Lift Control OpenFAST & Stabilization

- A model-free based control perspective -

LCO/St – version 1.8.2 (Oct. 2024)

USER GUIDE

Loïc Michel, Ph.D.
Ecole Centrale de Nantes
Laboratoire des Sciences du Numérique de Nantes – UMR 6004 CNRS
1 rue de la Noë
44300 Nantes, France

<https://github.com/LoicMichelControl/lcost.git>
(developed and tested under macOS)

Abstract : The Lift Control OpenFAST & Stabilization program, associated with Matlab® and Simulink®, aims to easily use the simulation software "OpenFAST" in order to develop advanced control algorithms dedicated to improve the stability and efficiency of the wind turbines dynamics. A particular model-free based control technique is experimented to control the aerodynamic lift at the blade scale in several aerodynamic conditions.

Table of Contents

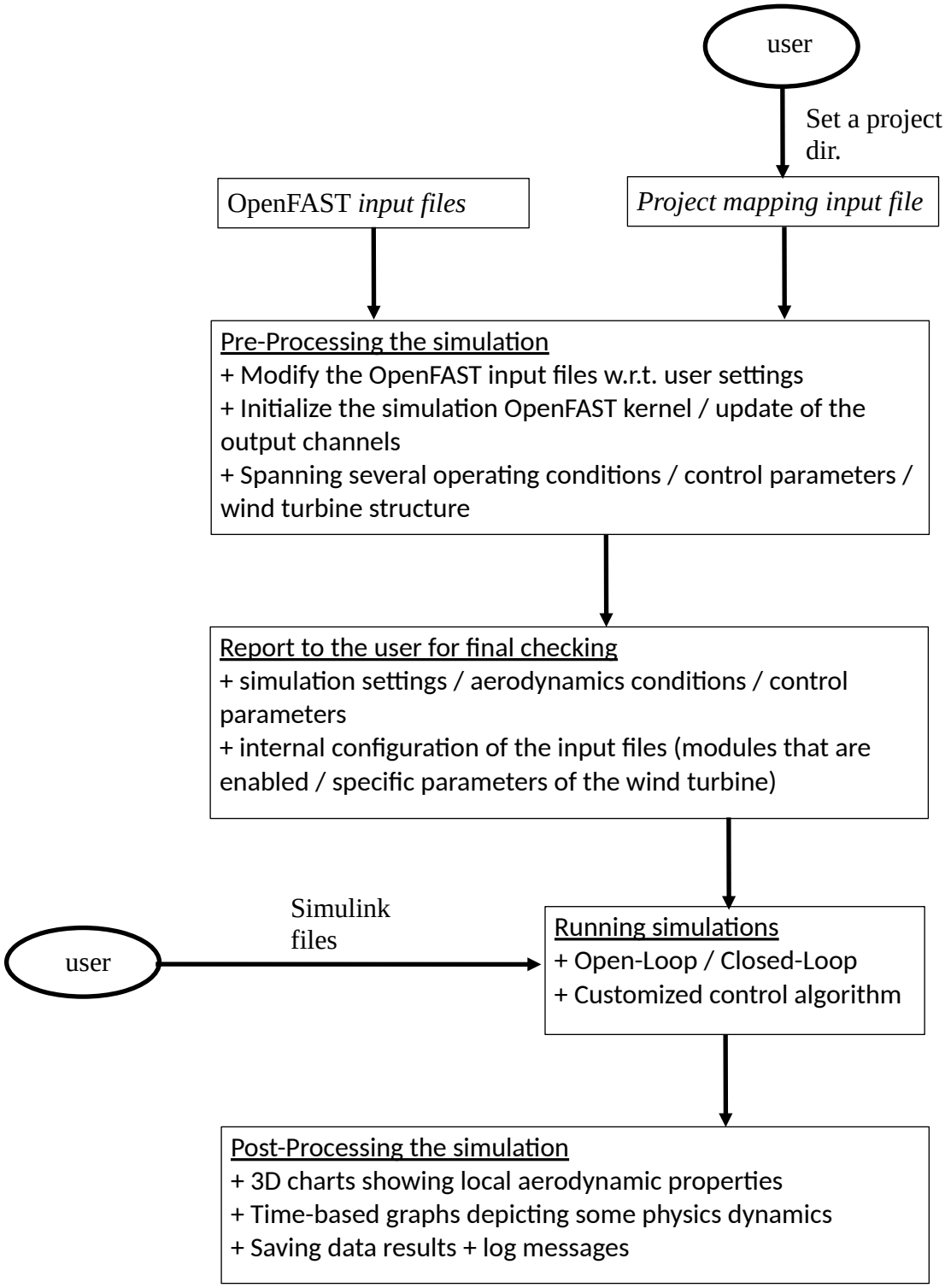
1. Introduction

1.1 Presentation

LCO/St is a Matlab toolbox that helps to use the simulation tool "OpenFAST" [1, 2] as a complete 'toolchain' to simulate the dynamic behavior of wind turbines in order to develop and investigate advanced 'control laws' algorithms. The user sets easily the simulation parameters, designs the control algorithms and LCO/St initializes, runs the OpenFAST simulation kernel and provides output results displayed as time-varying curves and 2D charts for rapid control prototyping. The software has been initially designed to evaluate firstly the efficiency of a recent model-free based control technique proposed by the author [3], for which this algorithm is currently implemented in the associated Simulink files: the current developments focus on the control of the aerodynamic lift at the blade scale of wind turbines. The architecture of the software allows spanning iteratively multiple aerodynamic conditions & control parameters as well as multiple wind turbine structures to compare the efficiency of the control algorithms in different situations. Plots / results and log messages are saved into a working / project directory.
The main purpose is to simulate the control of the aerodynamic lift and the Root Bending Momentum (RBM) at the blade scale of the wind turbine using Model-Free based nonlinear controller [3, 4].

1.2 Main features

+ A *project mapping input file* that sets the complete simulation framework including the aerodynamics operating conditions and the control law parameters.
  Set of a project (working) directory that will contain all saved data.
+ Customization of the input files required by OpenFAST (AeroDyn15, ElastoDyn, ServoDyn, InFlow, *fst file) by the user that sets some customized parameters (through the project mapping file) that are updated in the input files (via the associated Python tools which modify directly the corresponding files).
+ A Matlab/Simulink file makes the coupling (known as 'co-simulation') between the OpenFAST simulation and the control algorithm described in Simulink, so the wind turbine is virtually simulated in the Matlab environment including the interactions with the control laws. Hence, the parameters set by the user concern both the control algorithm in Simulink and the wind turbine simulation (via the OpenFAST kernel).
+ The outputs data provided by the simulator are indexed following an internal nomenclature of OpenFAST (output channels) and depend on the specific needs of the user. Any modification implies a particular internal re-indexation that is automatically updated by LCO/St in a transparent way for the user.
  The initialization procedure of the kernel allows updating such indexes for an efficient readability of the results afterwards.
+ All information regarding the simulation settings (choice of the control reference, aerodynamics conditions and parameters of the control law algorithm) as well as the state of the internal configuration, described in the input files (modules that are enabled / some specific parameters of the wind turbine), are reported to the user for final checking before processing the complete simulation.
+ The modularity allows spanning iteratively multiple aerodynamic conditions & control parameters as well as multiple wind turbine structure to compare the efficiency of the proposed control algorithms in these different situations.
+ The plotting features w.r.t. open-loop / closed-loop simulation include 3D charts (showing local aerodynamic properties) and Time-based graphs depicting some physics dynamics.
+ The results are saved in the project directory for being re-displayed or re-used in another simulations. The Matlab terminal messages are also logged into the project directory.



Work flow chart of LCO/St[1]

1 The slash '/' in the acronym is a small tribute to the 'CP/M' former OS created in 1974 <https://en.wikipedia.org/wiki/CP/M>.

2. LCO/St directories, files and models

2.1 Directories and main files

The main LCO/St directory (typically <src>) is the <root> directory of the software and every working project directory contains a *config file* (parameters of the project) and the resulting plots and log messages. See the "Installation" section to compile and install all required libraries (tested under macOS).

All operational Matlab script files, including the Simulink files, of LCO/St are located strictly in the <root> directory. The Python file to modify the OpenFAST input file is located in </internalPython> (see Section 2.2.d).

The main script of LCO/St to run is "RUN_LiftControl_Stabilization_v1_8". An additional script "Rplot_LiftControl_Stabilization_v1_8" allows plotting the results from prior computations.

The paths of some NREL models are located in the file "openFASTinit.m" and the following models are proposed: 5 MW model (mw5_case) and 15 MW model (mw15_case).

The *config file* contains the full parameters & simulator configuration (aerodynamic and control) to perform a simulation. A *config file* is associated to a single *project* for which a dedicated directory must be created, where all plots and log files will be saved.

The directory </lib> contains all libraries files needed to run OpenFAST. The libraries must be first unzipped from the package 'lib_OpenFAST.zip' (located in the root of the GitHub repository), for which the unzip instruction is proposed in the main LCO/St program.

To run a simulation after having configured the *project* (via the *config file*), just execute the main script RUN_LiftControl_Stabilization_v1.8" (current version 1.8).

2.2 Models

2.2.a Getting the NREL models

The models can be found in the <r-test> directory of the OpenFAST GitHub.

The 5 MW model has been copied in the LCO/St repository (the 15 MW must be unzipped), and has been used for the testing purpose whose preliminary results are presented in the Section "Practical examples".

2.2.b Preparation of the OpenFAST input files

Some mandatory changes are requested to prepare the OpenFAST input files to be used within LCO/St. The four concerned files are (using short name): "ElastoDyn", "InflowWind", "AeroDyn15" and "ServoDyn". Be careful since they may have different full name depending on the considered case (ex. 5 MW or 15 MW). For instance:

- The 5 MW turbine-case

Standard directory: /NREL-5MW/5MW_OC4Sem

```
"NRELOffshrBsline5MW_OC4DeepCwindSemi_ElastoDyn.dat"    ElastoDynFile
"NRELOffshrBsline5MW_InflowWind_Steady8mps.dat"    InflowFile
"NRELOffshrBsline5MW_OC3Hywind_AeroDyn15.dat"    AeroFile
"NRELOffshrBsline5MW_OC4DeepCwindSemi_ServoDyn.dat"    ServoFile
```

- The 15 MW turbine-case

Standard directory: /IEA-15MW/IEA-15-240-RWT-UMaineSemi

```
"IEA-15-240-RWT-UMaineSemi_ElastoDyn.dat" ElastoDynFile
"IEA-15-240-RWT-UMaineSemi_InflowFile.dat" InflowFile
"IEA-15-240-RWT-UMaineSemi_AeroDyn15.dat" AeroFile
"IEA-15-240-RWT-UMaineSemi_ServoDyn.dat" ServoFile
```

These original input files must be saved in the same directory in order to be first duplicated / modified and then processed by OpenFAST (in particular, the inflow file could be located in a different directory).

These original input files *dat + *fst (including the *fst file), are duplicated by LCO/St in the same directory prefixed with the '_' char, thus becoming the modified files (_*dat + _*fst) that could have been possibly modified by the Python script upon request by the user in the *config file*. Even if no modification are requested by the user, the files are duplicated to simplify the management[2].

All the names of the modified files, including the *fst file, are automatically preceded with the '_' char after being copied by LCO/St. The user must modify the original *fst file -> include the '_' char by hand for every file name and remove any prior path (since all files are located in the same directory). The modification of the *fst file must be made only one time since any further modifications in the modified input files do not impact the modified _*fst file. See example of modification below.

It is important that both the original files and the modified files are located in the same directory since LCO/St looks at the original files before dealing with the modified files.

2 This avoid mistakes to modify original files! LCO/St makes a copy of these files (*dat + *fst) that can be re-modified directly by the user through LCO/St while working in the project. There are no history list of the modification, so technically, the Python script is systematically called even if no changes are requested. See Section 2.2.d for more details.

Example of capture of the _*fst file for the 5 MW case (the modified files are in bold pink without prior path since they are located in the same directory).

---------- INPUT FILES ----------

"**_NRELOffshrBsline5MW_OC4DeepCwindSemi_ElastoDyn.dat**" EDFile - Name of file containing ElastoDyn input parameters (quoted string)

"../5MW_Baseline/NRELOffshrBsline5MW_BeamDyn.dat" BDBldFile(1) - Name of file containing BeamDyn input parameters for blade 1 (quoted string)

"../5MW_Baseline/NRELOffshrBsline5MW_BeamDyn.dat" BDBldFile(2) - Name of file containing BeamDyn input parameters for blade 2 (quoted string)

"../5MW_Baseline/NRELOffshrBsline5MW_BeamDyn.dat" BDBldFile(3) - Name of file containing BeamDyn input parameters for blade 3 (quoted string)

"**_NRELOffshrBsline5MW_InflowWind_Steady8mps.dat**" InflowFile - Name of file containing inflow wind input parameters (quoted string)

"**_NRELOffshrBsline5MW_OC3Hywind_AeroDyn15.dat**" AeroFile - Name of file containing aerodynamic input parameters (quoted string)

"**_NRELOffshrBsline5MW_OC4DeepCwindSemi_ServoDyn.dat**" ServoFile - Name of file containing control and electrical-drive input parameters (quoted string)

"**_NRELOffshrBsline5MW_OC4DeepCwindSemi_HydroDyn.dat**" HydroFile - Name of file containing hydrodynamic input parameters (quoted string)

During the initialization of the simulation, LCO/St verifies the format of the modified files in the _*fst file. In case of problem, the following message is displayed:

'Problem with the modified files, check that all sub-files *.dat are preceded with ''_'' in the fst file

2.2.c Preparation of the OpenFAST files for the lift / RBM control including wind turbine parameters monitoring

The same files "ElastoDyn", "InflowWind", "AeroDyn15" and "ServoDyn" must include some essential parameters to be compliant with the expected control of the aerodynamic lift and RBM at the blade scale. See the OpenFAST documentation regarding how to include additional output parameters.

