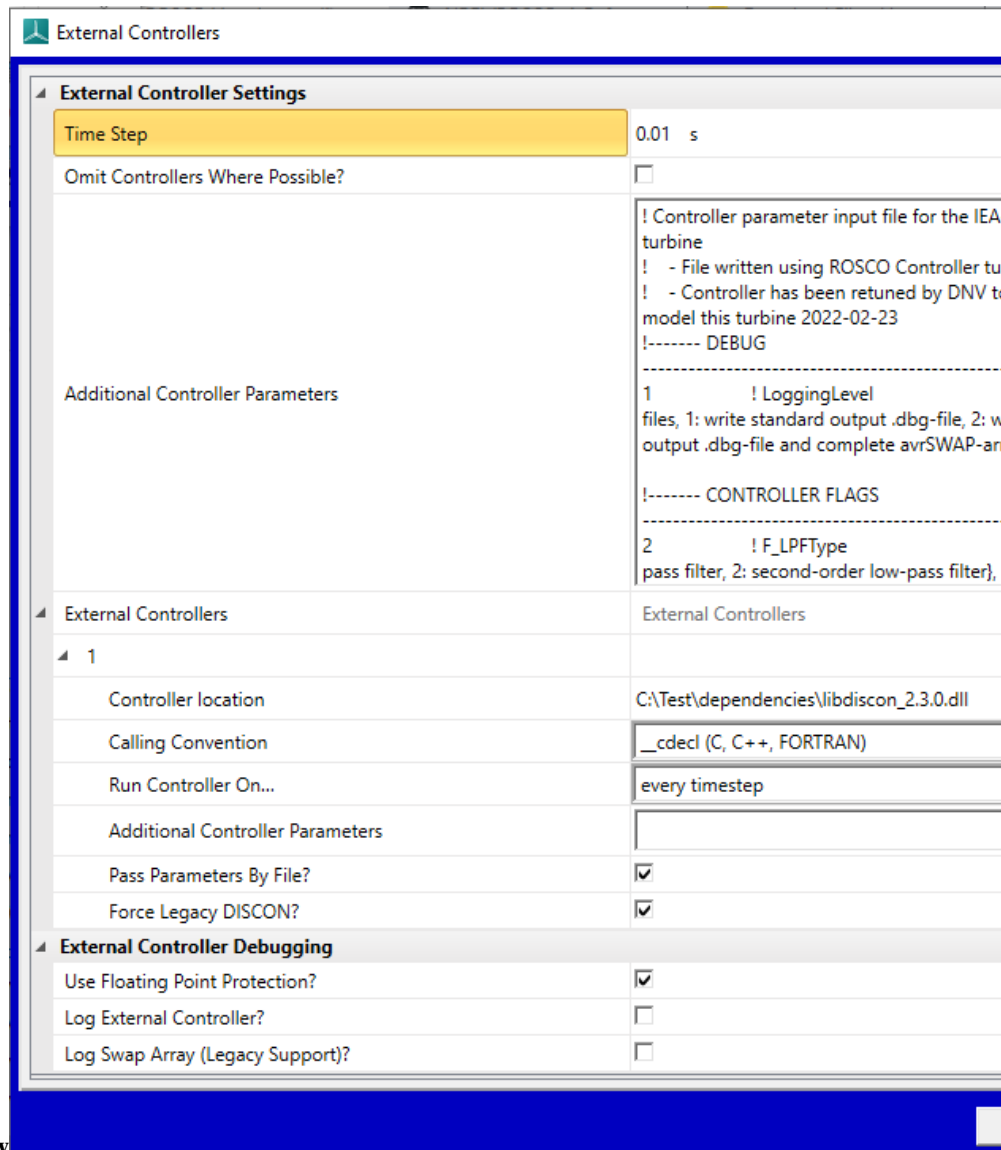
### 9.1 Bladed versions 4.6 to current (4.12)

In the Bladed External Controller dialog, fill in the fields as follows:

- *'Time step'* value non-critical as ROSCO adapts to whatever value is specified. Suggest 10ms.
- *'Additional Controller Parameters'* - copy all text from DISCON.IN for the relevant turbine and paste in to this field.

Notes:

- This must be the topmost field called *'Additional Controller Parameters'*, not the field with the same name lower down in external controller 1 settings.
- Do not add, remove or re-order any lines in this text as this will break ROSCO parsing of the content.
- Any paths such as *PerfFileName* must be absolute (eg *'C:ROSCOconfig.txt'*, not *'..config.txt'*) for use with Bladed.
- Add an external controller (click "+") and set
  - *'Controller location'* - path to the ROSCO libdiscon\_win32.dll
  - *'Calling convention'* - `__cdecl`
  - *'Additional Controller Parameters'* - blank
  - *'Pass parameters by file'* - must be ticked (this instructs Bladed to create a DISCON.IN file at runtime with the text from the Additional Controller Parameters window, and ROSCO reads from this file)
  - *'Force legacy discon'* - ticked



Example setup shown in the image below

## 9.2 Bladed 4.5 & earlier

In External Controller dialog,

- ‘*Communication interval*’ - ROSCO adapts to whatever value is specified. Suggest 10ms.
- ‘*Controller code*’ - path to the ROSCO libdiscon\_win32.dll
- ‘*Calling convention*’ - \_\_cdecl
- ‘*External Controller data*’ - copy the configuration text from DISCON.IN for the relevant turbine and paste in to this field.

Notes:

- Do not add, remove or re-order any lines in this text as this will break ROSCO parsing of the content
- Any paths such as PerfFileName must be absolute (eg ‘C:\ROSCOconfig.txt’, not ‘..config.txt’) for use with Bladed.

**Troubleshooting (all Bladed versions)**

Most error messages from ROSCO are not passed to the Bladed GUI for display or logging. They will be visible only in the transient DOS window that appears while a Bladed simulation is running. If there is a problem you will likely see only 'simulation terminated unexpectedly' in Bladed UI. To view error messages from ROSCO you will need to run Bladed from the command line. This will ensure that the console window where errors are displayed remains open to view the error message.

Instructions to run Bladed from the command line are available [here](#) on the Bladed Knowledge Base





**LICENSE**

Copyright 2021 NREL

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.