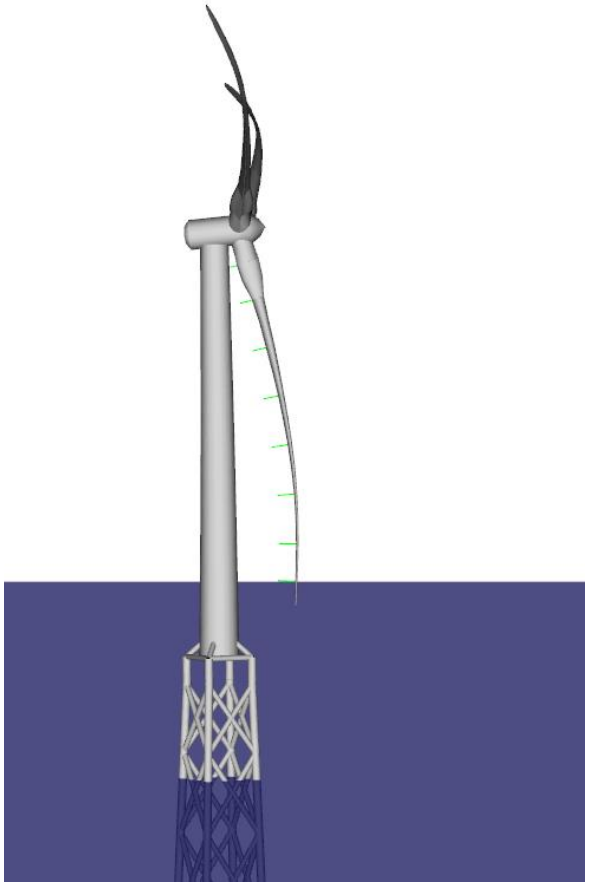
Risø-R-Report

How 2 HAWC2, the user's manual

Torben J. Larsen, Anders M. Hansen
Edited by the DTU Wind Energy HAWC2 Development Team

Risø-R-1597(ver. 12.9)(EN)



Risø National Laboratory
Technical University of Denmark
Roskilde, Denmark
October 2021

Authors: Torben Juul Larsen, Anders M. Hansen. Edited by the DTU Wind Energy HAWC2 Development Team
Title: How 2 HAWC2, the user's manual
Institute: Department of Wind Energy

Abstract:
The report contains the user's manual for the aeroelastic code HAWC2. The code is intended for calculating wind turbine response in time domain and has a structural formulation based on multi-body dynamics. The aerodynamic part of the code is based on the blade element momentum theory, but extended from the classic approach to handle dynamic inflow, dynamic stall, skew inflow, shear effects on the induction and effects from large deflections. It has mainly been developed within the years 2003-2006 at the aeroelastic design research programme at Risoe, National laboratory Denmark, but is continuously updated and improved.
This manual is updated for HAWC2 version 12.8 and wkin.dll version 2.8

Risø-R-1597(ver. 12.9)(EN)
October 2021
ISSN 0106-2840 ISBN
978-87-550-3583-6
Groups own reg. no.: 1110412-3

Technical University of Denmark
DTU Wind Energy
Frederiksborgvej 399
4000 Roskilde
Denmark
Telephone +45 45 46774004
bibl@risoe.dk
Fax +45 46774013

Contents

1 Preface

The HAWC2 code is a code intended for calculating wind turbine response in time domain. It has been developed within the years 2003-2006 at the aeroelastic design research programme at Risoe, National laboratory Denmark.

The structural part of the code is based on a multibody formulation where each body is an assembly of Timoshenko beam elements. The formulation is general which means that quite complex structures can be handled and arbitrary large rotations of the bodies can be handled. The turbine is modeled by an assembly of bodies connected with constraint equations, where a constraint could be a rigid coupling, a bearing, a prescribed fixed bearing angle etc. The aerodynamic part of the code is based on the blade element momentum theory, but extended from the classic approach to handle dynamic inflow, dynamic stall, skew inflow, shear effects on the induction and effects from large deflections. Several turbulence formats can be used. Control of the turbine is performed through one or more DLL's (Dynamic Link Library). The format for these DLL's is also very general, which means that any possible output sensor normally used for data file output can also be used as a sensor to the DLL. This allows the same DLL format to be used whether a control of a bearing angle, an external force or moment is placed on the structure. The code has internally at Risoe been tested against the older validated code HAWC, the CFD code Ellipsys and numerous measurements. Further on detailed verification is performed in the IEA annex 23 and annex 30 research project regarding offshore application. Scientific papers involving the HAWC2 is normally posted on the www.hawc2.dk homepage, where the code, manual and more can be downloaded. During the programming of the code a lot of focus has been put in the input checking so hopefully meaningful error messages are written to the screen in case of lacking or obvious erroneous inputs. However since the code is still constantly improved we appreciate feedback from the users – both good and bad critics are welcome. The manual is also constantly updated and improved, but should at the moment cover the description of available input commands.

2 Acknowledgements

The code has been developed primarily by internal funds from Risø National Laboratory – Technical University of Denmark, but the research that forms the basis of the code is mainly done under contract with the Danish Energy Authority. The structural formulation of the model is written by Anders M. Hansen as well as the solver and the linking between external loads and structure. The anisotropic FPM beam model is written by Christian Pavese, Taeseong Kim and Anders M. Hansen. The aerodynamic BEM module is written by Helge A. Madsen, Torben J. Larsen and Georg R. Pirrung. Three different stall models are implemented where the S.Ø. (Stig Øye) model is implemented by Torben J. Larsen, the mhh Beddoes model is written by Morten Hansen, Mac Gaunaa and Georg R. Pirrung and the ateflap model used for trailing edge flaps is written by Mac Gaunaa and Peter Bjørn Andersen and has later been rewritten by Leonardo Bergami. The near wake model has been developed by Georg R. Pirrung, Ang Li, Helge Aa. Madsen and Peter B. Andersen. The wind and turbulence module as well as the soil and DLL modules are written by Torben J. Larsen. The hydrodynamic module is written by Anders M. Hansen and Torben J. Larsen. The turbulence generator is written by Jacob Mann and the WAsP Team and converted into a DLL by Peter Bjørn Andersen. The dynamic wake meandering module is written by Helge A. Madsen, Gunner Larsen and Torben J. Larsen, and has been further maintained by Jaime Liew. The eigenvalue solver is implemented by Anders M. Hansen and John Hansen. The Gitlab repository including automatic testing and compilation was created by Mads M. Pedersen and Anders M. Hansen. Torben J. Larsen and Anders M. Hansen were the main authors of the manual up to version 4.7, and the main developers of HAWC2 up to version 12.8. Maintenance of the codebase, webpage and the manual is performed by the HAWC2 development team at DTU Wind Energy.

3 Contributors

Contributors to this manual and the HAWC2 code include but are not limited to:

Anders Melchior Hansen
Torben Juul Larsen
Peter Bjørn Andersen
Leonardo Bergami
Franck Bertagnolio
Kenneth Thomsen
Emmanuel Simon Pierre Branlard
Mikkel Friis-Møller
Christos Galinos
Mac Gaunaa
John Hansen
Morten Hartvig Hansen
Joachim Christian Heinz
Lars Christian Henriksen
Sergio González Horcas
Gunner Christian Larsen
Ang Li
Jaime Liew
Helge Aagaard Madsen
Jacob Mann
Taeseong Kim
Mads Mølgaard Pedersen
Christian Pavese
Georg Raimund Pirrung
Néstor Ramos García
Jennifer Rinker
Riccardo Riva
David Robert Verelst
Shaofeng Wang
Annop Wongwathanarat
Albert Meseguer Urban
Laura Voltá
Ozan Gözcü
Jenni Rinker

4 Getting started with HAWC2

This section contains some basic overview information and tips on debugging files when running HAWC2. A more detailed description of the format of the input file is discussed in Section 5.

4.1 Running HAWC2

HAWC2 is run by calling the HAWC2 executable from a Windows Command Prompt on the input file, which has a `.htc` file extension (see Section 5):

```
> <path to HAWC2 executable> <path to htc file>
```

For example, if the current working directory of the Command Prompt contains both your HAWC2 executable and an input file called `turbine_model.htc` (which is not a recommended folder structure, see below), the command to run HAWC2 would be

```
> HAWC2MB.exe turbine_model.htc
```

Important! Any relative paths in the `htc` file will be defined with respect to the current working directory of the Command Prompt, *not* with respect to the file's location.

4.2 Folder structure

HAWC2 does not assume any folder structure, so the executable and the input file can be located anywhere that is accessible by the Command Prompt. However, it is often best to separate different wind turbine models so that their results do not overwrite each other. It can also be nice to separate the HAWC2 executable from the input/output files in order to keep the directories as clean as possible.

One way to do this is to place HAWC2 and all its required DLLs in one directory and all of the files related to a specific turbine model in another directory. Let us demonstrate this with an example. Assume that we have placed the HAWC2 executable and all related DLLs in `C:\hawc2\`. We desire to run an htc file, called `input_a.htc`, that is located in `C:\Documents\turbine_models\prototype_a\htc\`. However, the htc file contains relative paths that are defined with respect to the `prototype_a\` directory. In this case, we must first change the working directory to the `prototype_a\` directory so that the relative paths in the htc file point to the correct files, and then we can call the HAWC2 executable on the input files using an absolute path. The commands for this example would be as follows:

```
> cd C:\Documents\turbine_models\prototype_a\
> C:\hawc2\HAWC2MB.exe .\htc\input_a.htc
```

4.3 Debugging models

Although HAWC2 is run from the Command Prompt, the errors that are printed to it when something goes wrong are often not illuminating to the average user. If something goes wrong with your model, you should first check the output log to see what warnings and errors are printed there. The output log is a text file ending in `.log`, and its location is determined by the `logfile` option in the `simulation` block in the htc file.

One of the most common errors for new users is having the wrong working directory in the Command Prompt, in which case the log file will state that it could not find the requested data files. Other common errors when running time-marching simulations include bad simulation

parameters that lead to non-convergence or incorrect definitions of body properties. Regardless, your first step when debugging a model should always be to look at the log file to determine what went wrong. If you cannot find the source of your problem, you can email the HAWC2 support address (hawc2@windenergy.dtu.dk) to ask for help.

Important! HAWC2 is a flexible software with many different simulation options, so building a model from the ground up is complicated and not recommended. We recommend starting from a working model (see the HAWC2 website to download a working wind turbine model) and incrementally making changes as needed.

5 General input layout

HAWC2 takes as input a text file with an `.htc` file extension. The HAWC2 input format is written in a form that forces the user to write the input commands in a structured way so aerodynamic commands are kept together, structural commands the same, etc. The order of the blocks does not matter.

The input commands are divided into command blocks, which are defined using a begin-end syntax. Each line must end with a semi colon ";" which gives the possibility for writing comments and the end of each line after the semi colon. The command lines can be written with any desired mix of capital or small letters because inside the code all lines are transformed into small letters. This could be important if something case-sensitive is written (e.g., the name of a subroutine within a DLL).

Important! All lines in an htc file must end with a semicolon, even if they are empty. You may insert whitespace between blocks to improve readability by having a line that is just a semicolon.

In the next chapters, the input commands are explained for every part of the code. The commands are separated into "main block" commands (namely, those that belong to a begin-end command block that is not part of a higher-level begin-end block) and "sub command blocks" (those that belong to a begin-end block included within another block). An example is printed below.: "simulation" is a main command block and "newmark" is a sub command block.

```
begin simulation;
  time_stop    100.0 ;
  solvertype   2 ;    (sparse newmark)
;
  begin newmark;
    beta      0.27;
    gamma     0.51;
    deltat    0.02;
  end newmark;
end simulation;
```

5.1 Continue_in_file option

A feature from version 6.0 and newer is the possibility of continuing reading of the main input file into another. The command word continue_in_file followed by a file name causes the program to open the new file and continue reading of input until the command word exit. When exit is read the reading will continue in the previous file. An infinite number of file levels can be used. The HAWC2 input format is written in a form that forces the user to write the input commands in a structured way so aerodynamic commands are kept together, structural commands the same etc.

Command name	Explanation
continue_in_file	1. File name (and path) to sublevel input file
exit	End of input file. Input reading is continued in higher level input file.

6 HAWC2 version handling

The HAWC2 code is still frequently updated and version handling is therefore of utmost importance to ensure quality control. For every new released version of the code a new version number is hard coded in the source. This number can be found by executing the HAWC2.exe file without any parameters. The version number is echoed to screen. The same version number is also written to every result file no matter whether ASCII or binary format is chosen.

All information covering the different code versions has been made. These data are listed in appendix D.

7 Coordinate systems

The global coordinate system is located with the z-axis pointing vertical downwards. The x and y axes are horizontal to the side. When wind is submitted, the default direction is along the global y-axes. Within the wind system meteorological u,v,w coordinates are used, where u is the mean wind speed direction, v is horizontal and w vertical upwards. When x,y,z notation is used within the wind coo. this refers directly to the u,v,w definition. Every substructure and body (normally the same) is equipped with its own coordinate system with origo in node1 of this structure. The structure can be arbitrarily defined regarding orientation within this coordinate system. Within a body a number of structural elements are present. The orientation of coordinate systems for these elements are chosen automatically by the program. The local z axis is from node 1 to 2 on the element. The coordinate system for the blade structures must be defined with the z axis pointing from the blade root and outwards, x axis in the tangential direction of rotation and y axis from the pressure side towards the suction side of the blade profiles. This is in order to make the linkage between aerodynamics and structure function.

In order to make a quick check of the layout of the structure the small program “animation.exe” can be used (this requires than an animation file has been written using the command animation in the Simulation block). The view option in this program is handled by keyboard hotkeys:

Animation Hotkeys:

```
translate: (shift)+{x,y,z}
rotate: arrow keys
rotate about line-of-sight: ctrl+left/right
zoom in: ctrl+up
zoom out: ctrl+down
amplify displacement (only for animation of natural frequencies): +
decrease displacement (only for animation of natural frequencies): -
```

If the animation does not start, press “s”

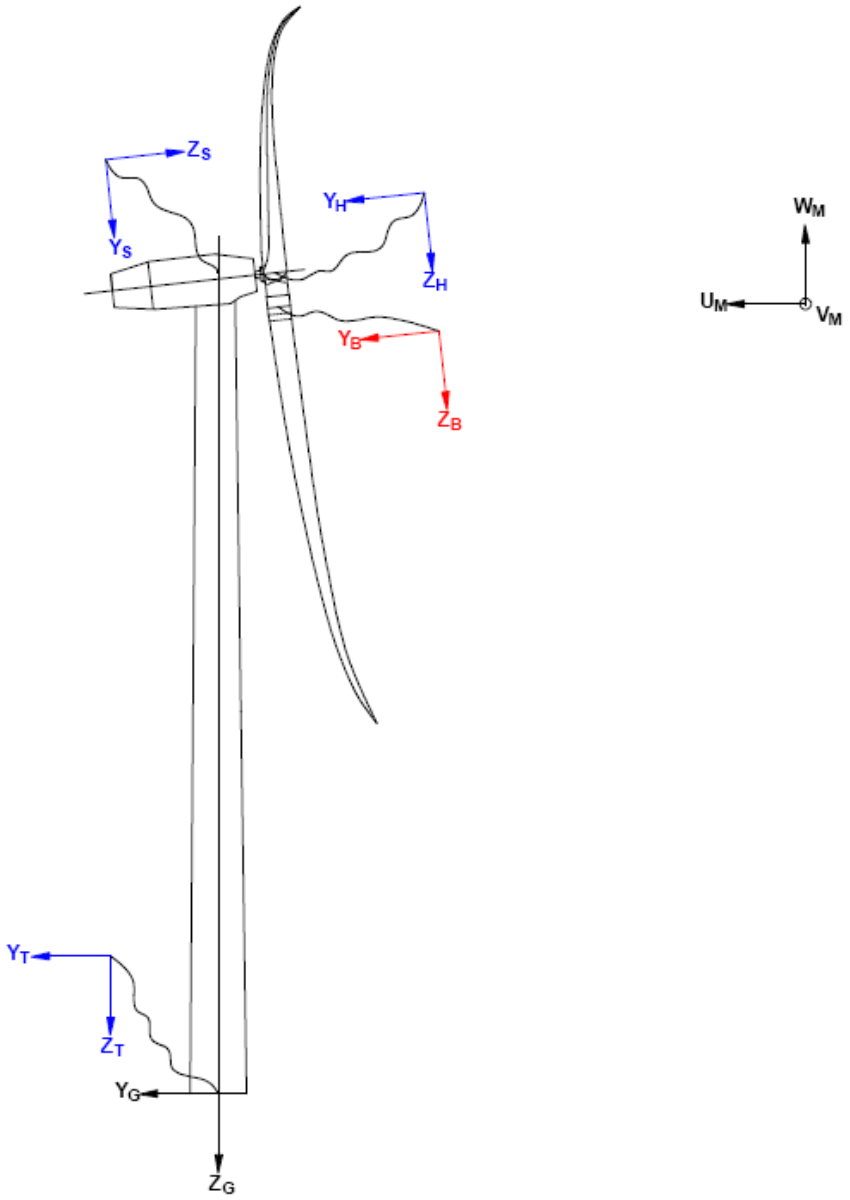


Figure 1: Illustration of coordinate system as result of user input from example in appendix A: Example of main input file. There are two coordinate systems in black which are the default coordinate systems of global reference and default wind direction. The blue coordinate systems are main body coordinate systems attached to node 1 of the substructure, the orientation of these are fully determined by the user. The red coordinate systems are also defined by the user, but in order to make the linkage between aerodynamic forces and structure work these have to have the z from root to tip, x in chordwise direction and y towards the suction side.

8 Simulation

8.1 Main command block - Simulation

Obl.	Command name	Explanation
*	time_stop	1. Simulation length [s]
	solvertype	1. Solver type (1=dense newmark (default), 2=sparse newmark (faster and recommended. New in version 12.7))
	solver_relax	1. Relaxation parameter on increment within a timestep. Can be used to make difficult simulation run through solver when parameter is decreased, however on the cost of simulation speed. Default=1.0
	on_no_convergence	Parameter that informs solver of what to do if convergence is not obtained in a time step. 1. 'stop': simulation stops – default. 'continue': simulation continues, error message is written.
	convergence_limits	Convergence limits that must be obtained at every time step. 1. epsresq, residual on internal-external forces, default=10.0 2. epsresd, residual on increment, default=1.0 3. epsresg, residual on constraint equations, default=1E-7
	max_iterations	1. Number of maximum iterations within a time step.
	animation	Included if animation file is requested 1. Animation file name incl. relative path. E.g. ./animation/animation1.dat
	visualization	Included if simulation visualization file is requested 1. Visualization file name incl. relative path. E.g. ./visualization/example.hdf5;
	logfile	Included if a logfile is requested internally from the htc command file. 1. Logfile name incl. relative path. E.g. ./logfiles/log1.txt
	log_deltat	If specified, iteration statitics is written to the log(file) every log_deltat seconds. Otherwise a log line is printed every time step. 1. Time between output to logfile [s], e.g. 2.5

8.2 Sub command block - newmark

Obl.	Command name	Explanation
	beta	1. beta value (default=0.27)
	gamma	1. gamma value (default=0.51)
*	deltat	1. time increment [s]
	symmetry	1. Solver assumtion regarding mass, damping and stiffness matrices (1=symmetric (default), 2=assymetric (recommended for offshore structures). When hydrodynamic loading is applyed this parameter will automatically change to 2.)

9 Structural input

9.1 Main command block - new_htc_structure

Obl.	Command name	Explanation
	beam_output_file_name	Write the beam properties for all bodies. 1. File name including relative path to file where the beam data are listed (output) (example ./info/beam.dat)
	body_output_file_name	Write the initial conditions and inertia matrix for all bodies. 1. File name including relative path to file where the body data are listed (output) (example ./info/body.dat)
	struct_inertia_output_file_name	For all bodies, write the inertia matrix, with respect to the center of gravity, in global and local coordinates. 1. File name including relative path to file where the global inertia information data are listed (output) (example ./info/inertia.dat)
	body_matrix_output	Write the assembled stiffness, damping and mass matrices for all bodies. 1. Folder name where the bodies structural matrices are listed (example ./info/body).
	element_matrix_output	Write the elements stiffness, damping and mass matrices. 1. File name including relative path to file where the elements structural matrices are listed (example ./info/element.dat).
	constraint_output_file_name	Write the initial conditions of the constraints in global coordinates. 1. File name including relative path to file where the constraint data are listed (output). (example ./info/constraint.dat)
	body_eigenanalysis_file_name	Do the eigenanalysis for all bodies (not recommended). Write the damped frequency, natural frequency and logarithmic decrement.
	structure_eigenanalysis_file_name	Do the eigenanalysis for the entire structure. Write the damped frequency, natural frequency, logarithmic decrement and animation of the mode shapes. 1. File name including relative path to file where the results of an complete turbine eigenanalysis are listed (example ./info/eigen_all.dat). Animation files are placed in the same directory of the file name. 2. Optional parameter determining if structural damping is included in the eigenvalue calculation or not. (0=damping not included, most robust method, 1=damping included default)
	system_eigenanalysis	Do the eigenanalysis for the entire structure, including external systems attached, eg. mooring lines. Constraint equations are also fully included in the analysis. Write the damped frequency, natural frequency, logarithmic decrement and animation of the mode shapes. 1. File name including relative path to file where the results of an complete turbine eigenanalysis are listed (example ./info/eigen_all.dat). Animation files are placed in the same directory of the file name. 2. (optional) Parameter determining if structural damping is included in the eigenvalue calculation or not. (0=damping not included, most robust method, 1=damping included default) 3. (optional) Number of modes outputted.

		4. (optional) Time for when the eigenanalysis is carried out. Eg. after a settling of a floating system.

9.2 Sub command block - main_body

This block can be repeated as many times as needed. For every block a new body is added to the structure. A main body is a collection of normal bodies which are grouped together for bookkeeping purposes related to input output. When a main body consist of several bodies the spacing the name of each body inherits the name of the master body and is given an additional name of '_#', where # is the body number. An example could be a main body called 'blade1' which consist of two bodies. These are then called 'blade1_1' and blade1_2' internally in the code. The internal names are only important if (output) commands are used that refers to the specific body name and not the main body name.

Obl.	Command name	Explanation
*	name	1. Main_body identification name (must be unique)
*	type	1. Element type used (options are: timoschenko)
*	nbodies	1. Number of bodies the main_body is divided into (especially used for blades when large deformation effects needs attention). Equal number of elements on each body, eventually extra elements are placed on the first body.
*	node_distribution	1. Distribution method of nodes and elements. Options are: "uniform" nnodes. Where uniform ensures equal element length and nnodes are the node numbers. "c2_def", which ensures a node a every station defined with the sub command block c2_def.
	damping	Original damping model that can only be used when the shear center location equals the elastic center to ensure a positive definite damping matrix. It is recommended to use the damping_posdef command instead. Rayleigh damping parameters containing factors that are multiplied to the mass and stiffness matrix respectfully. ! Pay attention, the mass proportional damping is not contributing when a mbdy consist of multiple bodies ! 1. M_x 2. M_y 3. M_z 4. K_x 5. K_y 6. K_z NOTE: This damping model cannot be used with the Fully Populated Matrix ("FPM 1", see below) beam element!
	damping_posdef	Rayleigh damping parameters containing factors. M_x, M_y, M_z are constants multiplied on the mass matrix diagonal and inserted in the damping matrix. K_x, K_y, K_z are factors multiplied on the moment of inertia I_x, I_y, I_z in the stiffness matrix and inserted in the damping matrix. Parameters are in size approximately the same as the parameters used with the original damping model written above. ! Pay attention, the contribution from mass proportional damping is limited when a mbdy consist of multiple bodies ! 1. M_x 2. M_y

		3. M_z 4. K_x 5. K_y 6. K_z NOTE: This damping model cannot be used with the Fully Populated Matrix ("FPM 1", see below) beam element!
	damping_aniso	Mixed mass/stiffness proportional and stiffness proportional damping parameters containing factors. η_x^m, η_y^m, η_t^m are constants multiplied on a mixed mass/stiffness matrix diagonal and inserted in the damping matrix. η_x^s, η_y^s, η_t^s are factors multiplied on the moment of inertia I_x, I_y, I_z in the stiffness matrix and inserted in the damping matrix. ! Pay attention, the mass proportional damping is not contributing when a mbdy consist of multiple bodies ! Damping_aniso will give a similar damping to damping_posdef if 1) only stiffness proportional damping is used (first three coefficients in both models are zero) and 2) the 4th and 5th parameters are swapped ($n_y^s = K_x$ and $n_x^s = K_y$) ! See the command for the corrected version of damping_aniso below ! 1. η_x^m 2. η_y^m 3. η_t^m 4. η_x^s 5. η_y^s 6. η_t^s
	damping_aniso_v2	Identical usage as damping_aniso, but a minor bug in the torsional damping computation has been fixed.
	damping_file	Pre-generated damping read from file - the file can be generated by the metod described in Section C. 1. File name.
	copy_main_body	Command that can be used if properties from a previously defined body shall be copied. The name command still have to be present, all other data are overwritten. 1. Main_body identification name of main_body that is copied.
	gravity	1. Specification of gravity (directed towards zG). NB! this gravity command only affects the present main body. Default=9.81 [m/s^2]
	concentrated_mass	Concentrated masses and inertias can be attached to the structure. The offset distance from the node to the center of mass is given in the body's coordinates system. The moments and products of inertia is given around the center of mass in the body's coordinates system. 1. Node number to which the inertia is attached. 2. Offset distance x-direction [m] 3. Offset distance y-direction [m] 4. Offset distance z-direction [m] 5. Mass [kg] 6. I_{xx} [kg m^2] 7. I_{yy} [kg m^2] 8. I_{zz} [kg m^2] 9. I_{xy} [kg m^2] – optional 10. I_{xz} [kg m^2] – optional

		11. I_{yz} [kg m^2] – optional
	external_bladedata_dll	Blade structural data are found in an external encrypted dll. If this command is present only these other command lines need to be present (name, type, nbodies, node_distribution and a damping command line). 1. Company name (that has been granted a password, eg. dtu). 2. Password for opening this specific dll, eg. test1234 3. path and filename for the dll. eg. ./data/encr_blade_data.dll

9.2.1 Sub sub command block – timoschenko_input

Block containing information about location of the file containing distributed beam property data and the data set requested.

Obl.	Command name	Explanation
*	filename	1. Filename incl. relative path to file where the distributed beam input data are listed (example ./data/hawc2_st.dat)
	FPM	Logic command for Fully Populated Matrix beam element: 1. Write "1" to read a structural input file based on the fully populated stiffness matrix. Write "0" for the original beam model If the command is neglected, HAWC2 will assume that the structural input file is based on the original beam model
*	set	1. Set number 2. Sub set number

9.2.2 Sub sub command block – c2_def

In this command block the definition of the centerline of the main_body is described (position of the half chord, when the main_body is a blade). The input data given with the sec commands below is used to define a continous differentiable line in space using akima spline functions. This centerline is used as basis for local coordinate system definitions for sections along the structure. If two input sections are given it is assumed that all points are on a straight line. If three input sections are given points are assumed to be on the line consisted of two straight lines. If four or more input sections are given points are assumed to be on an akima interpolated spline. This spline will include a straight line if a minimum of three points on this line is defined.

Obl.	Command name	Explanation
*	nsec	Must be the present before a "sec" command. 1. Number of section commands given below
*	sec	Command that must be repeated "nsec" times. Minimum 4 times. 1. Number 2. x-pos [m] 3. y-pos [m] 4. z-pos [m] 5. θ_z [deg]. Angle between local x-axis and main_body x-axis in the main_body x-y coordinate plane. For a straight blade this angle is the aerodynamic twist. Note that the sign is positive around the z-axis, which is opposite to traditional notation for etc. a pitch angle.

Here is an illustration of how a blade can be defined with respect to discretisation of bodies,

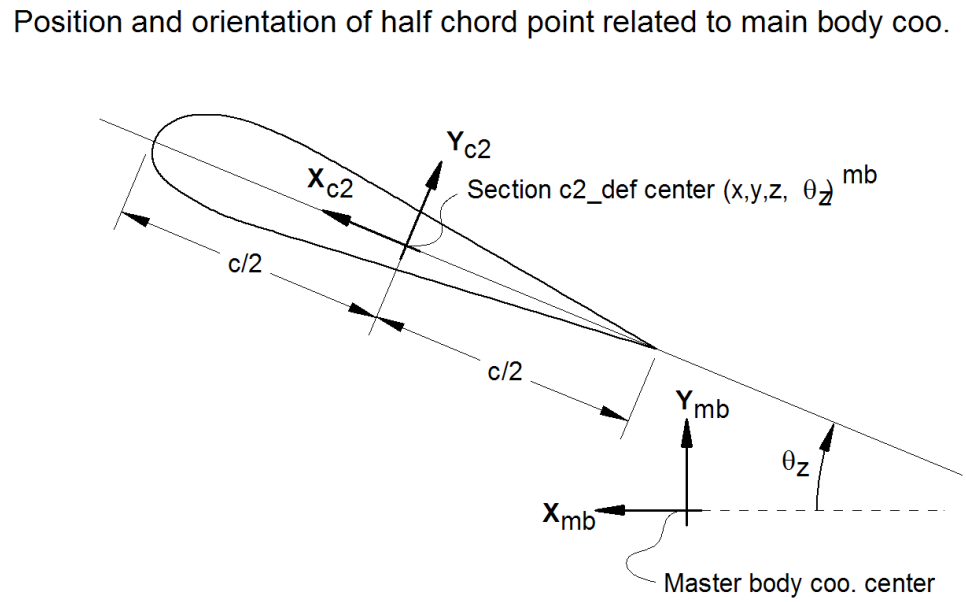
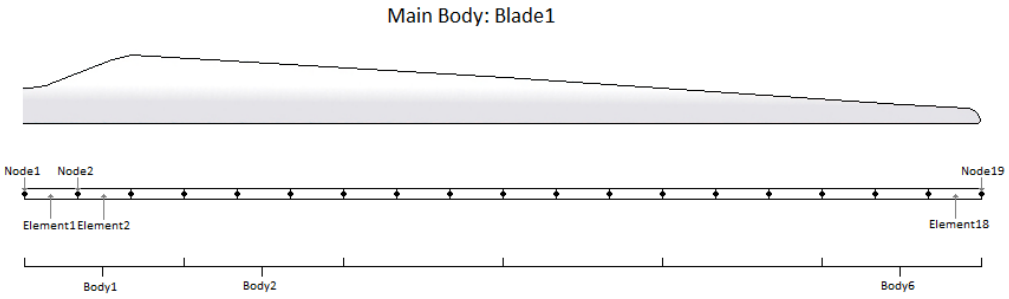


Figure 2: Illustration of c2_def coordinate system related to main body coordinates. The blade z-coordinate has to be positive from root towards the tip.

nodes and elements.



Here is an example of this written into the htc-input file.

```
begin main_body;
name        blade1 ;
type        timoschenko ;
nbodies     6 ;
node_distribution    c2_def;
damping_posdef   1.17e-4 5.77e-5 6.6e-6 6.6e-4 5.2e-4 6.5e-4 ;
begin timoschenko_input ;
filename ./data/st_file.txt ;
  FPM 0;  (optional, when parameter is 0)
  set 1 1 ;                set subset
end timoschenko_input;
begin c2_def;              Definition of centerline (main_body coordinates)
  nsec 19 ;
sec 1 -0.0000 0.0000 0.000  0.000 ;
sec 2 -0.0041 0.0010 3.278 -13.590 ;
sec 3 -0.1048 0.0250 6.556 -13.568 ;
sec 4 -0.2582 0.0492 9.833 -13.564 ;
sec 5 -0.4694 0.0587 13.111 -13.546 ;
sec 6 -0.5689 0.0957 16.389 -11.406 ;
```

```
sec 7 -0.5455 0.0883 19.667 -10.145 ;
sec 8 -0.5246 0.0732 22.944 -9.043 ;
sec 9 -0.4362 0.0669 26.222 -7.843 ;
sec 10 -0.4644 0.0554 29.500 -6.589 ;
sec 11 -0.4358 0.0449 32.778 -5.447 ;
sec 12 -0.4859 0.0347 36.056 -4.234 ;
sec 13 -0.3759 0.0265 39.333 -3.545 ;
sec 14 -0.3453 0.0130 42.611 -2.223 ;
sec 15 -0.3156 0.0084 45.889 -1.553 ;
sec 16 -0.2791 0.0044 49.167 -0.934 ;
sec 17 -0.2675 0.0017 52.444 -0.454 ;
sec 18 -0.1785 0.0003 55.722 -0.121 ;
sec 19 -0.1213 0.0000 59.000 -0.000 ;
 end c2_def ;
end main_body;
```

Format definition of file with distributed beam properties (st file) The format of this file, which in the old HAWC code was known as the hawc_st file, is changed slightly for the HAWC2 new_htc_structure format. The file is a text file in which the structural parameters are organized into main sets and sub sets. The main set is located after a "#" sign followed by the main set number. Within a main there can be as many subsets as desired. They are located after a "$" sign followed by the local set number. The next sign of the local set number is the number of lines in the following rows that belong to this sub set.