The following list provides the essential and optional (prepared for future investigations) output parameters that should be (at least) included in the OpenFAST input files. From the version 1.8, LCO/St allows adding easily new output parameters to be automatically post-processed[3]: any expected output parameter must be declared first in the corresponding input files of OpenFAST, and then both in the OpenFASTinit.m and the Simulink files (see Section 3.1 for more details).

3 An iterative automatic procedure allows including additional parameters to be processed by LCO/St, without having to re-declare them separately in the source code of LCO/St (this limitation has been removed from v1.8).

Alternatively, any parameter can be monitored directly using the Simulink scope.

1 - "Azimuth" (ElastoDyn)
2 - "Wind1VelX" (InflowWind)
3 - "BldPitch1" (ElastoDyn)
4 - "BldPitch2" (ElastoDyn)
5 - "BldPitch3" (ElastoDyn)
6 - "B1N1Alpha" (AeroDyn15)
7 - "RootMyb1" (ElastoDyn)
8 - "RootMyb2" (ElastoDyn)
9 - "RootMyb3" (ElastoDyn)
10 - "RotSpeed" (ElastoDyn)
11 - "GenSpeed" (ElastoDyn)
12 - "GenPwr" (ServoDyn)
13 - "GenTq" (ServoDyn)
14 - "NacYaw" (ElastoDyn)
15 - "PtfmPitch" (ElastoDyn)
16 - "PtfmRoll" (ElastoDyn)
17 - "PtfmYaw" (ElastoDyn)
18 - "B1N1Fl" (AeroDyn15)
19 - "B1N2Fl" (AeroDyn15)
20 - "B1N3Fl" (AeroDyn15)
21 - "B2N1Fl" (AeroDyn15)
22 - "B2N2Fl" (AeroDyn15)
23 - "B2N3Fl" (AeroDyn15)
24 - "B3N1Fl" (AeroDyn15)
25 - "B3N2Fl" (AeroDyn15)
26 - "B3N3Fl" (AeroDyn15)
27 - "B1N1Cl" (AeroDyn15)
28 - "B2N1Cl" (AeroDyn15)
29 - "B3N1Cl" (AeroDyn15)
30 - “Fl” (AeroDyn15)
31 - “Fy” (AeroDyn15)

According the documentation of OpenFAST, (see 4.8.2.2.13. Nodal Outputs) the particular parameters 30-31 must be declared in two steps in the AeroDyn15 file. An example with the specific format, that must be strictly followed (but the parameters can be changed), is given below:

```
 OutList The next line(s) contains a list of output parameters.
"B1N1Fl, B1N2Fl, B1N3Fl, B2N1Fl, B2N2Fl, B2N3Fl, B3N1Fl, B3N2Fl, B3N3Fl, B1N1Cl,
B2N1Cl, B3N1Cl, B1N1Alpha"
END
---------------------- NODE OUTPUTS --------------------------------------------
3           BldNd_BladesOut - Blades to output
99           BldNd_BlOutNd - Blade nodes on each blade (currently unused)
 OutList_Nodal The next line(s) contains a list of output parameters.  See OutListParameters.xlsx,
ElastoDyn_Nodes tab for a listing of available output channels, (-)
"Fl"
“Fy”
END
------------------------------------------------------------------------------
```

In this example, the output parameters "Fl" and "Fy" are requested to be *vectorized* i.e. the values of Fl and Fy are extracted at each node of the blade, thus creating output matrices. This will be of interest to compute the *lift surface*, which provides a two dimensional map of the evolution of the spatial lift (at each node of the blade) according to the time.

IMPORTANT: In the ServoDyn file, in line 77, the correct library to run the internal servo-control from the turbine side must be provided if used (especially the correct extension with respect to the OS). These libraries are not used in this work.

For example: "../5MW_Baseline/ServoData/DISCON_OC3Hywind.[lib_extension]"

2.2.d Modification of the OpenFAST internal parameters via Python script

The Python(v3) must be installed and configured. The possibility to use Python with LCO/St is disabled by default in the LCO/St package, since the user has to setup properly its Python installation. The Python toolbox, available at: <https://github.com/OpenFAST/python-toolbox> allows modifying in a very convenient way the internal parameters of OpenFAST through a Python script[4].

Download the package via `git clone http://github.com/OpenFAST/python-toolbox install` in the LCO/St main folder and proceed with the installation `python3 -m pip install -e . --break-system-packages` (be careful to the '`-e .`' [there is an extra point alone]) and the modified file ''fast_input_LCOS.py'' is called directly by LCO/St [5].

Warning 1: this Python script requires the 'pandas' package. Check your Python package with `python3 -c 'import pandas'` since an error can occur even if the package 'pandas' has been already installed (probably due to some conflicts if you have many Python versions installed on your system[6]). It can be installed through `python3 -m pip install pandas`.
Warning 2: the file path format is different between OS and may cause severe troubles to call the Python script if not used under macOS. Please contact the author for any assistance (see Section "Acknowledgement and credits").

This Python script is called internally to LCO/St. The script is called:
- 1/ as requested by the user to modify the internal parameters of OpenFAST and the user has only to set a list of pertinent parameters to be modified in the input file of LCO/St (see Section 3.2-8).
- 2/ to modify the initial condition of the blade pitch angle before running simulations (see Section 3.2-4).

The user has to check the proper installation of the 'pandas' library first. The use of Python script is disabled by default in the OpenFASTinit.m file of LCO/St (see Section 4.1).

4 Note that this script can be either invoked directly by Matlab through Python3 or it can be also compiled using for instance Pyinstaller v6.6.0. For the simplicity a direct processing through Python3 is offered to the user.

5 A typical call of this function via LCO/St is `python3 fast_input_file.py /internalPython/_NRELOffshrBsline5MW_OC4DeepCwindSemi_ServoDyn.dat' 'CompElast' 1 0` where the first argument is the name of the (modified) input file to be modified, the sec. Argument is the OpenFAST keyword, and the third argument is the value associated to the keyword. The last argument is for debug purpose and should be =0.

6 Under macOS and the homebrew manager, any version of Python can uninstalled using 'brew uninstall python@[version name]'.

2.3 The Simulink files

The Simulink files (in short, SLX files) make the *bridge* between the control law, calculated under Matlab, and the simulation of the wind turbine using the OpenFAST kernel.

The library **FAST_Sfunc.mexmaci64** must be present in the main LCO/St directory[7]. A Python script is provided in the GitHub repository to automatically download, compile and install the OpenFAST complete environment for Simulink, see the "Installation" section.

Four SLX files are provided depending on the type of simulation:

- open-loop file (OpenLoop_Kernel_1A.slx) that aims to initialize the *bridge* and run open-loop to provide lift / RBM trajectories
- open-loop file (OpenLoop_Kernel_1B.slx) that aims to initialize the *bridge* and run open-loop to provide lift / RBM trajectories + C_L curve outputs
- closed-loop / lift control (ClosedLoop_Kernel_LiftCntrl_1A.slx)
- closed-loop / RBM control (ClosedLoop_Kernel_RBMCntrl_1A.slx)
- closed-loop / lift control by Individual Pitch Control (ClosedLoop_Kernel_Lift_IPCCntrl_1A.slx)

The output parameters of OpenFAST are indexed following the *channels index / numbers* saved in the 'sum' file provided at the end of any OpenFAST regular simulation. This file is read by LCO/St during the initialization to assign index / channels to the associated variables in Simulink. Details will be given in Section 3.1-C-D-E.

3. Preparing a simulation

3.1 Setting the OpenFASTinit.m and Simulink variables

The OpenFASTinit is a script containing the paths and some basic SLX variables. This OpenFASTinit file and the SLX files must be properly parametrized before running a simulation using LCO/St.

A/ The SLX file contains some inputs that must be set prior to any simulations. The following parameters should be set directly in the OpenFASTinit file:

```
GenTorque = 1000;
ElectTorque = 1000;
YawPosition = 0;
YawRate = 0;
ShaftBreaking = 0;
AirfoilCommand = zeros(1,3)';
CableDelta_Commande = zeros(1,20)';
CableDelta_dot_Commande = zeros(1,20)';
```

The values are given here in example and should be set according to the user needs.

B/ The OpenFAST paths containing the paths to the NREL models should be set also in the OpenFASTinit file.

7 The access to the library could be blocked under macOS when executed for the first time.

The <FAST_InputPath> contains the path to the main files of OpenFAST and the <FAST_InputPath_2> contains the path to the secondary files that are also requested by OpenFAST. The example below shows the two usual directory configurations for the 5 MW and the 15 MW cases:

```
if ( mw5_case == 1 ) % 5 MW

FAST_InputPath = './NREL-5MW/5MW_OC4Semi/'; % -main directory
FAST_InputPath_2 = './NREL-5MW/5MW_Baseline/'; %-sec. directory
FAST_aerodyn = '*AeroDyn_blade.dat';
FAST_inflow = '*InflowWind_Steady8mps.dat';

end

if ( mw15_case == 1 ) % 15 MW

FAST_InputPath = './IEA-15MW/IEA-15-240-RWT-UMaineSemi/'; % -main directory
FAST_InputPath_2 = './IEA-15MW/IEA-15-240-RWT/'; %-sec. directory
FAST_aerodyn = '*AeroDyn15_blade.dat';
FAST_inflow = '*InflowFile.dat';

end
```

Also, the "FAST_aerodyn" and "FAST_inflow" variables contain the proper name of the AeroDyn and InflowFile files that can change according to the studied case (it avoids renaming the file). In the proposed example, a choice can be made by the user to select either the 5 MW case of the 15 MW case.

C/ Listing of requested channels in the OpenFASTinit

As described in Sections 2.2.c and 2.3, the output parameters that are requested by the user must be firstly declared in the input files of OpenFAST by the user and then, they are read from the "sum" file (including their assigned index/ channel) while LCO/St initializes the SLX files.

Below is an example of a "sum" file for the fifth first channels:

```
Requested Channels in FAST Output File(s)
-----------------------------------------
Number  Name                 Units              Generated by
------  -------------------  -----------------  ------------
     1  Time                 (s)                OpenFAST
     2  Wind1VelX            (m/s)              InflowWind
     3  Wind1VelY            (m/s)              InflowWind
     4  Wind1VelZ            (m/s)              InflowWind
     5  Azimuth              (deg)              ElastoDyn
```

The channel number is very important since it assigns the correct signals to the associated names in Simulink[8]. Nevertheless, the SLX files require a first initialization for which no prior channel have been defined[9]. Hence, it is absolutely necessary to predefine the channels in the OpenFASTinit file and assign accordingly the channels in the SLX file.

The following format is used for the prior indexing in the OpenFASTinit file: the name of all channels in the "sum" file must be followed by '_index' such as the indexation reads:

(indexed) channel_name [Matlab/SLX] = channel_name [sum file] + '_index'

For example, the list of the required outputs parameters in Section 2.2.c is defined as index-based channels in the OpenFASTinit file and the associated SLX file. The default value of the initialized channels must be set to '1'.

```
Wind1VelX_index = 1;
Azimuth_index = 1;
RotSpeed_index = 1;
BldPitch1_index = 1;
BldPitch2_index = 1;
BldPitch3_index = 1;
PtfmRoll_index = 1;
PtfmPitch_index = 1;
PtfmYaw_index = 1;
GenPwr_index = 1;
GenTq_index = 1;
B1N1Fl_index = 1;
B2N1Fl_index = 1;
B3N1Fl_index = 1;
B1N2Fl_index = 1;
B2N2Fl_index = 1;
B3N2Fl_index = 1;
B1N3Fl_index = 1;
B2N3Fl_index = 1;
B3N3Fl_index = 1;
B1N1Alpha_index = 1;
B1N1Cl_index = 1;
B2N1Cl_index = 1;
B3N1Cl_index = 1;
RootMyb1_index = 1;
RootMyb2_index = 1;
RootMyb3_index = 1;
```

8 The output of a simulation in Simulink is typically a matrix whose rows are the channels requested by the user and indexed explicitly in the "sum" file.

9 The instruction 'Simulink.findVars(SLX file)' lists all variables registered in a Simulink file but a first initialization is required of all Workspace-based variables when starting a first Simulink run. Consequently, it is necessary to declare all 'index' variables that correspond to the expected channels of the 'sum' file.

D/ Misleading between the 'sum' file and the Simulink / OpenFASTinit declarations

LCO/St can detect (from v1.8) any difference between the output channels declared in the OpenFASTinit file and the "sum" file since OpenFASTinit can instantiate more channels than the "sum" file and vice-versa. Any misleading is described in the following two examples:

The example #1, for which LCO/St displays a warning, illustrates the case where the 'sum' file contains more output channels than declared in the OpenFASTinit file.

WARNING: the following outputs (from the 'sum' file) HAVE NOT been assigned in SLX:

〔Wind1VelY〕〔Wind1VelZ〕〔GenSpeed〕 〔PtfmSurge〕〔PtfmSway〕

This warning indicates here that these five output channels are requested by the user (and properly instantiated in the 'sum' file) but these channels have not been declared in OpenFASTinit / Simulink, making them ignored in LCO/St.

The example #2, for which LCO/St displays a warning, illustrates the case where the OpenFASTinit file contains more output channels than declared in the "sum" file.

WARNING: the following outputs (may contain extra output) have not been requested in the 'sum' file:

〔RootMyb2_index〕 〔RootMyb3_index〕 〔RootMyc1_index〕 〔RootMyc2_index〕
〔RootMyc3_index〕

This warning indicates here that these five output channels have been declared in OpenFASTinit / Simulink but they do not have been requested properly by the user when configuring the OpenFAST model.

