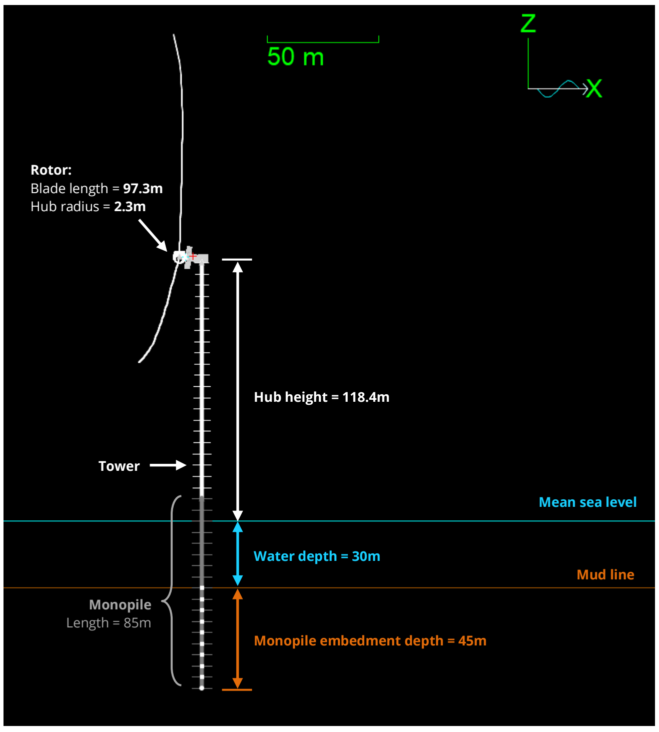
K02 10MW fixed-bottom OWT

This example models the 10MW reference wind turbine (RWT), developed as part of the International Energy Agency's (IEA) Wind Task 37. The system design basis is documented by the Systems Engineering Wind Energy - WP2.1 Reference Wind Turbines technical report, which can be accessed from the open-source IEA-10.0-198-RWT GitHub repository.

The model of the offshore wind turbine (OWT) is shown below, where the turbine takes the form of a three-bladed rotor, with variable-speed and collective blade-pitch control capabilities.

The turbine hub is connected to the nacelle, which houses a direct-drive synchronous generator. The nacelle is supported by a tower which is mounted upon a fixed-bottom monopile foundation embedded to a depth of 45m below the seabed.

Note: the model properties considered in this example are based on our interpretation of the information and data available at the time of writing. If you use, or refer to, this model as part of your own analysis requirements then you must carry out the appropriate checks to confirm the implemented data are correct.



Python installation requirements

As this example uses Python scripts, Python 3 is required to view and run the simulation. It is also possible to view this model using the demo version of OrcaFlex which is available for download from the Orcina website: https://www.orcina.com/orcaflex/demo/.

Note, the OrcaFlex installation package includes the option to install a compact embedded Python distribution. If this option is selected when installing OrcaFlex, a separate installation of Python is not required for embedded Python features e.g. Python external functions and user defined results. The Python Interface: Installation help page provides further details about this subject.

Controller modelling

Controller modelling for the turbine is supported through OrcaFlex external functions, which are used as a wrapper for the NREL Reference Open Source Controller (ROSCO), dynamic-link library (DLL). This example uses a **Python** external function.

Below is a summary of the controller setup:

- Included amongst the example files is a Python module (*BladedControllerWrapper.py*) that defines the *BladedController* Python OrcaFlex external function class. This function is responsible for calling the ROSCO DLL and returning controller output to OrcaFlex as required, switching between generator torque and blade pitch, depending on the data name calling it.
- In the OrcaFlex model, blade pitch and generator torque are then both specified by the *Controller* external function data source which uses the *BladedController* function.
- Object tags are used to identify the DLL to be wrapped by the external function and its associated input file. Both of which are included amongst the example files. These tags are assigned to the *10MW RWT* turbine object, on the *tags* page of the turbine data form.
- The *ControllerDLL* tag specifies the path, relative to the model, of the 64-bit, v2.7.0, ROSCO DLL (*libdiscon.dll*). Although this example makes use of a 64-bit version of the DLL, a 32-bit version is also available for download from the ROSCO GitHub repository.
- The *InputFile* tag specifies the path, relative to the model, of the ROSCO parameter input file (*IEA-10.0-198-RWT_DISCON.IN*). This can be downloaded from the IEA-10.0-198-RWT GitHub repository. The contents of the input file can be viewed and edited using a text file editor.
- The external function is responsible for calling the DLL, once per time step. When called, the DLL updates its state and calculates all the necessary controller values. A swap array is used to communicate between the external function and the DLL. Before the DLL is called, the appropriate elements (slots) in the swap array are populated with the values to be input to the calculation. The swap array is then passed into the DLL when it is called. After it has been called, the DLL will have calculated the control values which are then read out of the swap array and returned to OrcaFlex through the external function. Please note, the wrapper only populates the swap array sufficiently to support the ROSCO control modes used by this example. If other controller modes are needed, then the wrapper may need to be further extended.
- In this example, the ROSCO wind speed estimator is used, i.e. `WE_Mode` is "2" in the parameter input file. This introduces a further DLL dependency: the rotor performance file. The `PerfFileName` variable of the ROSCO input file specifies name and location of this file, as a path relative to the input file. It is included amongst the example files (*IEA-10.0-198-RWT_Cp_Ct_Cq.txt*) and can be downloaded from the IEA-10.0-198-RWT GitHub repository.