There are two types st_file:

- The st_file for the original HAWC2 beam element. Input parameters for this model are reported in Table 1 HAWC2 original beam element structural data.
- The st_file for the new anisotropic FPM beam element. Input parameters are reported in Table 2 New HAWC2 anisotropic beam element structural data.

Please note! The first column in the datasets, the curved-length distance from the main body's first node, *is normalized by HAWC2* using the curved length defined by the x, y and z coordinates given in the c2_def block in the htc file. In other words, if your curved length in the st file goes from 0 to 100 but the curved length defined by the c2_def coordinates has a max curved length of 50, then the st-file curved length will be normalized such that it goes from 0 to 50 and a warning will be printed in the log file. The curved length in the st file should start from 0. We recommend having consistent curved lengths in the st and htc files; consider using the `beam_output_file_name` to verify the lengths. For more information on how HAWC2 handles differing node locations in the htc file and st file, please see the structural module in the HAWC2 training course.

In general all centers are given according to the $C_{1/2}$ center location and all other are related to the principal bending axes. For the anisotropic beam element, centers are given according to the $C_{1/2}$ center location, but the cross sectional stiffness matrix is given at the elastic center rotated along the principal bending axes.

A small explanation about radius of gyration (also called radius of inertia) and the area moment of inertia (related to stiffness) is shown below in N.5 and N.11

An example of a st original beam formulation input file can be seen on the next page. The most important features to be aware of are colored with red.

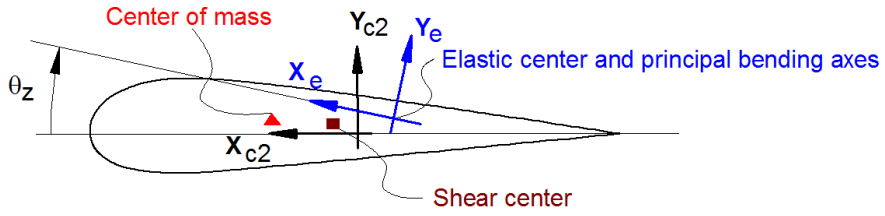


Figure 3: Illustration of structural properties that in the input files are related to the c2 coordinate system.

- **N.5** r_{ix} $[m]$ Radius of inertia. Related to the Moment of Inertia I_{xx} $[kg\ m^2]$, which gives the rotation inertia, resistance to change in rotation rate:

$$I_{xx} = \int r_{ix}^2 \mathrm{d}m \quad \rightarrow \quad r = \sqrt{\frac{I_{xx}}{m}} = \sqrt{\frac{I_x}{\mathrm{A}}}$$

- **N.11** I_x $[m^4]$ Area moment of inertia with respect to x_e. It's the second moment of area $I_x = \int y^2 \mathrm{d}A$. Multiplied by Young's modulus E gives the flapwise bending stiffness:

$$\mathrm{Stiffn}_{\mathrm{flap}} = E \cdot I_x = \frac{M}{\mathrm{d}^2 w / \mathrm{d}^2 x}$$

$$\mathrm{Stiffn}_{\mathrm{edge}} = E \cdot I_y$$

$$\mathrm{Stiffn}_{\mathrm{tors}} = \mathbf{G} \cdot \mathbf{K}$$

Table 3: HAWC2 original beam element structural data

Column	Parameter
1	r, curved length distance from main_body node 1 [m]. HAWC2 normalizes this by the curved length defined in c2_def.
2	m, mass per unit length [kg/m]
3	x_m, x_{c2}-coordinate from $C_{1/2}$ to mass center [m]
4	y_m, y_{c2}-coordinate from $C_{1/2}$ to mass center [m]
5	r_{ix}, radius of gyration related to elastic center. Corresponds to rotation about principal bending x_e axis [m]
6	r_{iy}, radius of gyration related to elastic center. Corresponds to rotation about principal bending y_e axis [m]
7	x_s, x_{c2}-coordinate from $C_{1/2}$ to shear center [m]. The shear center is the point where external forces only contributes to pure bending and no torsion.
8	y_s, y_{c2}-coordinate from $C_{1/2}$ to shear center [m]. The shear center is the point where external forces only contributes to pure bending and no torsion.
9	E, modulus of elasticity $[N/m^2]$
10	G, shear modulus of elasticity $[N/m^2]$
11	I_x, area moment of inertia with respect to principal bending x_e axis $[m^4]$. This is the principal bending axis most parallel to the x_{c2} axis
12	I_y, area moment of inertia with respect to principal bending ye axis $[m^4]$
13	K, torsional stiffness constant with respect to ze axis at the shear center $[m^4/rad]$. For a circular section only this is identical to the polar moment of inertia.
14	k_x shear factor for force in principal bending x_e direction [-]
15	k_y, shear factor for force in principal bending ye direction [-]
16	A, cross sectional area $[m^2]$
17	θ_z, structural pitch about z_{c2} axis. This is the angle between the x_{c2} -axis defined with the c2_def command and the main principal bending axis x_e. [deg]
18	x_e, x_{c2}-coordinate from $C_{1/2}$ to center of elasticity [m]. The elastic center is the point where radial force (in the z-direction) does not contribute to bending around the x or y directions.
19	y_e, y_{c2}-coordinate from $C_{1/2}$ to center of elasticity [m]. The elastic center is the point where radial force (in the z-direction) does not contribute to bending around the x or y directions.

```
1 main data sets available
-----------------------------------
Here is space for comments etc
.
.
.
-----------------------------------
#1 Main data set number 1 - an example of a shaft structure
-----------------------------------
More comments space
```

r	m	x_cg	y_cg	ri_x	ri_y	x_sh	y_sh	E	G	I_x	I_y	K	k_x	k_y	A	theta_s	x_e	y_e
[m]	[kg/m]	[m]	[m]	[m]	[m]	[m]	[m]	[N/m^2]	[N/m^2]	[N/m^4]	[N/m^4]	[N/m^4]	[-]	[-]	[m^2]	[deg]	[m]	[m]
$1 10 Sub set number 1 with 10 data rows																		
0.00	100	0	0	224.18	224.18	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
0.10	100	0	0	224.18	224.18	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
0.1001	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
1.00	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
1.90	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
2.00	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
3.00	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
3.20	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
4.00	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
5.0191	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0

```
-----------------------------------
More comments space
```

r	m	x_cg	y_cg	ri_x	ri_y	x_sh	y_sh	E	G	I_x	I_y	K	k_x	k_y	A	theta_s	x_e	y_e
[m]	[kg/m]	[m]	[m]	[m]	[m]	[m]	[m]	[N/m^2]	[N/m^2]	[N/m^4]	[N/m^4]	[N/m^4]	[-]	[-]	[m^2]	[deg]	[m]	[m]
$2 10 As dataset 1, but stiff																		
0.00	100	0	0	224.18	224.18	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
0.10	100	0	0	224.18	224.18	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
0.1001	1	0	0	0.2	0.2	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
1.00	1	0	0	0.2	0.2	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
1.90	1	0	0	0.2	0.2	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
2.00	1	0	0	0.2	0.2	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
3.00	1	0	0	0.2	0.2	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
3.20	1	0	0	0.2	0.2	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
4.00	1	0	0	0.2	0.2	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
5.0191	1	0	0	0.2	0.2	0	0	2.10E+16	8.10E+15	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0

```
-----------------------------------
More comments space
```

r	m	x_cg	y_cg	ri_x	ri_y	x_sh	y_sh	E	G	I_x	I_y	K	k_x	k_y	A	theta_s	x_e	y_e
[m]	[kg/m]	[m]	[m]	[m]	[m]	[m]	[m]	[N/m^2]	[N/m^2]	[N/m^4]	[N/m^4]	[N/m^4]	[-]	[-]	[m^2]	[deg]	[m]	[m]
$3 10 as data set 1 but changed mass properties																		
0.00	1000	0	0	2.2418	2.2418	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
0.10	1000	0	0	2.2418	2.2418	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
0.1001	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
1.00	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
1.90	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
2.00	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
3.00	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
3.20	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
4.00	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0
5.0191	1	0	0	0.2	0.2	0	0	2.10E+11	8.10E+10	1.00E+02	1.00E+02	0.05376	0.52	0.52	0.59	0	0.0	0.0

Table 4: New HAWC2 anisotropic beam element structural data

Column	
1	r, curved length distance from main_body node 1 [m]. HAWC2 normalizes this by the curved length defined in c2_def.
2	m, mass per unit length [kg/m]
3	x_m, x_{c2}-coordinate from $C_{1/2}$ to mass center [m]
4	y_m, y_{c2}-coordinate from $C_{1/2}$ to mass center [m]
5	r_{ix}, radius of gyration related to elastic center. Corresponds to rotation about principal bending x_e axis [m]
6	r_{iy}, radius of gyration related to elastic center. Corresponds to rotation about principal bending y_e axis [m]
7	θ_z, structural pitch about z_{c2} axis. This is the angle between the x_{c2} -axis defined with the c2_def command and the main principal bending axis x_e.
8	x_e, x_{c2}-coordinate from $C_{1/2}$ to center of elasticity [m]. The elastic center is the point where radial force (in the z-direction) does not contribute to bending around the x or y directions.
9	y_e, y_{c2}-coordinate from $C_{1/2}$ to center of elasticity [m]. The elastic center is the point where radial force (in the z-direction) does not contribute to bending around the x or y directions.
10	K_{11}, element 1,1 of the Cross sectional stiffness matrix [N]. REMEMBER: the cross sectional stiffness matrix is given at the elastic center rotated along the principal bending axes.
11	K_{12}, element 1,2 of the Cross sectional stiffness matrix [N].
12	K_{13}, element 1,3 of the Cross sectional stiffness matrix [N].
13	K_{14}, element 1,4 of the Cross sectional stiffness matrix [Nm].
14	K_{15}, element 1,5 of the Cross sectional stiffness matrix [Nm].
15	K_{16}, element 1,6 of the Cross sectional stiffness matrix [Nm].
16	K_{22}, element 2,2 of the Cross sectional stiffness matrix [N].
17	K_{23}, element 2,3 of the Cross sectional stiffness matrix [N].
18	K_{24}, element 2,4 of the Cross sectional stiffness matrix [Nm].
19	K_{25}, element 2,5 of the Cross sectional stiffness matrix [Nm].
20	K_{26}, element 2,6 of the Cross sectional stiffness matrix [Nm].
21	K_{33}, element 3,3 of the Cross sectional stiffness matrix [N].
22	K_{34}, element 3,4 of the Cross sectional stiffness matrix [Nm].
23	K_{35}, element 3,5 of the Cross sectional stiffness matrix [Nm].
24	K_{36}, element 3,6 of the Cross sectional stiffness matrix [Nm].
25	K_{44}, element 4,4 of the Cross sectional stiffness matrix [Nm^2].
26	K_{45}, element 4,5 of the Cross sectional stiffness matrix [Nm^2].
27	K_{46}, element 4,6 of the Cross sectional stiffness matrix [Nm^2].
28	K_{55}, element 5,5 of the Cross sectional stiffness matrix [Nm^2].
29	K_{56}, element 5,6 of the Cross sectional stiffness matrix [Nm^2].
30	K_{66}, element 6,6 of the Cross sectional stiffness matrix [Nm^2].

An example of a st anisotropic beam formulation input file can be seen on the next page.

```
-----------------
#1  user_cpav generated blade
```

r [0]	m [1]	x_cg [2]	y_cg [3]	ri_x [4]	ri_y [5]	x_e [6]	y_e [7]	K_11 [8]	K_12 [9]	K_13 [10]	K_14 [11]	K_15 [12]
$1 31												
0.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
3.3333333333e-01	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
6.6666666667e-01	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
1.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
1.3333333333e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
1.6666666667e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
2.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
2.3333333333e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
2.6666666667e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
3.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
3.3333333333e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
3.6666666667e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
4.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
4.3333333333e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
4.6666666667e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
5.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
5.3333333333e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
5.6666666667e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
6.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
6.3333333333e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
6.6666666667e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
7.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
7.3333333333e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
7.6666666667e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
8.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
8.3333333333e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
8.6666666667e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
9.0000000000e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
9.3333333333e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
9.6666666667e+00	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0
1.0000000000e+01	8.5380000000e-02	0.0000000000e+00	0.0000000000e+00	3.4793504562e-01	2.1906122240e-01	0.0000000000e+00	0.0000000000e+00	8.8560000000e+04	0.0000000000e+00	0.0000000000e+00	0.0000000000e+00	0.0

9.2.3 Sub sub command - damping_distributed

In this command block, Rayleigh damping parameters can be defined as function of blade length, hence damping parameters can be different at root of tip of a blade.

Obl.	Command name	Explanation
*	nsec	Number of input lines
*	sec	This command must be repeated nsec times. 1. r/R. Non-dim distance from node 1 to node N. 2. k_x Stiffness proportional damping around x 3. k_y Stiffness proportional damping around y 4. k_z Stiffness proportional damping around z

9.2.4 Sub sub command – damping_posdef_distributed

In this command block, Rayleigh damping parameters can be defined as function of blade length, hence damping parameters can be different at root of tip of a blade.

Obl.	Command name	Explanation
*	nsec	Number of input lines
*	sec	This command must be repeated nsec times. 1. r/R. Non-dim distance from node 1 to node N. 2. k_x Stiffness proportional damping around x 3. k_y Stiffness proportional damping around y 4. k_z Stiffness proportional damping around z

9.2.5 Sub sub command – visualization_profile

This command block is used together with the command name visualization in the main command block simulation. Default profiles are:

- Blade: An aerodynamic profile where thickness <95%, otherwise a cylinder. Dimensions as specified in the aerodynamic blade layout file.

- Other bodies: Cylinder. The diameter is calculated from the mass and inertia specified in the structural data

Obl.	Command name	Explanation
*	type	Profile type. (options are: “cylinder”, “cube” and “blade”)
*	nsec	Number of visualization sections
*	sec	This command must be repeated nsec times. 1. Distance from root [m or % or any other unit of choice (scaled relative to the largest number)] 2. Diameter (cylinder), width (cube), chord (blade) [m] 3. (not needed for cylinder), height (cube) [m], thickness (blade) [%]

9.3 Sub command - orientation

In this command block the orientation (regarding position and rotation) of every main_body are specified.

9.3.1 Sub sub command - base

The orientation of a main_body to which all other bodies are linked – directly or indirectly.

Obl.	Command name	Explanation
*	mbdy (old command name body still usable)	1. Main_body name that is declared to be the base of all bodies (normally the tower or foundation)
*	inipos	Initial position in global coordinates. 1. x-pos [m] 2. y-pos [m] 3. z-pos [m]
♣	mbdy_eulerang (old command name body_eulerang still usable)	Command that can be repeated as many times as needed. All following rotation are given as a sequence of euler angle rotations. All angle can be filled in (rotation order x,y,z), but it is recommended only to give a value different from zero on one of the angles and reuse the command if several rotations are needed. 1. θ_x [deg] 2. θ_y [deg] 3. θ_z [deg]
♣	mbdy_eulerpar (old command name body_eulerpar still usable)	The rotation is given as euler parameters (quaternions) directly (global coo). 1. r_0 2. r_1 3. r_2 4. r_3
♣	mbdy_axisangle (old command name body_axisangle still usable)	Command that can be repeated as many times as needed. A version of the euler parameters where the input is a rotation vector and the rotation angle of this vector. 1. x-value 2. y-value 3. z-value 4. angle [deg]
	mbdy_ini_rotvec_d1	Initial rotation velocity of main body and all subsequent attached bodies. A rotation vector is set up and the size of vector (the rotational speed) is given. The coordinate system used is main_body coo. 1. x-value 2. y-value 3. z-value 4. Vector size (rotational speed [rad/s])

♣ One of these commands must be present.

9.3.2 Sub sub command - relative

This command block can be repeated as many times as needed. However the orientation of every main_body should be described.

Obl.	Command name	Explanation
*	mbdy1	1. Main_body name to which the next main_body is attached.

	(old command name body1 still usable)	2. Node number of body1 that is used for connection. (“last” can be specified which ensures that the last node on the main_body is used).
*	mbdy2 (old command name body2 still usable)	1. Main_body name of the main_body that is positioned in space by the relative command. 2. Node number of body2 that is used for connection. (“last” can be specified which ensures that the last node on the main_body is used).
♣	mbdy2_eulerang (old command name body2_eulerang still usable)	Command that can be repeated as many times as needed. All following rotation are given as a sequence of euler angle rotations. All angle can be filled in (rotation order x,y,z), but it is recommended only to give a value different from zero on one of the angles and reuse the command if several rotations are needed. Until a rotation command is specified body2 has same coo. as body1. Rotations are performed in the present body2 coo. system. 1. θ_x [deg] 2. θ_y [deg] 3. θ_z [deg]
♣	mbdy2_eulerpar (old command name body2_eulerpar still usable)	The rotation is given as euler parameters (quaternions) directly (global coo). 1. r_0 2. r_1 3. r_2 4. r_3
♣	mbdy2_axisangle (old command name body2_axisangle still usable)	Command that can be repeated as many times as needed. A version of the euler parameters where the input is a rotation vector and the rotation angle of this vector. Until a rotation command is specified main_body2 has same coo. as main_body1. Rotations are performed in the present main_body2 coo. system. 1. x-value 2. y-value 3. z-value 4. angle [deg]
	mbdy2_ini_rotvec_d1 (old command name body2_ini_rotvec_d1 still usable)	Initial rotation velocity of main body and all subsequent attached bodies. A rotation vector is set up and the size of vector (the rotational speed) is given. The coordinate system used is main_body2 coo. 1. x-value 2. y-value 3. z-value 4. Vector size (rotational speed [rad/s])
	relpos	Vector from coupling node of mbdy 1 to coupling node of mbdy 2 in mbdy1 coo system in case a certain distance between these nodes is required. (Default for overlapping coupling nodes, this vector is (0,0,0))

		1. x-value 2. y-value 3. z-value

9.4 Sub command - constraint

In this block constraints between the main_bodies and to the global coordinate system are defined.

9.4.1 Sub sub command – fix0

This constraint fix node number 1 of a given main_body to ground.

Obl.	Command name	Explanation
*	mbdy (old command name body still usable)	Name of main body that is fixed to ground at node 1
	disable_at	Time to which constraint can be disabled 1. t_0
	enable_at	Time to which constraint can be enabled 1. t_0

9.4.2 Sub sub command – fix1

This constraint fix a given node on one main_body to another main_body's node.

Obl.	Command name	Explanation
*	mbdy1 (old command name body1 still usable)	1. Main_body name to which the next main_body is fixed. 2. Node number of main_body1 that is used for the constraint. ("last" can be specified which ensures that the last node on the main_body is used).
*	mbdy2 (old command name body2 still usable)	1. Main_body name of the main_body that is fixed to main_body1. 2. Node number of main_body2 that is used for the constraint. ("last" can be specified which ensures that the last node on the main_body is used).
	disable_at	Time to which constraint can be disabled 1. t_0
	enable_at	Time to which constraint can be enabled 1. t_0

9.4.3 Sub sub command – fix2

This constraint fix a node 1 on a main_body to ground in x,y,z direction. The direction that is free or fixed is optional.

Obl.	Command name	Explanation
*	mbdy (old command name body still usable)	1. Main_body name to which node 1 is fixed.
*	dof	Direction in global coo that is fixed in translation 1. x-direction (0=free, 1=fixed) 2. y-direction (0=free, 1=fixed) 3. z-direction (0=free, 1=fixed)

9.4.4 Sub sub command – fix3

This constraint fix a node to ground in t_x,t_y,t_z rotation direction. The rotation direction that is free or fixed is optional.

Obl.	Command name	Explanation
*	mbdy (old command name body still usable)	1. Main_body name to which node 1 is fixed. 2. Node number
*	dof	Direction in global coo that is fixed in rotation 1. tx-rot.direction (0=free, 1=fixed) 2. ty-rot.direction (0=free, 1=fixed) 3. tz-rot.direction (0=free, 1=fixed)

9.4.5 Sub sub command – fix4

Constraint that locks a node on a body to another node in translation but not rotation with a pre-stress feature. The two nodes will start at the defined positions to begin with but narrow the distance until fully attached at time T.

Obl.	Command name	Explanation
*	mbdy1 (old command name body1 still usable)	1. Main_body name to which the next main_body is fixed. 2. Node number of main_body1 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	mbdy2 (old command name body2 still usable)	1. Main_body name of the main_body that is fixed to body1. 2. Node number of main_body2 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
	time	1. Time for the pre-stress process. Default=2sec
	disable_at	Time to which constraint can be disabled 1. t_0
	enable_at	Time to which constraint can be enabled 1. t_0

9.4.6 Sub sub command – bearing1

Constraint with properties as a bearing without friction. A sensor with same identification name as the constraint is set up for output purpose.

Obl.	Command name	Explanation
*	name	1. Identification name
*	mbdy1 (old command name body1 still usable)	1. Main_body name to which the next main_body is fixed with bearing1 properties. 2. Node number of main_body1 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	mbdy2 (old command name body2 still usable)	1. Main_body name of the main_body that is fixed to body1 with bearing1 properties. 2. Node number of main_body2 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	bearing_vector	Vector to which the free rotation is possible. The direction of this vector also defines the coo to which the output angle is defined. 1. Coo. system used for vector definition (0=global,1=mbdy1,2=mbdy2) 2. x-axis 3. y-axis 4. z-axis
	sensor_offset_deg	User defined initial bearing angle in degrees. Used for sensor (output). 1. θ_0 [deg]
	sensor_offset_rad	User defined initial bearing angle in radians. Used for sensor (output). 1. θ_0 [rad]
	disable_at	Time to which constraint can be disabled 1. t_0
	enable_at	Time to which constraint can be enabled 1. t_0

9.4.7 Sub sub command – bearing2

This constraint allows a rotation where the angle is directly specified by an external dll action command.

Obl.	Command name	Explanation
*	name	1. Identification name
*	mbdy1 (old command name body1 still usable)	1. Main_body name to which the next main_body is fixed with bearing2 properties. 2. Node number of main_body1 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	mbdy2 (old command name body2 still usable)	1. Main_body name of the main_body that is fixed to main_body1 with bearing1 properties. 2. Node number of main_body2 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	bearing_vector	Vector to which the rotation occur. The direction of this vector also defines the coo to which the output angle is defined. 1. Coo. system used for vector definition (0=global,1=mbdy1, 2=mbdy2) 2. x-axis 3. y-axis 4. z-axis
	sensor_offset_deg	User defined initial bearing angle in degrees. Used for sensor (output) and control (input). 1. θ_0 [deg]
	sensor_offset_rad	User defined initial bearing angle in radians. Used for sensor (output) and control (input). 1. θ_0 [rad]
	disable_at	Time to which constraint can be disabled 1. t_0
	enable_at	Time to which constraint can be enabled 1. t_0

9.4.8 Sub sub command – bearing3

This constraint allows a rotation where the angle velocity is kept constant throughout the simulation.

Obl.	Command name	Explanation
*	name	1. Identification name
*	mbdy1 (old command name body1 still usable)	1. Main_body name to which the next main_body is fixed with bearing3 properties. 2. Node number of main_body1 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	mbdy2 (old command name body2 still usable)	1. Main_body name of the main_body that is fixed to body1 with bearing3 properties. 2. Node number of main_body2 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	bearing_vector	Vector to which the rotation occur. The direction of this vector also defines the coo to which the output angle is defined. 1. Coo. system used for vector definition (0=global,1=body1,2=body2) 2. x-axis 3. y-axis 4. z-axis
*	omegas	1. Rotational speed [rad/sec]

9.4.9 Sub sub command – bearing4

This constraint is a cardan shaft constraint. Locked in relative translation. Locked in rotation around one vector and allows rotation about the two other directions.

Obl.	Command name	Explanation
*	name	1. Identification name
*	mbdy1 (old command name body1 still usable)	1. Main_body name to which the next main_body is fixed with bearing3 properties. 2. Node number of main_body1 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	mbdy2 (old command name body2 still usable)	1. Main_body name of the main_body that is fixed to body1 with bearing3 properties. 2. Node number of main_body2 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	bearing_vector	Vector to which the rotation is locked. The rotation angle and velocity can be outputted around the two perpendicular directions. 1. Coo. system used for vector definition (0=global,1=mbdy1, 2=mbdy2) 2. x-axis 3. y-axis 4. z-axis

9.4.10 Sub sub command – bearing5

This constraint is a spherical constraint. Locked in relative translation. Free in rotation around all three axis, but only sensor on the main rotation direction.

Obl.	Command name	Explanation
*	name	1. Identification name
*	mbdy1 (old command name body1 still usable)	1. Main_body name to which the next main_body is fixed with bearing3 properties. 2. Node number of main_body1 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	mbdy2 (old command name body2 still usable)	1. Main_body name of the main_body that is fixed to body1 with bearing3 properties. 2. Node number of main_body2 that is used for the constraint. (“last” can be specified which ensures that the last node on the main_body is used).
*	bearing_vector	Vector to which the rotation is locked. The rotation angle and velocity can be outputted around the two perpendicular directions. 1. Coo. system used for vector definition (0=global,1=mbdy1, 2=mbdy2) 2. x-axis 3. y-axis 4. z-axis

10 DLL control

This block contains the possible Dynamic Link Library formats accessible for the user. The DLL's are mainly used to control the turbine speed and pitch, but since the DLL format is very general, other use is possible too e.g. external loading of the turbine. Since the HAWC2 core has no information about external stiffness or inertia we have experienced some issues with the solver if the DLL includes high stiffness terms or especially large inertia terms. The new `type2_dll` interface is slightly more stable related to the solver than the `hawc_dll` interface.

10.1 Main command block – dll

There are two DLL mechanisms available: `hawc_dll` and `type2_dll`. Both have two different interfaces (as documented in more detailed in the following sections 10.3 and 10.4) and have one other important distinction: a `hawc_dll` is updated in each aero-structure iteration, i.e. typically multiple times per time step while the `type2_dll` is only updated once per time step.

10.2 Important note about DLL file names

For both DLL interfaces the user needs to refer to the location of the specific DLL in use. Since version 12.9 HAWC2 is available for 3 different architectures (Windows 32-bit, Windows 64-bit and Linux 64-bit). To facilitate easy use of the same htc file across the different architectures, the intention is that with a single htc input file a user should be able to run on win32, win64 and linux without modifications. To this end, HAWC2 is using the following strategy:

- Determine what file name extension to use:
  - Win32: `.dll`
  - Win64: `_64.dll` (recommended), `.dll`
  - Linux: `.so`
- Find the correct path of the dll:
  - Absolute path (if the absolute path is specified)
  - Relative path relative to:
    - Current working directory (cwd)
    - The location of the HAWC2 executable
- On Linux, paths and file names are case sensitive (in contrast to Windows). Functionality to mimic the Windows behaviour on Linux has therefore been added. This functionality tries the following:
  - Load the exact specified filename (Note the automatic convertion to lower case and the exceptions described below)
  - Find the first filename that case-insentively matches the specified filename. This is done using `find /my/dir -maxdepth 1 -type f -ipath '*my_dll_name.so'`. Note: "first" may be arbitrary. Hence, avoid to have multiple files with the same name except for their case (e.g. `my_hawc2_dll.so` and `My_HAWC2_dll.so`) in the same folder.
    Note: With thousands of parallel simulations this behaviour may be problematic for the file system. Every use of `find`-command is therefore printed to the log file and in case the usage can be avoided by specifying the correct case-sensitive filename a warning is printed too.
- All input in the htc file(s) are converted to lower case with the following exceptions:
  - single-quoted strings, e.g. `'dont_CHANGE_case.dll'`

  - htc lines starting with `filename`
  - htc lines starting with `continue_in_file`
- Note that the log file will always report which files have been loaded so in case of doubt inspect that.

Each DLL needs to be compiled for each of the three different platforms independently, but with this functionality, the same input htc file, e.g.

```
...
begin dll;
  begin type2_dll;
    name 'MyDLL';
    filename  ./my_folder/MyDLL.dll ;
    ...
  end type2_dll;
end dll;
...
```

will load and use the correct dll on all platforms if the three files, `MyDLL.dll` (win32 compilation), `MyDLL_64.dll` (win64 compilation) and `MyDLL.so` (linux compilation) is put in `my_folder`.

10.3 Sub command block – `hawc_dll`

In the `hawc_dll` format a subroutine within an externally written DLL is setup. In this subroutine call two one-dimensional arrays are transferred between the HAWC2 core and the DLL procedure. The first contains data going from the HAWC2 core to the DLL and the other contains data going from the DLL to the core. It is very important to notice that the data is transferred between HAWC2 and the DLL in every time step and every iteration. The user should handle the iteration inside the DLL.

Two more subroutines are called if they are present inside the dll file:

The first is an initialisation call including a text string written in the init_string in the commands below. This could be the name of a file holding local input parameters to the data transfer subroutine. This call is only performed once. The name of this subroutine is the same name as the data transfer subroutine defined with the command dll_subroutine below with the extra name '_init', hence is the data transfer subroutine is called 'test', the initialisation subroutine will be 'test_init'.

The second subroutine is a message exchange subroutine, where messages written in the DLL can be send to the HAWC2 core for logfile writing. The name of this subroutine is the same name as the data transfer subroutine defined with the command dll_subroutine below with the extra name '_message', hence is the data transfer subroutine is called 'test', the initialisation subroutine will be 'test_message'.

The command block can be repeated as many times as desired. Reference number to DLL is same order as listed, starting with number 1. However it is recommended to refer the DLL using the name feature which in many cases can avoid confusion.

Obl.	Command name	Explanation
	name	1. Reference name of this DLL (to be used with DLL output commands)
*	filename	1. Filename incl. relative path of the DLL (example ./DLL/control.dll)
*	dll_subroutine	1. Name of subroutine in DLL that is addressed (remember to specify the name in the DLL with small letters!)
*	arraysizes	1. size of array with outgoing data 2. size of array with ingoing data
	deltat	1. Time between dll calls. Must correspond to the simulation sample frequency or be a multiple of the time step size. If deltat=0.0 or the deltat command line is omitted the HAWC2 code calls the dll subroutine at every time step.
	init_string	1. Text string (max 256 characters) that will be transferred to the DLL through the subroutine 'subroutine_init'. Subroutine is the name given in in the command dll_subroutine. No blanks can be included.

10.4 Sub command block – `type2_dll`

This dll interface is an updated slightly modified version of the `hawc_dll` interface. In the `type2_dll` format a subroutine within an externally written DLL is setup. In this subroutine call two one-dimensional arrays are transferred between the HAWC2 core and the DLL procedure. The first contains data going from the HAWC2 core to the DLL and the other contains data going from the DLL to the core. It is very important to notice that the data are transferred between HAWC2 and the DLL in the first call of every time step where the out-going variables are based on last iterated values from previous time step. The sub command output and actions are identical for both the `hawc_dll` and the `type2_dll` interfaces.

In the dll connected with using the `type2_dll` interface two subroutines should be present. An initialization routine called only once before the time simulation begins, and an update routine called in every time step. The format in the calling of these two subroutines are identical where two arrays of double precision is exchanged. The subroutine uses the cdecl calling convention.