This second example is probably more critical than the first one since Simulink will simulate output(s) assigned to incorrect index, making therefore the result(s) irrelevant! An illustrative example will be given in Section 3.2-4-A.

E/ Requested outputs in the SLX files

In the SLX files, the bloc "Select the outputs' dispatches each index-based channel to the corresponding output variable.

As stated before, each channel must have a workspace constant bloc in Simulink containing its corresponding **channel_name 〔Matlab/SLX〕** and properly associated to its corresponding output.

The following format is used for the regular outputs from the OpenFAST kernel. The name of all channels in the "sum" file must be followed by '_out' such as the output channel reads:

(output) channel_name 〔Matlab/SLX〕 = channel_name 〔sum file〕 + '_out'

3.2 Configuration of the parameters

A working project is assigned to a (single) dedicated directory that contains a (single) *config file,* all resulting plots and log files. The directory is given in relative path format with respect to the LCO/St root folder.

The *config file* is an (internal) script that defines all the parameters of an OpenFAST-based lift / RBM control simulation.
It parametrizes some of the features of LCO/St according to the needs of the user.

The first step is to setup properly a simulation project through the *config file*. The entries of the c*onfig file* are divided into sections:

- 1. ================ Modification of the OpenFAST input files

The parameter selectModififyConfigFile allows the user to decide how to manage the input files (see Section 2.2.b).

If the modified input files have been already created:

- selectModififyConfigFile = 0 ignores any modification and continues running the LCO/St program;
- selectModififyConfigFile = 1 forces systematically the modifications of the input files with respect to the 'InputSettings' vector defined in the Section 3.2-8;
- selectModififyConfigFile = 2 allows the user choosing which option (ignoring or forcing modifications) and LCO/St asks what to do.

If the modified files have not been created previously, the modification procedure is launched.

- 2. ================ Closed-loop Trajectories settings

The parameter SelectTrajectoryComputation allows the user to choose the type of computation, that manages either a simple open-loop computation (definition of a lift / RBM trajectory and possible lift surface computation) or a closed-loop trajectory involving possibly a prior open-loop computation.

The parameter can be set to (see Sections 3.2-4):

- SelectTrajectoryComputation = 0 requires a pitch trajectory and performs only an open-loop simulation including possibly a lift computing surface (depending on the ComputeSurface parameter, see Section 3.2-5);
- SelectTrajectoryComputation = 1 requires a lift / RBM trajectory and runs directly a closed-loop simulation considering this trajectory profile;
- SelectTrajectoryComputation = 2 requires a pitch trajectory, computes first an open-loop to deduce a lift / RBM trajectory, then performs a closed-loop simulation using the lift / RBM trajectory previously calculated.

Three parameters allow choosing the type of reference for the closed-loop:
- LiftTrajectoryRef = 0 no lift control;
- LiftTrajectoryRef = 1 sets the direct output of the lift (typically B1N1Fl) as the trajectory reference;

- LiftTrajectoryRef = 2 sets the filtered output of the lift (typically the filtered B1N1Fl (through Filter_K)) as the trajectory reference;
- LiftTrajectoryRef = 3 sets the direct output as a constant (whose value is taken as the average of the second half B1N1Fl of the open-loop simulation).

- RBMTrajectoryRef = 0 no RBM control;
- RBMTrajectoryRef = 1 sets the direct output of the RBM (typically RootMyb1) as the trajectory reference;
- RBMTrajectoryRef = 2 sets the filtered output of the RBM (typically the filtered RootMyb1 (through Filter_K)) as the trajectory reference;
- RBMTrajectoryRef = 3 sets the direct output of the RBM as a constant (whose value is taken as the average of the second half of RootMyb1 of the open-loop simulation).

At least one type of reference must be selected (> 0) and LCO/St displays an error in case of no type of reference selected.

The IPC_mode parameter selects the simulation of the lift control in the Individual Pitch Control mode, see Sections 3.2-4-C and 4.2.5 for more details.

-3. ================ Wind Speed settings

A uniform wind inflow is set using a "Uniform Wind Data" file to generate a customized wind speed profile (see Section 4.11.4.1 – 4 of OpenFAST documentation). Only the 'x' direction is considered.

The *config file* allows the user to set directly his own wind profile, that will be internally processed by LCO/St for OpenFAST processing.

The vectors wind_speed_x and wind_speed_t designate respectively the wind speed profile and the specific associated points of time.
For instance:

```
wind_speed_x = [13 13 15 15];
wind_speed_t = [0, 1500, 2000, 3000];
```

illustrate a wind speed profile that evolves from 13 m/s until 1500 sec, to 15 m/s from 2000 sec.

The power law coefficient PwrLaw for the wind profile is updated at the same time as the wind speed profile defined previously.

-4. ================ Trajectory Definition

Depending on the parameter SelectTrajectoryComputation, the user has to choose among the following definitions:

Def. A: the *pitch trajectory* in open-loop: the pitch angle is imposed and the lift / RBM evolves accordingly to its own dynamics with respect to the wind turbine environment;
Def. B: the *direct reference trajectory* in closed-loop: the lift / RBM ref. profile is applied directly as the reference of the closed-loop;

Def. C: the *pitch-based natural reference trajectory* in closed-loop: a first open-loop computation of the lift, given a pitch profile, is performed (similarly to the A/ definition) and

the computed lift is used afterwards as a reference trajectory for the lift / RBM closed-loop (similarly to the B/ definition).

Remark that in the *config file*, Def. B and Def. C are respectively referred to as 'external' lift trajectory and 'internal' lift trajectory[10].

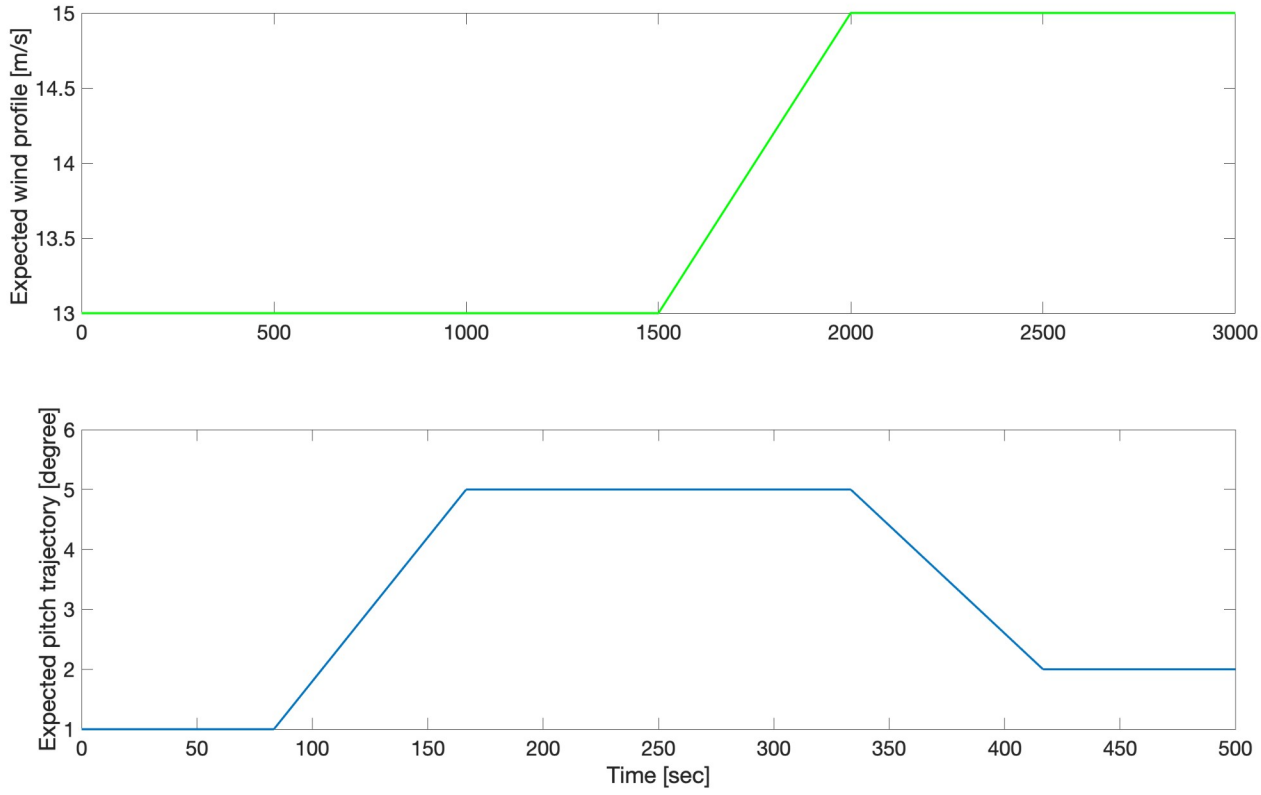
In more details:

Def-A/ If SelectTrajectoryComputation = 0, the wind turbine blade is directly driven by the pitch.

For instance, the following Point_OpenLoop_Pitch_1-2-3 parameters[11]:

```
Point_OpenLoop_Pitch_1 = 1;
Point_OpenLoop_Pitch_2 = 5;
Point_OpenLoop_Pitch_3 = 2;
```

define a pitch blade profile evolving from 1° , to 5° and ending at 2° . The following figure is plotted by LCO/St and illustrates an example of the wind speed profile and the pitch blade profile as (both) defined in the *config file.*



10 This was the original definitions of the trajectories in previous versions of LCO/St.

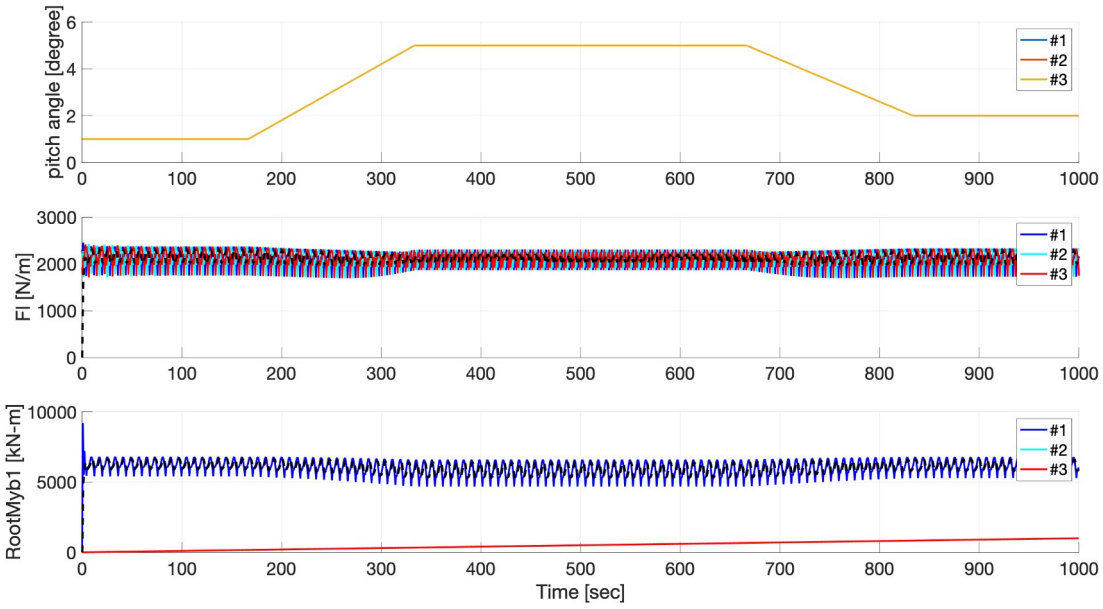
11 The trajectory is divided into five time-segments (over 6 points). In order to reproduce a slope-based variation of the trajectory, the user gives only three points (associated to three steps), for which the trajectory passes by these three steps over the full time-segment. Hence, in future versions of LCO/St, the number of steps should be increased and probably define new type of trajectories.

This figure allows the user to verify the set-points that have been defined in the *config file* before launching the simulations.

Remark that, in this A-definition, the trajectory reference defines the pitch profile (to modify directly the blade pitch).

The results of the open-loop computation are illustrated by graphs showing the evolution of the lift / RBM according to the time, and with respect to the wind and pitch profiles.

As an incorrect example, the following figure illustrates, in particular, an open-loop computation for which an incorrect indexation has been detected by LCO/St, following the example #2 given in Section 3.1-D, where:



the 〔RootMyb2_index〕〔RootMyb3_index〕 have been declared in Simulink but have not been assigned by OpenFAST. It is very important to observe, on the third subplot, that the RootMyb waves are not similar between the three blades.

Def-B/ If SelectTrajectoryComputation = 1, the controlled wind turbine blade tracks the Lift / RBM trajectory.

Regarding the lift tracking, for instance, the following Point_ClosedLoop_Lift_1-2-3 parameters:

```
Point_ClosedLoop_Lift_1 = 2100;
Point_ClosedLoop_Lift_2 = 2100;
Point_ClosedLoop_Lift_3 = 2100;
```

define a constant *reference lift profile*, which will be applied directly as the ref. trajectory of the closed-loop.

Regarding the RBM tracking, for instance, the set of Point_ClosedLoop_RBM_1-2-3 parameters:

Point_ClosedLoop_RBM_1 = 9000;
Point_ClosedLoop_RBM_2 = 9000;
Point_ClosedLoop_RBM_3 = 9000;

defines a constant *reference RBM profile,* which will be applied directly as the ref. trajectory of the closed-loop.

The same type of figure is plotted in the case B/ to allow the user to check the expected trajectory as lift or RBM reference.
Remark that, in this B-definition, the trajectory reference defines the lift / RBM profile that is used directly as the tracking reference for the closed-loop.