Obl.	Command name	Explanation
	name	1. Reference name of this DLL (to be used with DLL output commands)
*	filename	1. Filename incl. relative path of the DLL (example ./DLL/control.dll)
*	dll_subroutine_init	1. Name of initialization subroutine in DLL that is addressed (remember to specify the name in the DLL with small letters!)
*	dll_subroutine_update	1. Name of subroutine in DLL that is addressed at every time step (remember to specify the name in the DLL with small letters!)
*	arraysizes_init	1. size of array with outgoing data in the initialization call 2. size of array with ingoing data in the initialization call
*	arraysizes_update	1. size of array with outgoing data in the update call 2. size of array with ingoing data in the update call
	deltat	1. Time between dll calls. Must correspond to the simulation sample frequency or be a multiple of the time step size. If deltat=0.0 or the deltat command line is omitted the HAWC2 code calls the dll subroutine at every time step.

when using the type2_dll interface the values transferred to the DLL in the initialization phase is done using a sub command block called init. The commands for this subcommand block is identical to the output subcommand explained below, but only has the option of having the constant output sensor available. An example is given for a small dll that is used for converting rotational speed between high speed and low speed side of a gearbox:

```
begin dll;
  begin type2_dll;
    name hss_convert;
    filename  ./control/hss_convert.dll ;
    arraysizes_init  3 1 ;
    arraysizes_update  2 2 ;
    begin init;
      constant 1 2.0 ;     number of used sensors - in this case only 1
      constant 2 35.110;   gearbox ratio
      constant 3 35.110;   gearbox ratio
    end init;
    begin output;
      constraint bearing1 shaft_rot 2 only 2 ;   rotor speed in rpm
      constraint bearing1 shaft_rot 3 only 2 ;   rotor speed in rad/s
    end output;
;
    begin actions;
;      rotor speed in rpm * gear_ratio
;      rotor speed in rad/s * gear_ratio
    end actions;
  end type2_dll;
end dll;
```

10.5 Sub command block – init

In this block type2_dlls can be initialized by passing constants to specific channels.

Obl.	Command name	Explanation
*	constant	Constants passed to the dll. 1. Channel number 2. Constant value

10.6 Sub command block – output

In this block the same sensors are available as when data results are written to a file with the main block command output, see section 17. The order of the sensors in the data array is continuously increased as more sensors are added.

10.7 Sub command block – actions

In this command block variables inside the HAWC2 code is changed depending of the specifications. This command block can be used for the hawc_dll interface as well as the type2_dll interface. An action commands creates a handle to the HAWC2 model to which a variable in the input array from the DLL is linked.

!NB in the command name two separate words are present.

Obl.	Command name	Explanation
	aero beta	The flap angle beta is set for a trailing edge flap section (is the mhhmagf stall model is used). The angle is positive towards the pressure side of the profile. Unit is [deg] 1. Blade number 2. Flap section number
	aero bem_grid_a	1. Number of points
	body force_ext	An external force is placed on the structure. Unit is [N]. 1. body name 2. node number 3. componet (1 = F_x, 2 = F_y, 3 = F_z)
	body moment_ext	An external moment is placed on the structure. Unit is [Nm]. 1. body name 2. node number 3. component (1 = M_x, 2 = M_y, 3 = M_z)
	body force_int	An external force with a reaction component is placed on the structure. Unit is [N]. 1. body name for action force 2. node number 3. component (1 = F_x, 2 = F_y, 3 = F_z) 4. body name for reaction force 5. Node number
	body moment_int	An external moment with a reaction component is placed on the structure. Unit is [N]. 1. body name for action moment 2. node number 3. component (1 = M_x, 2 = M_y, 3 = M_z) 4. body name for reaction moment 5. Node number
	body bearing_angle	A bearing either defined through the new structure format through bearing2 or through the old structure format (spitch1=pitch angle for blade 1, spitch2=pitch angle for blade 2,...). The angle limits are so far [0-90deg]. 1. Bearing name
	mbdy force_ext	An external force is placed on the structure. Unit is [N]. 1. main body name 2. node number on main body 3. component (1 = F_x, 2 = F_y, 3 = F_z), if negative number the force is inserted with opposite sign. 4. coordinate system (possible options are: mbdy name, ”global”, ”local”). “local” means local element coo on the inner element (on the element indexed 1 lower that the node number). One exception if node number =1 then the element nr. also equals 1.
	mbdy moment_ext	An external moment is placed on the structure. Unit is [Nm]. 1. main body name 2. node number on main body 3. component (1 = M_x, 2 = M_y, 3 = M_z), if negative number the moment is inserted with opposite sign.

		4. coordinate system (possible options are: mbdy name,"global","local"). "local" means local element coo on the inner element (on the element indexed 1 lower that the node number). One exception if node number =1 then the element nr. also equals 1.
	mbdy force_int	An internal force with a reaction component is placed on the structure. Unit is [N]. 1. main body name for action force 2. node number on main body 3. component (1 = F_x, 2 = F_y, 3 = F_z), if negative number the force is inserted with opposite sign. 4. coordinate system (possible options are: mbdy name, "global", "local"). "local" means local element coo on the inner element (on the element indexed 1 lower that the node number). One exception if node number =1 then the element nr. also equals 1. 5. main body name for reaction force 6. Node number on this main body
	mbdy moment_int	An internal force with a reaction component is placed on the structure. Unit is [Nm]. 1. main body name for action moment 2. node number on main body 3. component (1 = M_x, 2 = M_y, 3 = M_z), if negative number the moment is inserted with opposite sign. 4. coordinate system (possible options are: mbdy name,"global","local"). "local" means local element coo on the inner element (on the element indexed 1 lower that the node number). One exception if node number =1 then the element nr. also equals 1. 5. main body name for reaction moment 6. Node number on this main body
	constraint bearing2 angle_deg	The angle of a bearing2 constraint is set. The angle limits are so far [± 90 deg]. 1. Bearing name
	constraint bearing3 angle_deg	The angle of a bearing3 constraint is set. The angle limits are so far [± 90 deg]. 1. Bearing name
	constraint bearing3 omegas	The angular velocity of a bearing3 constraint is set. 1. Bearing name
	body printvar	Variable is just echoed on the screen. No parameters.
	body ignore	1. Number of consecutive array spaces that will be ignored
	mbdy printvar	Variable is just echoed on the screen. No parameters.
	mbdy ignore	1. Number of consecutive array spaces that will be ignored
	general printvar	Variable is just echoed on the screen. No parameters.
	general ignore	1. Number of consecutive array spaces that will be ignored
	stop_simulation	Logical switch. If value is 1 the simulation will be stopped and output written.
	wind printvar	Variable is just echoed on the screen. No parameters.
	wind windspeed_u	External contribution to wind speed in u-direction [m/s]
	wind winddir	External contribution to the wind direction (turb. box is also rotated) [deg]
	quake comp	1. Degree of freedom
	ext_sys control	1. Name of external system

10.8 hawc_dll format example written in FORTRAN 90

```
subroutine test(n1,array1,n2,array2)
implicit none
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS:'test'::test
integer*4           :: n1, &    ! Dummy integer value containing the array size of array1
                       n2       ! Dummy integer value containing the array size of array2
real*4,dimension(10) :: array1  ! fixed-length array, data from HAWC2 to DLL
                                ! - in this case with length 10
real*4,dimension(5)  :: array2  ! fixed-length array, data from DLL to HAWC2
                                ! - in this case with length 5

! Code is written here

end subroutine test

!-------------------------------------------------------

Subroutine test_init(string256)
Implicit none
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS:'test_init'::test_init
Character*256 :: string256

! Code is written here

End subroutine test_init

!-------------------------------------------------------

Subroutine test_message(string256)
Implicit none
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS:'test_message'::test_message
Character*256 :: string256

! Code is written here

End subroutine test_message
```

10.9 hawc_dll format example written in Delphi / Lazarus / Pascal

```
library test_dll;

type
  array_10 = array[1..10] of single;
  array_5  = array[1..5] of single;
  ts       = array[0..255] of char;

Procedure test(var n1:integer;var array1 : array_10;
               var n2:integer;var array2 : array_5);stdcall;
// n1 is a dummy integer value containing the size of array1
// n2 is a dummy integer value containing the size of array2
begin
  // Code is written here

end;

//-------------------------------------------------------------

Procedure test_init(var string256:ts; length:integer);stdcall;
var
  init_str:string[255]
begin
  init_str=strpas(string256);
  // Code is written here
  writeln(init_str);
end;

//-------------------------------------------------------------

Procedure test_message(var string256:ts; length:integer);stdcall;
var
  message_str:string;
begin
  // Code is written here
  message_str:='This is a test message';
  strPCopy(string256,message_str);
end;

exports test,test_init,test_message;

begin
  writeln('The DLL pitchservo.dll is loaded with succes');

  // Initialization of variables can be performed here
end;

end.
```

10.10 hawc_dll format example written in C

```
extern "C" void __declspec(dllexport) __cdecl test(int &size_of_Data_in,
float Data_in[], int &size_of_Data_out, float Data_out[])
{
 for (int i=0; i<size_of_Data_out; i++) Data_out[i]=0.0;
//
 printf("size_of_Data_in  %d: \n",size_of_Data_in);
 printf("Data_in          %g: \n",Data_in[0]);
 printf("size_of_Data_out %d: \n",size_of_Data_out);
 printf("Data_out         %g: \n",Data_out[0]);

}

extern "C" void __declspec(dllexport) __cdecl test_init(char* pString, int length)
{
// Define buffer (make room for NULL-char)
const int max_length = 256;
char buffer[max_length+1];
//
// Print the length of pString
printf("test_init::length = %d\n",length);
//
// Transfer string
int nchar = min(max_length, length);
memcpy(buffer, pString, nchar);
//
// Add NULL-char
buffer[nchar] = '\0';
//
// Print it...
printf("%s\n",buffer);
}

extern "C" void __declspec(dllexport) __cdecl test_message(char* pString, int max_length)
{
// test message (larger than max_length)
char pmessage[] = "This is a test message "
 "and it continues and it continues and it continues "
 "and it continues and it continues and it continues "
 "and it continues and it continues and it continues "
 "and it continues and it continues and it continues "
 "and it continues and it continues and it continues ";

// Check max length  -  transfer only up to max_length number of chars
int nchar = min((size_t)max_length, strlen(pmessage));  // nof chars to transfer
    // (<= max_length)
memcpy(pString, pmessage, nchar);
//
// Add NULL-char if string space allows it (FORTRAN interprets a NULL-char as
// the end of the string)
if (nchar < max_length) pString[nchar] = '\0';
}
```

10.11 type2_dll written in Delphi / Lazarus / Delphi

```
library hss_convert;

uses
  SysUtils,
  Classes,
  Dialogs;

Type
  array_1000 = array[0..999] of double;
Var
  factor : array of double;
  nr : integer;
{$R *.res}

procedure initialize(var InputSignals: array_1000;var OutputSignals: array_1000); cdecl;
var
  i : integer;
begin
  nr:=trunc(inputsignals[0]);
  if nr>0 then begin
    setlength(factor,nr);
    for i:=1 to nr do
      factor[i-1]:=Inputsignals[i];
    outputsignals[0]:=1.0;
  end else outputsignals[0]:=0.0;
end;

procedure update(var InputSignals: array_1000;var OutputSignals: array_1000); cdecl;
var
  i : integer;
begin
  for i:=0 to nr-1 do begin
    OutputSignals[i] := InputSignals[i]*factor[i];
  end;
end;

exports Initialize,Update;

begin
  // Main body

end.
```

10.12 type2_dll written in C

```
extern "C" void __declspec(dllexport) __cdecl initialize(dfloat *Data_in, dfloat *Data_out)
{ for (int i=0; i<8; i++) Data_out[0]+=Data_in[i];
}

extern "C" void __declspec(dllexport) __cdecl update(dfloat *Data_in, dfloat *Data_out)
{ for (int i=0; i<25; i++) Data_out[0]+=Data_in[i];
Data_out[8]=123;
}
```

10.13 type2_dll format example written in FORTRAN 90

```
subroutine initialize(array1,array2)
implicit none
!DEC$ ATTRIBUTES DLLEXPORT, C, ALIAS:'initialize'::initialize
real*8,dimension(1000) :: array1  ! fixed-length array, data from HAWC2 to DLL
                                  ! - in this case with length 1000
real*8,dimension(1)    :: array2  ! fixed-length array, data from DLL to HAWC2
                                  ! - in this case with length 1

! Code is written here

end subroutine initialize

!-------------------------------------------------------

subroutine update(array1,array2)
implicit none
!DEC$ ATTRIBUTES DLLEXPORT, C, ALIAS:'update'::update
real*8,dimension(1000) :: array1  ! fixed-length array, data from HAWC2 to DLL
                                  ! - in this case with length 1000
real*8,dimension(100)    :: array2  ! fixed-length array, data from DLL to HAWC2
                                  ! - in this case with length 100

! Code is written here

end subroutine update
```

11 Wind and Turbulence

11.1 Main command block -wind

Obl.	Command name	Explanation
*	wsp	1. Mean wind speed in center [m/s]
*	density	1. Density of the wind [kg/m3]
*	tint	Turbulence intensity [-].
*	horizontal_input	This command determines whether the commands above should be understood as defined in the global coordinate system (with horizontal axes) or the meteorological coordinates system (u,v,w) witch can be tilted etc. 1. (0=meteorological, 1=horizontal)
*	center_pos0	Global coordinates for the center start point of the turbulence box, meteorological coordinate system etc. (default should the hub center) 1. x_G [m] 2. y_G [m] 3. z_G [m]
*	windfield_rotations	Orientation of the wind field. The rotations of the field are performed as a series of 3 rotations in the order yaw, tilt and roll. When all angles are zero the flow direction is identical to the global y direction. 1. Wind yaw angle [deg], positive if the wind comes from the right side when sitting in the nacelle and looking upwind (i.e. in the -y_G direction). 2. Terrain slope angle [deg], positive when the wind comes from below. 3. Roll of wind field [deg], positive when the wind field is rotated according to the turbulence u-component.
*	shear_format	Definition of the mean wind shear 1. Shear type 0=none. !This option sets the mean wind speed to zero ! $\bar{u}(z) = 0$ 1=constant $\bar{u}(z)$ = wsp. The value is taken from the `wsp` parameter. 2=logarithmic $\bar{u}(z) = u_0 \frac{\log \frac{-z_0^G + z^M}{r_0}}{\log \frac{-z_0^G}{r_0}}$ 3=power law $\bar{u}(z) = u_0 \left(\frac{-z_0^G + z^M}{-z_0^G} \right)^{\alpha}$ 4=linear $\bar{u}(z) = u_0 \frac{\partial u}{\partial z}$ 2. Parameter used together with shear type (case of shear type: 0=dummy, 1=dummy, 2=r_0, 3= a, 4=d_u/d_z at center)
*	turb_format	1. Turbulence format (0=none, 1=mann, 2=flex)

Obl.	Command name	Explanation
*	tower_shadow_method	1. Tower shadow model (0=none, 1=potential flow – default, 2=jet model, 3=potential_2 (flow where shadow source is moved and rotated with tower coordinates system). Please see section, page 68 for sub block commands.
	scale_time_start	1. Starting time for turbulence scaling [s]. Stop time is determined by simulation length.
	wind_ramp_factor	Command that can be repeated as many times as needed. The wind_ramp_factor is used to calculate a factor that is multiplied to the wind speed vectors. Can be used to make troublefree cut-in situations. Linear interpolation is performed between t_0 and t_{stop}. 1. time start, t_0 2. time stop, t_{stop} 3. factor at t_0 4. factor at t_{stop}
	wind_ramp_abs	Command that can be repeated as many times as needed. The wind_ramp_abs is used to calculate a wind speed that is added to the wind speed u-component. Can be used to make wind steps etc. Linear interpolation is performed between t_0 and t_{stop}. 1. time start, t_0 2. time stop, t_{stop} 3. wind speed at t_0 4. wind speed at t_{stop}
	user_defined_shear	1. Filename incl. relative path to file containing user defined shear factors (example ./data/shear.dat)
	user_defined_- shear_turbulence	1. Filename incl. relative path to file containing user defined shear turbulence factors (example ./data/shearturb.dat)
	met_mast_wind	1. Filename incl. relative path to file containing time series of wind components in meteorological coordinates. The file should have four columns of data: time, v_u, v_v and v_w.
	iec_gust	Gust generator according to IEC 61400-1 1. Gust type 'eog' = extreme operating gust $u(z,t) = u(z,t) - 0.37A \sin\left(\frac{3\pi(t-t_0)}{T}\right)\left(1 - \cos\frac{2\pi(t-t_0)}{T}\right)$ 'edc' = extreme direction change $\theta(t) = 0.5\phi_0\left(1 - \cos\left(\frac{\pi(t-t_0)}{T}\right)\right)$ 'ecg' = extreme coherent gust $u(z,t) = u(z,t) + 0.5A\left(1 - \cos\left(\frac{\pi(t-t_0)}{T}\right)\right)$ 'ecd' = extreme coherent gust with dir. change $u(z,t) = u(z,t) + 0.5A\left(1 - \cos\left(\frac{\pi(t-t_0)}{T}\right)\right)$ $\theta(t) = 0.5\phi_0\left(1 - \cos\left(\frac{\pi(t-t_0)}{T}\right)\right)$

Obl.	Command name	Explanation
		'ews' = extreme wind shear $$vw_{res} = \sqrt{y_M^2 + z_M^2}$$ $$u(z,t) = u(z,t) + vw_{res}\, A \left(1 - \cos\left(\frac{2\pi(t-t_0)}{T}\right)\right)$$ $$* \cos\left(\text{atan2}\left(y^M, -z^M\right) - \phi_0\right)$$ even though the 'ews' expressions do not match the expressions in the standard completely, it gives identical results provided a mutual power law shear is used and the A parameter is set to $$A = \frac{2.5 + 0.2\beta\sigma_1\left(\frac{D}{\Lambda_1}\right)^{\frac{1}{4}}}{D}$$ and the parameter φ_0 is set to 0, 90, 180, 270 [deg] respectively. Note that: • Y_M and Z_M refer to the horizontal and vertical wind speeds respectively (expressed in meteorological coordinates, or V_M and W_M in figure **??**). • D refers to the rotor diameter. 2. Amplitude A [m/s]. For the 'eog', 'edc', 'ecd' this corresponds to the parameter 'V_{gust}', '0', 'V_{cg}' respectively, in the IEC61400-1 standard. 3. Angle φ_0 [deg] 4. Time start, t_0 [s] 5. Duration T [s]

11.2 Sub command block - mann

Block that must be included if the mann turbulence format is chosen. Normal practice is to use all three turbulence components (u,v,w) but only the specified components are used. In 2008 the turbulence generator was linked to the code so mannturbulence can be created without using external software. The command create_turb_parameters will search for turbulence files with names given below, but if these are not found the turbulence will be created.

A short explanation of the parameters L and $\alpha\varepsilon^{\frac{2}{3}}$ and its relation to the IEC61400-1 ed. 3 standard is given:

The fundamentals of the Mann model is isotropic turbulence in neutral atmospheric conditions. The energy spectrum is given based on the Von Karman spectrum (1). In isotropic turbulence, the properties of turbulence like variance and turbulent length scale is identical for all three direction corresponding to vortex structures being circular.

$$E(k) = \alpha\varepsilon^{\frac{2}{3}} L^{\frac{5}{3}} \frac{(Lk)^4}{\left(1 + (Lk)^2\right)^{\frac{17}{6}}} \tag{1}$$

The relation between wave number k and frequency f is related through the mean wind speed $\bar{U}$.

$$k = \frac{2\pi f}{\bar{U}} \tag{2}$$

However, atmospheric conditions are not isotropic and the vortex structures become more elliptic in shape with longer length scale and higher variance level in the u direction. In the

Mann model, this is accounted for using rapid distortion theory quantified through a shear blocking factor Γ. A Γ parameter of 0 corresponds to isotropic turbulence, whereas a higher Γ value is used for non-isotropic turbulence. The relation between non-isotropic and isotropic properties as function of Γ can be seen in Figure 5. For neutral atmospheric conditions (often referred to as "normal" conditions) it is recommended to use $\Gamma = 3.9$ in combination with a length scale of $L = 0.8\Lambda_1$. Λ_1 is defined as the wavelength where the longitudinal power spectral density is equal to 0.05. According to the IEC61400-1 the wavelength Λ_1 shall be considered as a constant of 42m above a height of 60m, or $0.7z$ otherwise (z being the height). In the Mann generation of turbulence a length scale L has to be used. This is the length scale of the Von Karman spectrum (1) and therefore different than the length scale used in the Kaimal formulation (3). The energy spectrum of Kaimal is formulated

$$E(f) = \sigma^2 \frac{4L/\bar{U}}{\left(1 + 6fL/\bar{U}\right)^{\frac{5}{3}}} \tag{3}$$

where the input parameters are given based on the table values in

	Velocity component index (k)		
	1	2	3
Standard deviation σ_k	σ_1	0,8 σ_1	0,5 σ_1
Integral scale, L_k	8,1 Λ_1	2,7 Λ_1	0,66 Λ_1

Figure 4: Information about Kaimal length scales and standard deviation ratio from the IEC61400-1

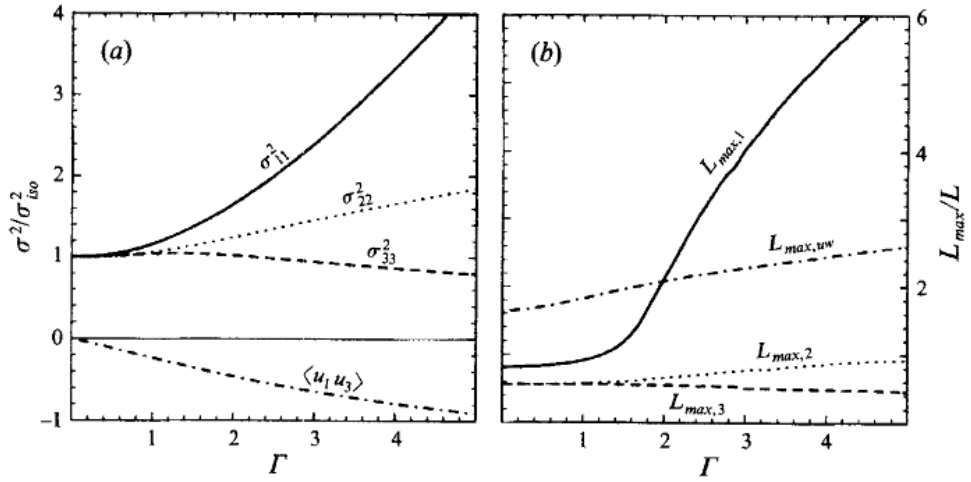


Figure 5: Turbulence characteristics compared to isotropic conditions as function of gamma parameter, Mann.. Left: Relation between variance is changed for higher shear distortions. Right: The relation between length scales are also changed for non-isotropic turbulence. It is recommended to use $\Gamma = 3.9$ for normal atmospheric conditions. This is also the requirement in the IEC61400-1 standard. Isotropic conditions are obtained using Γ=0.

The result of using $\Gamma = 3.9$ is that the structure of the turbulence corresponds to the normal atmospheric conditions, but the actual level of turbulence is also affected as seen in Figure 4. It is not straight forward to give the exact analytical relationship between the input parameter $\alpha\varepsilon^{\frac{2}{3}}$ and the final longitudinal variance and it is therefore very practical to introduce a turbulence scaling factor SF. This turbulence scaling factor is calculated based on the actual variance level in the box (normally extracted in the center of the box of longitudinal turbulence) and the target

variance σ^2_{target} based on the requested turbulence intensity $\sigma = Ti\ \bar{U}$. In this case of rescaling, which is the normal usage, the input value for $\alpha\varepsilon^{\frac{2}{3}}$ can be any arbitrary value except for zero.

$$SF = \sqrt{\frac{\sigma^2_{\text{target}}}{\sigma^2}} \tag{4}$$

The scale factor is to be multiplied to every values in the turbulence box for all the u,v and w directions. This is done automatically inside HAWC2.

Obl.	Command name	Explanation
	create_turb_parameters	With this command, the code will search for turbulence files with names given below, but if these are not found the turbulence will be created based on the given parameters. 1. Length scale L (L=33.6 according to the IEC standard at 42m and above) 2. $\alpha\varepsilon^{2/3}$ (when rescaling applied, 1.0 is normal practice) 3. γ (3.9 for neutral atmospheric conditions) 4. Seed number (any integer will do) 5. High frequency compensation (1=point velocity only represent local value which is closest to anemometer measurements, recommended in most cases, 0=point velocity represents average velocity in grid volume)
	filename_u	1. Filename incl. relative path to file containing mann turbulence u-component (example ./turb/mann-u.bin)
	filename_v	1. Filename incl. relative path to file containing mann turbulence v-component (example ./turb/mann-v.bin)
	filename_w	1. Filename incl. relative path to file containing mann turbulence w-component (example ./turb/mann-w.bin)
*	box_dim_u	1. Number of grid points in u-direction 2. Length between grid points in u-direction [m]
*	box_dim_v	1. Number of grid points in v-direction 2. Length between grid points in v-direction [m]
*	box_dim_w	1. Number of grid points in w-direction 2. Length between grid points in w-direction [m]
	std_scaling	Ratio between standard deviation for specified component related to turbulence intensity input specified in main wind command block. If the std_scaling command is omitted, the SF is determined based on the u-variance, the SF for v and w direction are kept equal to u-direction (recommended) 1. Ratio to u-direction (default=1.0) 2. Ratio to v-direction (default=0.8) 3. Ratio to w-direction (default=0.5)
	scaling_method	If the std_scaling command is used, this command specifies which method is used to scale the turbulent velocity components. If one of the dont_scale or factor_scaling command is used, this command is ignored.

Obl.	Command name	Explanation
		1. (1=scaling is based on a standard deviation of the Mann box by convecting a point along the x coordinate at the velocity u_mean at the y-z center of the box – default, 2=scaling is based on a standard deviation calculated using the entire Mann box)
	dont_scale	If this command is used the normal scaling to ensure the specified turbulence intensity is bypassed. 1. (0=scaling according to specified inputs – default, 1=raw turbulence field used without any scaling)
	factor_scaling	If this command is used constant, scaling factors are applied. 1. Scaling factor in u-direction, F_u 2. Scaling factor in v-direction, F_v 3. Scaling factor in w-direction, F_w

11.3 Sub command block - flex

Block that must be included if the flex turbulence format is chosen.

Obl.	Command name	Explanation
*	filename_u	1. Filename incl. relative path to file containing flex turbulence u-component (example ./turb/flex-u.int)
*	filename_v	1. Filename incl. relative path to file containing flex turbulence v-component (example ./turb/flex-v.int)
*	filename_w	1. Filename incl. relative path to file containing flex turbulence w-component (example ./turb/flex-w.int)
	std_scaling	Ratio between standard deviation for specified component related to turbulence intensity input specified in main wind command block. 1. Ratio to u-direction (default=1.0) 2. Ratio to v-direction (default=0.7) 3. Ratio to w-direction (default=0.5)

11.4 File description of a user defined shear

In this file a user defined shear used instead, or in combination with one of the default shear types (logarithmic, exponential...). When the user defined shear is used the name and location of the datafile must be specified with the *wind – user_defined_shear* command. This command specifies the location of the file and activates the user defined shear. If this shear is replacing the original default shear the command *wind – shear_format* must be set to zero!

Only one shear can be present in a single file. The shear describes the mean wind profile of the u, v and w component of a vertical cross section at the rotor. The wind speeds are normalized with the mean wind speed defined with the command *wind – wsp*.

Line number	Description
1	Headline (not used by HAWC2)
2	Information of shear v-component. #1 is the number of columns, NC #2 is the number of rows, NR
3	Headline (not used by HAWC2)

Line number	Description
4..+NR	Wind speed in v-direction, normalized with u-mean. # NC columns
1	Headline (not used by HAWC2)
+1..+NR	Wind speed in u-direction, normalized with u-mean. # NC columns.
1	Headline (not used by HAWC2)
+1..+NR	Wind speed in w-direction, normalized with u-mean. # NC columns
1	Headline (not used by HAWC2)
+1..+NC	Horizontal position of grid points (meteorological coo)
1	Headline (not used by HAWC2)
+1..+NR	Vertical position of grid points (meteorological coo)

11.5 Example of user defined shear file

```
# User defined shear file
3 4  # nr_v, nr_w     array sizes
# shear_v component, normalized with U_mean
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
# shear_u component, normalized with U_mean
1.0 1.0 1.0
1.0 1.0 1.0
1.0 1.0 1.0
1.0 1.0 1.0
# shear_w component, normalized with U_mean
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
# v coordinates
-50.0
0.0
50.0
# w coordinates (zero is at ground level)
0.0
60.0
100.0
200.0
```

11.6 File description of a user defined shear turbulence

The same file format is used as for *user_defined_shear* (see above). Instead of a normalized mean wind speed component, an additional turbulence scale factor is given by the user. The defined scale factors are applied on top of (multiplied with) the normal turbulence scaling coming from the turbulence model/box.

user_defined_shear_turbulence is an ad hoc and inconsistent scaling of a consistent turbulence field to obtain a non-homogeneous turbulence field, with the turbulence intensity varying with height. The HAWC2 developers do not recommend the use of this functionality, and users

should be aware that by using the *user_defined_shear_turbulence* the correlation properties between the various components in space of the generated turbulence box will no longer be valid as originally intended (for example when using the Mann turbulence model). This feature will allow users to easily alter turbulence boxes with the 'cost' it no longer holds a reasonable physical representation of a turbulent wind field.

11.7 Example of user defined shear turbulence file

```
# User defined shear turbulence file
3 4  # nr_v, nr_w     array sizes
# std_v component (to be multiplied with turbulence scaling)
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
# std_u component (to be multiplied with turbulence scaling)
1.0 1.0 1.0
1.0 1.0 1.0
1.0 1.0 1.0
1.0 1.0 1.0
# std_w component (to be multiplied with turbulence scaling)
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
# v coordinates
-50.0
0.0
50.0
# w coordinates (zero is at ground level)
0.0
60.0
100.0
200.0
```

11.8 Sub command block - wakes

Block that must be included if the Dynamic Wake Meandering model is used to model the wind flow from one or more upstream turbines. The model is described, calibrated and validated in [1, 2], where [2] contains both a recalibration and a validation against measurements. In order to make the model function, two Mann turbulence boxes must be used. One for the meandering turbulence – which is a box containing atmospheric turbulence, but generated with a course resolution in the v,w plane (grid size of 1 rotor diameter). It is important that the turbulence vectors at the individual grid points represent a mean value covering a grid cube. It is also important that the total size of the box is large enough to cover the different wake sources including their meandering path. The resolution in the u-direction should be as fine a possible. The used length scale should correspond to normal turbulence condition. The other turbulence box that is needed is a box representing the micro scale turbulence from the wake of the upstream turbine itself. The resolution of this box should be fine (e.g. 128x128 points) in the v,w plane which should only cover 1 rotor diameter. The resolution in the u direction should also be fine, but a short length of the box (e.g. 2.5Diameter) is OK, since the turbulence box is reused. The length scale for this turbulence is significantly shorter than for the other boxes since it represents turbulence from tip and root vortices mainly. A length scale of 1/16 rotor diameter

seems appropriate.

The two turbulence boxed are included by the following sub commands

```
begin mann_meanderturb;
    (parameters are identical to the normal Mann turbulence box, see above)
end mann_meanderturb;

begin mann_microturb;
    (parameters are identical to the normal Mann turbulence box, see above)
end mann_microturb;
```

The rest of the wake commands are given in the following table.