Def-C/ If SelectTrajectoryComputation = 2, the controlled wind turbine blade tracks the Lift / RBM trajectory through a prior computation of the reference lift from a pitch profile.

For instance, the pitch profile is given by the following parameters:

Point_ClosedLoop_Pitch_1 = 1;
Point_ClosedLoop_Pitch_2 = 2;
Point_ClosedLoop_Pitch_3 = 2;

Remark that, in this C-definition, the trajectory reference is also defined, like in Def-A/, as a pitch profile, that computes first a "natural" lift profile, used afterward as a reference to track the lift in closed-loop.

The same type of figure is plotted in the case C/ to allow the user to check the expected pitch trajectory.
Finally, note that the A et C definitions compute firstly an open-loop simulation from a pitch profile given by the user. In the Definition A, LCO/St displays only the results of the lift / RBM, whereas in the Definition C, the resulting lift / RBM, computed from the open-loop, is used as the tracking reference for the closed-loop computation.

The individual pitch control (IPC), managed by the parameter IPC_mode can be run only in the Definition-C.

In addition, few more parameters are given:

+ The parameter PitchBlade_0 updates the initial condition of the pitch of the blade in the ElastoDyn file thanks to the Python script (in the same manner that the OpenFAST input files can be modified by LCO/St according to the needs of the user). See Section 2.2.d.

+ The parameter ControlledNode defines the #node on the blade used for the control feedback.

+ The parameter LiftStepRef allows incrementing the tracking reference to induce small variations on the controlled lift (available only for the B-definition): the initial lift / RBM, given by the first point Point_ClosedLoop_Lift/RBM_1 is incremented with the value LiftStepRef every LiftStepRef_stepTime (a period of 500 sec is fixed by default in the *config file*).

-5. ================ Lift Surface Computation

The lift surface computation allows plotting the lift force "Fl" as well as the "Fy" force in 2D (evolution of the Fl or Fy along the blade) with respect to the time.

The lift surface computation is only available in open-loop (SelectTrajectoryComputation = 0).

-6. ================ Closed-loop settings

In this section are given the parameters of the model-free based control algorithm, developed and experimented by the author. The paper that presents the control law including some basic examples can be found on arXiv at <https://arxiv.org/abs/1202.4707> and the most recent application in CFD-based lift control of the blade can be found at <https://hal.science/hal-04520407v1/>[12].

The parameters manage:
- the control law (PMC parameters);
- the Filter_K time-constant filtering for output averaging (AVG output in SLX);
- the ratePitchLimit rate limitation for the pitch (in closed-loop).

Below is given the equation of the control law (screenshot from the report [5]):

A model-free control, without derivative term but with an integrator associated to a forgetting factor was proposed.

$$u_k = \Psi_k \, . \int_0^t K_i (y_k^* - y_k) \, d\tau \tag{1}$$

where k is the discrete iteration index and Ψ_k is a time series that "adjusts" online the gain of the integrator. The term Ψ is given by

$$\Psi_k = \Psi_{k-1} + K_p (K_\alpha e^{-K_\beta . k} - y_k) \tag{2}$$

In (1)-(2), u_k is the control output ; y_k^* is the output reference trajectory; y_k is the output of the controlled system; K_p, K_I, K_α and K_β are real positive tuning gains.

-7. ================ Which output(s) to plot?

This section allows choosing which outputs to plot of the closed-loop with respect to some categories.
Each plot_x variable can select of disable the associated plot.

```
plot_1  % plot 1 - Wind1VelX + pitch angle #1 + Traj. Profile
plot_2  % plot 2 - GenPwr + Gen Tq + GenSpeed
plot_3  % plot 3 - Azimuth + Ptfm angles
```

12 There are currently no systematic rules to tune the control. Please contact the author for any assistance!

plot_4 % plot 4 - Fl#1 + pitch angle
plot_5 % plot 5 - Fl#1 - Fl#2 - Fl#3
plot_5b % plot 5b - Fl#1 + Fl#2 + Fl#3
plot_6 % plot 6 - AVG Fl#1 + pitch angle
plot_7 % plot 7 - pitch angle #1 - #2 - #3
plot_8 % plot 8 - RootMyb #1 + blade pitch
plot_9 % plot 9 - RootMyb #1 - RootMyb #2 - RootMyb #3
plot_10 % plot 10 - AVG Fl#1 - AVG Fl#2 - AVG Fl#3

The modification of the plots can be done in the **ModulePlot_vA** script in the <root> directory.

Additionally, extra_plot, if it exists in the *config file*, can refers to an external script (located in the <root> directory) allowing additional customized plots. The variable extra_plot should be assigned to the name of the script and LCO/St runs this script directly after the regular plots.

-8. ================ Set of input parameters to be modified

In this section, the input parameters of OpenFAST that the user wants to modify are listed in the matrix following format.

InputSettings(**cnt_Input**,:) = ["name of the parameter","value"];

where name of the parameter is the official name given in the OpenFAST nomenclature and the corresponding value.

The counter **cnt_Input** increments the list and should be updated after each matrix instantiation.

-9. ================ SIMULINK FILES

The following variables are assigned to the corresponding SLX files.

OpenLoop_SLXfile (the open-loop case)

ClosedLoop_SLXfile_Lift (the closed-loop case for the lift control)

ClosedLoop_SLXfile_RBM (the closed-loop case for the RBM control)

ClosedLoop_SLXfile_Lift_IPC (the closed-loop case for the lift-based control in the IPC mode)

-10. ================ PLOT THE C_L curve

This optional section refers to the plot of the simulated C_L curves with respect to the selected node ControlledNode. This extra plot is also available in closed-loop provided that the plot_11 variable (in section 7) is enabled.

The variable Cl_curve_Node contains the theoretical C_L curves that the user must copy in order to be compared with the simulation. LCO/St does not read the associated input files that contains these technical data about the blades. See Section 4.2.3 for more details.

If Cl_curve_Node is not present in the *config file*, then the C_L curves are not plotted.

4. Running a simulation

The main script is "RUN_LiftControl_Stabilization_v1_8". An additional script "Rplot_LiftControl_Stabilization_v1_8" allows plotting the results from prior computations.

Having configured properly the *config file*, a simulation can be started provided few general parameters to set.

TMax specifies the simulation time.
OnlyReadingInput only checks OpenFAST & SLX files. No computation is performed.
Start_time_plotting specifies the starting time of the plots[13]
KeepOUTfile saves the 'out' file of OpenFAST in the project folder.

Note that the time-step 'DT' of the simulation can not be modified through LCO/St. A direct modification of the modified *fst file is necessary (located in l. 7).

4.1 General notifications by Matlab terminal

Some of the important messages, displayed by LCO/St, are listed below.

The first part of the messages, before initializing a simulation, concerns mainly the management of the input files (original and modified files – Section 2.2.b).

```
 ---------------------------------------
 Internal files summary :
** Initial files:
Found NRELOffshrBsline5MW_AeroDyn_blade.dat...
Found NRELOffshrBsline5MW_OC3Hywind_AeroDyn15.dat...
Found NRELOffshrBsline5MW_OC4DeepCwindSemi_ElastoDyn.dat...
Found 5MW_OC4Semi_WSt_WavesWN.fst...
** Modified files:
Found _NRELOffshrBsline5MW_OC3Hywind_AeroDyn15.dat...
Found _NRELOffshrBsline5MW_OC4DeepCwindSemi_ElastoDyn.dat...
Found _5MW_OC4Semi_WSt_WavesWN.fst...
Found _5MW_OC4Semi_WSt_WavesWN.sum...
 ---------------------------------------
```

+ If modifications / duplication of the input files are requested, the following message appears:

```
RUN the modification of the OpenFAST files (w.r.t. the 'InputSettings' vector)
```

13 Minor improvement introduced in v1.8.0-b.

+ If modifications / duplication of the input files are not requested, the following message appears:

IGNORE the modification of the OpenFAST files (w.r.t. the 'InputSettings' vector)

+ The use of the Python script is disabled by default in the OpenFASTinit file (see Section 2.2.d), the parameter UsePython has to be enabled (= 1) by the user.

The status of the ServoDyn variables is then displayed:

--------- SERVO-DYN VARIABLES ---------

Checking the co-simulation with Simulink ... OK

WARNING : Variable-speed control disabled - GenModel = 1

WARNING : CPC mode selected (Ptch_Cntrl = 0)

the program pauses a couple of seconds to allow the user checking these important parameters, and in particular the status of the IPC mode.

Then, a reading of the input files is done to display some parameters (related to those subjected to modification by the user):

--------- AeroDyn15 Config ---------

TwrPotent : 1

TwrShadow : 1

TwrAero : True

--------- Degrees of Freedom selected (in ElastoDyn file) ---------

FlapDOF1 -> True

FlapDOF2 -> True

...

PtfmYDOF -> False

--------- Current ElastoDyn Angles :

ShftTilt = -5.000000

PreCone_1 = -2.500000

PreCone_2 = -2.500000

PreCone_3 = -2.500000

--------- Current ElastoDyn IC Pitch :

BlPitch(1) = 0.000000

BlPitch(2) = 0.000000

BlPitch(3) = 0.000000

followed by a review of additional parameters:

--------- Coeff. of the power law (from the input file) 0.300000

--------- Current ElastoDyn RotorSpeed : 7.550000

--------- Current DT selected : 0.012500

--------- Current WindType = 2 with 'Power Law coeff.' = 0.300000

---- Number of Nodes in the blade 19 with 1 Output Node(s) extracted (in AeroDyn15 file) :〔9〕

and the current module configuration:

--------- Current module configuration

CompElast = 1

CompInflow = 1

CompAero = 2

CompServo = 1

CompHydro = 0

CompSub = 0

CompMooring = 0

CompIce = 0

MHK = 0

The input files are checked (one more time) and the current *fst file used in the SLX files is displayed:

--------- INPUT FILE CHECKING ---------

Copied NRELOffshrBsline5MW_OC4DeepCwindSemi_ServoDyn.dat -> _NRELOffshrBsline5MW_OC4DeepCwindSemi_ServoDyn.dat !

Copied NRELOffshrBsline5MW_OC3Hywind_AeroDyn15.dat -> _NRELOffshrBsline5MW_OC3Hywind_AeroDyn15.dat !

Copied NRELOffshrBsline5MW_OC4DeepCwindSemi_ElastoDyn.dat -> _NRELOffshrBsline5MW_OC4DeepCwindSemi_ElastoDyn.dat !

Copied 5MW_OC4Semi_WSt_WavesWN.fst -> _5MW_OC4Semi_WSt_WavesWN.fst !

==========

Using FAST InputFileName : /Users/lincoln/Library/Mobile Documents/com~apple~CloudDocs/OpenFAST_Work/RUN_v_1.8/NREL-5MW/5MW_OC4Semi/_5MW_OC4Semi_WSt_WavesWN.fst

==========

the name of the SLX file for the open-loop computation is displayed:

Setting initial conditions of the next simulation to: Using SLX file for the Open-Loop : OpenLoop_Kernel_1A

The second part of the messages, after the SLX initialization, displays information about the SLX index and output variables.

The first group is a list of the available SLX variables extracted from the Matlab workspace:

Available output Slx variables (from the workspace):

〔Avg_B1N1Fl_out〕〔Avg_RootMyb1_out〕〔Azimuth_out〕〔B1N1Alpha_out〕〔B1N1Fl_out〕

〔B2N1Fl_out〕〔B3N1Fl_out〕〔BldPitch1_out〕〔BldPitch2_out〕〔BldPitch3_out〕

〔Timetable: Fl_node_out〕

〔Timetable: Fx_node_out〕

〔Timetable: Fy_node_out〕

〔GenPwr_out〕〔GenTq_out〕〔PtfmPitch_out〕〔PtfmRoll_out〕〔PtfmYaw_out〕

〔RootMyb1_out〕〔RootMyb2_out〕〔RootMyb3_out〕〔RotSpeed_out〕〔Wind1VelX_out〕

The second group is a list of the registered output channels (read from the 'sum' file):

Reading Channels (from the "sum" file)...

〔Time -> Time_index = 1〕

〔Wind1VelX -> Wind1VelX_index = 2〕

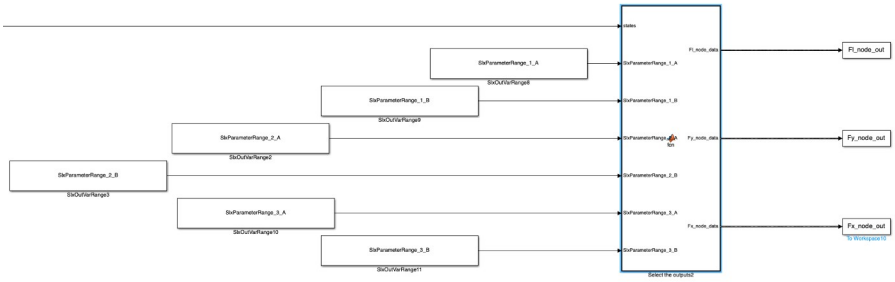
...

〔GenTq -> GenTq_index = 147〕

〔Fl -> (SlxParameterRange_1_A = 32):(SlxParameterRange_1_B = 32)〕

〔Fy -> (SlxParameterRange_2_A = 89):(SlxParameterRange_2_B = 89)〕

It is important that the user checks again the variable assignments for the vectorized nodes (see Section 2.2.c) and verify, in the SLX scheme, that the outputs are properly set with respect to the channel index. For instance, in the following SLX diagram:



the 'Fl_node_out' is assigned to SlxParameterRange_1_A/B and the 'Fy_node_out' is assigned to SlxParameterRange_2_A/B which is correct!

Finally, the *third group* ends the checking by listing the possible incorrect index, as discussed in Section 3.1-D.

It is very very important, while setting up an OpenFAST model for the first time or after major modifications of the files (SLX, input files …), that the user checks properly the input and output channels / index.

4.2 Starting a simulation: general flow and examples

Below are some general considerations:

+ In the *config file*, the user can define a name [file_name] to his project in the save_filename variable.

+ The user must choose a trajectory definition (see Section 3.2-4) in the *config file*.

+ The terminal of Matlab is logged after each simulation, hence saving all notifications in the file '[file_name]_check_parameters.txt'.

+ The variable OnlyReadingInput checks only OpenFAST & SLX files without doing any computations. It saves all notifications in the project directory and in the file '[file_name]_check_parameters.txt'.

Having checked the input files, LCO/St runs:

- a first / blank SLX simulation in order to set all index channel;

- a second SLX simulation to get the values of 'Fl' and 'RootMyb' in order to set the initial conditions of the filters (that gives the AVG trajectories) for the open-loop computation.

- the third computation, depending on the trajectory Definition, performs either the open-loop computation (Def. A and C) or the direct closed-loop computation (Def. B). See Section 3.2-4 for the details about the trajectory definitions.

All results, that have been selected for the output plot (see section 3.2-7), are saved in a *result*-mat file, that can be re-read in the script Rplot_LiftControl_Stabilization_v1_8.

Some detailed examples are given in the Section "Practical examples" to illustrate how the non-linear control works with respect to the Definition B and C.

IMPORTANT: The *fig and -mat files are cleaned once a simulation is started (except the file checking via OnlyReadingInput).

4.2.1 Auto-Save of the open-loop mode

In the case of open-loop (Def-A) or closed-loop computation (Def-C) (by selecting either SelectTrajectoryComputation = 0 or SelectTrajectoryComputation = 2), the open-loop computation is saved, as a *generator*-mat file, in the project folder and can be recalled when recomputing the closed-loop.

It is not available for the direct closed-loop (SelectTrajectoryComputation = 1) since no prior open-loop computation is required in this mode.

Restoring a prior open-loop computation

The possibility to re-use a prior (saved) open-loop computation would be useful for closed-loop (re)computation based on the same operating conditions. If a generator file exists already in the project folder, LCO/St asks the user either to re-use this generator to perform the closed-loop computation or restart a new open-loop computation.

As illustrated below, few information are displayed about the saved computation (DT, controlled node, wind speed profile, reference points) are displayed so the user can decide what to do.

```
-------------------------------------------------------
Verify existing generator file:
          wind_speed_x: [13 13 13 13]
   NodeLift_Index_cntrl: 9
                    DT: 0.0125
Trajectory reference points: 1.000000 1.000000 1.000000
 Do you want to re-build the generator file (default "0") ?
```

If the user chooses to recompute the closed-loop (re-build the generator file), then the open-loop is recomputed based on the data provided in the *config file*. Otherwise, the data that have been previously saved are used for the closed-loop computation.

4.2.2 The open-loop / lift surface computation mode

Let us consider the example config_file_OpenLoop_Surface in the directory /ex_1_5MW_OpenLoop_Surface.

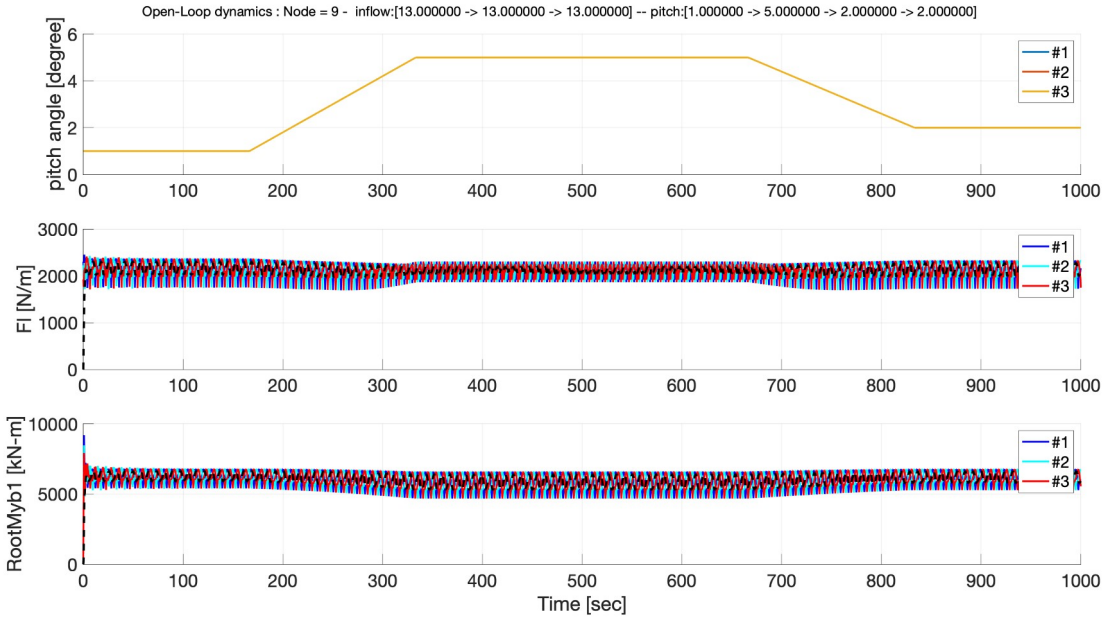
Given the following wind speed profile and pitch profile:

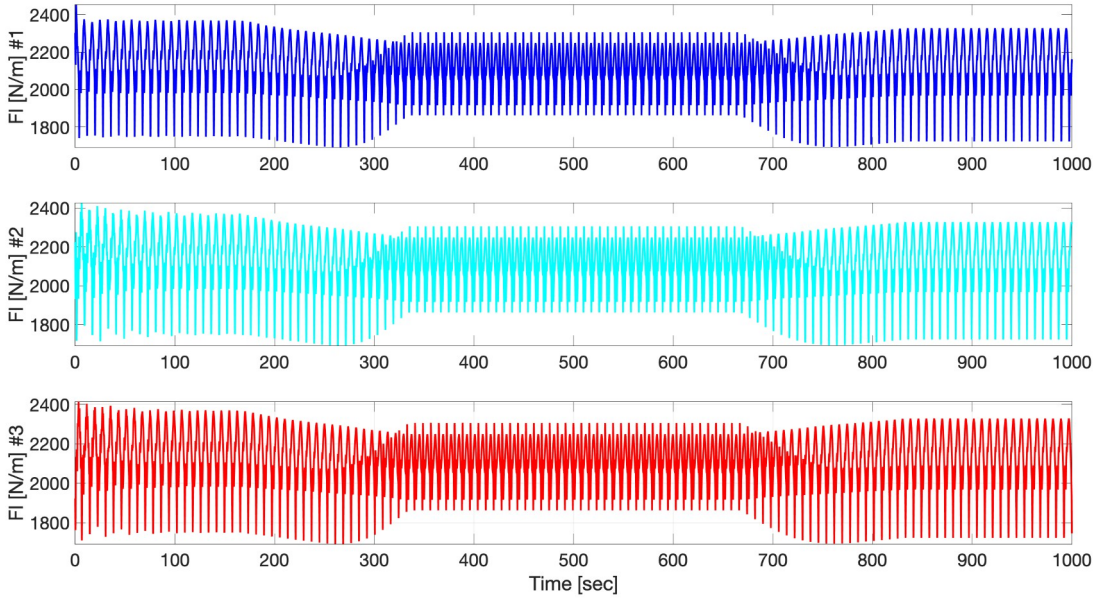
```
wind_speed_x = [13 13 13 13];
wind_speed_t = [0, 1500, 2000, 3000];

Point_OpenLoop_Pitch_1 = 1;
Point_OpenLoop_Pitch_2 = 5;
Point_OpenLoop_Pitch_3 = 2;
```

The following figures illustrate the resulting computation of the lift and the RBM.

In the current configuration of the plots, a first figure displays the pitch angle + the 3 lifts (for each blade) at the ControlledNode node + the 3 RBM (for each blade).



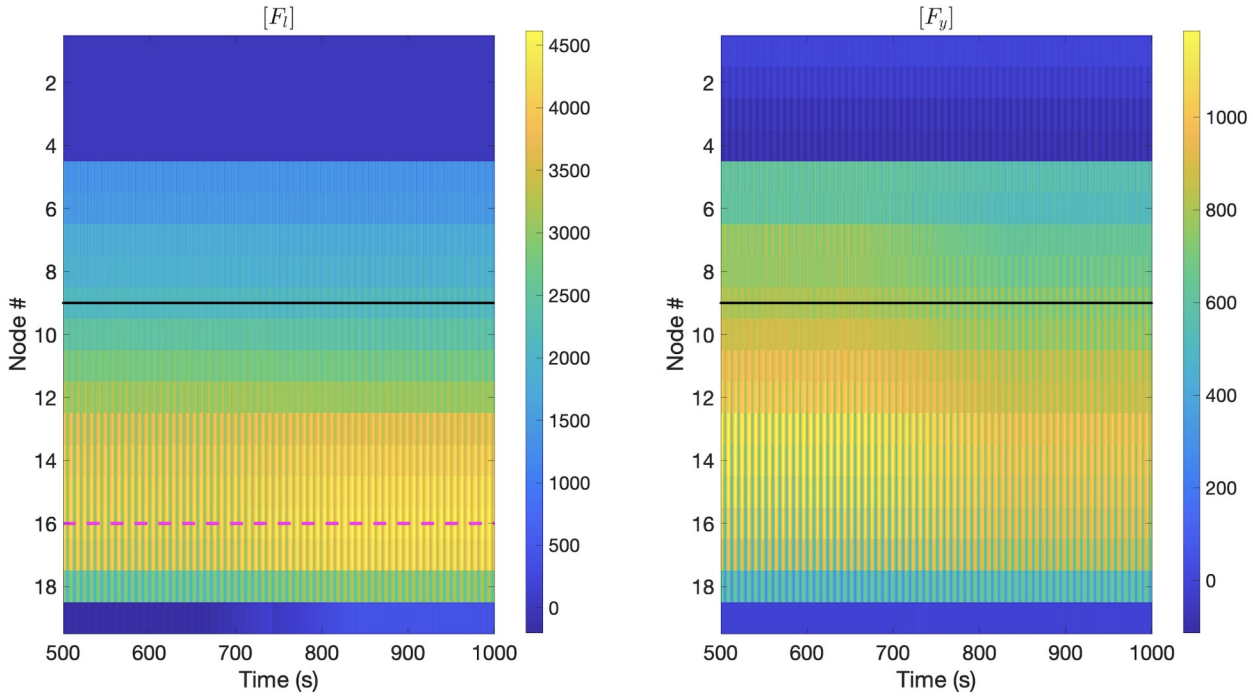


The surface computation can be invoked additionally in open-loop.

The lift surface computation allows plotting the 2D lift force "Fl" as well as the 2D force "Fy" (along the blade) with respect to the time[14]. To avoid displaying the transient, half of the full time interval is considered meaning [Tmax/2, Tmax].

14 In this current version of LCO/St, it is not possible to change easily the output to plot in 2D since it requires to reorganize the matrices that contain the 2D data. The user can have a look at the script ComputeLiftSurface_v2b to make modifications.

The proposed representation of the graphs is (vertically) the spatial node along the blade vs. the time (horizontally). The following figure illustrate a lift surface computation and this example can be found in the directory <Examples/ComputeSurface> in the GitHub repository.



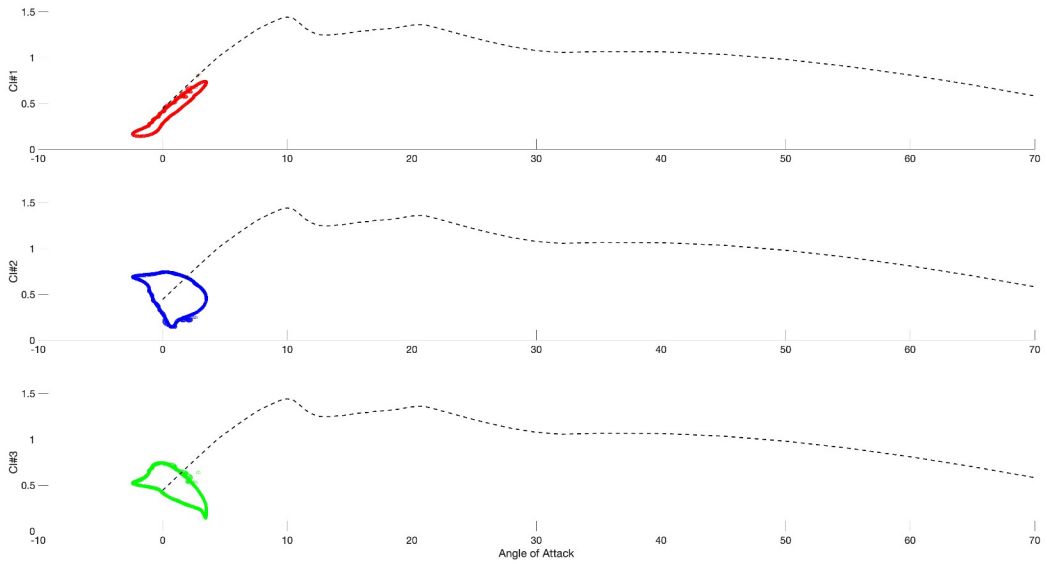
The node #0 corresponds to the root of the blade and the last node corresponds to the full length of the blade.
Note that the total number of nodes depends on the model of the wind turbine under study (the user should verify the node configuration before running a simulation using a new wind turbine model since no information are given by LCO/St).

4.2.3 The open-loop / C_L curves plotting

This feature[15] allows plotting the C_L curves and compared with theoretical curves provided in the input files: the user should refer to the "AeroDyn_blade" file in order to open the correct file, in the blade section, that contains the CL curves with respect to the ControlledNode.