Obl.	Command name	Explanation
*	nsource	1. Number of wake sources. If 0 is used the wake module is by-passed (no source positions can be given in this case).
*	source_pos	Command that must be repeated nsource times. This gives the position of the wake source (hub position) in global coordinates. Wake source position given for down stream turbines are however not used in the simulations since they don't affect the target turbine. 1. x-pos [m] 2. y-pos [m] 3. z-pos [m]
*	op_data	Operational conditions for the wake sources. This command can be repeated nsource times to independently set the operation data of individual sources. If op_data appears once, the same operation data is used for all sources. 1. Rotational speed [rad/s] 2. Collective pitch angle [deg]. Defined positive according to the blade root coo, with z-axis from root towards tip. Note, this is opposite to the traditional notation for a pitch angle.
	ble_parameters	Parameters used for the BLE model used for developing the wake deficit due to turbulent mixing. 1. k_1 [-], default=0.10 2. k_2 [-], default=0.008 3. clean-up parameter (0=intermediate files are kept, 1=intermediate files are deleted), default=1
	microturb_factors	Parameters used for scaling the added wake turbulence according to the deficit depth and depth derivative. 1. k_{m1} [-], factor on deficit depth, default=0.60 2. k_{m2} [-], factor on depth derivative, default=0.25
	multiple_deficit_method	Command that is used for choosing the best approach for handling multiple deficit 1. method (1=MAX operator (default), 2=Direct summation) In general it is recommended to use the MAX operator when the ambient free wind speed is below rated and the direct summation approach above rated wind speed.
	tint_meander	Turbulence intensity of the meander turbulence box. If this command is not used then the default turbulence intensity from the general wind commands is used (normal use) 1. Turbulence intensity [-]

Obl.	Command name	Explanation
	use_specific_deficit_file	File with the deficits used in the correct downstream distance is used instead of the build in deficit generator. The wind speed deficits are non-dim with the mean wind speed. 1. Filename incl. path (e.g. ./data/deficit.data)
	write_ct_cq_file	File including the local axial and tangential forces (non-dim) as function of blade radius is written. 1. Filename incl. path (e.g. ./res/ct_cq.data)
	write_final_deficits	File with the deficits used in the correct downstream distance is written. The windspeed deficits are non-dim with the mean wind speed. 1. Filename incl. path (e.g. ./res/ct_cq.data)

11.9 File description of a user defined wake deficit file

When another flow solve has been used to find the non-dim turbulence deficit, eg. using an actuator disc approach, this can replace the deficit otherwise calculated internally. This method cannot be used together with multiple deficits as only one deficit can be read.

Line number	Description
1	#1 Any single character (eg. #) #2 The number of rows (NR) #3 (optional) The rotor diameter. If not included, the diameter of the reference turbine is used.
2..+NR	Deficit non-dim with ambient free mean wind speed. #1 Radius (non-dim with rotor radius) #2 Deficit (non-dim with free mean wind speed). In the free

11.10 Example of user defined wake deficit file

```
# 121 178.0
0.000000000E+00 8.276891200E-01
2.500000000E-02 8.486243600E-01
5.000000000E-02 8.809613720E-01
7.500000000E-02 9.007844070E-01
1.000000000E-01 8.957724550E-01
1.250000000E-01 8.660702830E-01
1.500000000E-01 8.303410890E-01
1.750000000E-01 8.044380440E-01
2.000000000E-01 7.895593800E-01
2.250000000E-01 7.786515560E-01
2.500000000E-01 7.691674220E-01
2.750000000E-01 7.618372330E-01
3.000000000E-01 7.572012850E-01
3.250000000E-01 7.550918200E-01
3.500000000E-01 7.542137030E-01
3.750000000E-01 7.518827010E-01
4.000000000E-01 7.456746090E-01
4.250000000E-01 7.357259740E-01
4.500000000E-01 7.250309980E-01
4.750000000E-01 7.168460970E-01
5.000000000E-01 7.119492260E-01
5.250000000E-01 7.088296670E-01
```

```
5.500000000E-01 7.057605130E-01
5.750000000E-01 7.021459650E-01
6.000000000E-01 6.983228280E-01
6.250000000E-01 6.947171830E-01
6.500000000E-01 6.913423360E-01
6.750000000E-01 6.879199230E-01
7.000000000E-01 6.842943230E-01
7.250000000E-01 6.806519720E-01
7.500000000E-01 6.773263690E-01
7.750000000E-01 6.744196220E-01
8.000000000E-01 6.716445590E-01
8.250000000E-01 6.684818930E-01
8.500000000E-01 6.644046880E-01
8.750000000E-01 6.592242170E-01
9.000000000E-01 6.529686490E-01
9.250000000E-01 6.445576730E-01
9.500000000E-01 6.324201240E-01
9.750000000E-01 6.173566910E-01
1.000000000E+00 5.982423590E-01
1.028634580E+00 5.679249380E-01
1.058116050E+00 5.982195030E-01
1.088469450E+00 7.292761710E-01
1.119720570E+00 9.095984580E-01
1.151895960E+00 1.014958390E+00
1.185022960E+00 1.022114240E+00
1.219129700E+00 1.017341600E+00
...
8.903031630E+00 1.000285950E+00
9.165402860E+00 1.000213540E+00
9.435533870E+00 1.000143160E+00
9.713654150E+00 1.000066170E+00
1.000000000E+01 1.000018010E+00
```

11.11 Sub command block – tower_shadow_potential

Block that must be included if the potential flow tower shadow model is chosen.

Obl.	Command name	Explanation
*	tower_offset	The tower shadow has its source at the global coordinate z axis. The offset is the base point for section 1 1. Offset value (default=0.0)
*	nsec	Command that needs to present before the radius commands. 1. Number of datasets specified by the radius command.
*	radius	Command that needs to be listed nsec times. 1. z coordinate [m] 2. Tower radius at z coordinate [m]

11.12 Sub command block – tower_shadow_jet

Block that must be included if the model based on the boundary layer equations for a jet is chosen. This model is especially suited for downwind simulations.

Obl.	Command name	Explanation
*	tower_offset	The tower shadow has its source at the global coordinate z axis. The offset is the base point for section 1 1. Offset value (default=0.0)
*	nsec	Command that needs to present before the radius commands. 1. Number of datasets specified be the radius command.
*	radius	Command that needs to be listed nsec times. 1. z coordinate [m] 2. Tower radius at z coordinate [m] 3. Cd drag coefficient of tower section (normally 1.0 for circular section, but this depends heavily on the reynold number)

11.13 Sub command block – tower_shadow_potential_2

Block that must be included if the tower shadow method 3 is chosen. This potential model is principally similar to the potential flow model described previously but differs in the way that the shadow source is moved and rotated in space as the tower coordinate system is moving and rotating. It is also possible to define several tower sources e.g. if the tower is a kind of tripod or quattropod. Just include more tower_shadow_potential_2 blocks if more sources are required.

The coordinate system that the shadow method is linked to is specified by the user, e.g. the mbdy coordinate from the tower main body. To make sure that the tower source model is always linked in the same way as the tower (could be tricky since the tower is fully free to be specified along the x,y or z axis or a combination) the base coordinate system for the shadow model is identical to the coordinates system obtained by the local element coordinates, where the z axis is always pointing from node 1 towards node 2. This is the reason that the tower radius input has to specified with positive z-values, see below.

Obl.	Command name	Explanation
*	tower_mbdy_link	Name of the main body to which the shadow source is linked. 1. mbdy name
*	nsec	Command that needs to present before the radius commands. 1. Number of datasets specified by the radius command.
*	radius	Command that needs to be listed nsec times. 1. z coordinate [m] (allways positive!) 2. Tower radius at z coordinate [m]

11.14 Sub command block – tower_shadow_jet_2

Block that must be included if the tower shadow method 4 is chosen. This jet model is principally similar to the jet model described previously but differs in the way that the shadow source is moved and rotated in space as the tower coordinate system is moving and rotating. It is also possible to define several tower sources e.g. if the tower is a kind of tripod or quattropod. Just include more tower_shadow_jet_2 blocks if more sources are required.

The coordinate system that the shadow method is linked to is specified by the user, e.g. the mbdy coordinate from the tower main body. To make sure that the tower source model is always linked in the same way as the tower (could be tricky since the tower is fully free to be specified along the x,y or z axis or a combination) the base coordinate system for the shadow model is identical to the coordinates system obtained by the local element coordinates, where the z axis is always pointing from node 1 towards node 2. This is the reason that the tower radius input has to specified with positive z-values, see below.

Obl.	Command name	Explanation
*	tower_mbdy_link	Name of the main body to which the shadow source is linked. 1. mbdy name
*	nsec	Command that needs to present before the radius commands. 1. Number of datasets specified by the radius command.
*	radius	Command that needs to be listed nsec times. 1. z coordinate [m] 2. Tower radius at z coordinate [m] 3. C_d drag coefficient of tower section (normally 1.0 for circular section, but this depends heavily on the reynold number)

11.15 Sub command block – user_wind_dll

A user defined DLL can be used to provide additional wind velocity on top of what is already defined by wind input in HAWC2. During simulation, HAWC2 calls the DLL with position as argument, and the DLL must provide the wind velocity in that position on return. Apart from the position, HAWC2 also parses time and user-specified arguments to the DLL - the user-specified arguments are defined in the same output block format as is used for type2_dlls and hawc_dlls and as regular output. See Section B for further details.

Obl.	Command name	Explanation
*	filename	Path and name of DLL.
	dll	deprecated alternative to filename.
*	subroutine	Subroutine name to call in DLL.
	refsys	Reference coordinates for position (in) and velocity (in/out). 0. meteorological coordinates (default) 1. global coordinates
	begin output; <output block> end output;	Output block definition which can be used to provide additional user-specified input to the DLL, see example in Section B . Note that the only output types that can be used are: - `general`, - `dll`, - `constraint`, and - `mbdy`.

11.16 Sub command block – turb_export

With this sub command block, a mann format turbulence box including information from shear, wakes, tower shadow etc. is written. Same data point positions are used as specified in the turbulence module including the parameters specified for the originally used mann turbulence box.

Obl.	Command name	Explanation
*	filename_u	Filename of turbulence box with axial turbulence 1. File name
*	filename_v	Filename of turbulence box with lateral turbulence 1. File name
*	filename_w	Filename of turbulence box with vertical turbulence 1. File name
	samplefrq	1. Sample frequency
	time_start	1. Time at which the the turbulence recording will start

Obl.	Command name	Explanation
	nsteps	1. Number of steps between output
	box_dim_v	1. Number of points in v-direction 2. Distance between points in v-direction
	box_dim_w	1. Number of points in w-direction 2. Distance between points in w-direction

11.17 How the wind speed is constructed

The wind speed is finally constructed based on the following user inputs (and in the meteorological coordinate system:

```
wsp = action_windspeed_u + gust
   + (wsp_mean*wind_ramp_factor+wind_ramp_abs)*shear_factor
   + (wsp_mean*wind_ramp_factor+wind_ramp_abs)*user_defined_shear
   + met_mast_wind + dwm_deficit_u*wind_ramp_factor
   + dwm_turb*wind_ramp_factor
   + turb * scaling * user_defined_shear_turbulence * wind_ramp_factor
   + user_wind_dll velocity
```

The above commands are explained in more detail in the sections above. Some additional clarifications are as follows:

- `action_windspeed_u` corresponds to the DLL action command `wind windspeed_u`.
- `wsp_mean` is the mean wind speed as set by the `wsp` command.
- `shear_factor` is the determined by the shear type as set by the `shear_format` command.
- `scaling` is affected by the commands `std_scaling`, `dont_scale`, and/or `factor_scaling`. See also the description in the Mann section above.
- `dwm_deficit_u` is the velocity deficit in the wake as given by the Dynamic Wake Meandering model (DWM).
- `dwm_turb` is the added turbulence due to the wake as given by the DWM model.

After transforming to the global coordinate system, the tower shadow deficit is added as follows:

```
wspG = wspG*tower_shadow_factor
```

12 Aerodynamics

12.1 Main command block - aero

This module set up parameters for the aerodynamic specification of the rotor. It is also possible to submit aerodynamic forces to other structures as example the tower or nacelle, but see chapter (Aerodrag) regarding this. The module can be added as many times as requested if multiple aerodynamic rotors are needed.

Obl.	Command name	Explanation
(*)	name	Name of rotor (in case of multiple rotors defined this is obligatory.)
*	nblades	Must be the first line in aero commands! 1. Number of blades
*	hub_vec	Link to main-body vector that points downwind from the rotor under normal conditions. This corresponds to the direction from the pressure side of the rotor towards the suction side where the coordinate system is normally taken from the main shaft system.. 1. mbdy name or 'old_input' if old_htc_structure format is applied. 2. mbdy coo. component (1=x, 2=y, 3=z). If negative the opposite direction used. Not used together with old_htc_structure input (specify a dummy number). 3. Node number (optional). Node number on mbdy where rotor center is located. 'last' can also be used (default if no value is present).
*	link	Linker between structural blades and aerodynamic blades. There must be same number of link commands as nblades! 1. blade number 2. link chooser – options are - mbdy_c2_def (used with new structure format) - blade_c2_def (used with old structure format, see description below in this chapter) 3. mbdy name (with new structure format), not used to anything with old structure format.
*	ae_filename	1. Filename incl. relative path to file containing aerodynamic layout data (example ./data/hawc2_ae.dat)
*	pc_filename	1. Filename incl. relative path to file containing profile coefficients (example ./data/hawc2_pc.dat)
*	induction_method	1. Choice between which induction method that shall be used (0=none, 1=normal BEM dynamic induction, 2= Near Wake induction method, 3= VAWT)
	only_update_r_mono_incr	1. Should the induction model be updated if the blade radius doesn't monotonically increase towards the tip? Then some assumptions in the aerodynamic induction models are no longer valid and the crashes may occur. (0=always update, 1=only update if the radius is increasing monotonically, otherwise keep values from last time step)(default=0)
*	aerocalc_method	1. Choice between which aerodynamic load calculation method that shall be used. (0=none, 1=normal)
	aerosections	Number of aerodynamic calculation points at a blade. The distribution is performed automatically using a cosine transformation which gives closest spacing at root and tip. 1. Number of points at each blade.

Obl.	Command name	Explanation
	aero_distribution	1. Distribution method of aerodynamic calculation points. Options are: - “default” number. The distribution is performed automatically using npoints position with a cosine transformation which gives closest spacing at root and tip. - “ae_file” set. The distribution is given with same spacing as values in the ae_file with set number set..
*	ae_sets	Set number from ae_filename that is linked to blade 1,2,...,nblades 1. set for blade number 1 2. set for blade number 2 . . . nblades. set for blade number nblades
*	tiploss_method	1. Choice between which tip-loss model that shall be used (0=none, 1=prandtl (default))
*	dynstall_method	1. Choice between which dynamic stall model that shall be used (0=none, 1=Stig Øye method, 2=MHH Beddoes method, 3=Gaunaa-Andersen method with Deformable Trailing Edge Flap’s)
	3d_correct_method	Airfoil Cl values from the pc_file is modified for 3D effects. 1. Correction method (1=Snel method for correction of Cl values)
	external_bladedata_dll	Blade structural data are found in an external encrypted dll. If this command is present the following command lines shall not be present (ae_filename, pc_filename and ae_sets). 1. Company name (that has been granted a password, eg. dtu). 2. Password for opening this specific dll, eg. test1234 3. path and filename for the dll. eg. ./data/encr_blade_data.dll
	output_profile_coef_filename	Interpolated profile coefficients at all aerodynamic calculation points are written into a data file. This command can not be used in combination with encrypted_profile_coef_filename. 1. path and filename for the dll. eg. ./res/aero_profiles.dat

12.2 Sub command block – dynstall_so

Block that may be included if the Stig Øye dynamic stall method is chosen. If not included defaults parameters are automatically used.

Obl.	Command name	Explanation
	dclda	1. Linear slope coefficient for unseparated flow (default=6.28)
	dcldas	1. Linear slope coefficient for fully separated flow (default=3.14)
	alfs	1. Angle of attack [deg] where profile flow is fully separated. (default=40)
	alrund	1. Factor used to generate synthetic separated flow Cl values (default=40)
	taufak	1. Time constant factor in first order filter for F function (default=10.0). Internally used as tau=taufak*chord*vrel

12.3 Sub command block – dynstall_mhh

This Block may be included if the MHH Beddoes dynamic stall method is chosen [3, 4], otherwise default values are used. The MHH model is the recommended model for a turbine without trailing edge flaps. It consists of an attached flow part that covers the Theodorsen effect as well as torsion rate terms and added mass terms, as well as a dynamic stall part that simulates trailing edge stall.

Due to the torsion rate and added mass terms, the lift coefficients predicted by the MHH Beddoes dynamic stall model can reach very high values. The torsion rate lift coefficient is $c_{l,tors} = \pi T_0 \dot{\theta}$ (Eq (5) in [4]), and the added mass lift coefficient $c_{l,acc} = -\pi T_0 \frac{\ddot{y}}{U}$ (Eq (13) in [4]). The term T_0 in these equations is $T_0 = c/(2U)$ with the chord c and the relative velocity U; $\dot{\theta}$ is the rate of rotation of the airfoil and $\ddot{y}$ is the acceleration of the airfoil perpendicular to the chord. The lift coefficient from Eq (5) has a relative velocity in the denominator, the lift coefficient from Eq (13) a relative velocity squared. Both can reach very large values if the relative velocity is close to zero. However because they are multiplied by the relative velocity squared to compute the lift force, these large values will not result in large lift forces. Thus if the code predicts very large lift coefficients, the relative velocity and, most importantly, the lift forces should be investigated. If the lift forces are reasonable, then it is safe to assume that the large lift coefficient is not problematic but instead correctly modeling the aerodynamic forces due to torsion rate or added mass.

Obl.	Command name	Explanation
	a1	1. Coefficients of the exponential potential flow step response approximation: Phi(s)=1-A1*exp(-b1*s)-A2*exp(-b2*s). (default= 0.165)
	a2	1. Coefficients of the exponential potential flow step response approximation: Phi(s)=1-A1*exp(-b1*s)-A2*exp(-b2*s). (default= 0.335)
	b1	1. Coefficients of the exponential potential flow step response approximation: Phi(s)=1-A1*exp(-b1*s)-A2*exp(-b2*s). (default= 0.0455)
	b2	1. Coefficients of the exponential potential flow step response approximation: Phi(s)=1-A1*exp(-b1*s)-A2*exp(-b2*s). (default= b2=0.300)
	update	Choice between update methods: 1. 1 (default)=>update aerodynamics all iterations all timesteps; 0=>only update aerodynamics first iteration each new timestep
	taupre	1. Non-dimensional time-lag parameters modeling pressure time-lag. Default value =1.5
	taubly	1. Non-dimensional time-lag parameters modeling boundary layer time-lag. Default value=6.0
	only_potential_model	1. 0(default)=>run full MHH beddoes model; 1=>Potential flow model dynamics superposed to steady force coefficients;
	max_cl_attached	1. Maximum value of lift coefficient for attached flow.

12.4 Sub command block – dynstall_ateflap

This sub-block should be included if the ATEFlap dynamic stall model is chosen (dynstall_method number 3). The dynamic stall model is similar to the MHH model, expanded to account for steady and dynamic effects of trailing edge flap deflections; the model is described in L. Bergami and M. Gaunaa, *ATEFlap Aerodynamic model, a dynamic stall model including the effects of trailing edge flap deflection* (Risoe-R-1792(EN), Risoe DTU, February 2012). The model requires a .ds input file containing pre-processed steady aerodynamic data for the blade

sections containing a flap (see following paragraphs for the file specifications). Sections without any flap are attributed steady input data according to the aerodynamic layout specified in the ae_filename.

Obl.	Command name	Explanation
*	flap	Mandatory command to define a flap section. The flap is defined on all the blades of the rotor. Command syntax: 1. Radius r_start [in m]. Starting point of flap section. 2. Radius r_end [in m]. Ending point of flap section (should be > r_start). 3. Filename incl. relative path to .ds file containing pre-processed aerodynamic steady input data. See .ds file specifications in the following paragraph. N.B. The location along the blade refer to the 'stretched' blade, distances are given along the half-chord line (as the layout in ae_file). A maximum of 99 flap sections can be defined.
	ais	Coefficients for the indicial response exponential function: 1. A1 (default= 0.1784) 2. A2 (default=0.07549) 3. A3 (default=0.3933) Default coefficients describe the step response of a NACA 64-418 profile, where t/c=0.18.
	bis	Coefficients of the exponential potential flow step response approximation: 1. B1 (default= 0.8000) 2. B2 (default= 0.01815) 3. B3 (default= 0.1390) Default coefficients describe the step response of a NACA 64-418 profile, where t/c=0.18.
	taupre	1. Non-dimensional time-lag parameters modelling pressure time-lag. Default value =1.5
	taubly	1. Non-dimensional time-lag parameters modelling boundary layer time-lag. Default value=6.0
	only_potential_model	1. 0(default)=>run full ATEFlap model; 1=>Potential flow model dynamics superposed to steady force coefficients;
	update	Choice between update methods: 1. 1 (default)=>update aerodynamics all iterations all timesteps; 0=>only update aerodynamics first iteration each new timestep
	hystar	1. Camberline coef. (default= -4.675844E-003)
	fylestar	1. Camberline coef. (default= +4.155446E-004)
	fdydxle	1. Camberline coef. (default= +7.236104E-003)
	gdydxle	1. Camberline coef. (default= +3.309147E-003)

The camber line coefficients describe the camber line deformation shape induced by the flap; they are computed according to the thin-airfoil model described in Gaunaa's Wind Energy journal article *Unsteady two-dimensional potential-flow model for thin variable geometry airfoils*. Hystar and fylestar are dimensionless parameters corresponding to the shape integrals Hy and FyLE normalized by the half-chord length. The default coefficients refer to a 10% chord length flap with a continuous deformation shape, describing a circular arc, whose chord forms an angle of 1 degree with the horizontal axis.

12.5 Sub command block – aero_noise

If this command block is used, aero-acoustic calculations are performed. The blade is discretized spanwise into elementary blade sections corresponding to the aerodynamic calculation points of the main command block – aero, i.e. as defined by the command 'aerosections'. Aerodynamic noise is calculated for each of these blade sections and subsequently added at the observer location(s) assuming incoherent noise sources. Only geometrical spreading is considered for the noise propagation between blade sections and observer. Details of the implementation for the turbulent inflow, trailing edge and stall noise models can be found in Bertagnolio et al, *A combined aeroelastic-aeroacoustic model for wind turbine noise: verification and analysis of field measurements*, Wind Energy (20), 2017. As for the loading-thickness noise model, the implementation is described in Bertagnolio et al, *A temporal wind turbine model for low-frequency noise*, InterNoise (Conf. Proc.), 2017.

Obl.	Command name	Explanation
	noise_mode	1. Noise mode (0=no noise calculation, 1=compute noise at each time-step on the fly, 2=store aerodynamic data for later noise calculation as post-processing (using option 3 or 4), 3=compute noise at each time-step using stored data, 4=compute steady-state noise using stored data and rotor disk azimuthal sector averaging yielding large time-saving) (default=0)
	noise_start_end_time	Start and end time for noise computation. 1. Start time, t_0 [s] 2. End time, t_1 [s] (default: at all time)
	noise_deltat	1. Time-step for noise calculation (default: at each HAWC2 time-step)
	noise_azimuth_sectors	1. Number of rotor disk azimuthal sectors when running noise_mode=4 (default=16)
	atmospheric_pressure	1. Atmospheric pressure [Pa] (default=101325.)
	temperature	1. Temperature [deg. Celsius] (default=20.)
	octave_bandwidth	1. Octave band frequency centers used for defining noise spectra. Options are: 1, 3, 12 and 24 (default=3)
	spl_min_max_frq	Minimum and maximum computed frequency for integrated sound pressure level calculations. 1. Minimum frequency, fr_{min} [Hz] 2. Maximum frequency, fr_{max} [Hz] (default: all octave band frequency centers are used)
	turbulent_inflow_noise	1. Turbulent inflow noise model (0=using Von Karman turbulence spectra, 1=using Mann atmospheric turbulence model) (default=0)
	turbulent_inflow- _thickness_correction	1. Turbulent inflow thickness correction (0=none, 1=correction is added to turbulent inflow noise) (default=0)
	mann_turbulence_parameters	Mann turbulence parameters. 1. L: turbulent integral length (default=29.7m) 2. $\alpha\varepsilon^{2/3}$: energy level (default=1.0) 3. γ: anisotropy factor (default=3.7) If any value is negative, then its default value is assumed.
	surface_roughness	1. Surface roughness, z0 (If specified, it is used to re-define the Mann turbulence parameters)
	trailing_edge_noise	1. Trailing edge model (0=none, 5=TNO 'frba' model, 31=Amiet 'frba' model, 41=Amiet 'asfi' model) (default=0)
*	bldata_filename	1. Filename incl. relative path defining tabulated input data for trailing edge noise model.
	trailing_edge_serration	Trailing edge serration model parameters.

Obl.	Command name	Explanation
		1. R_1 Inboard radius [m] 2. R_2 Outboard radius [m] 3. L_{ser} Serration periodic span length [m] 3. H_{ser} Serration crest to trough height [m]
	stall_noise	1. Stall noise model (0=none, 1=Amiet based model, 2=Full formulation) (default=0)
	stall_separation	Stall separation definition. 1. Stall separation (1=tabulated and given in bldata_filename, 2=use dynamic stall model, 3=forced separation location) (default=1) 2. Forced separation location (x/C[-]: if positive on suction side, if negative on pressure side)
	tip_noise	1. Tip noise model (0=none, 1=not implemented yet!!!) (default=0)
	loading_noise	1. Loading-thickness noise model (0=none, 1=based on tabulated Cl, 2=based on Cp distribution from tabulated data, 3=based on Cl from HAWC2 aerodynamics) (default=0) This model does not work with noise_mode=4.
	loading_data_filename	1. Filename incl. relative path defining tabulated input data for loading-thickness noise.
*	xyz_observer	Position of observer in global reference system. 1. x [m] 2. y [m] 3. z [m] More than one observer is allowed (but must be <256).
	output_filename	1. Filename incl. relative path for output log file.

12.6 Sub command block – bemwake_method

Dynamic inflow settings used to calculate the dynamic induction. If not included defaults parameters are automatically used.

Obl.	Command name	Explanation
	nazi	1. Number of azimuthal points in the induction grid. A high number increased accuracy but slow down the simulation time. Default is 16.
	fw	Dynamic time constants and mixing ratio contribution for the far wake part of the induction. 1. Mixing ratio, default is 0.4 2. k_3 (poly. coef. for r/R sensitivity) default=0.0 3. k_2 (poly. coef. for r/R sensitivity) default=-0.4751 4. k_1 (poly. coef. for r/R sensitivity) default=0.4101 5. k_0 (poly. coef. for r/R sensitivity) default=1.921
	nw	Dynamic time constants and mixing ratio contribution for the near wake part of the induction. 1. Mixing ratio, default is 0.6 2. k_3 (poly. coef. for r/R sensitivity) default=0.0 3. k_2 (poly. coef. for r/R sensitivity) default=-0.4783 4. k_1 (poly. coef. for r/R sensitivity) default=0.1025 5. k_0 (poly. coef. for r/R sensitivity) default=0.6125
	a-ct-filename	Filename for a user defined relation bewteen a and ct.
	custom_tiploss	Filename for a user defined tip/root loss factor.

Obl.	Command name	Explanation
		Filestructure: One number in the first line gives the number of radial stations specified in the file. Following lines have two numbers: non-dimensional radius $0 < r/R < 1$ followed by tip/root loss factor $0 < F_{custom} < 1$. It is applied on the $a = f(CT)$ relation as $a = f(C_T/(FF_{custom}))$ where F is the regular tip loss factor. This allows e.g. implementation of a user defined root loss model by specifying F_{custom} going from 0 at the root towards 1 at or before the tip. In that way F_{custom} and the regular tiploss factor F can be used together.
	radial_induc	1. Radial induction model (0=none , 1=Radial induction model described in Section 2.8 of [5])(default=0)

12.7 Sub command block – nearwake_method

The near wake model implementation in HAWC2 couples the lifting line theory based near wake model for trailed vorticity with the modified HAWC2 BEM as a far wake model. Inherently included in the trailed vorticity computations are the influences of the tip and root vortices; a 'root-loss' model is otherwise not included in HAWC2. The model is described in [6, 7] and has been shown to improve the dynamic blade loading in the presence of turbulence, blade vibrations and flap actuations.

In case of strong load gradients on the blade due to for example flaps at fixed angle or other aerodynamic devices activating the near wake model leads to an improved steady state load distribution. When used in this case with a prescribed point distribution along the blade (defined in the ae-file) then sudden changes in the point density (for example close to the flap) should be avoided as they can lead to numerical instability of the model. As with any vortex model, care should be taken when operating in deep stall conditions, such as extreme yaw conditions in standstill.

Obl.	Command name	Explanation
	only_one_nw_function	Dynamic accuracy, see Section 6 in [7] for details. (0=2 exponential functions used; 1=default, 1 exponential function used: minimally lower accuracy but almost twice as fast)
	only_axial_nw	(0=default, near wake model used for both axial and tangential induction; 1= near wake model used for axial induction only)
	fast_nwm	(0=full iteration loop of the near wake model; 1= default, helix angle and vortex filament length fixed during iteration loop, almost identical results, much faster)
	fixed_kfw	kfw (0<kfw<1). This coupling factor will be used and is fixed during the computation. Not using this command means the coupling factor will be computed automatically and dynamically updated each time step (default, see Section 5 in [6] for details)
	r_core	r_core determines the vortex core radius (default=0: no vortex core is used). The implementation is in beta version and not validated.

12.8 Sub command block – vawtwake_method

VAWT wake method parameters.

Obl.	Command name	Explanation
	nazi	1. Number of azimuthal points in the induction grid. A high number increased accuracy but slow down the simulation time.
	fw	Dynamic time constants and mixing ratio contribution for the far wake part of the induction. 1. Mixing ratio, default is 0.4 2. k_3 (poly. coef. for r/R sensitivity) default=0.0 3. k_2 (poly. coef. for r/R sensitivity) default=-0.4751 4. k_1 (poly. coef. for r/R sensitivity) default=0.4101 5. k_0 (poly. coef. for r/R sensitivity) default=1.921
	nw	Dynamic time constants and mixing ratio contribution for the near wake part of the induction. 1. Mixing ratio, default is 0.6 2. k_3 (poly. coef. for r/R sensitivity) default=0.0 3. k_2 (poly. coef. for r/R sensitivity) default=-0.4783 4. k_1 (poly. coef. for r/R sensitivity) default=0.1025 5. k_0 (poly. coef. for r/R sensitivity) default=0.6125

12.9 Data format for the aerodynamic layout

The format of this file which in the old HAWC code was known as the hawc_ae file is changed slightly for the HAWC2 input format. The position of the aerodynamic center is no longer an input value, since the definition is that the center is located in $C_{1/4}$ with calculated velocities in $C_{3/4}$.

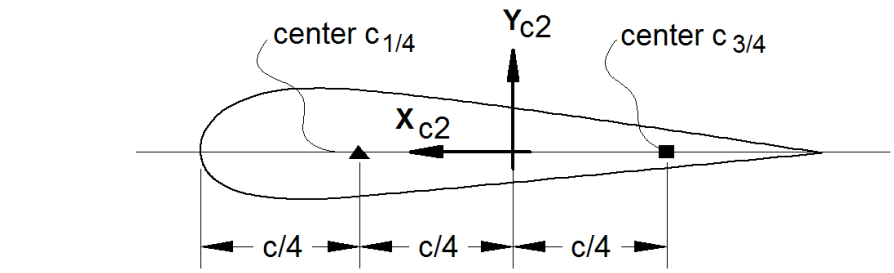


Figure 6: Illustration of aerodynamic centers $C_{1/4}$ and $C_{3/4}$

The format of the file is specified in the following two tables

Line number	Description
1	#1: Nset, Number of datasets present in the file. The format of each data set can be read below. The datasets are repeated without blank lines etc.
2	#1: Set number. #2: Nrows, Number of data rows for this set
3..2+Nrows	Data row according to Table 5

Table 27: Format of main data structure for the aerodynamic "_ae" blade layout file

The content of the colums in a data row is specified in the table below.

Column	Parameter
1	r, curved length distance from main_body node 1 [m]
2	chord length [m]
3	thickness ratio between profile height and chord [%]
4	Profile coefficient set number
(5)	Optional column. When present, it includes a dynamic stall model selector. It is then possible to bypass or change dynamic stall model for different part of the blade. Numbers are identical to the one used in the command "aero dynstall_method"

Table 28: Format of the data rows for the aerodynamic "_ae" blade layout file

12.10 Example of an aerodynamic blade layout file

```
1 Number of datasets in the file.
1 25  Set nr, nrows.
0 2.42 100 1  Radius [m] chord[m] thick[%] PC [-]
1.239 2.42 100 1
1.24 2.42 99.9 1
3.12 2.48 96.4 1
5.24 2.65 80.5 1
7.24 2.81 65.0 1
9.24 2.98 51.6 1
11.24 3.14 40.3 1
13.24 3.17 32.5 1
15.24 2.99 28.4 1
```

```
17.24 2.79 25.6 1
19.24 2.58 23.7 1
20.44 2.46 22.8 1
23.24 2.21 20.9 1
25.24 2.06 20.0 1
27.24 1.92 19.4 1
29.24 1.8 19.0 1
31.24 1.68 18.7 1
33.24 1.55 18.6 1
35.24 1.41 18.3 1
37.24 1.18 17.9 1
38.24 0.98 17.3 1
39.24 0.62 16.3 1
39.64 0.48 15.7 1
40.00 0.07 14.8 1
```

12.11 Data format for the profile coefficients file

The format of this file which in the old HAWC code was known as the hawc_pc file has not been changed for the HAWC2 code.

The format of the file is specified in the following two tables

Line number	Description
1	#1: Nset, Number of datasets present in the file. The format of each data set can be read below. The datasets are repeated without blank lines etc.
2	#1: Nprofiles. Number of profiles included in the data set. There must be more than 1 Nprofiles. First profile is the thinnest, last profile is the thickest (continously increasing order).
3	#1: Profile number. #2: Nrows. #3: Thickness in percent of chord length
4..3+Nrows	Data row according to Table 7

Table 29: Format of main data structure for the profile coefficients file

The content of the colums in a data row is specified in table below.

Column	Parameter
1	α, angle of attack [deg]. Starting with -180.0, ending with +180.0
2	C_l lift coefficient [-]
3	C_d drag coefficient [-]
4	Cm moment coefficient [-]

Table 30: Format of the data rows for the profile coefficients file

12.12 Example of the profile coefficients file "_pc file"

```
1 Airfoil data for the nrel 5 mw turbine
8
1   127  17 DU17 airfoil with an aspect ratio of 17.  Original -180 to 180deg
-180.00    0.000   0.0198   0.0000
```

```
-175.00    0.374   0.0341   0.1880
-170.00    0.749   0.0955   0.3770
-160.00    0.659   0.2807   0.2747
-155.00    0.736   0.3919   0.3130
-150.00    0.783   0.5086   0.3428
-145.00    0.803   0.6267   0.3654
-140.00    0.798   0.7427   0.3820
-135.00    0.771   0.8537   0.3935
-130.00    0.724   0.9574   0.4007
-125.00    0.660   1.0519   0.4042
-120.00    0.581   1.1355   0.4047
-115.00    0.491   1.2070   0.4025
-110.00    0.390   1.2656   0.3981
-105.00    0.282   1.3104   0.3918
-100.00    0.169   1.3410   0.3838
 -95.00    0.052   1.3572   0.3743
 -90.00   -0.067   1.3587   0.3636
 -85.00   -0.184   1.3456   0.3517
 -80.00   -0.299   1.3181   0.3388
 -75.00   -0.409   1.2765   0.3248
 -70.00   -0.512   1.2212   0.3099
 -65.00   -0.606   1.1532   0.2940
 -60.00   -0.689   1.0731   0.2772
 -55.00   -0.759   0.9822   0.2595
```

12.13 Data format for the flap steady aerodynamic input (.ds file)

This file contains the pre-processed steady data required by the ATEFlap dynamic stall model. Steady lift, drag and moment coefficients are given as function of angle of attack and flap deflection, together with the fully separated and fully attached lift, and the separation function values required by the Beddoes-Leishmann dynamic stall model. The input file can be generated automatically through an external pre-processing application, as for instance the "Preprocessor for ATEFlap Dynamic Stall Model, v.2.04". Please refer to the application documentation for further details.

The format of the file is specified in the following two tables:

Line number	Description
1	Free for comments
2	Free for comments
3	#1: Aoa0 [rad]. Angle of attack returning a null steady lift
4	Free for comments
5	#1: dCl/dAoa [1/rad]. Gradient of the steady lift function with respect to angle of attack variations
6	Free for comments
7	#1: dCl/dBeta [-]. Gradient of the steady lift function with respect to flap deflection variations
8	Free for comments
9	#1: Nrows. Total number of the following data-rows.
10...9+Nrows	Data rows, as specified in following table.

Table 31: Format of main data structure for the .ds flap steady aerodynamic input file

The content of the columns in a data row is specified in table below.

Column	Parameter
1	α, Angle Of Attack [deg]. Starting with -180.0, ending with +180.0. External loop (changes value after going through all the beta flap deflection values, i.e. 100 rows)
2	Beta, flap deflection. Starting from -49 to +50. Internal loop (changes at every data row)
3	C_l st. Steady lift coefficient [-]
4	C_l att. Fully attached lift coefficient [-]
5	C_l fs. Fully separated lift coefficient [-]
6	C_d drag coefficient [-]
7	C_m moment coefficient [-]
8	f . Steady value of the separation function [-]

Table 32: Format of the data rows for the .ds flap steady aerodynamic input file

12.14 Example of a .ds flap steady aerodynamic input file

```
Input file for Flap dyn.stall model. Generated with Delphi preprocessor
.Linear Region: Aoa Cl0 [rad]:
-0.06523855
.Linear Region: dCl / dAoa [1/rad]:
6.60081861
.Linear Region: dCl / dBeta [1/deg]:
0.0435375
   . Polars: 1.Aoa | 2.Beta | 3.Clst | 4.Cl Att | 5.Cl fs | 6.Cd  | 7.Cm  | 8.F
36100
-180   -49   -0.22013   -20.5241432   -0.22013   0.0199118108   0.0451649986   0
-180   -48   -0.22013   -20.5241432   -0.22013   0.0199118108   0.0451649986   0
  ...  ...
-180   +50    0.21096   -20.088768     0.21096   0.0199443996   -0.0431930013   0
-179   -49  ...
-179   -48  ...
  ...  ...
+180   +50  ...
```

12.15 Data format for the user defined a-ct relation

The format of the file is specified in the following two tables

Line number	Description
1. nrad interp	Nrad interpolation. Interpolation method can either be “linear” or “akima”
2. nazi	Data row according to Table 11

Table 33: Format of main data structure for the profile coefficients file

The content of the colums in a data row is specified in table below.

Column	Parameter
1	non-dim radius r/R
2	k_1 polynomium coef
3	k_2 polynomium coef
4	k_3 polynomium coef
5	k_4 polynomium coef

Table 34: Format of the data rows for the profile coefficients file

12.16 Main command block – blade_c2_def (for use with old_htc_structure format)

In this command block the definition of the centerline of the main_body is described (position of the half chord). This command shall be used as a main command even though it is only used together with the aerodynamic module. The reason for this is that it used to submit information that is usually given in the new_htc_structure format, which is also a main command block. The input data given with the sec commands below is used to define a continuous differentiable line in space using akima spline functions. This centerline is used as basis for local coordinate system definitions for sections along the structure. If a straight line is requested a minimum of three points of this line must be present.

Obl.	Command name	Explanation
*	nsec	Must be the present before a “sec” command. 1. Number of section commands given below
*	sec	Command that must be repeated “nsec” times 1. Number 2. x-pos [m] 3. y-pos [m] 4. z-pos [m] 5. θ_z [deg]. Angle between local x-axis and main_body x-axis in the main_body x-y coordinate plane. For a straight blade this angle is the aerodynamic twist. Note that the sign is positive around the z-axis, which is opposite to traditional notation for etc. a pitch angle.

13 Aerodrag (for tower and nacelle drag)

13.1 Main command aerodrag

With this module it is possible to apply aerodynamic drag forces at a given number of structures.

13.2 Subcommand aerodrag_element

Command block that can be repeated as many times as needed. In this command block aerodynamic drag calculation points are set up for a given main body.

Obl.	Command name	Explanation
*	body_name or mbdy_name	1. Main_body name to which the aerodynamic calculation points are linked.
*	aerodrag_sections	1. Distribution method: (“uniform” only possibility) 2. Number of calculation points (min. 2).
	nsec	This command must be present before the sec commands 1. Number of sections given below
	sec	This command must be repeated nsec times 1. Distance in [m] along the main_body c2_def line. Positive directed from node 1 to node “last”. 2. C_d drag coefficient (default=1.0) 3. Width of structure (diameter)
	update_states	Logical parameter that determines whethe the movement of the structure is included or not. 1. parameter (1=states are updated (default), 0=not updated)

*) Input commands that must be present

14 Hydrodynamics

14.1 Main command block - hydro

In this command block hydrodynamic forces calculated using Morison's formula is set up.

14.2 Sub command block – water_properties

Obl.	Command name	Explanation
*	gravity	1. Gravity acceleration (used for calculation of buoyancy forces). Default = 9.81 m/s^2
*	mudlevel	1. Mud level [m] in global z coordinates.
*	mwl	1. Mean water level [m] in global z coordinates.
*	rho	1. Density of the water [kg/m3]. Default=1000
	wave_direction	1. Wave direction [deg]. Direction is positive when the waves come forward from the right when looking towards the wind at default conditions.
	current	1. Current type (0=none (default), 1=constant, 2=power law $U(z) = U_0((z + mudlevel - mwl)/(mudlevel - mwl))^{\alpha}$ 2. Current velocity at mwl, u_0 3. type parameter. If type=2 then parameter is alfa 4. Current direction relative to wave direction [deg]. Positive direction if current comes from the right looking towards the incoming waves.
	water_kinematics_dll	1. Filename incl. relative path to file containing water kinematics dll (example ./hydro/water_kin.dll) 2. String sent to initialization of dll. This is typical the name of a local inputfile of the dll.

14.3 Sub command block – hydro_element

Command block that can be repeated as many times as needed. This command block set up hydrodynamic calculation points and link them to a main_body.

Obl.	Command name	Explanation
*	body_name or mbdy_name	1. Main_body name to which the hydrodynamic calculation points are linked.
*	hydrosections	1. Distribution method of hydrodynamic calculation points. Options are: "uniform" nnodes. Where uniform ensures equal distance of the calculation points. nnodes are number of calculation points. "auto" nint. Here calculations points are chosen as the postions of the structural nodes and the hydro dynamic input section given by the sec command. The parameter nint is a refinement parameter given nint extra calculation points in between the other points.
*	nsec	This command must be present before the sec commands 1. Number of sections given below
*	sec_type	Type of cross section (1=circular, 2=general)
*	sec	This command must be repeated nsec times and is different for each section type. Section type 1 – circular:

Obl.	Command name	Explanation
		1. Relative distance along the main_body c2_def line. Positive directed from node 1 to node "last". 2. C_a added mass coefficient (default=1.0) 3. C_d drag coefficient (default=1.0) 4. Cross sectional area [m2] 5. Cross sectional area to which C_a is related. (default=area for circular sections) [m2] 6. Width of construction perpendicular to flow direction [m] 7. drdz gradient(optional). For calculating the buoyancy also for conical sections the gradient expressing the change in radius with change of distance along the main_body c2_def line. Only important when buoyancy forces are included. 8. Axial drag C_d coefficient for concentrated force contribution (optional). Drag area is circular area defined by the local width. Contribution is quadratic regarding water velocity. 9. Axial added mass $C_{a,axial}$ coefficient for concentrated force contribution (optional). Force is computed (in each hydro element section with $C_{a,axial}$ different than 0) as: $\rho V_{ref} C_{a,axial}(water\_acc - body\_acc)$, with V_{ref} taken as half volume of sphere defined by the local width as diameter. 10. Axial drag C_d coefficient for concentrated force contribution (optional). Drag area is circular area defined by the local width. Contribution is linear regarding water velocity. 11. Internal cross sectional area for flooded members [m2] (optional). 0=member is not flooded. 12. Torque friction coefficient Cf (optional). For rotating cylinders around local z-direction. $M_z = \frac{1}{16}\rho\pi D^4\omega^2 \mathrm{C_f}$
	buoyancy	1. Specification whether buoyancy forces are included or not. 0=off (default), 1=on (remember to define the 7th parameter in the sec input line.
	update_states	1. Specification whether the hydrodynamic sections are updated in time with respect to pos, vel, acc and orientations, or simply considered to remain fixed. 0=not updated, 1=updated (default)
	update_kinematics	1. Specification whether the water kinematics are updated during iterations or only once per time step. 0=only updated once per time step, 1=full update (default).

Here is an example of this written into the htc-input file.

```
begin HYDRO_ELEMENT ;
 mbdy_name cylinder ;
 buoyancy 1 ;
 update_states  1 ; (0: no dynamic interaction, 1: fully coupled solution
 hydrosections auto 4 ; dist, of hydro calculation points from 1 to nsec
 nsec 2; z    Ca Cd A      Aref   width dr/dz Cd_a_(quad) Ca_a Cd_a_lin Aif
 sec     0.0  1  1  3.404  3.404  2.082 0.0   0.0         0.0  0.0      3.023;
 sec     5.0  1  1  3.404  3.404  2.082 0.0   0.0         0.0  0.0      3.023;
end HYDRO_ELEMENT ;
```

This example shows a flooded cylindrical element (l=5 m, d= 2,082 m and t=60mm).

14.4 Description of the water_kinematics_dll format.

```
subroutine init(inputfile,t0,t1,dt) implicit none
character*(*) :: inputfile
real*8 :: t0 ! start time for simulation
real*8 :: t1 ! stop time for simulation
real*8 :: dt ! time increment
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS:'init'::init
end subroutine init

!-------------------------------------------------------------------------------
subroutine set_new_time(time)
implicit none
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS:'set_new_time'::set_new_time
real*8                :: time

end subroutine set_new_time

!-------------------------------------------------------------------------------
subroutine get_sea_elevation(posxy_h,elevation)
implicit none
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS:'get_sea_elevation'::get_sea_elevation
real*8,dimension(2) :: posxy_h         ! horizontal position coordinates
real*8            :: elevation ! water height above mean water           ! level, positive u
end subroutine get_sea_elevation

!-------------------------------------------------------------------------------
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS:'get_kinematics'::get_kinematics
real*8,dimension(3) ::  pos_h,& vel_h,&
acc_h
real*8 :: pres
end subroutine get_kinematics
```

14.5 User manual to the standard wkin.dll version 2.8.3

The wkin.dll which is delivered along with the HAWC2 code needs a separate inputfile. The format for these inputs are the same as the HAWC2 main inputfile with usage of begin..end clauses, semi colon separators, exit command etc. Command words are described below.

All command words written below has to be included in an begin .. end clause called wkin_input:

```
begin wkin_input;
...
end wkin_input;
exit;

Version info:
 1.0 TJUL Basic edition by TJUL
 1.1 ANMH  Wave field can be read by file and used directly through fft
conversion
 1.2 TJUL Directional spreading included
 1.3 ANMH Bug corrected regarding read on seed number using iregular
waves
```

```
 1.4 TJUL Pierson-Moscowitz spectrum added as option
Stream function wave added
  Possible pre processing of wave field to speed up simulation
time and enable many more coeffients
 1.5 TJUL Bug in stream function wave. Static pressure was included -
now removed
 1.6 TJUL Bug in stream fuction wave. lateral position was applied
instead of vertical in kinematics look-up!!!
 1.7 TKIM New wave format for precalculated (high order) wave fields
 1.8 ANMH  Update in deterministic iregular waves+bugfix
 1.9 TJUL New option for white noise wave exitation
 2.0 TJUL Bug fix of version 1-9. Version 1-9 had some debug
statements included that could meas up the time.
2.1 ANMH Ported to intel
            ANMH     Correction for high wave numbers in deterministic irregular
waves
            TJUL Embedded stream function wave, phase velocity used insted
of group velocity with respect to pregenerated waves
2.2   TJUL     Bug fix. Tightended criteria for jonswap spectrup min-max.
Use of real*8 in all internal memory related variables
2.3  TJUL    Bug fix. PM spectrum ireg waves
2.4 TJUL    Update so embedded stream function wave is ensured to be inside the requested time
2.5 TJUL    Bugfix in randomnumber generator. Problem occured in version 2.1 until 2.4
SHFE    Bugfix in embedded steam function wave
2.6 TJUL    Embedded stream function wave updated for manual input of Tp
2.7 ANMH    Bugfix regarding embedded stream function wave
SHFE    Bugfix (stretching first, then embed stream function wave)
2.8 SHFE    New feature to write out the pregenerated wave field
SHFE Change PM spectra from Tz type to Tp type
SHFE Solve the memory issue when pregenerate large scale wave field
SHFE Fix issue with long filenames
2.8.3 SHFE McCamy Fuchs correction is applied on water particle acceleration
```

14.6 Main commands in the wkin.dll

Obl.	Command name	Explanation
*	wavetype	1. Type of wave used. (0=regular airy, 1=irregular airy, 2=deterministic irregular airy, 3=regular stream function, 4=general wavemode format)
*	wdepth	1. Water depth [m]. Positive value.

14.7 Sub command reg_airy

Command that need to be present if the wavetype equals 0 in the main command.

Obl.	Command name	Explanation
*	stretching	1. Wheeler stretching of waves. (0=off, 1=on)
*	wave	1. Wave height H [m] 2. Wave period T [s] 3. Wave phase shift [deg] (optional)
	ignore_water_surface	Allow the lookup of the wave kinematics above the waterline if requested in the output.

14.8 Sub command ireg_airy

Command that need to be present if the wavetype equals 1 in the main command.

Obl.	Command name	Explanation
*	stretching	1. Wheeler stretching of waves. (0=off, 1=on)
*	spectrum	1. Base spectrum used. (1=jonswap, 2= Pierson Moscowitz)
	mccamyfuchs	1. McCamy Fuchs correction on water particle acceleration. (0=off, 1=on) 2. Representative radius [m]
	jonswap	Jonswap spectrum formulation 1. Significant wave height H_s [m] 2. Wave period T_p [s] 3. γ parameter [-]. A typical value is 3.3
	pm	Pierson-Moscowitz spectrum 1. Significant wave height H_s [m] 2. Wave period T_p [s]
	wn	White noise. 1. Target variance level [m^2] 2. f_0, minimum frequency 3. f_1, maximumn frequency
*	coef	1. Number of coefficients. Normally 200 are used even though higher values are recommended in general. A speed issue... 2. Seed number. A positive integer value. 3. Phase shift for all wave components [deg] (optional).
	spreading	1. Spreading model. (0=none, 1=K_{2s} model also referred to as K_n model) 2. Spreading parameter. If model=1 the parameter is s, a positive integer. The higher value, the less spreading.
	pregen	Pre-generation of a wave field (default is on). Using this option the irregular wave field is calculated during initialization phase and only table look-up is done during the time simulation phase. Very fast and still accurate. 1. Pregen option. (0=traditional approach (slow), 1=pregenerated wave field used (default))
	embed_sf	Embed stream function wave in time series at the time when the otherwise largest wave occurs. The wave kinematics is blended into the iregular waves before and after. 1. Wave height H [m] 2. Wave period T [s]. Default = Peak wave period T_p. (optional) 3. Truncated transition period T0 [s]. Default = 0. (optional)

14.8.1 Sub sub command pregen_field

Command that used to define the resolution of the pregenerated wave field if this feature is activated where the pregen equals to 1 (default). The whole command block is optional.

Obl.	Command name	Explanation
	wave_filename	1. File name for writing (file not existed) or reading (file existed) pregenerated wave field.
	y_resolution	1. Field dimensions in lateral direction. Default is 1. 2. Lateral grid length. Default is 0.

Obl.	Command name	Explanation
	t_resolution	1. The time step used for the pregenerated wave field. Default = 1/10 of maximum wave period.
	z_resolution	1. Field dimensions in vertical direction. Default is 10 points in z direction.
	x_range	1. extra simulated wave train in meters before and after requested time interval. Default is 100 m.

14.9 Sub command det_airy

Command that need to be present if the wavetype equals 2 in the main command. This command is used when water kinematics needs to be calculated based on a measured elevation time series.

Obl.	Command name	Explanation
*	file	1. File name for measured wave elevation.
*	nsamples	1. Number of lines present in wave elevation file
*	nskip	1. Number of lines to skip before reading of wave elevation file
*	columns	1. Column number for time sensor in file. 2. Column number for wave elevation in file.
	stretching	1. Wheeler stretching of waves. (0=off, 1=on (default))
*	cutoff_frac	1. Fraction of total energy which is discarded in the low and high frequency ranges. Default 1E-5
	pregen	Pre-generation of a wave field (default is on). Using this option the irregular wave field is calculated during initialization phase and only table look-up is done during the time simulation phase. Very fast and still accurate. 1. Pregen option. (0=traditional approach (slow), 1=pregenerated wave field used (default))
	x_range	1. extra simulated wave train in meters before and after requested time interval. Default is 100 m.
	wave_filename	1. File name for writing (file not existed) or reading (file existed) pregenerated wave field.

14.10 Sub command strf

Stream function wave input.

Obl.	Command name	Explanation
*	wave	1. Significant wave height H_s [m] 2. Wave period T [s] 3. Current speed U [m/s]

14.11 Sub command wavemods

Command that need to be present if the wavetype equals 4 in the main command. This command is used when water kinematics needs to be calculated based on a measured elevation time series.

Obl.	Command name	Explanation
*	datafile_y	1. Name of datafile where wave kinematic data is present for the horizontal (wave) direction
*	datafile_z	1. Name of datafile where wave kinematic data is present for the vertical direction

Obl.	Command name	Explanation
*	datafile_nd	1. Number of depth locations
*	datafile_depth	1. Minimum water depth (m)
*	datafile_nt	1. Number of time steps in datafile
*	datafile_t0	1. Time for when wave data is extracted in the datafiles
*	ncol_y	1. Number of columns in datafile1 (time+eta+vel+acc)
*	ncol_z	2. Number of columns in datafile2 (time+vel+acc)

An example of input files with wave kinematics data for the wavemods option is given below. Please note the following:

- The first 9 lines are general comment lines
- Line 10 lists the relative depths, and the number of relative depths must match datafile_Nd in the wavemods subcommand
- Each row starting at Line 12 corresponds to a single time step, and there should be at least datafile_Nt rows before the end of the file
- The datafile columns correspond to time, eta (the distance between the wave height and the MSL; not present in the vertical-component input file), datafile_Nd velocities, and then datafile_Nd accelerations

Example of datafile_y (horizontal wave component):

```
Wave kinematics input to Flex5 Monopile ver. 2.1
General comment line
Wave load program "WaveKin" ver. 1.0
Echo file : Outfile.dat
Name of Case
Wave Description
slope 1:25
50  water depth
 3 No rel. depths N
0.000 0.500 1.000
 T      eta   u[1]..u[N]     a[1]..a[N]
0.000  -0.645   -0.022  -0.027  -0.047  -0.018   -0.022  -0.035
0.063  -0.659   -0.023  -0.029  -0.049  -0.017   -0.021  -0.032
0.126  -0.671   -0.025  -0.030  -0.051  -0.016   -0.020  -0.030
(etc)
```

Example of datafile_z (vertical wave component):

```
Wave kinematics input to Flex5 Monopile ver. 2.1
General comment line
Wave load program "WaveKin" ver. 1.0
Echo file : Outfile.dat
Name of Case
Wave Description
slope 1:25
50  water depth
 3 No rel. depths N
0.000 0.500 1.000
 T      u[1]..u[N]     a[1]..a[N]
```

```
0.000  -0.022  -0.027  -0.047  -0.018   -0.022  -0.035
0.063  -0.023  -0.029  -0.049  -0.017   -0.021  -0.032
0.126  -0.025  -0.030  -0.051  -0.016   -0.020  -0.030
(etc.)
```

14.12 Wkin.dll example file

```
begin wkin_input ;
  wavetype 1 ;       0=regular, 1=irregular, 2=deterministic
  wdepth  220.0 ;
;
  begin reg_airy ;
    stretching 0;    0=none, 1=wheeler
    wave 9 12.6;     Hs,T
  end;
;
  begin ireg_airy ;
    stretching 0;     0=none, 1=wheeler
    spectrum   1;    (1=jonswap)
    jonswap 9 12.6 3.3 ; (Hs, Tp, gamma)
    coef 200 1 ;      (coefnr, seed)
    spreading 1 2;      (type(0=off 1=on), s parameter (pos. integer min 1)
  end;
;
  begin det_airy ;
    stretching 0;     0=none, 1=wheeler
    file ..\waves\elevation.dat ;
    nsamples 32768 ;
    nskip 1 ;
    columns 1 5 ;     time column, elevation column
  end;
;
  begin wavemods;
    datafile_y  ./wavedata/wavekin_y.dat;
    datafile_z  ./wavedata/wavekin_z.dat;
    datafile_nt  900;  number of time steps in file
    datafile_nd  3;  number of relative water depths
    datafile_t0  50;   start time for data extraction
    datafile_depth  50 ; minimum water depth
    ncol_y 8; Number of data columns in file
    ncol_z 7; Number of data columns in file
  end;
end;
;
exit ;
```

15 Soil module

15.1 Main command block - soil

In this command block soil spring/damper forces can be attached to a main body. The formulation is performed so it can be used for other external distributed spring/damper systems than soil.

15.2 Sub command block – soil_element

Command block that can be repeated as many times as needed. In this command block the distributed soil spring/damper system is set up for a given main body.

Obl.	Command name	Explanation
*	mbdy_name	1. Main_body name to which the soil calculation points are linked.
*	datafile	1. Filename incl. relative path to file containing soil spring properties (example ./soil/soildata.dat)
*	soilsections	1. Distribution method: (“uniform” only possibility) 2. Number of section (min. 2).
	damping_k_factor	1. Rayleigh kind of damping. Factor the linear stiffness coefficients are multiplied with to obtain the damping coefficients. When the factor is 1.0 the vibration is critically damped for the rigid mainbody connected to the spring and dampers.
♣	set	1. Set number in datafile that is used.

*) Input commands that must be present

♣) Command can be repeated as many times as desired.

15.3 Data format of the soil spring datafile

In the file (which is a text file) different distributed springs can be defined. Each set is located after the “#” sign followed by the set number. Within a set the following data needs to be present.

line 1	“spring type”	(can be “axial”, “lateral” or “rotation_z”)
line 2	“nrow ndefl”	(nrow is number of rows, ndefl is number of deflections (colums)
line 3.. 3+nrow	“z_global F(1) F(2),..., F(ndefl)”	First colum is the spring location (global z coordinate). The following colums are Force/length at the different deflection stations. First deflection must be zero. The forces are assumed symmetrical around the zero deflection.

An example is given below:

```
This is a nonlinear soil spring demonstration file
#1
lateral              (axial/lateral)
5 4         nrow ndefl
0.0  0.1  0.2  1.0   x1 x2 x3 ..... [m]
0.0   0  15  20  500   Z_G  F_1 F_2 F_3 .... F_ndefl [kN/m]
10.0  0  15  20  500
```

```
20.0  0  15  20  500
30.0  0  15  20  500
40.0  0  15  20  500
#2
axial                (axial/lateral)
5 4          nrow ndefl
0.0  0.1  0.2  1.0   x1 x2 x3 ..... [m]
0.0   0  150  200  5000  Z_G  F_1 F_2 F_3 .... F_ndefl [kN/m]
10.0  0  150  200  5000
20.0  0  150  200  5000
30.0  0  150  200  5000
40.0  0  150  200  5000
#3
rotation_z           (axial/lateral/rotation_z)
5 4          nrow ndefl
0.0  0.1  0.2  1.0   x1 x2 x3 ..... [rad]
0.0   0  150  200  5000  Z_G M_1 M_2 M_3 .... M_ndefl [kNm/m]
10.0  0  150  200  5000
20.0  0  150  200  5000
30.0  0  150  200  5000
40.0  0  150  200  5000
```

16 External forces

16.1 Main command block – Force

16.1.1 Sub command - Base

This command block can be used to specify a user-defined constant external force and/or moment on a node on the structure.

Obl.	Command name	Explanation and parameters
	name	1. Name used to reference the force from output sensors
	mbdy	1. Name of mainbody.
	node	1. Node number.
	force	External force in global coordinates 1. Fx [N] 1. Fy [N] 1. Fz [N]
	moment	External moment in global coordinates 1. Mx [Nm] 1. My [Nm] 1. Mz [Nm]

16.1.2 Sub command - DLL

This command block can be used when a user defined external force is applied to the structure. The main difference between this DLL format and the normal DLL control interface (used with external controllers) is that added stiffness is calculated initially leading to a more robust a fast solution of the coupled system. This force module can with good results be applied for external equivalent soil-springs or hydrodynamic forces for floating constructions or mooring lines.

Obl.	Command name	Explanation and parameters
*	filename	1. Filename incl. relative path to the external DLL (example ./dll/force.dll)
	dll	deprecated alternative to filename
	init	1. Name of subroutine in the DLL that is called before the simulation starts. 2. String passed to the init subroutine.
*	update	1. Name of subroutine in the DLL that is called at each time step.
*	mbdy	1. Name of main body to which force dll is coupled.
*	node	1. Node number of main body to which force dll is couple

16.2 Example of a DLL interface written in fortran90

```
!
! Demonstration of force DLL
!
SUBROUTINE DemoForceDLL(time,x,xdot,xdot2,amat,omega,omegadot,F,M)
!DEC$ ATTRIBUTES DLLEXPORT::DemoForceDLL
!DEC$ ATTRIBUTES ALIAS:'demoforcedll' :: DemoForceDLL
! input
DOUBLE PRECISION                :: time    ! time
DOUBLE PRECISION ,DIMENSION(3)  :: x       ! global pos. of reference node
```

```
DOUBLE PRECISION ,DIMENSION(3)   :: xdot    ! global vel. of reference node
DOUBLE PRECISION ,DIMENSION(3)   :: xdot2   ! global acc. of reference node
DOUBLE PRECISION ,DIMENSION(3)   :: omega   ! angular vel. of ref. node
      ! (global base)
DOUBLE PRECISION ,DIMENSION(3)   :: omegadot ! angular acc. of ref. node
! (global base)
DOUBLE PRECISION ,DIMENSION(3,3) :: amat     ! rotation matrix (body ->
!                  global)
! output
DOUBLE PRECISION ,DIMENSION(3)   :: F        ! External force in reference
! node (global base)
DOUBLE PRECISION ,DIMENSION(3)   :: M        ! External moment in reference
! node (global base)
! locals
LOGICAL, SAVE                    :: bInit = .FALSE. ! Initialization flag
DOUBLE PRECISION                 :: mass  = 0.d0    ! Point mass
!
! Initialise on first call
IF (.NOT.bInit) THEN
  bInit = .TRUE.
  ! Open file and read mass
  OPEN(10,FILE="DemoForceDLL_mass.dat")
  READ(10,*) mass
  CLOSE(10)
ENDIF
!
! Calc. force
F = mass*((/0.d0,0.d0,9.81d0/) - xdot2)
M = 0.d0
!
END SUBROUTINE DemoForceDLL
```

16.3 Example of a DLL interface written in Lazarus / Pascal

```
library force_dll;

Type
  vect = array[0..2] of double;
  mat  = array[0..2,0..2] of double;

procedure update( var time:double;var x:vect;var xdot:vect;var xdot2:vect;
var amat:mat;var omega:mat;var omegadot:vect;
var F,M:vect);stdcall;

// Example of applying a step up force in the x-direction:
begin
if time < 10 then
  F[0] := 0.0;
if time >= 10 then
  F[0] := 20000.0;
if time >= 20 then
  F[0] := 40000.0;
end;
```

```
exports update;

begin
  writeln('The DLL force_dll.dll is loaded with succes');
end.
```

17 Output

This command output can either be a main command block or a sub command block within the hawc_dll command block. In the tables below two special columns are introduced. One is only option and the other label option. When the check mark is ‘yes’ in only option it is possible to use only one of the fields if more than one sensor was defined through the command. The sensor that is used is determined by the number following the only command word, see example below.

```
constraint bearing1 shaft_rot 2 only 2;
```

If the only command (and the following number) was omitted two sensors was defined; one for the angle and one for the velocity. With the only command only the velocity sensor is used in the output since the following number is 2.