The following figure illustrates a plot of the C_L curves with respect to each blade in open-loop.

15 This feature is introduced in v1.8.0-b.



In this figure, for the three blades #1, #2 and #3, the theoretical C_L curves are plotted in dot and the simulated curves are in red, blue and green according to the blade.

4.2.4 Closed-loop – pitch-based natural reference trajectory

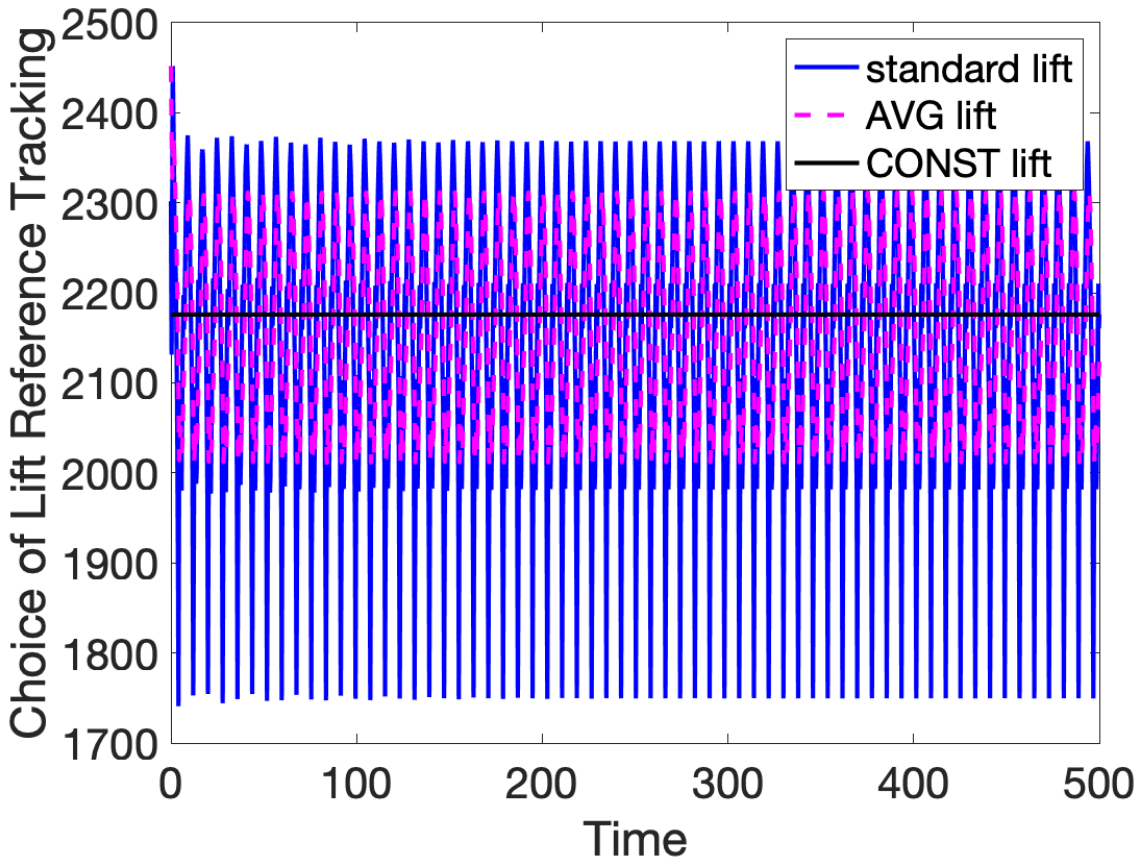
Remind that, in the Definition C (see Section 3.2-4-C), a first computation of the lift in open-loop, given a pitch profile, is performed and the resulting lift is used afterwards as a reference trajectory for the lift / RBM closed-loop.

1. The pitch profile, as defined by the user, is plotted accordingly to the wind speed profile (see section 3.2-4-A).

2. The open-loop trajectory is computed and lift / RBM trajectories are plotted according to the choice of the user LiftTrajectoryRef or RBMTrajectoryRef in the *config file* (see Section 3.2-2).

3. The lift / RBM trajectory, computed from the open-loop, is applied as the tracking reference for the closed-loop.

The following figure illustrates the resulting lift / RBM trajectories obtained from the open-loop computation, which could be used as reference lift / RBM trajectory in closed-loop.



Remark: LCO/St does not allow changing the expected closed-loop trajectory while the program is running. Therefore, the user should first change the closed-loop trajectory reference LiftTrajectoryRef or RBMTrajectoryRef (see Section 3.2-2) in the *config file*, restart the program and restore the previous (saved) open-loop to change the selected trajectory.

The selected lift trajectory given by the open-loop computation is applied as a reference trajectory in closed-loop to control for example the 'B1N1Fl' output, which is in feedback with the reference trajectory, in the SLX file. The controlled output can be changed in the SLX file.

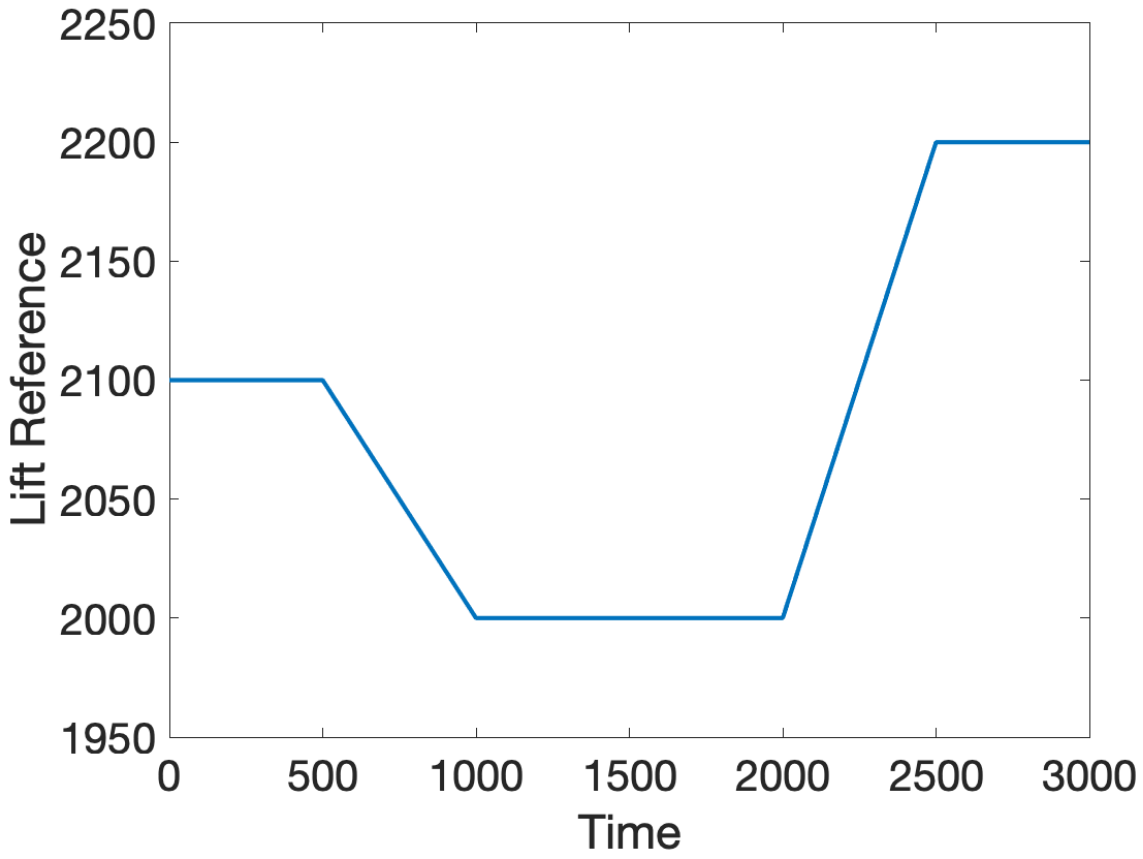
Note that the CONST lift trajectory (calculated from the mean of the output lift) is similar to use the Definition B with a constant profile. The advantage of using first the CONST trajectory via the C Definition is that it gives an appropriate physical value of the lift that corresponds to the given pitch profile.

Some examples will be illustrated in the Section "Practical examples".

4.2.5 Closed-loop – lift reference trajectory

Remind that, in the Definition B (see Section 3.2-4-B), the lift / RBM profile is used as a reference trajectory for the lift / RBM closed-loop. There is no prior open-loop computing.

1. The lift / RBM profile, as defined by the user, is plotted accordingly to the wind speed profile, for instance illustrated by the following figure:



In this profile, the lift is piecewise defined (built in the same manner as the pitch profile in the A-C/ definitions).

2. The lift / RBM trajectory profile is applied direclty as the tracking reference for the closed-loop.

Some examples will be illustrated in the Section "Practical examples".

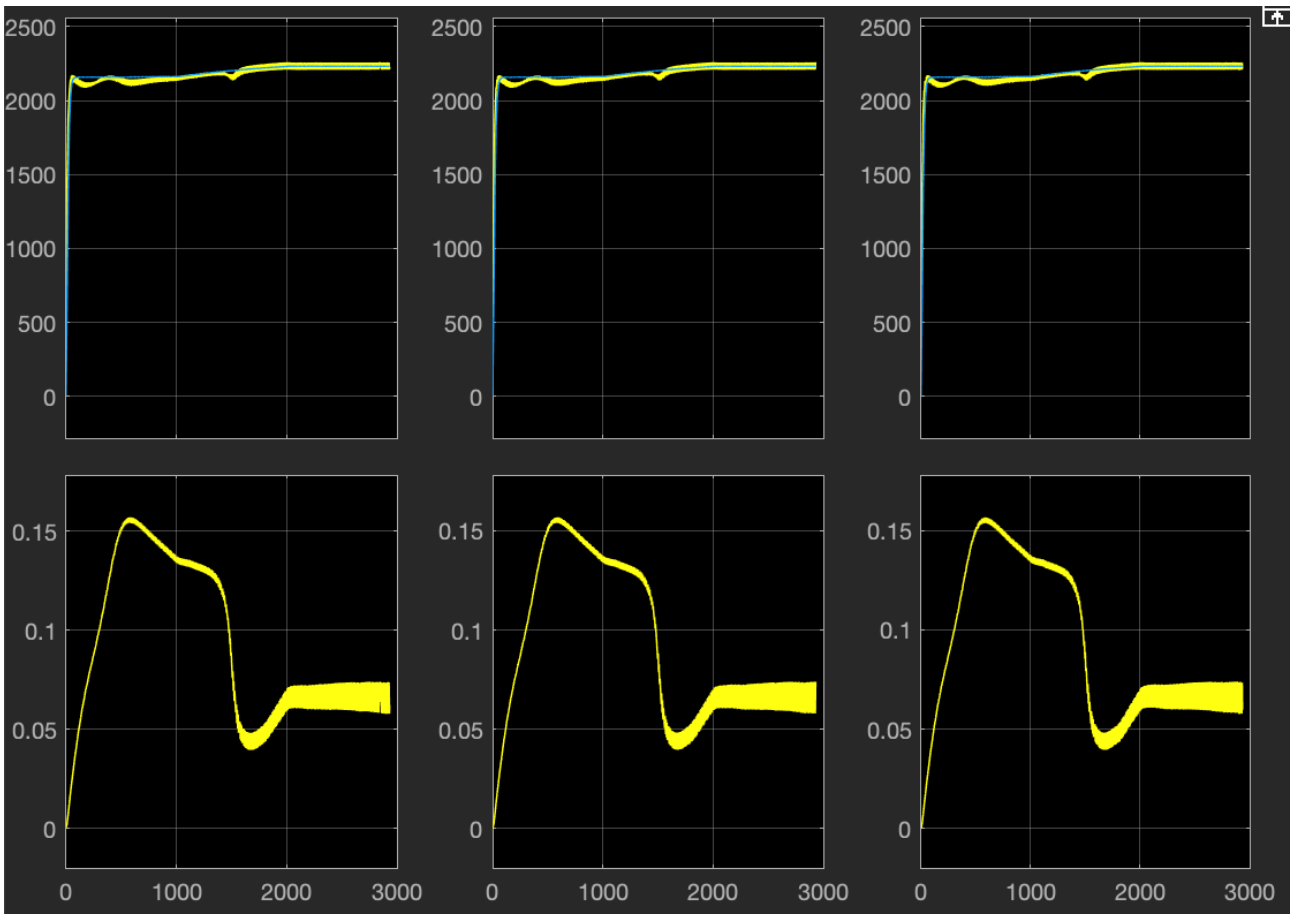
4.2.6 Individual pitch control mode

The parameter IPC_mode, defined in the *config file* (see Section 3.2-2), updates in the servoDyn file[16] the parameter that enables/disables the IPC mode, thanks to the Python script. The Python script is called in the same manner that the OpenFAST input files and the Initial Condition on the pitch can be modified by LCO/St according to the needs of the user (see Section 2.2.d).

The three references needed to track the lift / RBM of each blade are computed from the open-loop. The C-definition (SelectTrajectoryComputation = 2) is automatically set internally if the IPC mode is requested in the *config file*.

The figure below illustrates the IPC mode from a Simulink plot screenshot. The upper row shows the lift associated to each blade. The lower row shows the corresponding blade pitches.

16 It is assumed that modifying the 'Ptch_Cntrl' parameter in the Servodyn file is sufficient to consider the "Individual Pitch Control" mode.



Note that no SLX version regarding the RBM control has been released for the IPC mode.

4.2.7 Plotting the results

The script "RPlot_LiftControl_Stabilization_v1_8" allows plotting the results from a *result-*mat file. The script plots the sames figures as requested by the *config file* and the user is asked to save the plots [17].