With the label option it is possible to make a user defined label of the sensor which is written in the sensor list file. The label command is the # symbol. Everything after the # symbol is used as a label. An example of this could be

```
dll inpvec 1 1 # This is a dummy label ;
```

With the *$calc()* option, the output value of output sensors can be manipulated by various math operations. This feature can be used e.g. to offset time sensor, or to scale forces from kN to N, or to do more complex operations. The $calc() must be placed after the output line, either before or after the # symbol, e.g.

```
dll inpvec 1 1 $calc(*1000) # This is a dummy label ;
```

The operation string inside $calc() is composed of sets of:

1 Operation key describing the math operation (e.g. ’-’,’+’,’*’,’/’),
2 then a (optional, dependent on operation) number <val>,
3 and then ’=’ character (to separate operations)(this can be omitted for last operation)

E.g. $calc(-100=*5) added to sensor line x will return (x-100)*5 in the x sensor output.

Other math operations available (other than -+*/) are:

power,	$calc(pow<val>)	:	returns $x^{<val>}$
signed power,	$calc(sgnpow<val>)	:	returns sign(x) * abs($x^{<val>}$)
absolute,	$calc(abs)	:	returns abs(x)
sine,	$calc(sin)	:	returns sin(x)
cosine,	$calc(cos)	:	returns cos(x)
tangens,	$calc(tan)	:	returns tan(x)

17.1 Commands used with results file writing

When the output command is used for output files (the most normal purpose) some information regarding file name and format needs to be given.

Obl	Command	Explanation
*	filename	1. Filename incl. relative path to outputfile without extension (example ./res/output)
	data_format	ASCII or compressed binary output can be chosen. Default is the ASCII format if nothing is specified. 1. format (‘hawc_ascii’=ASCII format,

Obl	Command	Explanation
		'hawc_binary'=compressed binary format, 'flex_int'=compressed binary format, 'gtsdf'=General time series data format (hdf5 based compressed binary), 'gtsdf64'=General time series data format (hdf5 based binary)) 2. optional for 'flex_int', time [s] to subtract from the time channel.
	buffer	Buffer size in terms of time steps. When the buffer is full the data are written to data file. Only used together with the 'hawc_ascii',' gtsdf' and 'gtsdf64' formats. Default is 3000 time steps 1. 1. buffer size
	deltat	Time interval between outputs [s]. If 'deltat' is smaller than simulation time step, output is made each time step.
	time	Time start t_0 and stop t_1 for output is defined. Default is the entire simulation length if nothing is specified. 2. t_0 3. t_1

17.2 File format of HAWC_ASCII files

Results are written to an ascii formatted data file with the name assigned to the filename variable (eg. filename ./res/resfil). The data file will have the extension .dat as a standard. The description of the sensors in the data file is given in another textfile with same filename as the data file but the extension .sel. An example could be: ./res/resfil.dat and ./res/resfil.sel.

In the .sel-file, line numer 9 specifies the following parameters: Number of scans, Number of sensors, Duration of output file, Data format (ASCII/BINARY). Example:

```
10 96 20.000 ASCII
```

From line number 13 and onwards, the sensors are specified with the following information: Sensor number, Variable description, unit, Long description. Example:

```
5      bea1 angle_speed              rad/s      pitch1 angle speed
```

Full example of the .sel file:

```
_____________________________________________________________________________
  Version ID : HAWC2MB 4.3w
                                                          Time : 14:23:28
                                                          Date : 22:11.2006
_____________________________________________________________________________
  Result file : ./res2_rev0/case41c_nohydro.dat
_____________________________________________________________________________
   Scans    Channels    Time [sec]      Format
     4500      199         90.000       ASCII

  Channel   Variable Description

     1      Time                          s          Time
     2      bea1 angle                    deg        shaft_rot angle
```

```
    3      bea1 angle_speed              rpm        shaft_rot angle speed
    4      bea1 angle                    deg        pitch1 angle
    5      bea1 angle_speed              rad/s      pitch1 angle speed
    6      bea1 angle                    deg        pitch2 angle
    7      bea1 angle_speed              rad/s      pitch2 angle speed
    8      bea1 angle                    deg        pitch3 angle
    9      bea1 angle_speed              rad/s      pitch3 angle speed
-------------------------------------------------------------------------
```

17.3 File format of HAWC_BINARY files

In this file format results are written to a binary unformatted data file with the name assigned to the filename variable (eg. filename ./res/resfil). The data file will have the extension .dat as a standard. The description of the sensors in the data file is given in another textfile with same filename as the data file but the extension .sel. An example could be: ./res/resfil.dat and ./res/resfil.sel.

The data are scaled to standard 2-byte integers, with a range of 32000 using a scalefactor. The scalefactor is determined for each output sensor

$$s = \frac{\max(|max|, |min|)}{32000}$$

where *max* and *min* are the largest and lowest number in the original data for the sensor. These scale factors are written in the end of the accompanying .sel file. When converting a binary number to the actual number its just a matter of multiplying the binary numbers of a sensor with the corresponding scalefactor.

In the accompanying text file, which has the extension .sel-file, information of the content in the datafile is stored. In line number 9 the following parameters are specified: Number of scans, Number of sensors, Duration of output file, Data format (ASCII/BINARY). Example:

```
10 96 20.000 ASCII
```

From line number 13 and onwards, the sensors are specified with the following information: Sensor number, Variable description, unit, Long description. Example:

```
5      bea1 angle_speed              rad/s      pitch1 angle speed
```

From line number 9+nsensors+5 and upwards the scalefactors are written.

Full example of the .sel file:

```
-------------------------------------------------------------------------
  Version ID : HAWC2MB 4.3
                                                          Time : 14:23:28
                                                          Date : 22:11.2006
-------------------------------------------------------------------------
  Result file : ./res2_rev0/case41c_nohydro.dat
-------------------------------------------------------------------------
   Scans    Channels    Time [sec]      Format
     4500        9         90.000       ASCII

  Channel   Variable Description
```

```
     1      Time                          s          Time
     2      bea1 angle                    deg        shaft_rot angle
     3      bea1 angle_speed              rpm        shaft_rot angle speed
     4      bea1 angle                    deg        pitch1 angle
     5      bea1 angle_speed              rad/s      pitch1 angle speed
     6      bea1 angle                    deg        pitch2 angle
     7      bea1 angle_speed              rad/s      pitch2 angle speed
     8      bea1 angle                    deg        pitch3 angle
     9      bea1 angle_speed              rad/s      pitch3 angle speed
_____________________________________________________________________________________
Scale factors:
  1.56250E-04
  5.61731E-03
  4.41991E-04
  1.00000E+00
  1.00000E+00
  1.00000E+00
  1.00000E+00
  1.00000E+00
  1.00000E+00
```

An important thing to notice is that in the binary data file all sensors are stored sequentially, i.e. all data for sensor 1, all data for sensor 2, etc. This way of storing the data makes later reading of a sensor extra fast since all data for a sensor can be read without reading any data for the other sensor.

A small matlab code for reading the binary HAWC2 format can be seen below.

```
function sig = ReadHawc2Bin(FileName,path);
% Reads binary HAWC2 results file
% ------------------------------------
% [t,sig] = ReadFlex4(FileName,Ch);
% filename should be without extension
% ------------------------------------
% BSKA 26/2-2008
% ------------------------------------
ThisPath = pwd; cd(path(1,:))

% reading scale factors from *.sel file
fid = fopen([FileName,'.sel'], 'r'); fgets(fid); fgets(fid);
fgets(fid); fgets(fid); fgets(fid); fgets(fid); fgets(fid);
fgets(fid);
tline = fscanf(fid,'%d');
N = tline(1); Nch = tline(2); Time = tline(3); fclose(fid);
ScaleFactor = dlmread([FileName,'.sel'],'',[9+Nch+5 0 9+2*Nch+4
0]);

% reading binary data file
fid = fopen([FileName,'.dat'], 'r'); sig =
fread(fid,[N,Nch],'int16')*diag(ScaleFactor); fclose(fid);

cd(ThisPath)
```

17.4 File format for gtsdf and gtsdf64 files

The file formats and reading and writing examples of the gtsdf and gtsdf64 file types and are described here: `https://gitlab.windenergy.dtu.dk/toolbox/WindEnergyToolbox/blob/master/wetb/gtsdf/General%20Time%20Series%20Data%20Format.pdf`

A reference Python implementation to read and write gtsdf files is available in the open source Wind Energy Toolbox: `https://gitlab.windenergy.dtu.dk/toolbox/WindEnergyToolbox/blob/master/wetb/gtsdf/gtsdf.py`

17.5 mbdy (main body output commands)

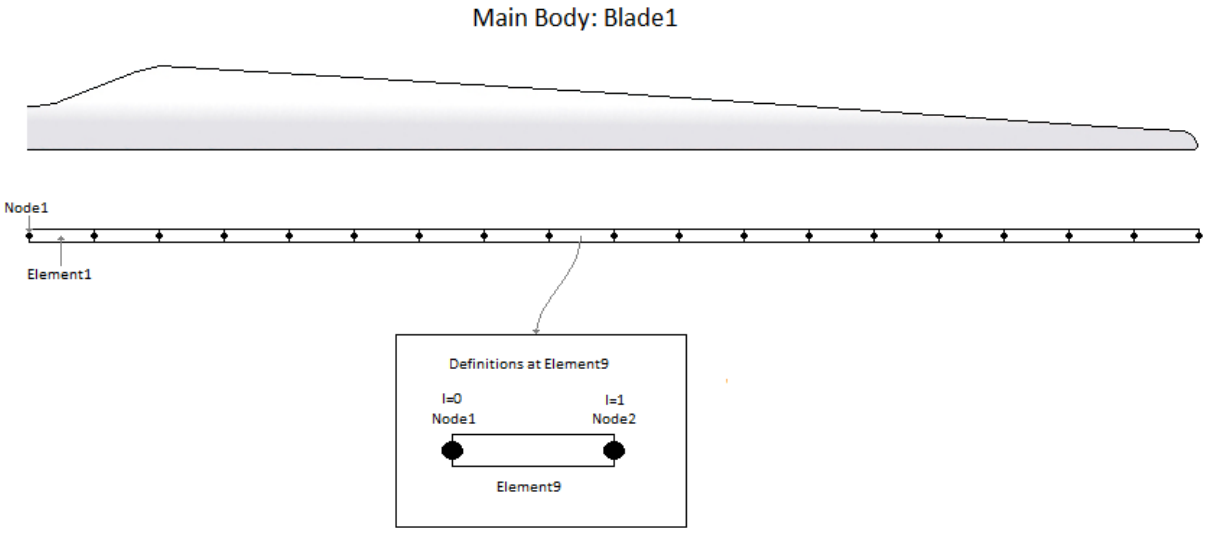
Command 1	Command 2	Explanation	Only option	Label option
mbdy	forcevec	F_x, F_y, F_z shear force vector, see definition in figure 7. 1. Main_body name 2. Element number 3. Node number on element (1 or 2) 4. Main_body name of which coordinate system is used for output. “global” and “local” can also be used. Local is around local beam main bending directions.	yes	yes
mbdy	momentvec	M_x, M_y, M_z moment vector, see definition in figure 7. 1. Main_body name 2. Element number 3. Node number on element (1 or 2) 4. Main_body name of which coordinate system is used for output. “global” and “local” can also be used. Local is around local beam main bending directions.	yes	yes
mbdy	forcemomentvec_interp	F_x, F_y, F_z, M_x, M_y, M_z interpolated shear force and moment vector defined to output. This sensor can write out an interpolated set of cross sectional forces and moments independent of the node discretization. It can also write out in local deformed c2_def coordinates and therefore breaks the limit of using element coordinates. 1. Main_body name 2. Position of location outputted: ‘c2def’ or ‘default’ (default = elastic center). 3. Name of mbdy used for output coordinate system: mbdy_name, ‘global’, ‘local_aero’ or ‘local_element’ 4. Distance along c2_def to output location 5. Sign multiplied to output: 1.0 or -1.0	yes	yes
mbdy	state	Vector with 3 components of either position, velocity or acceleration of a point on an element defined to output. If ‘acg’ is used, the acceleration including the gravity contribution is written. 1. State: ‘pos’, ‘vel’, ‘acc’, ’acg’ (“pos”=position, “vel”=velocity, “acc”=acceleration)	yes	yes

Command 1	Command 2	Explanation	Only option	Label option
		2. Main_body name 3. Element number 4. Relative distance from node 1 to node 2 on element 5. Main_body name of which coordinate system is used for output. “global” can also be used.		
mbdy	state_at	Vector with 3 components of either position, velocity or acceleration of a point on an element defined to output. The point is offset from the element z axis by an x and y distance in element coordinates. 1. State: ‘pos’, ‘vel’, ‘acc’, ’acg’ 2. Main_body name 3. Element number 4. Relative distance from node 1 to node 2 on element 5. Main_body name of which coordinate system is used for output. “global” can also be used. 6. x-coordinate offset [m] 7. y-coordinate offset [m]	yes	Yes
mbdy	state_at2	Vector with 3 components of either position, velocity or acceleration of a point on an element defined to output. The point is offset from the c2_def centerline z axis by an x and y distance in local c2def centerline coordinates. 1. State: ‘pos’, ‘vel’, ‘acc’, ’acg’ 2. Main_body name 3. Element number 4. Relative distance from node 1 to node 2 on element 5. Main_body name of which coordinate system is used for output. “global” can also be used. 6. x-coordinate offset [m] 7. y-coordinate offset [m]	yes	Yes
mbdy	state_rot	Vector with components of either axis and angle (angle [rad], r_1,r_2,r_3), euler parameters (quaternions r_0,r_1,r_2,r_3), euler angles, rotation velocity (-vector) or rotation acceleration (-vector) of a point on an element defined to output. For the sensor eulerang_xyx a set of euler angles are created based on the orientation matrix. Be aware that the method used is only valid for rotations in the intervals	yes	Yes

Command 1	Command 2	Explanation	Only option	Label option
		(θ_x ±180°, θ_y ±90°, θ_x ±180°). The method proj_ang can be used to see how much a blade tip rotates around the pitch axis, but be aware that the angles are how the element is oriented and not necesarily how the local chord is rotated. With the command proj_ang the angles are obtained from the local element orientation 3x3 matrix T_e, seen from the chosen coordinate system using the Atan2 functions (rot_x=atan2[T_e(2,3),T_e(3,3)], rot_y=atan2[T_e(3,1),T_e(1,1)], rot_z=atan2[T_e(1,2),T_e(2,2)]). 1. State : 'axisangle', 'eulerp', 'eulerang_xyz', 'omega', 'omegadot' or proj_ang 2. Main_body name 3. Element number 4. Relative distance from node 1 to node 2 on element 5. Main_body name of which coordinate system is used for output. "global" can also be used.		
mbdy	statevec_new	This sensor writes out the position vector and orientation vector for a point on the structure. The orientation vector is a direction vector to which the structure is rotated and the vector length is the size of this rotation. There is a direct relation between this vector and the 3x3-orientation matrix, but it is easier to overview as each single element corresponds to a 2D projected rotation (rot_x, rot_y, rot_z). Furthermore it can write out the orientation of the local deformed c2_def coordinates system and therefore breaks the limit of only looking at element orientations. 1. Main_body name 2. Position of location outputted: 'c2def' or 'default' (default = elastic center). 3. Name of mbdy used for output coordinate system: mbdy_name or 'global' 4. State: 'elastic' or 'absolute'. Elastic means that initial location is subtracted results 5. Distance along c2_def to output location 6. Sign multiplied to output: 1.0 or -1.0 7. x-coordinate offset from center to a point where location is outputted (local c2def coo) [m] 8. y-coordinate offset from center to a point where location is outputted (local c2def coo) [m]	yes	Yes
mbdy	wind	This sensor writes out the global or relative wind velocity components for a point on a main body. The measurement point follows the structure rigid body motions and elastic deflections.	yes	Yes

Command 1	Command 2	Explanation	Only option	Label option
		This output channel can be important if the wind measurement point moves long distances during the analysis. For example floating wind turbines can move dozens meters during a simulation. 1. Main_body name 2. Element number on the main body 3. Relative distance from node 1 to node 2 on the element 4. Wind velocity measurement method: 'global' or 'relative'. Relative means the point velocity is substracted from the global wind speed 5. x-coordinate offset of the point [m] 6. y-coordinate offset of the point [m]		

This illustration shows how the sensors are placed on an element in terms of local nodes and relative distance.



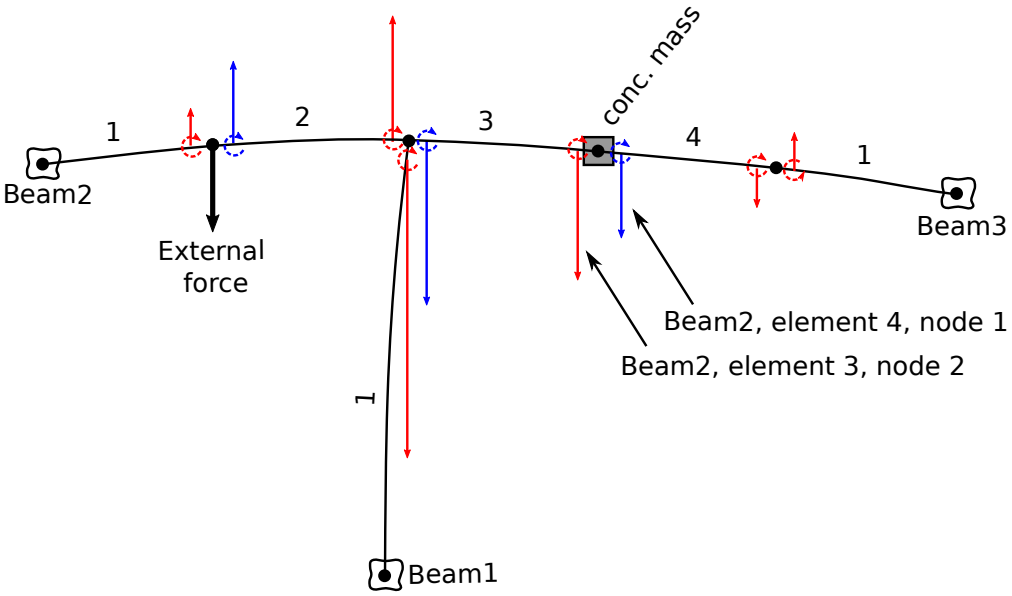


Figure 7: The "mbdy forcevec" and "mbdy momentvec" sensor definitions depend on argument 3, "node number on element", which must be 1 or 2.
For node number 1 (element start node), the sensors output the forces and moments (blue in figure) that the element and the succeeding structure (excluding concentrated masses and external forces attached to the node) applies to the preceding structure.
For node number 2 (element end node), the sensors output the forces and moments (red in figure) that the succeeding structure (including concentrated masses and external forces attached to the node) applies to the element and the preceding structure.

17.6 Constraint (constraint output commands)

17.6.1 bearing1

Command 1	Command 2	Explanation	Only option	Label option
constraint	bearing1	Bearing angle and angle velocity defined to output 1. bearing1 name 2. unit of output (1:angle [unit=rad, range -π:π], vel [rad/s]; 2:angle [unit=deg, range 0:360], vel [rpm]; 3:angle [unit=deg, range 0:360], vel [rad/s]); 4:angle [unit=deg, range -180:180], vel [rad/s]; 5:angle [unit=deg, range -180:180], vel [deg/s])	Yes	No

17.6.2 bearing2

Command 1	Command 2	Explanation	Only option	Label option
constraint	bearing2	Bearing angle and angle velocity defined to output 1. bearing2 name 2. unit of output (1:angle [unit=rad, range -π:π], vel [rad/s]; 2:angle [unit=deg, range 0:360], vel [rpm]; 3:angle [unit=deg, range 0:360], vel [rad/s]); 4:angle [unit=deg, range -180:180], vel [rad/s]; 5:angle [unit=deg, range -180:180], vel [deg/s])	Yes	No

17.6.3 bearing3

Command 1	Command 2	Explanation	Only option	Label option
constraint	bearing3	Bearing angle and angle velocity defined to output 1. bearing3 name 2. unit of output (1:angle [unit=rad, range -π:π], vel [rad/s]; 2:angle [unit=deg, range 0:360], vel [rpm]; 3:angle [unit=deg, range 0:360], vel [rad/s]); 4:angle [unit=deg, range -180:180], vel [rad/s]; 5:angle [unit=deg, range -180:180], vel [deg/s])	Yes	No

17.6.4 bearing4

Rotation angle and velocity of the two axis perpendicular to the cardan shaft torsion axis are outputted.

Command 1	Command 2	Explanation	Only option	Label option
constraint	bearing4	Bearing angle and angle velocity defined to output 1. bearing4 name 2. unit of output (1:angle [unit=rad, range -π:π], vel [rad/s]; 2:angle [unit=deg, range 0:360], vel [rpm];	Yes	No

Command 1	Command 2	Explanation	Only option	Label option
		3:angle [unit=deg, range 0:360], vel [rad/s]); 4:angle [unit=deg, range -180:180], vel [rad/s]; 5:angle [unit=deg, range -180:180], vel [deg/s])		

17.7 aero (aerodynamic related commands)

Command 1	Command 2	Explanation	Label option
aero	time	Simulation time to output. No parameters.	No
aero	azimuth	Azimuth angle of selected blade. Zero is vertical downwards. Positive clockwise around blade root y-axis. Unit [deg] 1. Blade number	No
aero	omega	Rotational speed of rotor. Unit [rad/s]. See additional explanations below table.	No
aero	vrel	Relative velocity in x-y local aerodynamic plane. Unit [m/s] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	vrel_3d	Relative velocity in x-y-z local aerodynamic space. Unit [m/s] 3. Blade number 4. Radius [m] (nearest inner calculation point is used)	No
aero	alfa	Angle of attack in x-y local aerodynamic plane. Unit [deg] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	alfadot	Pitch rate term (z-axis rotation) in local aerodynamic plane, as used for non-circulatory contributions. Unit [rad/s] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	sideslip	Side slip angle (from radial flow of BEM expansion) 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	beta	Flap deflection angle (matching the deflection specified by the flap control .dll): 1. Blade number 2. Flap number, according to the order defined in the dynstall_ateflap sub-command block.	No
aero	cl	Instantaneous lift coefficient. Unit [-] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	cd	Instantaneous drag coefficient. Unit [-]	No

Command 1	Command 2	Explanation	Label option
		1. Blade number 2. Radius [m] (nearest inner calculation point is used)	
aero	cm	Instantaneous moment coefficient. Unit [-] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	lift	Lift force at calculation point. Unit [kN/m] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	drag	Drag force at calculation point. Unit [kN/m] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	moment	Aerodynamic moment at calculation point. Unit [kNm/m] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	secforce	Aerodynamic force at calculation point. Local aero coo. Unit [kN/m] 1. Blade number 2. Dof number (1=F_x, 2=F_y, 3=F_z) 3. Radius [m] (nearest inner calculation point is used) 4. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar) Note that 4th input argument is optional (default=1)	No
aero	secmoment	Aerodynamic moment at calculation point. Local aero coo. Unit [kNm/m] 1. Blade number 2. Dof number (1=M_x, 2=M_y, 3=M_z) 3. Radius [m] (nearest inner calculation point is used)	No
aero	int_force	Integrated aerodynamic forces from tip to calculational point. NB the integration is performed around the $C_{3/4}$ location. Unit [kN] 1. Coordinates system (1=local aero coo, 2=blade ref. system, 3=global, 4=rotor polar) 2. Blade number 3. Dof number (1=F_x, 2=F_y, 3=F_z) 4. Radius [m] (nearest inner calculation point is used)	No
aero	int_moment	Integrated aerodynamic moment from tip to calculational point. NB the integration is performed around the $C_{3/4}$ location. Unit [kNm] 1. Coordinates system (1=local aero coo, 2=blade ref. system, 3=global, 4=rotor polar)	No

Command 1	Command 2	Explanation	Label option
		2. Blade number 3. Dof number (1=M_x, 2=M_y, 3=M_z) 4. Radius [m] (nearest inner calculation point is used)	
aero	torque	Integrated aerodynamic forces of all blades to rotor torsion. Unit [kNm]. No parameters. See additional explanations below table.	No
aero	thrust	Integrated aerodynamic forces of all blades to rotor thrust. Unit [kN]. No parameters	No
aero	position	Position of calculation point. Unit [m]. Please be aware that if the blade ref system is used, the orientation is in the blade coo, but the origo is still in the hub center. 1. Coordinates system (1=local aero coo, 2=blade ref. system, 3=global, 4=rotor polar) 2. Blade number 3. Dof number (1=M_x, 2=M_y, 3=M_z) 4. Radius [m] (nearest inner calculation point is used)	No
aero	power	Integrated aerodynamic forces of all blades to rotor torsion multiplied by the rotor speed. Unit [kW]. No parameters. See additional explanations below table.	No
aero	rotation	Orientation of calculation point. Unit [deg]. 1. Blade number 2. Dof number (1=θ_x, 2=θ_y, 3=θ_z) 3. Radius [m] (nearest inner calculation point is used) 4. Coordinates system (1=blade_ref. coo, 2=rotor polar coo.)	No
aero	rotation_e	Orientation of calculation point. Unit [deg]. 1. Blade number 2. Dof number (1=θ_x, 2=θ_y, 3=θ_z) 3. Radius [m] (nearest inner calculation point is used) 4. Coordinates system (1=blade_ref. coo, 2=rotor polar coo.)	No
aero	velocity	Velocity of calculation point. Unit [m/s]. 1. Coordinates system (1=local aero coo, 2=blade ref. system, 3=global, 4=rotor polar) 2. Blade number 3. Dof number (1= V_x, 2=V_y, 3=V_z) 4. Radius [m] (nearest inner calculation point is used)	No
aero	acceleration	Acceleration of calculation point. Unit [m/s2]. 1. Coordinates system (1=local aero coo, 2=blade ref. system, 3=global, 4=rotor polar) 2. Blade number 3. Dof number (1= V_x, 2=V_y, 3=V_z) 4. Radius [m] (nearest inner calculation point is used)	No

Command 1	Command 2	Explanation	Label option
aero	tors_e	Aeroelastic torsional twist minus initial static twist of a blade section. 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	windspeed	Free wind speed seen from the blade. Unit [m/s] 1. Coordinate system (1=local aero coo, 2=blade ref. system, 3=global, 4=rotor polar) 2. Blade number 3. Dof number (1= V_x, 2=V_y, 3=V_z) 4. Radius [m] (nearest inner calculation point is used)	No
aero	wsp_rotor_avg (New in 12.6.14)	Rotor average free wind speed (excluding tower top motion). Unit [m/s] 1. Coordinate system (1=global; 2=rotor with y perpendicular to the rotor plane, for zero yaw and tilt equivalent to global coordinate system) 3. Dof number (1= V_x, 2=V_y, 3=V_z)	No
aero	spinner_lidar	Sensor emulating a spinner mounted lidar 1. Measurement type (1=single point, 2=volume average) 2. Scan type (1=circular scan, 2=horizontal line (sine sweep), 3=horizontal line (linear sweep), 4=circular 2D scan) 3. Focus length [m] 4. Measurement angle [deg] 5. Scanning velocity [rev/sec] 6. Velocity fraction (2D scan) 7. Aperture radius (volume) [m] 8. Number of points in volume scan 9. Wavelength (Volumen) [m]	No
aero	induc	Local induced velocity at calculation point. Unit [m/s] 1. Coordinates system (1=local aero coo, 2=blade ref. system, 3=global, 4=rotor polar) 2. Blade number 3. Dof number (1= V_x, 2=V_y, 3=V_z) 4. Radius [m] (nearest inner calculation point is used)	No
aero	induc_sector_ct	Thrust coefficient at a position on the rotor. Unit [-] 1. Radius [m] 2. Azimuth angle (zero downwards) [deg]	No
aero	induc_sector_cq	Torque coefficient at a position on the rotor. Unit [-] 1. Radius [m] 2. Azimuth angle (zero downwards) [deg]	No
aero	induc_sector_a	Axial induction coefficient at a position on the rotor. Unit [-] 1. Radius [m] 2. Azimuth angle (zero downwards) [deg]	No

Command 1	Command 2	Explanation	Label option
aero	induc_sector_am	Tangential induction coefficient at a position on the rotor. Unit [-] 1. Radius [m] 2. Azimuth angle (zero downwards) [deg]	No
aero	induc_a_norm	Axial velocity used in normalization expression of rotor thrust coefficients. The average axial wind velocity excl. induction. Unit [m/s]. No parameters.	No
aero	induc_am_norm	Tangential velocity used in normalization expression of torque coefficient. Average tangential velocity at a given radius. Unit [m/s]. 1. Radius [m]	No
aero	inflow_angle	Angle of attack + rotation angle of profile related to polar coordinates (not pitching). Unit [deg] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	dcldalfa	Gradient $dCl/d\alpha$. Unit [deg^{-1}] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	dcddalfa	Gradient $dCd/d\alpha$. Unit [deg^{-1}] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	gamma	Circulation strength at calculation point. Unit [m2/s] 1. Blade number 2. Radius [m] (nearest inner calculation point is used)	No
aero	lambda	Tip speed ratio, Unit [−]	No
aero	windspeed_boom	Free wind speed seen by a boom mounted on a blade section. Coordinate system used “blade ref. system”. Unit [m/s]. 1. Blade number 2. Radius [m] (nearest inner calculation point is used) 3. Boom-length X, measured from half chord point positive towards LE [m] 4. Boom-length Y, measured from half chord point positive towards pressureside [m]	No
aero	actuatordiskload	Actuator disk load provide normalized load export for the Actuator Disk Model. 1. DOF (1=Ft, 2=Fa, 3=Fr) 2. Radius [m] (nearest inner calculation point is used)	No
aero	grid_radius_nd	Aerodynamic calculation point non-dim radius r/R 1. Number of radial stations outputted (should normally correspond to number of aerodynamic calculation points on a blade)	No

Command 1	Command 2	Explanation	Label option
aero	vawt_induc_x	Induction for a VAWT outputted in tangential polar coordinates 1. disc number 2. azimuth number	No
aero	vawt_induc_y	Induction for a VAWT outputted in radial polar coordinates 1. disc number 2. azimuth number	No
aero	nacelle_lidar	Model of a single-beam CW nacelle-mounted lidar. Laser beam is approximated by a line with a Lorentzian shaped weighting function. Outputs are: 1. Line-of-sight velocity [m/s] 2. Doppler spectrum variance [m2/s2] 3. Global x position of focus point 4. Global y position of focus point 5. Global z position of focus point Input are: 1. Mounting distance from rotor center in global x coordinates [m] 2. Mounting distance from rotor center in global y coordinates [m] 3. Mounting distance from rotor center in global z coordinates [m] 4. Half-cone opening angle of beam [deg] 5. Azimuth angle of beam measured (clockwise as seen from turbine) from vertical up postion [deg] 6. Focus length measured from rotor center (along the beam) [m] 7. Rayleigh length of beam (Gamma) [m] 8. half-width of integration interval over probe volume [Gamma] 9. Number of integration points (recommendation: 100) [−] 10. Beam identifier number [−]	No
aero	effective_wind_speed	Estimation of rotor effective wind speed as a (weighted) average of longitudinal wind speeds within the rotor area: $v_{eff} = \sqrt[n]{\frac{\int_0^{2\pi}\int_0^R v_u^n(r,\varphi)w(r,\varphi)rdrd\theta}{\int_0^{2\pi}\int_0^R w(r,\varphi)rdrd\theta}}$. Unit [m/s] Inputs are: 1. Number of blades [-] 2. Rotor radius (R) [m] 3. Tip speed ratio at rated wind speed [-] (only used when input 9 is equal to 3) 4. Exclusion of root part [R] (only used when input 9 is equal to 3) 5. Normal measurement distance from rotor plane [m] 6. Width of turbulence box [m] (use the values from the turbulence box block)	No

Command 1	Command 2	Explanation	Label option
		7. Number of integration points along width/height [-] 8. power to weight wind speed with (n) [-] 9. Weighting method: 1: arithmetic mean 2: dCpdr weight w/o losses 3: dCpdr weight with (tip and root) losses 10. Optimum axial induction factor (only used when input 9 is equal to 2 or 3)	

Multi-rotor simulation

For multi-rotor simulation, three commands are used: - Command 1: aero_mr - Command 2: as command 2 from above table - Command 3: name of rotor given in main command block 'aero'

Additional explanations on aero omega, aero torque and aero power

aero omega Gives the 'aerodynamic' rotor speed, which is an average rotational speed of the blades. In general, it is very similar to the shaft rotor speed, but it may be different especially in case of vibrations in a drivetrain mode. In this case, the blades move edgewise collectively, which means that the 'aero' rotor speed will oscillate around the shafts rotational speed with the frequency of the respective drivetrain mode.

aero torque Aerodynamic torque from integration of the sectional torque over all aerodynamic blade sections. Depending on the aerodynamic (number of aerosections) and structural (number of sections in the c2_def) this torque may differ by up to a few percent from the computed mechanical torque at the shaft. Differences may indicate that a refinement of the aerodynamic and/or structural discretization of the blades is necessary. Further, the aerodynamic torque may be very different from the shaft torque if the rotor speed is not constant. For example in case of an increasing rotor speed, such as due to a gust, a part of the aerodynamic torque will accelerate the rotor and will thus not be felt at the shaft.

aero power This aerodynamic power is computed as the product of the aerodynamic rotor speed and aerodynamic torque above. Therefore it will be different from a mechanical power (shaft moment multiplied by shaft rotational speed) in any of the cases described above: vibrations of the drivetrain including collective edgewise blade vibrations, insufficient aerodynamic or structural discretization and non-constant rotor speed.

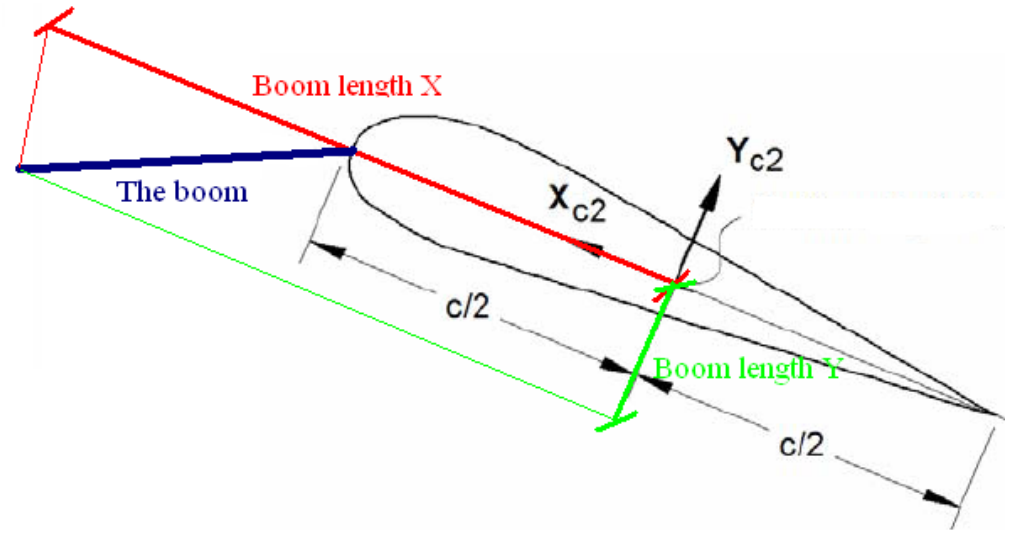


Figure 8: Illustration of the boom coordinates used by the "windspeed_boom" command.

17.8 wind (wind output commands)

Command 1	Command 2	Explanation	Only option	Label option
wind	free_wind	Wind vector V_x, V_y, V_z, (wind as if the turbine didn't exist). 1. Coordinate system (1=global, 2=non rotating rotor coordinates (x always horizontal, y always out-of-plane)) 2. x-pos (global coo) 3. y-pos (global coo) 4. z-pos (global coo)	Yes	Yes
wind	free_wind_center_pos0	Wind vector V_x, V_y, V_z, (wind as if the turbine didn't exist). 1. Coordinate system (1=global, 2=non rotating rotor coordinates (x always horizontal, y always out-of-plane)) _center_pos0	Yes	Yes
wind	free_wind_hor	Horizontal wind component velocity [m/s] and direction [deg] defined to output. Dir=0 when wind equals y-dir. 1. Coordinate system (1=global, 2=non rotating rotor coordinates (x always horizontal, y always out-of-plane)) 2. x-pos (global coo) 3. y-pos (global coo) 4. z-pos (global coo)	Yes	Yes
wind	free_wind_-hor_center_pos0	Horizontal wind component velocity [m/s] and direction [deg] defined to output. Dir=0 when wind equals y-dir. 1. Coordinate system (1=global, 2=non rotating rotor coordinates (x always horizontal, y always out-of-plane))	Yes	Yes
wind	free_wind_shadow	As sensor "free_wind", but with tower shadow included.	Yes	Yes

Command 1	Command 2	Explanation	Only option	Label option
		1. Coordinate system (1=global, 2=non rotating rotor coordinates (x always horizontal, y always out-of-plane)) 2. x-pos (global coo) 3. y-pos (global coo) z-pos (global coo)		

17.9 wind_wake (wind wake output commands)

Command 1	Command 2	Explanation	Only option	Label option
wind_wake	wake_pos	Position of the wake deficit center after the meandering proces to the downstream end position. x,y and z position is written in meteorological coordinates $(x, y, z)_M = (u, v, w)$ with origo in the position defined with center_pos0 in the general wind commands. 1. wake source number	Yes	Yes

17.10 dll (DLL output commands)

Command 1	Command 2	Explanation	Label option
dll	inpvec	Value from DLL input vector is defined to output 1. DLL number 2. array index number	yes
dll	outvec	Value from DLL output vector is defined to output 1. DLL number 2. array index number	yes
dll	hawc_dll	Special output commands for the “hawc_dll” format. With this command the dll name can be used in the output definitions 1. string. Reference name of the dll given in the begin – end hawc_dll input definitions. 2. string. “outvec” or “inpvec” can be used. Same definition as previously written above. 3. Channel number in the in or out going array.	yes
dll	type2_dll	Special output commands for the “type2_dll” format. With this command the dll name can be used in the output definitions 1. string. Reference name of the dll given in the begin – end hawc_dll input definitions. 2. string. “outvec” or “inpvec” can be used. Same definition as previously written above. 3. Channel number in the in or out going array.	yes
dll	sensor_id	Name of sensor_id defined for other output sensor 1. Sensor number if sensor id refers to a vector	

17.11 hydro (hydrodynamic output commands)

Command 1	Command 2	Explanation	Only option	Label option
hydro	water_surface	Water surface level at a given horizontal location is defined to output (global coordinates). Unit [m] 1. x-pos 2. y-pos	No	No
hydro	water_vel_acc	Water velocity V_x, V_y, V_z, and acceleration A_x, A_y, A_z vectors defined to output. Unit [m/s] and [m/s2]. 1. x-pos 2. y-pos 3. z-pos	Yes	No
hydro	water_pressure	Dynamic water pressure from Bernoulli's equation, in a given hydro element calculation point. Unit [MPa]. 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	fm	Radial inertia force (FK + hydro mass) F_x, F_y, F_z contribution from Morisons formula in a given calculation point. Unit [kN/m]. Note: added mass is by default computed in the right hand side of the EOM, so the hydro mass term is only accounting for the fluid acceleration term. 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	fd	Radial drag force F_x, F_y, F_z contribution from Morisons formula in a given calculation point. Unit [kN/m] 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	fb	Buoyancy force (distributed along element) F_x, F_y, F_z contribution in a given calculation point. Unit [kN/m] 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	mb	Buoyancy moment (distributed along element) M_x, M_y, M_z contribution in a given calculation point. Unit [kNm/m] 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No

Command 1	Command 2	Explanation	Only option	Label option
hydro	cfb	Concentrated axial buoyancy force F_x, F_y, F_z contribution in a given calculation point. Unit [kN]. Note: in case of auto hydro sections =0, if the specified radius is not a hydro or structural node, the sensor will output the nearest node information. 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	cmb	Concentrated buoyancy moment M_x, M_y, M_z contribution in a given calculation point. Unit [kNm]. Note: in case of auto hydro sections =0, if the specified radius is not a hydro or structural node, the sensor will output the nearest node information. 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	cfm	Concentrated force from axial hydro mass F_x, F_y, F_z contribution in a given calculation point. Unit [kN]. Added mass is by default computed in the right hand side of the EOM, so the hydro mass term is only accounting for the fluid acceleration term, and computed as $\rho V_{ref} C_{a,axial} water\_acc$, with V_{ref} taken as half volume of sphere defined by the local width (of the specified element radius) as diameter. Note: in case of auto hydro sections =0, if the specified radius is not a hydro or structural node, the sensor will output the nearest node information. 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	cfdrag	Concentrated force from axial damping F_x, F_y, F_z contribution in a given calculation point. Unit [kN]. Note: in case of auto hydro sections =0, if the specified radius is not a hydro or structural node, the sensor will output the nearest node information. 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	fdyn	Dynamic wave pressure force (distributed) F_x, F_y, F_z contribution in a given calculation point. Unit [kN/m] 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No

Command 1	Command 2	Explanation	Only option	Label option
hydro	cfdyn	Concentrated axial dynamic wave pressure force F_x, F_y, F_z contribution in a given calculation point. Unit [kN]. Note: in case of auto hydro sections =0, if the specified radius is not a hydro or structural node, the sensor will output the nearest node information. 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	secfrc	Total hydro distributed force F_x, F_y, F_z contribution in a given calculation point. Unit [kN/m] 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	secmom	Total hydro distributed moment M_x, M_y, M_z contribution in a given calculation point. Unit [kNm/m] 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No
hydro	cfrc	Total hydro axial concentrated force F_x, F_y, F_z contribution in a given calculation point. Unit [kN]. Note: in case of auto hydro sections =0, if the specified radius is not a hydro or structural node, the sensor will output the nearest node information. 1. hydro element number 2. radius (in [m], axial distance from 1st node) 3. coordinate system (1=global, 2=local hydro sec coo)	No	No

17.12 general (general output commands)

Command 1	Command 2	Explanation	Label option
general	constant	A constant value is send to output 1. constant value	No
general	step	A step function is created. This function changes from f_0 to f_1 at time t_0. 1. t_0 [sec] 2. f_0 3. f_1	No
general	step2	A step function is created. This function changes from f_0 to f_1 between time t_0 and t_1 using linear interpolation. 1. t_0 [sec] 2. t_1 [sec] 3. f_0 4. f_1	No

Command 1	Command 2	Explanation	Label option
general	step3	A step function is created. This function changes from f_0 to f_1 between time t_0 and t_1 using a continous sinus2 interpolation function. 1. t_0 [sec] 2. t_1 [sec] 3. f_0 4. f_1	No
general	time	The time is send to output. No parameters	No
general	deltat	The time increment is send to output. No parameters	No
general	harmonic	A harmonic function is send to output $F(t) = A\sin(2\pi f_0 t) + k$ 1. A 2. f_0 3. k	No
general	harmonic2	A harmonic function is send to output $F(t) = \begin{cases} 0 & t < t_0 \\ A\sin(2\pi f_0(t - t_0)) + k & t_0 \le t \le t_1 \\ 0 & t > t_1 \end{cases}$ 1. A 2. f_0 3. k 4. t_0 5. t_1	No
general	stairs	A series of steps resulting in a staircase signal is created. 1. t_0 time for first step change [s] 2. f_0 start value of function 3. Step size 4. Step duration [s] 5. Number of steps	No
general	status	A status flag (mainly for controller purpose) is written. A first time step and first iteration the output value is 0. During the rest of the simulation the value is 1 until last time step where the value is -1.	No
general	random	A randon (uniform distribution) is written 1. lower limit 2. upper limit 3. seed number	No
general	impulse	A step function which return to zero after a certain duration 1. t_0 time for impulse start [s] 2. Impulse duration [s] 3. f_0 impulse level	No
general	sensor_id	Sensor name. 1. Sensor number if sensor_id refers to a vector	No

18 Output_at_time (output at a given time)

This command is especially usefull if a snapshot of loads or other properties are required at a specific time. This is mostly used for writing calculated aerodynamic properties as function of blade location. The command block can be repeated as many times as needed (e.g. if outputs at more than one time is needed)

The command must be written with the following syntax

```
output_at_time keyword time
```

where *keyword* is the name of an output subcommand. Currently only the subcommand *aero* is supported. The last command word time is the time in seconds from simulation start to which the output are written.

18.1 aero (aerodynamic output commands)

The first line in the output_at block must be the information regarding which file the outputs are written (the filename command listed in the table below)

Command 1	Explanation	Label option
filename	Filename incl. relative path to output file (example ./output/output_at.dat). 1. filename	No
alfa	Angle of attack [deg]. 1. Blade number	No
alfadot	Pitch rate term (z-axis rotation) in local aerodynamic plane, as used for non-circulatory contributions. Unit [rad/s]. 1. Blade number	No
vrel	Relative velocity [m/s] 1. Blade number	No
cl	Lift coefficient [-] 1. Blade number	No
cd	Drag coefficient [-] 1. Blade number	No
cm	Moment coefficient [-] 1. Blade number	No
lift	Lift force L [N/m] 1. Blade number	No
drag	Drag force D [N/m] 1. Blade number	No
moment	Moment force M [Nm/m] 1. Blade number	No
secforce	Aerodynamic forces [kN/m] 1. Blade number 2. DOF number (1=x,2=y,3=z) 3. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar)	No
secmoment	Aerodynamic moments [kNm/m] 1. Blade number 2. DOF number (1=x,2=y,3=z) 3. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar)	No
int_force	Aerodynamic forces integrated from tip to given radius [kN] 1. Blade number	No

Command 1	Explanation	Label option
	2. DOF number (1=x,2=y,3=z) 3. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar)	
int_moment	Aerodynamic moment integrated from tip to given radius [kNm] 1. Blade number 2. DOF number (1=x,2=y,3=z) 3. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar)	No
inipos	Initial position of sections in blade coo [m] 1. Blade number 2. DOF number (1=x,2=y,3=z)	No
position	Actual position of section [m] 1. Blade number 2. DOF number (1=x,2=y,3=z) 3. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar)	No
velocity	Actual velocity of section [m/s] 1. Blade number 2. DOF number (1=x,2=y,3=z) 3. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar)	No
acceleration	Actual acceleration of section [m/s] 1. Blade number 2. DOF number (1=x,2=y,3=z) 3. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar)	No
ct_local	Local thrust coefficient [-]. Calculated based on the expression $C_t = \frac{F_{axial} B}{1/2 \rho 2 \pi r V_{inf}^2}$ 1. Blade number	No
cq_local	Local tangential force coefficient [-]. Calculated based on the expression $C_q = \frac{F_{tan} B}{1/2 \rho 2 \pi r V_{inf}^2}$ 1. Blade number	No
chord	Chord length [m] 1. Blade number	No
induc	Induced velocity [m/s] 1. Blade number 2. DOF number (1=x,2=y,3=z) 3. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar)	No
windspeed	Free windspeed (without induction but incl. tower shadow effects if used) [m/s] 1. Blade number 2. DOF number (1=x,2=y,3=z) 3. Coordinate system (1=aero, 2=blade, 3=global, 4=rotor polar)	No
inflow_angle	Angle of attack + rotation angle of profile related to polar coordinates (not pitching). Unit [deg] 1. Blade number	No
dcldalfa	Gradient $dCl/d\alpha$. Unit [deg^{-1}] 1. Blade number	No
dcddalfa	Gradient $dCd/d\alpha$. Unit [deg^{-1}] 1. Blade number	No
tiploss_f	The local tiploss factor (product of Prandtl and custom tiploss factor) 1. Blade number	No

An example of an output_at_time command block could be:

```
begin output_at_time aero 100;
```

```
  filename ./output_at_time ;
  alfa 1;
end output_at_time;
```

19 Input file encryption

19.1 DLL format

From version 11.6 it is possible to attach a DLL from where the blade data can be extracted. In doing so, the user is not able to inspect the blade input data from a readable text file. This approach requires a Fortran Intel compiler setup (when using a DTU provided template) from the data owner in order to compile the blade data into a DLL. The user refers to the external blade data in DLL format in the `new_htc_structure` and `aero` sections using the `external_bladedata_dll` command (as described earlier in this manual in the relevant sections).

19.2 Encrypted binary format

Starting from version 12.8, a new method to hide confidential input data is offered. This approach allows the data owner to encrypt normal HAWC2 input data files into files that only HAWC2 is able to read. The encryption is performed by an executable, which is provided by the HAWC2 support, hawc2@windenergy.dtu.dk, upon request. Note, the executable is only needed to generate the encrypted input data, i.e. no additional tools are needed by the end user to use the encrypted data files.

19.2.1 How to encrypt data files

Data files are encrypted from command line as follows:

```
> EncryptDataFile.exe mydatafile.dat
```

This will generate the encrypted file `mydatafile.dat.enc`

19.2.2 Using encrypted data files

To use the data file with HAWC2, simply replace the data file name in the htc file.

HAWC2 will recognize and decrypt `*.enc` files for the following types of files:

- Timoschenko input (`new_htc_structure/main_body/timoschenko_input/filename`)
- Aero dynamic layout (`aero/ae_filename`)
- Profile coefficients (`aero/pc_filename`)
- HTC sub files (`continue_in_file`). NOTE: No output options are disabled. The user may be able to output your confidential data via output sensors or other output options

19.2.3 Disabled output

When the aerodynamic layout or profile coefficients are encrypted, the following output options are unavailable:

- output:
  - cl, cd, cm
  - lift, drag, moment
  - induc

    - induc_sector_ct, induc_sector_cq
    - induc_sector_a, induc_sector_am
    - induc_a_norm
    - induc_am_norm
    - dcldalfa, dcddalfa
    - secforce, secmoment
    - int_force, int_moment
    - vawt_induc_x, vawt_induc_y
    - grid_all_ct, grid_all_induc_u
- Output_at_time:
    - cl, cd, cm
    - lift, drag, moment
    - ct_local, cq_local
    - chord, twist
    - dcldalfa, dcddalfa
    - induc
    - secforce, secmoment
    - int_force, int_moment
    - vawt_qr, vawt_qt, vawt_qn
    - vawt_induc_x, vawt_induc_y
- output_profile_coef_filename

When the Timoschenko input file is encrypted, the following output options are unavailable:

- new_htc_structure
    - beam_output_file_name
    - body_output_file_name
    - struct_inertia_output_file_name
    - body_matrix_output
    - element_matrix_output

References

[1] H. Aa. Madsen, G. C. Larsen, T. J. Larsen, N. Troldborg, and R. Mikkelsen. Calibration and Validation of the Dynamic Wake Meandering Model for Implementation in an Aeroelastic Code. *Journal of Solar Energy Engineering*, 132(4), 10 2010.

[2] T. J. Larsen, H. Aa. Madsen, G. C. Larsen, and K. S. Hansen. Validation of the dynamic wake meander model for loads and power production in the egmond aan zee wind farm. *Wind Energy*, 16(4):605–624, 2013.

[3] Morten H Hansen, Mac Gaunaa, and Helge Aa Madsen. *A Beddoes-Leishman type dynamic stall model in state-space and indicial formulations*. Risø-R-1354, Roskilde, Denmark, 2004.

[4] Georg R Pirrung and Mac Gaunaa. *Dynamic stall model modifications to improve the modeling of vertical axis wind turbines*. DTU Wind Energy E-0171, Roskilde, Denmark, 2018.

[5] H. Aa. Madsen, T. J. Larsen, G. R. Pirrung, A. Li, and F. Zahle. Implementation of the blade element momentum model on a polar grid and its aeroelastic load impact. *Wind Energy Science*, 5(1):1–27, 2020.

[6] G. R. Pirrung, H. Aa. Madsen, T. Kim, and J. Heinz. A coupled near and far wake model for wind turbine aerodynamics. *Wind Energy*, 2016.

[7] G. R. Pirrung, V. Riziotis, H. Aa. Madsen, M. Hansen, and T. Kim. Comparison of a coupled near- and far-wake model with a free-wake vortex code. *Wind Energy Science*, 2(1):15–33, 2017.