4.2.8 Known limitations

In the following list are given some current limitations that have been considered for future improvements.

+ A project is dedicated to a single configuration of the aerodynamic conditions & control law, which currently does not allow to choose among several configuration files. Hence, it does not allow easily to span multiple parameters to compare multiple resulting performances.

+ The pitch / lift / RBM profile that is created could be not correctly generated by the generator (with respect to the given key points), since unexpected trajectories could occur. This will be improved in future versions.

+ the lift / RBM profile is only defined for a single blade. There is currently no possibility to perform a direct closed-loop computation in the IPC mode based on lift / RBM profile since it is assumed that the trajectories are different with respect to the blade.

17 Minor improvement introduced in v1.8.0-b.

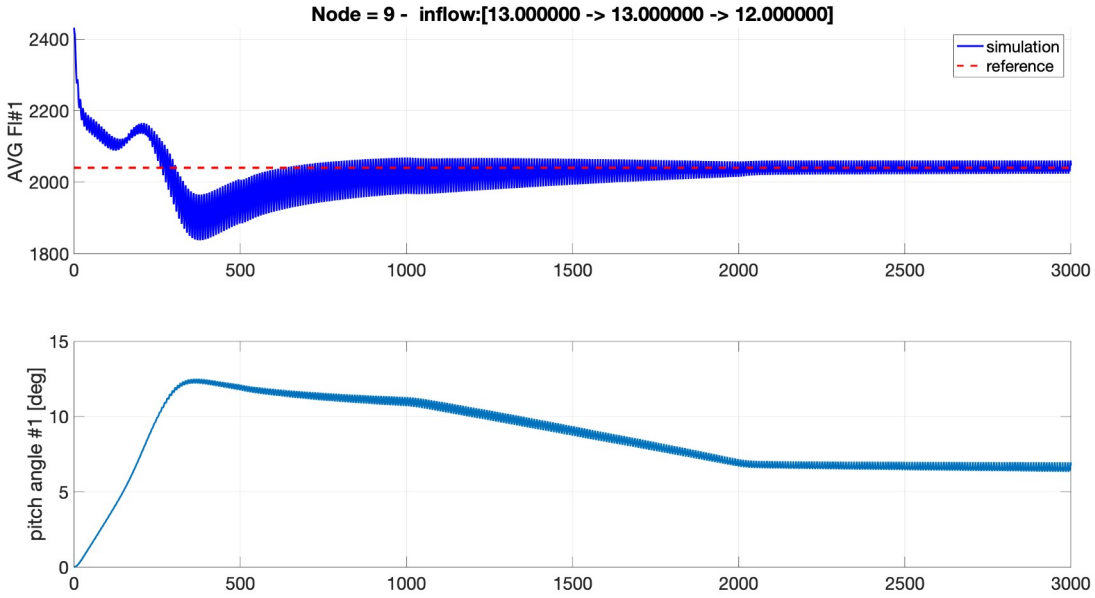
5. Practical examples

The following examples illustrate some lift & RBM control simulation considering the 5 MW model, the controlled node #9 and a zero IC pitch. These examples are located in the directory <Examples> in the GitHub repository and are given without any physical interpretation, just to illustrate some operating conditions of the control.

Some additional results that illustrate stochastic conditions of the wind inflow as well as the introduction of a very basic maximum lift tracking algorithm are summarized in the report [5].

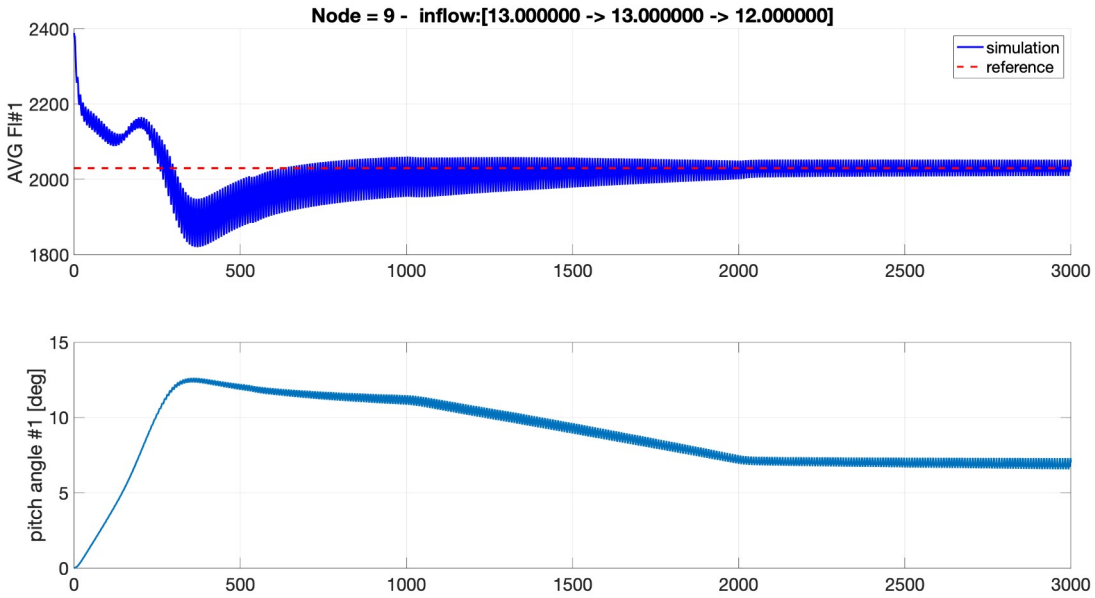
5.1 Lift-based control #1

The lift-based control has been tested under the Def-C, considering: the wind profile: 13 → 13 → 12 → 12 and the pitch profile: 1 → 1→ 1.
The following figure illustrates the controlled lift.



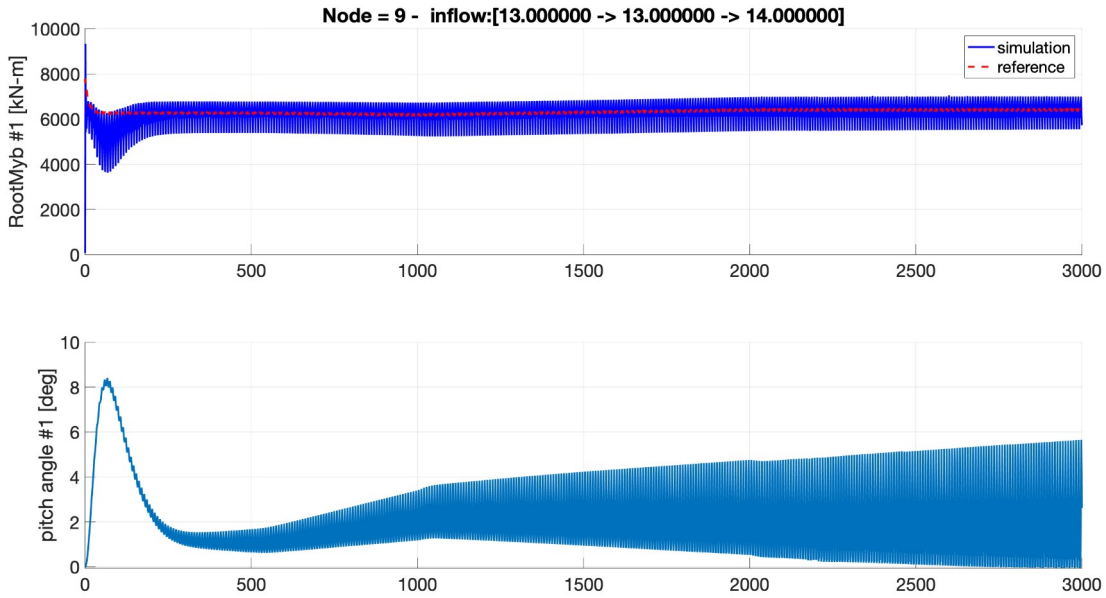
5.2 Lift-based control #2

The lift-based control has been tested under the Def-C, considering: the wind profile: $13 \rightarrow 13 \rightarrow 12 \rightarrow 12$ and the pitch profile: $1 \rightarrow 2 \rightarrow 2$.
The following figure illustrates the controlled lift.



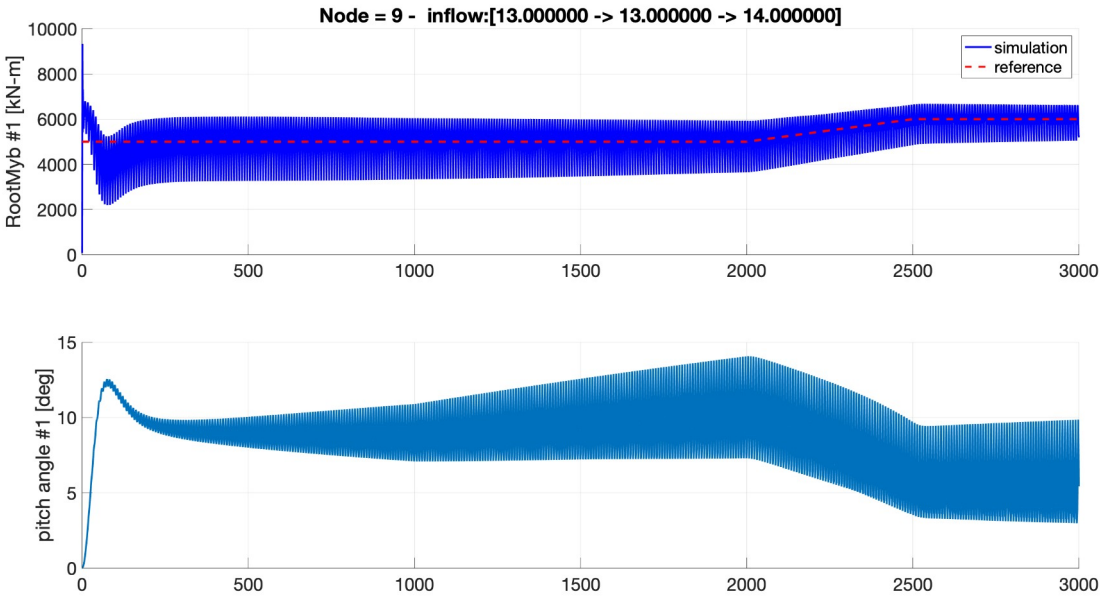
5.3 RBM-based control #1

The RBM-based control has been tested under the Def-C, considering: the wind profile: 13 → 13 → 14 → 14 and the pitch profile: 1 → 2 → 2.
The following figure illustrates the controlled RBM.



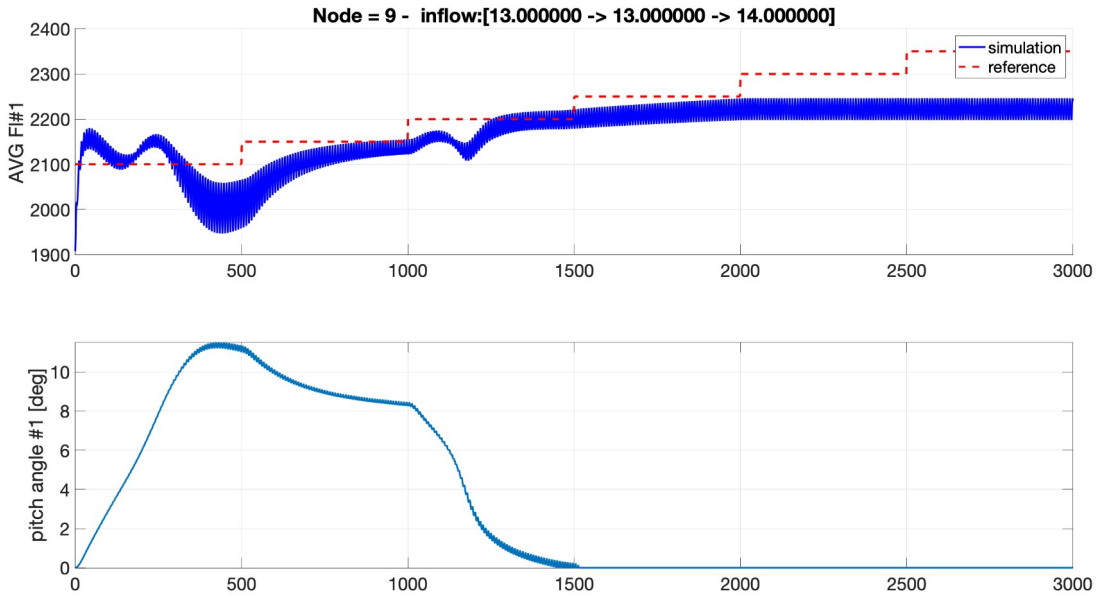
5.4 RBM-based control #2

The RBM-based control has been tested under the Def-B, considering: the wind profile: $13 \rightarrow 13 \rightarrow 14 \rightarrow 14$ and the RBM profile: $5000 \rightarrow 5000 \rightarrow 6000$. The following figure illustrates the controlled RBM.