A Example of main input file

```
begin Simulation;
  time_stop    100;
  solvertype   2 ;    (sparse newmark)
  on_no_convergence continue ;
  logfile ./log/oc3_monopile_phase_1.log ;
  animation ./animation/oc3_monopile_phase_1.dat;
;
  begin newmark;
    deltat    0.02;
  end newmark;
end simulation;
;
begin new_htc_structure;
  ; Optional - Calculated beam properties of the bodies are written to file:
  beam_output_file_name  ./log/oc3_monopile_phase_1_beam.dat;
  ; Optional - Body initial position and orientation are written to file:
  body_output_file_name  ./log/oc3_monopile_phase_1_body.dat;
;  body_eigenanalysis_file_name ./eigenfrq/oc3_monopile_phase_1_body_eigen.dat;
;  structure_eigenanalysis_file_name ./eigenfrq/oc3_monopile_phase_1_strc_eigen.dat ;
;---------------------------------------------------------------------------------------
;
  begin main_body;          monopile 30m
    name        monopile ;
    type        timoschenko ;
    nbodies     1 ;
    node_distribution     c2_def ;
    damping   4.5E-02 4.5E-02 8.0E-01 1.2E-03 1.2E-03 4.5E-04 ;
    begin timoschenko_input;
      filename ./data/Monopile.txt ;
      set 1 1 ;                set subset   1=flexible,2=stiff
    end timoschenko_input;
    begin c2_def;              Definition of centerline (main_body coordinates)
      nsec 7;
      sec 1 0.0 0.0 0.0    0.0 ;  x,y,z,twist      Mudline
      sec 2 0.0 0.0 -0.1    0.0 ;  x,y,z,twist
      sec 3 0.0 0.0 -10.0    0.0 ;  x,y,z,twist    50% between mudline and MSL
      sec 4 0.0 0.0 -15.0    0.0 ;  x,y,z,twist
      sec 5 0.0 0.0 -20.0    0.0 ;  x,y,z,twist    MWL
      sec 6 0.0 0.0 -25.0  0.0 ;
      sec 7 0.0 0.0 -30.0 0.0 ;                    Monopile flange
    end c2_def ;
  end main_body;
;
  begin main_body;         tower 80m
    name        tower ;
    type        timoschenko ;
    nbodies     1 ;
    node_distribution     c2_def ;
     damping_posdef   6.456E-4 6.45E-4 1.25E-3 1.4E-3 1.4E-3 1.25E-3 ;
     ;damping_posdef Mx My Mz Kx Ky Kz , M´s raises overall level, K´s raises high freguency level
     ;
    begin timoschenko_input;
```

```
      filename ./data/NREL_5MW_st.txt ;
      set 1 1 ;
    end timoschenko_input;
    begin c2_def;              Definition of centerline (main_body coordinates)
      nsec 8;
      sec 1 0.0 0.0 0.0    0.0 ;  x,y,z,twist
      sec 2 0.0 0.0 -10.0  0.0 ;
      sec 3 0.0 0.0 -20.0  0.0 ;
      sec 4 0.0 0.0 -30.0  0.0 ;
      sec 5 0.0 0.0 -40.0  0.0 ;
      sec 6 0.0 0.0 -50.0  0.0 ;
      sec 7 0.0 0.0 -60.0 0.0 ;
      sec 8 0.0 0.0 -77.6 0.0 ;
     end c2_def ;
    end main_body;
;
  begin main_body;
    name        towertop ;
    type        timoschenko ;
    nbodies     1 ;
    node_distribution     c2_def ;
;    damping_posdef   9.025E-06 9.025E-06 8.0E-05 8.3E-06 8.3E-06 8.5E-05 ;
    damping  2.50E-04  1.40E-04  2.00E-03  3.00E-05  3.00E-05  2.00E-04  ;
    ;
    ;Nacelle mass and inertia:
    concentrated_mass 2 0.0  1.9 0.21256 2.4E5 1741490.0  1.7E5 1741490.0 ;
begin timoschenko_input;
      filename ./data/NREL_5MW_st.txt ;
      set 2 1 ;
    end timoschenko_input;
    begin c2_def;              Definition of centerline (main_body coordinates)
      nsec 2;
      sec 1 0.0 0.0 0.0       0.0 ; x,y,z,twist
      sec 2 0.0 0.0 -1.96256  0.0 ;
    end c2_def ;
  end main_body;
;
  begin main_body;
    name        shaft ;
    type        timoschenko ;
    nbodies     1 ;
    node_distribution     c2_def ;
;   damping_posdef  7.00E-3  7.00E-03  7.00E-02  3.48E-04  3.48E-04  1.156E-03 ;
   damping_posdef  7.00E-3  7.00E-03  7.00E-02  6.5E-04  6.5E-04  1.84E-02 ;
    concentrated_mass 1 0.0 0.0 0.0 0.0 0.0 0.0 5025497.444 ;generator equivalent slow shaft
    concentrated_mass 5 0.0 0.0 0.0 56780 0.0 0.0 115926 ; hub mass and inertia;
begin timoschenko_input;
      filename ./data/NREL_5MW_st.txt ;
      set 3 1 ;
    end timoschenko_input;
    begin c2_def;              Definition of centerline (main_body coordinates)
      nsec 5;
      sec 1 0.0 0.0 0.0     0.0 ; Tower top x,y,z,twist
      sec 2 0.0 0.0 1.0     0.0 ;
```

```
      sec 3 0.0 0.0 2.0     0.0 ;
      sec 4 0.0 0.0 3.1071  0.0 ; Main bearing
      sec 5 0.0 0.0 5.0191  0.0 ; Rotor centre
    end c2_def ;
  end main_body;
;
  begin main_body;
    name        hub1 ;
    type        timoschenko ;
    nbodies     1 ;
    node_distribution     c2_def ;
    damping_posdef  2.00E-05  2.00E-05  2.00E-04  3.00E-06  3.00E-06  2.00E-05;
begin timoschenko_input;
      filename ./data/NREL_5MW_st.txt ;
      set 4 1 ;
    end timoschenko_input;
    begin c2_def;              Definition of centerline (main_body coordinates)
      nsec 2;
      sec 1 0.0 0.0 0.0     0.0 ; x,y,z,twist
      sec 2 0.0 0.0 1.5   0.0 ;
    end c2_def ;
  end main_body;
;
  begin main_body;
    name           hub2 ;
    copy_main_body hub1;
  end main_body;
;
  begin main_body;
    name           hub3 ;
    copy_main_body hub1 ;
  end main_body;
;
  begin main_body;
    name        blade1 ;
    type        timoschenko ;
    nbodies     9 ;
    node_distribution    c2_def;
;   damping   3.5e-2 5.5e-4 5.0e-4 3.0e-4 0.5e-3 5.5e-3  ;
    damping_posdef   1.16e-4 5.75e-5 6.1e-6 6.5e-4 5.1e-4 6.4e-4 ;
   begin timoschenko_input ;
      filename ./data/NREL_5MW_st.txt ;
      set 5 1 ;                set subset
    end timoschenko_input;
    begin c2_def;              Definition of centerline (main_body coordinates)
      nsec 19 ;
sec 1  0.0000 0.0000 0.000 0.000 ; x.y.z. twist
sec 2 -0.0041 0.0010 1.367 -13.308 ;
sec 3 -0.1058 0.0250 4.100 -13.308 ;
sec 4 -0.2502 0.0592 6.833 -13.308 ;
sec 5 -0.4594 0.1087 10.250 -13.308 ;
sec 6 -0.5699 0.1157 14.350 -11.480 ;
sec 7 -0.5485 0.0983 18.450 -10.162 ;
sec 8 -0.5246 0.0832 22.550 -9.011 ;
```

```
sec 9 -0.4962 0.0679 26.650 -7.795 ;
sec 10 -0.4654 0.0534 30.750 -6.544 ; 50% blade radius
sec 11 -0.4358 0.0409 34.850 -5.361 ;
sec 12 -0.4059 0.0297 38.950 -4.188 ;
sec 13 -0.3757 0.0205 43.050 -3.125 ;
sec 14 -0.3452 0.0140 47.150 -2.319 ;
sec 15 -0.3146 0.0084 51.250 -1.526 ;
sec 16 -0.2891 0.0044 54.667 -0.863 ;
sec 17 -0.2607 0.0017 57.400 -0.370 ;
sec 18 -0.1774 0.0003 60.133 -0.106 ;
sec 19 -0.1201 0.0000 61.500 -0.000 ;
   end c2_def ;
  end main_body;
;
  begin main_body;
    name           blade2 ;
    copy_main_body blade1;
  end main_body;
;
  begin main_body;
    name           blade3 ;
    copy_main_body blade1 ;
  end main_body;
;-------------------------------------------------------------------------------
;
  begin orientation;
    begin base;
      body   monopile;
      inipos        0.0 0.0 20.0 ;         initial position of node 1
      body_eulerang 0.0 0.0 0.0;
    end base;
;
    begin relative;
      body1  monopile last;         indtil videre antages der internt i programmet at der
    ;                               altid kobles mellen sidste knude body1 og første
    ;                               knude body 2
      body2  tower 1;
      body2_eulerang 0.0 0.0 0.0;
    end relative;
;
    begin relative;
      body1  tower last;
      body2  towertop 1;
      body2_eulerang 0.0 0.0 0.0;
    end relative;
;
    begin relative;
      body1  towertop last;
      body2  shaft 1;
      body2_eulerang 90.0 0.0 0.0;
      body2_eulerang 5.0 0.0 0.0;    5 deg tilt angle
      ;body initial rotation velocity x.y.z.angle velocity[rad/s]  (body 2 coordinates):
      body2_ini_rotvec_d1 0.0 0.0 -1.0 0.5 ;
    end relative;
```

```
;
    begin relative;
      body1  shaft last;
      body2  hub1 1;
      body2_eulerang -90.0 0.0 0.0;
      body2_eulerang 0.0 180.0 0.0;
      body2_eulerang 2.5 0.0 0.0;      2.5deg cone angle
    end relative;
;
    begin relative;
      body1  shaft last;
      body2  hub2 1;
      body2_eulerang -90.0 0.0 0.0;
      body2_eulerang 0.0 60.0 0.0;
      body2_eulerang 2.5 0.0 0.0;      2.5deg cone angle
    end relative;
;
    begin relative;
      body1  shaft last;
      body2  hub3 1;
      body2_eulerang -90.0 0.0 0.0;
      body2_eulerang 0.0 -60.0 0.0;
      body2_eulerang 2.5 0.0 0.0;      2.5deg cone angle
    end relative;
;
    begin relative;
      body1  hub1 last;
      body2  blade1 1;
      body2_eulerang 0.0 0.0 0;
    end relative;
;
    begin relative;
      body1  hub2 last;
      body2  blade2 1;
      body2_eulerang 0.0 0.0 0.0;
    end relative;
;
    begin relative;
      body1  hub3 last;
      body2  blade3 1;
      body2_eulerang 0.0 0.0 0.0;
    end relative;
;
  end orientation;
;------------------------------------------------------------------------
begin constraint;
;
    begin fix0;  fixed to ground in translation and rotation of node 1
      body monopile;
    end fix0;
;
    begin fix1;  fixed relative to other body in translation and rotation
      body1 monopile last;
      body2 tower 1;
```

```
    end fix1;
;
    begin fix1;
      body1 tower last ;
      body2 towertop 1;
    end fix1;
;
    begin bearing1;                        free bearing
     name  shaft_rot;
      body1 towertop last;
      body2 shaft 1;
      bearing_vector 2 0.0 0.0 -1.0;        x=coo (0=global.1=body1.2=body2) vector in body2
      ;                                     coordinates where the free rotation is present
   end bearing1;
;
     begin fix1;
       body1 shaft last ;
       body2 hub1 1;
     end fix1;
;
     begin fix1;
       body1 shaft last ;
       body2 hub2 1;
     end fix1;
;
     begin fix1;
       body1 shaft last ;
       body2 hub3 1;
    end fix1;
;
    begin bearing2;
      name pitch1;
      body1 hub1 last;
     body2 blade1 1;
bearing_vector 2 0.0 0.0 -1.0;
   end bearing2;
;
    begin bearing2;
      name pitch2;
      body1 hub2 last;
      body2 blade2 1;
bearing_vector 2 0.0 0.0 -1.0;
    end bearing2;
;
    begin bearing2;
      name pitch3;
      body1 hub3 last;
      body2 blade3 1;
bearing_vector 2 0.0 0.0 -1.0;
    end bearing2;
end constraint;
;
end new_htc_structure;
;---------------------------------------------------------------------
```

```
begin wind ;
  density                 1.25;
  wsp                     8  ;
  horizontal_input        1;
  windfield_rotations     0.0  0.0  0.0 ;    yaw, tilt, rotation
  center_pos0             0.0 0.0 -90.00;      hub_height
  shear_format            3 0.12;
  turb_format             1     ;  0=none, 1=mann,2=flex
  tower_shadow_method     1;
  tint                    0.06 ;
  scale_time_start        200;
  wind_ramp_factor   0.0 200 0.5 1.0 ;
  ;------------------------------------------------------------------
  begin tower_shadow_potential;
    tower_offset 0.0;
    nsec  2;
    radius      0.0   2.10;
    radius     -68.10 1.15;
  end tower_shadow_potential;
  ;------------------------------------------------------------------
  ; This next part is only to be include in case of wake effects being studied
  begin wakes;
    nsource 35;
    source_pos 2548 -2900 -90 ;
    source_pos 2123 -2417 -90 ;
    source_pos 1706 -1942 -90 ;
    source_pos 1281 -1458 -90 ;
    source_pos 857 975 -90 ; WT5
    source_pos 432 491 -90 ; WT6
    source_pos -425 -484 -90 ; WT8
    source_pos -850 -968 -90 ; WT9
    source_pos -1267 1458 -90 ;
    source_pos -1700 1935 -90 ;
    source_pos -2125 2419 -90 ;
    source_pos 3556 -2533 -90 ;
    source_pos 3131 -2049 -90 ;
    source_pos 2706 -1565 -90 ;
    source_pos 2281 1081 -90 ; WT16
    source_pos 1602 308 -90 ; WT17
    source_pos 1176 -176 -90 ; WT18
    source_pos 751 -660 -90 ; WT19
    source_pos 326 -1144 -90 ; WT20
    source_pos -99 -1627 -90 ; WT21
    source_pos 3915 -1427 -90 ;
    source_pos 3486 -943 -90 ;
    source_pos 3062 -455 -90 ;
    source_pos 2405 -292 -90 ; WT25
    source_pos 1927 -836 -90 ; WT26
    source_pos 1502 -1319 -90 ; WT27
    source_pos 1077 -1803 -90 ; WT28
    source_pos 652 -2287 -90 ; WT29
    source_pos 4235 -283 -90 ;
    source_pos 3813 205 -90 ;
    source_pos 3163 944 -90 ;
```

```
    source_pos 2679 1495 -90 ;
    source_pos 2254 1979 -90 ;
    source_pos 1829 2463 -90 ;
    source_pos 1404 2947 -90 ;
    op_data      1.4252392 2 ; 1.8 -23.1 ;1.87 0.0 rad/sec, pitch [grader] opstrøms;
    ble_parameters 0.10 0.008 0;
    begin mann_meanderturb ;
      create_turb_parameters 33.6 1 3.7 508 0.0 ;      L, alfaeps,gamma,seed, highfrq compensation
      filename_v    ./free_sector_monopile/wake-meander/wake_meand_turb_wsp8_s508_t1800v.bin ;
      filename_w    ./free_sector_monopile/wake-meander/wake_meand_turb_wsp8_s508_t1800w.bin ;
      box_dim_u   16384 1.7578125 ;
      box_dim_v     32 90 ;
      box_dim_w     32 90 ;
    end mann_meanderturb;
;
    begin mann_microturb ;
      create_turb_parameters 8.0 1.0 1.0 508 1.0 ;      L, alfaeps,gamma,seed, highfrq compensation
      filename_u    ./free_sector_monopile/wake-micro/wake_turb_wsp8_s508_t1800u.bin ;
      filename_v    ./free_sector_monopile/wake-micro/wake_turb_wsp8_s508_t1800v.bin ;
      filename_w    ./free_sector_monopile/wake-micro/wake_turb_wsp8_s508_t1800w.bin ;
      box_dim_u   128 1.0 ;
      box_dim_v     128 1.0 ;
      box_dim_w     128 1.0 ;
    end mann_microturb;
  end wakes;
  ;-----------------------------------------------------------------
  begin mann;
    create_turb_parameters 33.6 1 3.7 508 1.0 ;      L, alfaeps,gamma,seed, highfrq compensation
    filename_u    ./free_sector_monopile/turb/turb_wsp8_s508_t1800u.bin ;
    filename_v    ./free_sector_monopile/turb/turb_wsp8_s508_t1800v.bin ;
    filename_w    ./free_sector_monopile/turb/turb_wsp8_s508_t1800w.bin ;
    box_dim_u    16384 1.7578125 ;
    box_dim_v    32 3.75;
    box_dim_w    32 3.75;
  end mann;
end wind;;
begin aero ;
  nblades  3;
  hub_vec shaft -3 ;         rotor rotation vector (normally shaft component directed from
  ;                          pressure to suction side)
  link 1 mbdy_c2_def blade1;
  link 2 mbdy_c2_def blade2;
  link 3 mbdy_c2_def blade3;
  ae_filename        ./data/NREL_5MW_ae.txt;
  pc_filename        ./data/NREL_5MW_pc.txt;
  induction_method   1 ;     0=none, 1=normal
  aerocalc_method    1 ;     0=ingen aerodynamic, 1=med aerodynamic
  aerosections       30 ;
  ae_sets            1 1 1;
  tiploss_method     1 ;     0=none, 1=prandtl
  dynstall_method    2 ;     0=none, 1=stig øye method,2=mhh method
end aero ;
;
;-------------------------------------------------------------------------------------------------
```

```
 begin hydro;
   begin water_properties;
     rho 1027 ; kg/m^3
     gravity 9.81 ; m/s^2
     mwl 0.0 ;
     mudlevel 20.0 ;
     water_kinematics_dll ./wkin_dll.dll   ./htc_hydro/reg_airy_h6_t10.inp ;
 end water_properties;
;
   begin hydro_element;
     body_name monopile ;
     hydrosections uniform 50 ; distribution of hydro calculation points from sec 1 to nsec
     nsec 2;
     sec 0.0 1.0 1.0 28.27 28.27 6.0 ;  nr z Cm Cd V Vr width
     sec 30.0 1.0 1.0 28.27 28.27 6.0 ;  nr z Cm Cd V Vr width
   end hydro_element;
 end hydro;
;
;-------------------------------------------------------------------------------------------
begin dll;
  begin hawc_dll;
    filename  ./control/bladed2hawc.dll ;
    dll_subroutine regulation ;
    arraysizes  15 15 ;
    deltat  0.02;
    begin output;
      general time ;
      constraint bearing2 pitch1 1;  angle and angle velocity written to dll
      constraint bearing2 pitch2 1;  angle and angle velocity written to dll
      constraint bearing2 pitch3 1;  angle and angle velocity written to dll
      constraint bearing2 shaft_rot 1; angle and angle velocity written to dll  (slow speed shaft)
      wind free_wind 1 0.0 0.0 -90.55; local wind at fixed position: coo
      general constant 97.0 ;                  generator exchange ratio
    end output;
;
    begin actions;
      body moment_int shaft 1 3 towertop 2 ;
    end actions;
  end hawc_dll;
;
  begin hawc_dll;
    filename  ./control/pitchservo_pos.dll ;
    dll_subroutine servo ;
    arraysizes  15 15 ;
    deltat    0.02 ;
    begin output;
      general time ;                                                              1
      dll inpvec 1 2;                                                             2
      dll inpvec 1 3;                                                             3
      dll inpvec 1 4;                                                             4
      constraint bearing2 pitch1 1;  angle and angle velocity written to dll         5,6
      constraint bearing2 pitch2 1;  angle and angle velocity written to dll         7,8
      constraint bearing2 pitch3 1;  angle and angle velocity written to dll         9,10
    end output;
```

```
;
    begin actions;
      body bearing_angle pitch1;
      body bearing_angle pitch2;
      body bearing_angle pitch3;
    end actions;
  end hawc_dll;
;
   begin hawc_dll;
    filename  ./control/damper.dll ;
    dll_subroutine damp ;
    arraysizes  15 15 ;
    begin output;
      general time ;                                                                1
      general constant 5.0;
      general constant 10.0;
      general constant -1.0E1 ;
      mbdy state vel towertop 1 1.0 tower;
   end output;
;
    begin actions;
       mbdy force_ext towertop 2 1 towertop;
    mbdy force_ext towertop 2 2 towertop;
    end actions;
  end hawc_dll;
end dll;
;
;-------------------------------------------------------------------------------------
;
begin output;
  filename ./res/oc3_monopile_phase_1 ;
;  time 390.0 450.0 ;
  buffer 1 ;
  general time;
  data_format  hawc_binary;
;
  constraint bearing1 shaft_rot 2; angle and angle velocity
  constraint bearing2 pitch1 5;    angle and angle velocity
  constraint bearing2 pitch2 5;    angle and angle velocity
  constraint bearing2 pitch3 5;    angle and angle velocity
  aero omega ;
  aero torque;
  aero power;
  aero thrust;
  wind free_wind 1 0.0 0.0 -90.0; local wind at fixed position: coo
  hydro water_surface 0.0 0.0 ;        x,y   gl. pos
  mbdy momentvec towertop 1 2 towertop # yaw bearing ;
  mbdy forcevec  towertop 1 2 towertop # yaw bering ;
  mbdy momentvec shaft 4 1  shaft # main bearing ;
  mbdy momentvec blade1 3  1 blade1 # blade 1 root ;
  mbdy momentvec blade1 10 1 local # blade 1 50% local e coo ;
  mbdy momentvec hub1 1  2 hub1 # blade 1 root ;
  mbdy momentvec hub2 1  2 hub2 # blade 2 root ;
  mbdy momentvec hub3 1  2 hub3 # blade 3 root ;
```

```
  mbdy state pos towertop   1 1.0 global # tower top flange position ;
  mbdy state pos tower   1 0.0 global # tower MSL position ;
  mbdy state pos blade1  18 1.0 blade1 # blade 1 tip pos ;
  mbdy state pos blade2  18 1.0 blade2 # blade 2 tip pos ;
  mbdy state pos blade3  18 1.0 blade3 # blade 3 tip pos ;
  mbdy state pos blade1  18 1.0 global # blade 1 tip pos ;
  aero windspeed 3 1 1 63.0;   wind seen from the blade:
  ;                            coo(1=local ae,2=blade,3=global,4=rotor polar),
  aero windspeed 3 1 2 63.0;
  aero windspeed 3 1 3 63.0;
  aero alfa 1 45.0;
  aero alfa 2 45.0;
  aero alfa 3 45.0;
  mbdy momentvec towertop 1 1 tower # tower top -1: below top mass ;
  mbdy forcevec  towertop 1 1 tower # tower top -1: below top mass ;
  mbdy momentvec tower  1 1 tower # tower MSL ;
  mbdy forcevec  tower  1 1 tower # tower MSL ;

; mbdy statevec_new mbdyname center coo elastic/absolute r sign xy_vector:
  mbdy statevec_new blade1 c2def   blade1 elastic  88.0 1.d0 0.0 0.0
  mbdy statevec_new blade1 default blade1 elastic  88.0 1.d0 0.0 0.0 ;
  mbdy statevec_new blade1 c2def   blade1 absolute 88.0 1.d0 0.0 0.0 ;
  mbdy statevec_new blade1 default blade1 absolute 88.0 1.d0 0.0 0.0 ;
  mbdy statevec_new blade1 default global absolute 88.0 1.d0 0.0 0.0 ;

; mbdy forcemomentvec_interp mbdy_name center coo_mbdy curved_distance_from_orig sign
  mbdy forcemomentvec_interp blade1 default blade1 5 1.0 # blade1 R= 5 ;
  mbdy forcemomentvec_interp blade1 default blade1 55 1.0 # blade1 R=55  ;
  mbdy forcemomentvec_interp blade1 c2def local_aero 35 1.0 # blade1 R=35  ;
  mbdy forcemomentvec_interp blade1 c2def local_aero 60 1.0 # blade1 R=60  ;
  mbdy forcemomentvec_interp blade1 c2def local_element 50 1.0 # blade1 R=50  ;
; an example where the forces and moments are extracted at the c2def instead of the actual node:
  mbdy forcemomentvec_interp blade1 c2def blade1 5 1.0 # blade1 R= 5 ; ()
;
  dll outvec 1 1 # time;
  dll outvec 1 2 # pitch angle 1;
  dll outvec 1 3 # pitch vel 1;
  dll outvec 1 4 # pitch angle 2;
  dll outvec 1 5 # pitch vel 2;
  dll outvec 1 6 # pitch angle 3;
  dll outvec 1 7 # pitch vel 3;
  dll outvec 1 8 # gen. azi slow;
  dll outvec 1 9 # gen. speed slow;
  dll outvec 1 10 # free wind x;
  dll outvec 1 11 # free wind y;
  dll outvec 1 12 # free wind z;
  dll outvec 1 13 # gear ratio;
  dll inpvec 1 1 # Mgen slow;
  dll inpvec 1 2 # pitchref 1;
  dll inpvec 1 3 # pitchref 2;
  dll inpvec 1 4 # pitchref 3;
  dll inpvec 1 7 # F;
  dll inpvec 1 8 # Mechanical power generator [kW];
  dll inpvec 1 10 # Pitch rate [rad/s];
```

```
  dll inpvec 2 1 # pitch 1;
  dll inpvec 2 2 # pitch 2;
  dll inpvec 2 3 # pitch 3;
  dll outvec 2 1 # time;
  dll outvec 2 2 # pitchref 1;
  dll outvec 2 3 # pitchref 2;
  dll outvec 2 4 # pitchref 3;
  dll outvec 2 5 # pitch angle 1;
  dll outvec 2 6 # pitch speed 1;
  dll outvec 2 7 # pitch angle 2;
  dll outvec 2 8 # pitch speed 2;
  dll outvec 2 9 # pitch angle 3;
  dll outvec 2 10 # pitch speed 3;
end output;
;
exit;
```

B User guide for user-wind-dll

A user defined DLL can be used to provide additional wind velocity on top of what is already defined by wind input in HAWC2. During simulation, HAWC2 calls the DLL with position as argument, and the DLL must provide the wind velocity in that position on return. Apart from the position, HAWC2 also parses time and user-specified arguments to the DLL - the user-specified arguments are defined in the same output block format as is used for type2_dlls and hawc_dlls and as regular output.

B.1 Htc file input

Application of the DLL is defined inside the `begin wind` block as shown below.

```
begin wind ;
    .
    begin user_wind_dll ;
        filename 2-test.dll;
        subroutine wind_dll_getwindspeed ;
        refsys 0     ; Reference coordinates for position (in) and velocity (in/out),
                     ;   0=meteorological(default),
 ;   1=global
        begin output ;
            general constant 1.0                    ;
            dll inpvec 1 1                          ;
            constraint bearing1 shaft_rot 1 only 2  ;
            mbdy momentvec shaft 1 1 shaft only 3   ;
        end output ;
    end user_wind_dll ;
    .
end wind
```

The output arguments that can be used inside the `begin output` block are limited `general`, `dll`, `constraint`, and `mbdy`.

B.2 DLL interface definition

The DLL subroutine is called each iteration with these arguments: - time, (double). - position vector: Dependent on the key `refsys` in the `user_wind_dll` block (see above), the position vector provided is either meteorological or global coordinates, (double(3)). - Nof arguments in the `begin output` in the `user_wind_dll` block, (nargs, integer). - Argument vector defined in the `begin output` in the `user_wind_dll` block, (double(nargs)). - Wind velocity: On input, the vector contains the wind velocity contribution from the whatever is defined in the `begin wind` block, i.e. the sum of mean wind, wind shear, etc. On output, the vector must contain the extra(!!, NOT the total) wind contribution in the `refsys` coordinate system , (double(3))

The DLL subroutine interface is defined as follows:

```
interface
    subroutine user_wind_dll_call(time, pos, nargs, args, wsp)
    !dec$ attributes c :: user_wind_dll_call
    double precision    :: time         ! time
    double precision    :: pos(3)       ! position of lookup point (refsys coordinates)
    integer             :: nargs        ! nof user arguments (provided via dll output block)
```

```
    double precision    :: args(nargs)  ! user arguments (provided via dll output block)
    double precision    :: wsp(3)       ! lookup windspeed,
                                    !   on input : wind velocity in <pos> (refsys coord.)
                                    !   on output: user velocity contribution (refsys coord.)
    !dec$ attributes reference :: time, pos, nargs, args, wsp
    end subroutine
end interface
```

Note that the effect of tower shadow is applied after the call to the DLL.

B.2.1 FORTRAN example

```
subroutine wind_dll_getwindspeed(time,pos,nvar,var,wsp)
    !dec$ attributes c,dllexport, alias:"wind_dll_getwindspeed" :: wind_dll_getwindspeed
    !gcc$ attributes cdecl :: wind_dll_getwindspeed
    !gcc$ attributes dllexport :: wind_dll_getwindspeed
    ! variables
    integer nvar
    double precision time,pos(*),var(*),wsp(*)
    !dec$ attributes reference :: time, pos, var, wsp

    ! implementation
    print*,"nvar = ",nvar
    print*,"time = ",time
    print*,"pos  = ", pos(1:3)
    print*,"wsp  = ", wsp(1:3)
    wsp(1:3) = (/0.0, 0.0, 0.0/)
end subroutine wind_dll_getwindspeed
```

The DLL can be built from the FORTRAN code above using the GNU compiler syntax:

```
gfortran -shared -static -o <file>.dll -fno-underscoring <file>.f90
```

B.2.2 C example

```
#include <stdio.h>
__declspec(dllexport) void wind_dll_getwindspeed(double* time, double* pos, int* nvar, double*
{
    int i;
    //  implementation
    printf("nvar = %d\n", *nvar);
    printf("time = %f\n", *time);
    printf("pos  = (%f, %f, %f)\n", pos[0], pos[1], pos[2]);
    printf("wsp  = (%f, %f, %f)\n", wsp[0], wsp[1], wsp[2]);
    for (i = 0; i < 2; i++)
    {
        wsp[i] = 0.0;
    }
}
```

The DLL can be built from the C code above using the GNU compiler syntax: “‘ gcc -shared -static -o .dll .c

C Fit of structural damping

Please note that this feature is not easy to use, and some iterations must be foreseen in order to end at satisfactory result.

The aim of this feature is to develop a method to fit the damping parameters in a HAWC2 model in such a way that desired damping ratios are obtained for specified eigenmodes. Further, it is the aim that the formulation can be used in both HAWC2 and HAWCStab2.

The method is described below in Section C.1. It fits element stiffness matrices and saves them to file so that they can be used for bodies using the damping method "damping_file <damping file> ;" (where the <damping file> is generated by the method). This requires a special block inside the htc file which must be placed after the "new_htc_structure" block:

Obl.	Command name	Explanation
*	begin damping_fit ;	First line in damping fit block.
*	damping_file	Name of damping file. This file name MUST match the <damping file> used for the "damping_file" method in the "main_body" block.
	twin_bodies	The two body names given as arguments will share damping properties. This is used e.g. to specify the same damping for all the blades on the rotor. 1. Body name for 1st twin. 2. Body name for 2nd twin.
*	cmd_solver	1. Command executed by HAWC2 which does the actual fitting, (e.g. *python.exe damping_fit.py ;*). If you rely on a virtual Python environment, make sure to activate this first before running HAWC2 within this environment. Within this Python environment the `numpy` and `scipy` packages are required to be installed.
*	mode -	Repeated line for each mode to be fitted: 1. Mode number. 2. Damping ratio.
*	end damping_fit ;	Last line in damping fit block.

The example below shows the setup for fitting the damping of a single blade with requested damping ratios specified for the first six modes. 0.5% damping ratio (i.e. approx. 3% log.decr.) is requested for modes 1 and 2, 1% for modes 3 and 4, and 2% for modes 5 and 6. Note the use of the damping file (blade.dmp) in two locations which links the damping fit only to include blade damping in the fit. If other bodies were present in the example, the damping specified for those bodies would enter the total damping fit, but only the damping parameters for the blade will change the total damping.

```
;------------------------------------------------------------------------
begin new_htc_structure;
;------------------------------------------------------------------------
  begin main_body;
  name        blade1 ;
  type        timoschenko ;
  nbodies     10 ;
  node_distribution    c2_def;
  damping_file blade.dmp ;
  begin timoschenko_input ;
```

```
    filename ./data/DTU_10MW_RWT_Blade_st.dat;
    set 1 1 ;                set subset
  end timoschenko_input;
  begin c2_def;              Definition of centerline (main_body coordinates)
    nsec 27 ;
    sec  1 0.00000E+00 7.00600E-05 4.44089E-16 -1.45000E+01 ;
        ..
        ..
    sec 27 -8.98940E-02 -3.33685E+00 8.63655E+01 3.42796E+00 ;
    end c2_def ;
  end main_body;
;-------------------------------------------------------------------------
  begin orientation;
    begin base;
      body   blade1;
      inipos        0.0 0.0 0.0 ;         initial position of node 1
      body_eulerang 0.0 0.0 0.0;
    end base;
  end orientation;
;-------------------------------------------------------------------------
  begin constraint;
    begin fix0;  fixed to ground in translation and rotation of node 1
      body blade1;
    end fix0;
  end constraint;
end new_htc_structure;
;-------------------------------------------------------------------------
begin damping_fit ;
  damp_file blade.dmp ;
  cmd_solver C:\Users\anmh\Anaconda3\Scripts\conda.exe run python damping_fit.py ;
  mode 1 0.005 ; Damping ratio of mode 1
  mode 2 0.005 ; etc.
  mode 3 0.01 ;
  mode 4 0.01 ;
  mode 5 0.02 ;
  mode 6 0.02 ;
end damping_fit ;
;-------------------------------------------------------------------------
```

A fully functional example is available for download at `https://gitlab.windenergy.dtu.dk/hawc-users/examples/-/tree/master/hawc2/structure/damping_fit`.

C.1 Formulation

The linearised HAWC2 EOMs are of the usual 2nd order form:

$$\mathbf{M}\,\mathbf{x} + \mathbf{C}\,\mathbf{x} + \mathbf{K}\,\mathbf{x} = \mathbf{0} \tag{C.5}$$

The solution to the undamped eigenvalue problem ($\mathbf{C} = \mathbf{0}$) are defined by the eigenvectors $\mathbf{\Gamma}$ and diagonal eigenfrequency matrix $\mathbf{\Omega}$. The eigensolution fulfills the identity $\mathbf{M\Gamma\Omega^2} = \mathbf{K\Gamma}$. By using the eigenvectors as basis, $\mathbf{x}$ can be transformed as $\mathbf{x}(\mathbf{t}) = \mathbf{\Gamma}\,\alpha(\mathbf{t})$. By using the above

relations, C.5 can be manipulated as:

$$\ddot{\alpha} + \mathbf{\Gamma^{-1}M^{-1}C\Gamma}\,\alpha + \mathbf{\Omega^2}\,\alpha = \mathbf{0} \tag{C.6}$$

Note that the undamped part of C.6 is a diagonal system, and that the total set of equations can de uncoupled if the damping matrix part is also a diagonal matrix. If we choose this matrix as $2\,\zeta\mathbf{\Omega}$ (ζ is a diagonal matrix), then the system damping matrix $\mathbf{C}$ can be calculated as

$$\mathbf{C} = \mathbf{2\,\Gamma M}\zeta\mathbf{\Omega\Gamma^{-1}} \tag{C.7}$$

Unfortunately, such a damping matrix cannot directly be used in neither HAWC2 nor in HAWCStab2, so something else must be done. Instead, the damping of one mode at a time is formulated as function of element damping matrices:

$$\ddot{\alpha}_\mathbf{i} + (\gamma_\mathbf{i}{}^\mathbf{T}\mathbf{M}\gamma_\mathbf{i})^{-1}(\gamma_\mathbf{i}{}^\mathbf{T}\mathbf{C}\gamma_\mathbf{i})\,\alpha_\mathbf{i} + \mathbf{\Omega_i^2}\,\alpha_\mathbf{i} = \mathbf{0} \tag{C.8}$$

where all variables with sub-script i relate to the i'th eigenmode. The damping coefficient in (C.8) must then fulfill the equation

$$(\gamma_\mathbf{i}{}^\mathbf{T}\mathbf{M}\gamma_\mathbf{i})^{-1}(\gamma_\mathbf{i}{}^\mathbf{T}\mathbf{C}\gamma_\mathbf{i}) = \mathbf{2}\zeta_\mathbf{i}\,\mathbf{\Omega_i} \tag{C.9}$$

The system damping matrix, $\mathbf{C}$, is assembled based on element damping matrices $\mathbf{c_j}$ (for the j'th element), where the element damping matrices are defined as having the same eigenvectors as the element stiffness matrices. By using this formulation, the structure only dissipates energy when it is deformed and not during rigid body motion.

$$\mathbf{c_j} = \mathbf{v_j x_j v_j^T} \tag{C.10}$$

where $\mathbf{v_j}$ is the eigenvectors of the j'th stiffness matrix (or rather the six eigen vectors that have non-zero eigenvalues) and $\mathbf{x_j}$ is a 6×6 diagonal matrix containing the unknown damping parameters.

For each eigenmode that needs to be fitted, (5) provides one equation that needs to be fulfilled, and the unknowns are the six element damping parameters, $diag(\mathbf{x_j})$, for all elements. Further, in order for the element damping matrices to be positive semi-definite, $\mathbf{x_j} \geq \mathbf{0}$ for all diagonal components $\mathbf{x_j}$ and for all elements (all j). The equations in (C.9) are linear in $\mathbf{x}$ and the (one of many!) solution is found by solving the optimization problem

$$\min_{wrt.\mathbf{x}} \left(|\mathbf{W}\,(\mathbf{A}\,\mathbf{x} - \zeta)|^2 \right),\ s.t.\ \mathbf{x} \geq \mathbf{0} \tag{C.11}$$

where $\mathbf{x}$ are the element damping parameters for all elements collected in a vector, ζ are the (user-specified) damping ratios prescribed for the individual modes, $\mathbf{A}$ are the coefficients to $\mathbf{x}$ in accordance with (C.9), and $\mathbf{W}$ is a diagonal weighting matrix which is included in order to weigh the individual eigenmodes in the optimization.

C.2 HAWC2 implementation

Currently, the solution of (C.11) is handled by an external call to a python script outside of HAWC2. This means that HAWC2 calculates the matrices and vectors in (C.11) and exports those to a binary file (currently named *'dfit_a.bin'*). This binary file is then handled in a Python

script that reads the system, solves (C.11) for **x** and writes back the solution to file (currently named *'dfit_x.bin'*). This solution is then read back into HAWC2 and the resulting element damping matrices are calculated and written to file for subsequent use in HAWC2 simulations.

Note that even though only a few of the total number of eigenmodes have prescribed damping ratios (specified in the htc-file), all eigenmodes are included in the outputted binary file, however, the components in **W** associated with non-prescribed modes are all set to zero.

The Python script is listed below from which the individual binary file formats can be deduced, if needed. This script is part of the distributed HAWC2 files.

```
#-------------------------------------------------------------------------
# -*- coding: utf-8 -*-
"""
Damping fit for HAWC2

This script finds the parameters for structural HAWC2 damping type
"damp_file", based on the mode damping matrix, A, calculated by HAWC2
and written to file "dfit_a.bin".

Each row of the A matrix corresponds to a mode shape in the HAWC2 model,
ordered in increasing order of eigenfrequency, i.e. first row corresponds to
the mode with the lowest eigenfrequency. By multiplication with the damping
parameter vector, x, gives the damping rati0 vector, d = (A*x).

The purpose of this script is then to find the best fit of x which gives the
specified damping ratio for the individual modes using the constraint that
x>0 for all x.

The A matrix contains all modes, and not all modes can be fitted for any
damping level. Normally the first (say 10) modes are of interest. This is
handled by the weighting vector, w, below. See code below for further details.

"""


import numpy as np
import struct
from scipy.optimize import nnls

def damping_fit():

    wmin = 1.e-6

    # Read A matrix from file
    f=open('dfit_a.bin','rb')
    (nr,nc) = struct.unpack('ii',f.read(8))
    ntot = nr*nc
    data = np.zeros(ntot,dtype=np.dtype('f8'))
    for i in range(ntot):
        (data[i],) = struct.unpack('d',f.read(8))
    A = np.reshape(data,[nr,nc], order='F')

    # Read target damping for optimization
    d = np.zeros(nr,dtype=np.dtype('f8'))
    for i in range(nr):
        (d[i],) = struct.unpack('d',f.read(8))
```

```
    w = np.zeros(nr,dtype=np.dtype('f8'))
    for i in range(nr):
        (w[i],) = struct.unpack('d',f.read(8))
        if w[i] == 0.0:
            w[i] = wmin
    f.close()

    # Solve
    res = nnls(np.matmul(np.diag(w),A), np.matmul(np.diag(w),d))

    # Write results to file
    f = open('dfit_x.bin','wb')
    f.write(np.array([nc,1],dtype='i4'))
    f.write(res[0])
    f.close()

    # Check solution
    dfit = np.matmul(A,res[0])
    print(('*'+'{0:1s}'*36+'*').format('*'))
    print('*{:^36s}*'.format('Damping fit result'))
    print(('*'+'{0:1s}'*36+'*').format('*'))
    print('*{:^8s}{:^14s}{:^14s}*'.format('Mode','Target','Fit'))
    for i in range(nr):
        if w[i] > wmin:
            print('*{:^8d}{:^14.3e}{:^14.3e}*'.format(i+1,d[i],dfit[i]))
    print(('*'+'{0:1s}'*36+'*').format('*'))

    return res

#-----------
# DO IT....
#-----------
res = damping_fit()
#-------------------------------------------------------------------------
```

C.3 Usage considerations

C.3.1 Consistently use matched input and damping file

The result of the structural damping fitting procedure is a main body element damping matrix file that will match the user defined damping for the relevant modes. This file is specific for a given combination of nodes, number of bodies and structural input (st-file). If any changes are made in either of these inputs the element damping matrix file will have to be redefined based on the procedure outlined here. Users are especially cautioned to carefully track that the number of bodies used for generating the damping fit is also the same number of bodies used in subsequent simulations.

C.3.2 Tune on full or main body only models

It is more likely to obtain a good damping fit for many frequencies when tuning the damping for a HAWC2 model containing only the main body of interest. When a model with several main bodies is used (tower, blades, etc) the optimisation problem becomes inherently more difficult to solve. When using multiple main bodies, make sure to verify that the targeted damping ratios in the `damping_fit` section relate to the total systems modes (1st and 2nd modes likely to be

the tower, etc), as opposed to when using a model that only contains the main body of interest. The user is responsible for tracking which mode number relates to which body. For example, fitting the damping for tower modes while only adjusting the damping coefficients related to the blades is not likely to give meaningful results. It is therefore recommended to only list/target mode numbers of the body at interest, and leave out the others (especially rigid body modes) in the `damping_fit` section.

C.3.3 Number of modes to target

When fitting to a low number of modes a very good result can be expected. The more modes a user attempts to fit a damping value to, the more difficult the trade-off becomes. In those cases an advanced user could consider changing the weights **W** in the example script `damping_fit.py` (defined as `w`, see above) to obtain a specific trade-off in which some modes are allowed to differ more compared to others with respect to the requested target values.

D Code Version Data

The release notes from all previous HAWC2 releases are included as a text file in the all-in-one download package available on `http://tools.windenergy.dtu.dk/HAWC2/downloads`.

Risø's research is aimed at solving concrete problems in the society.

Research targets are set through continuous dialogue with business, the political system and researchers.

The effects of our research are sustainable energy supply and new technology for the health sector.