5.5 Lift-based control with increment #1

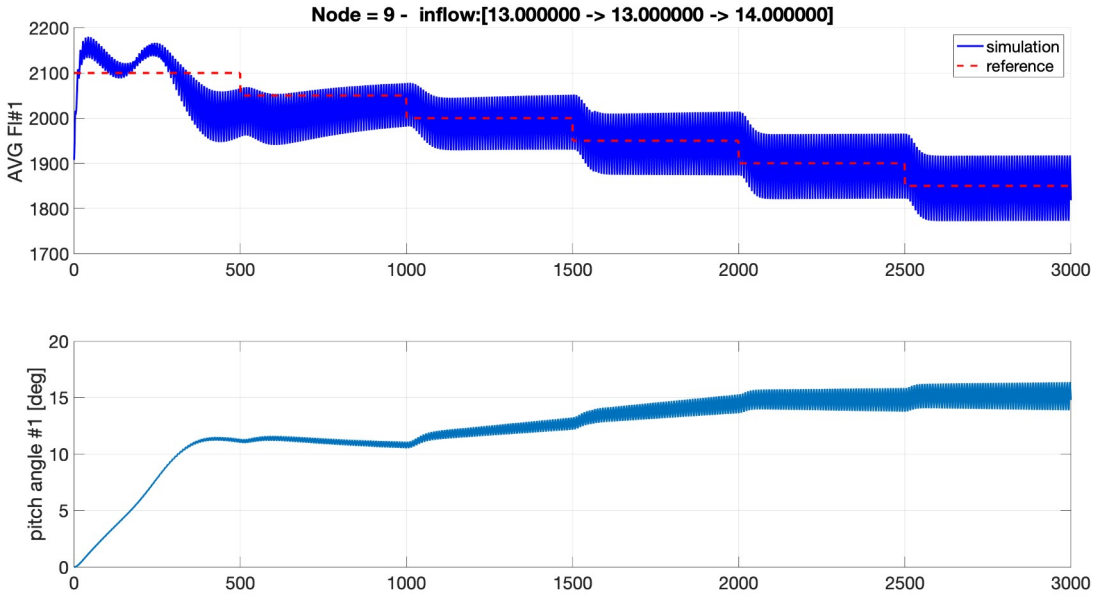
The lift-based control has been tested under the Def-B, considering: the wind profile: $13 \rightarrow 13 \rightarrow 14 \rightarrow 14$ with a lift profile starting from 2100 with an increment of $+50$. The following figure illustrates the controlled lift.



Remark that in this example, the pitch decreases and saturates to zero, highlighting the fact that this particular configuration is not be able to track the lift.

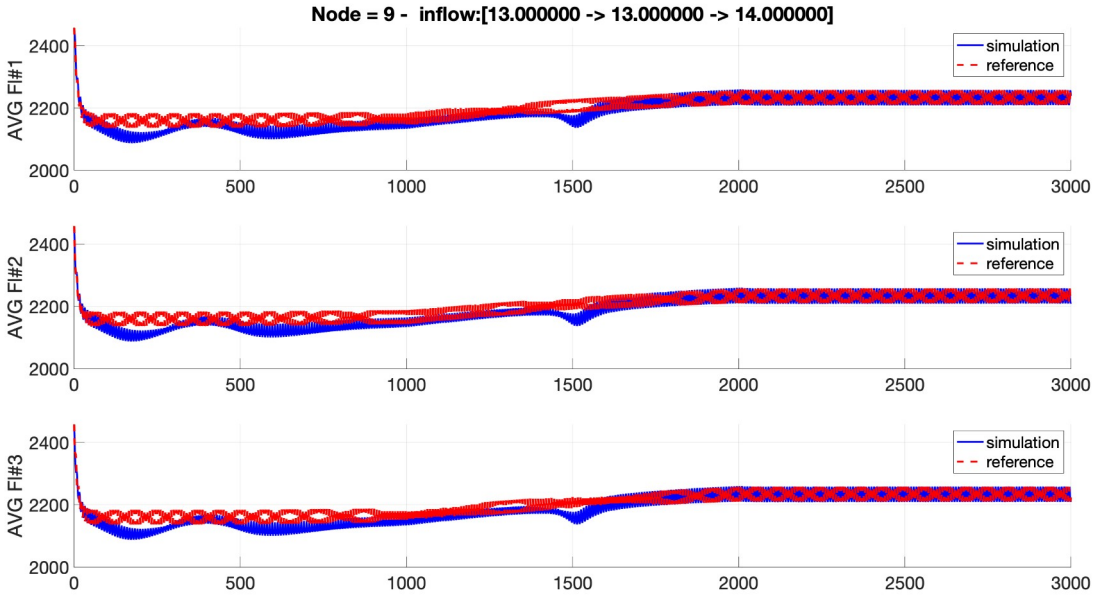
5.6 Lift-based control with increment #2

The lift-based control has been tested under the Def-B, considering: the wind profile: 13 → 13 → 14 → 14 with a lift profile starting from 2100 with an increment of -50 (symmetric from the previous case in 5.5).
The following figure illustrates the controlled lift.



5.7 Lift-based control in IPC mode

The lift-based IPC mode has been tested under the Def-C, considering: the wind profile: 13 → 13 → 14 → 14 and the pitch profile: 1 → 0.5 → 0.5. The following figure illustrates the resulting controlled lift.



6. Installation

LCO/St has been developed and tested under macOS v14 using Matlab / Simulink version = R2023b including the "Control System Toolbox" and the "DSP System Toolbox". The OpenFAST software, including its Simulink API, has been recompiled using gcc 14.0.1.

A Python script has been prepared by our colleague Dr. P. Molinaro that is extremely useful to install OpenFAST (and its dependencies) in a very straightforward manner. The script can be found in the <install> directory in the GitHub repository. To use the script (under macOS), Xcode Command Line Tools >= 15.0, gcc >= 14.0, clang >= 15.0 should work (depending on your system configuration).

Having compiled all dependencies and the OpenFAST software, the </lib> directory that is located in </product-openfast-library-3.5.3> must be entirely copied in the </root> folder of LCO/St. In addition, the library **FAST_Sfunc.mexmaci64** must be moved (or copied) in the </root> folder of LCO/St directory.

The official NREL libraries needed to include the internal wind turbine controllers can be downloaded at <https://github.com/OpenFAST/nrel-5mw-controllers>.

Acknowledgments and credits

The author is grateful to Dr. Jean-Christophe Gilloteaux and Dr. Thomas Potentier (INNOSEA company) and M. Maximilien André (ECN/LHEEA lab) for their assistance and support regarding the utilization of OpenFAST.
The author is also grateful to Dr. Pierre Molinaro for his assistance regarding the compilation and installation of OpenFAST.
The author is also sincerely thankful to Prof. Jean-Pierre Barbot (ECN/LS2N & ENSEA/QUARTZ) for his constant guidance and support.

The Python toolbox has been developed by Dr. E. Branlard (2023), available at: <https://github.com/OpenFAST/python-toolbox>. The Python script has been modified to allow a direct call, including arguments pass, through LCO/St. No modification of the original code has been made.

This work has been fully carried out through the Carnot GOWIBA project, funded by the Carnot Institute Marine Engineering Research for "Sustainable, Safe and Smart Seas".

This is a work under progress. Please contact the author for any question at <loic.michel54@gmail.com>. Any contributions to this project are welcome!

References

〔1〕 OpenFAST website <https://www.nrel.gov/wind/nwtc/openfast.html>

〔2〕 E. Branlard GitHub main repository <https://github.com/ebranlard>

〔3〕 L. Michel, A para-model agent for dynamical systems, preprint arXiv:1202.4707, 2018.

〔4〕 L. Michel, E. Guilmineau, C. Braud, F.Plestan, J.-P. Barbot. Model-free based pitch control of a wind turbine blade section: aerodynamic simulation. European Control Conference 2024, Jun 2024, Stockholm (Suède), Sweden.

〔5〕 L. Michel, Model-free based pitch control of a wind turbine blade section in the OpenFAST environment - Some technical remarks, preprint, Oct. 2024.

List of revisions

v1.0 : Kernel to run a complete OpenLoop Lift reference build + Closed-loop control
v1.1.a : Addition of a triphase (0-120-240) generator + minor improvements (incl. kernel plot)
v1.1.b (beta) : Addition of a switching generator => partially checked on 29/12/23 : switching generator ok but not tested in control
v1.2 : Inclusion of the third-party code (written by M. André / ECN-LHEEA) to process the 'outb' file via OpenFAST internal Matlab procedure (=> needs a modification of the data files of OpenFAST to allow additional real-time outputs.) + major modifications of the management of the I/O files (code of the switching ref. operational but not fully tested)
v1.2.1 : Add the ReadNode function that extract information from OpenFAST internal file + parameter checking to re-build generator
v1.2.1.b : external checking of the PMC control (due to unstabilities observed with v1.1.b) => PMC single is unstable despite apparently same operating conditions with the previous SLX file ('Lift_PMC_v1')=> give up with single lift simulation
v.1.3 : Add of the tracking of the averaged lift (2 control types) :
=> this version can run both "shape-based control" & "averaged-based control"
=> bug with apparently the reference of the shaping that makes the closed-loop diverging. -> solved because of the pre-cone and ShftTilt parameters => should be set to no zero !!!!! Limitation of the generation of the ref. (via the usual blocks under Matlab) for the Avg tracking -> to be enhanced through next version
v1.4 : Inclusion of a script-based reference generator for both "shape-based control" & "averaged-based control" + upgrade of the file management -> seems operationnal : avg-mode & shape-mode are running without errors Triphase management / switching generator needs to be checked and improved (not fully tested from v1.2 !!) seems to have a Matlab bug plot when trying to plot huge (time) vectors -> to be solved urgently !!
v1.4.1 : proposes a novel structure to post-process the data (using cells: 'ProcessingSlxData_v2_cell') and enhance the plotting tasks though prior slot variables -> would allow to undersample the results in order to relax the plotting tasks ;) -> update files (at least 'ProcessingSlxData') and 'PlotKernel' to v2 -> needs to be finalized! this version gives results for both shaping-based control and AVG-control but it is requested 1/ to perform a spatial-avg measure of the lift on the blade 2/ to simplify the config of the wind turbine -> update needed in next v1.5 ;)
v1.4.2 : minor update to prepare the next version -> improve the ReadNode_v2 to allow more flexibility in reading the OpenFAST files + update of the out/outb cleaning and saving method + preparation of the 15 MW case :)
v1.5 : auto-extraction of data from the OpenFAST file (through auto-reading of the files) The v3b version of ReadData may not be useful -> to be checked later. Also : add the computing surface of the lift and major update of the Slx file (v5) that implies a major revision of the structure of the init. & execution / post-processing of Slx variables and results => this version (v1.5.0) is operational for a complete 'Lift-shaped based control' of the lift (not checked of avg-based -> to be be checked later in really necessary !) -> prepare next sub-version including python data-file manager ;)
v1.6 : Update the OpenFAST kernel to 3.5.0. Add the python data-file manager to help changing every openFAST parameter in a more easier way -> request Python lib. Update of the structure of the code to better match with duplication of the config files / check of the modified parameters. -> this version is operational for the 'shaping-based' control. A small improvement of the management of the output plot should be done in the next minor version (to separate the power management from the lift). The SurfaceLift computation is operational but the loop has been disabled in this version since it requires more precise needs. Also minor improvement of the reference trajectory by defining only 3 points -> Point_Ref_1 -> Point_Ref_2 -> Point_Ref_3
v1.6.1 : The Loop to compute iteratively (w.r.t blade pitch and wind speed) the Surface Lift is operational but uses the same initial condition on the pitch depsite the variations (which can be a problem). The surface is limited to the lift surface display -> a need to compute the Power extracted from the blade requires an additional surface -> will be handled in the minor v(1.b).
v1.6.1b : Improve the Surface Lift plotting (incl. power/Torque computation and review of the display); add security to the output 'basic' variables check in 'SimulinkInternalVariables_check' if some of them are not properly declared in the 'ConfigOutputChannel'.
Add the modification of the I.C. Pitch into the ElastoDyn file prior to any simulation

v1.6.2 : Some minor changes in the main regarding the computation of the lift surface (put in the main program). Ready for control in the simplified wind turbine mode: gives interesting results that may be extended to the normal case → prevision of fusionning the Slx file into a single one and consider 'ControlType' only to switch between open-loop and closed-loop → seems easier to manage -> only project directory 'path' to consider. The results obtained in this version highlight the possibility to control both the simplified case (where everything is constant) and the more standard case (oscillating) by considering driving the avg value in a free-wheeling control context. The Bending Root Momentum is also of interest to be controlled alternatively to the lift. The next version should be focused on 'merging' control strategies towards king of unified control incl. the lift and the BRM.
v1.6.3 : as pre-RC 1.7 -> will be transfered to v1.7 once ready: Introduces some essential changes in the management of the simulation including new definition of lift and RBM trajectories w.r.t. open-loop simulation. Three operating cases are considered
Open-loop / Closed-loop with internal trajectory (OL-based) / Closed-loop with external (user defined trajectory). The LiftSurface computation is maintained for future developments.
v1.7a : Remove of the generator file management - validate the pre-RC 1.7 to obtain results on the lift and RBM control subjected to oscillations.
v1.7b : Improved generator file management to allow a direct display of stored previous simulation condition. Solved a bug related to the update of the time vector that was not properly saved and re-assigned from the Simulink file (used time-base associated to the trajectory)
v1.7b-1 : sub-version that runs the 5 MW case and add the constant trajectory reference (taken from the last value of the filtered reference in open-loop) - fix the sum file instantiation - fix the generator file verification -> if no generator, then print (no generator)
v1.7.2 : fix (again) the management of the modified files – simplify the SurfaceComputing (due to possible not useful variables) – fix bugs in the SelectTrajectoryComputation == 1. This version can run a priori the 5-MW case and the 15-MW case
v1.7.2b : sub-version to run IPC mode - solve the problem of not checking the generator trajectory if SelectTrajectoryComputation == 1
v1.7.2c : sub-version that introduces the lift-stepping as external reference (increasing the lift step by step)
v1.8 : full review and update of the complete code - fix the limitation concerning the wind inflow file (can set the wind type and the profile) - check also the plotting module to read results and plot results in the same format as the main module. some small improvements of the code management within the config file.
v1.8.2 : fix some minor bugs and add some minor improvements including the plot management script (plot kernel) and the detection of using GenDOF (allowing the use of VScontrol), introduce the C_L plotting and add the max. lift tracking algorithm.