For more detail about using OrcaFlex external functions to wrap the ROSCO DLL, please see our guide to using external functions to model turbine controllers. This includes documentation of the swap array, information on further modifying behaviour with object tags, a native code external function wrapper, and some simple guidance on debugging issues.

Building the model

Turbine object

We have gathered the necessary input data for the turbine object from the available 10MW RWT documentation and supporting online resources.

This example document does not provide any further details about modelling the turbine rotor itself. For a more detailed discussion on this topic, please refer to our K01 5MW spar FOWT example.

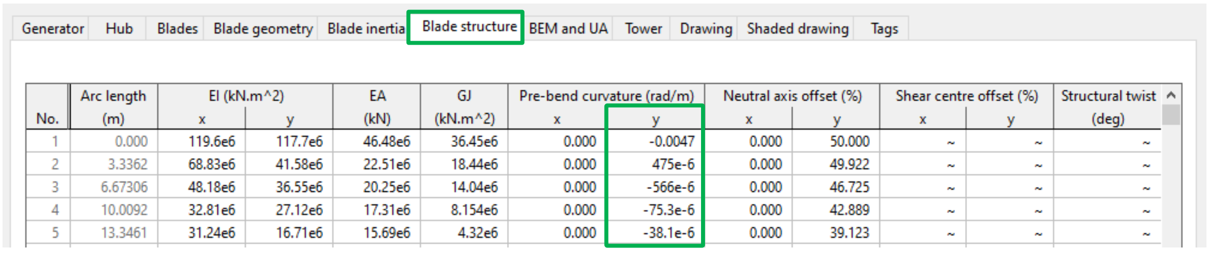
Blade pre-bend

One important turbine feature, not discussed in the above example, is blade pre-bend. This allows for representation of blades which are not straight in their unstressed state.

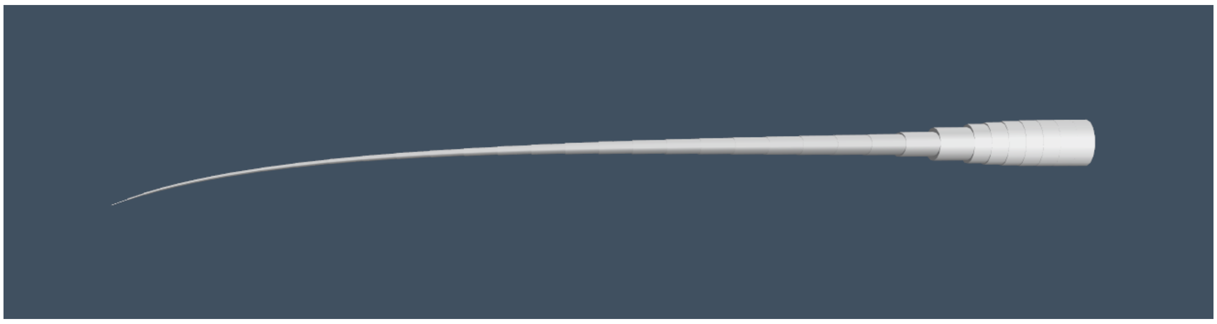
The blade pre-bend feature is very similar to pre-bend for OrcaFlex line objects, where it is possible to specify the local curvature components in radians per unit length.

On the *blade structure* page of the turbine data form, we have specified the *pre-bend curvature* (*y*) values necessary to model the blade pre-bend for the 10MW RWT. Here, negative curvature values are specified to pre-bend the blades out of the rotor plane and away from the tower. Note that OrcaFlex applies pre-bend irrespective of any structural twist in the blades.

Further details about this feature may be found on the Modelling, data and results | Turbines | Turbine data | Blade profile page of the OrcaFlex help.



	Arc length	EI (kN.m^2)		EA	GJ	Pre-bend curvature (rad/m)		Neutral axis offset (%)		Shear centre offset (%)		Structural twist
No.	(m)	x	y	(kN)	(kN.m^2)	x	y	x	y	x	y	(deg)
1	0.000	119.6e6	117.7e6	46.48e6	36.45e6	0.000	-0.0047	0.000	50.000	~	~	~
2	3.3362	68.83e6	41.58e6	22.51e6	18.44e6	0.000	475e-6	0.000	49.922	~	~	~
3	6.67306	48.18e6	36.55e6	20.25e6	14.04e6	0.000	-566e-6	0.000	46.725	~	~	~
4	10.0092	32.81e6	27.12e6	17.31e6	8.154e6	0.000	-75.3e-6	0.000	42.889	~	~	~
5	13.3461	31.24e6	16.71e6	15.69e6	4.32e6	0.000	-38.1e-6	0.000	39.123	~	~	~

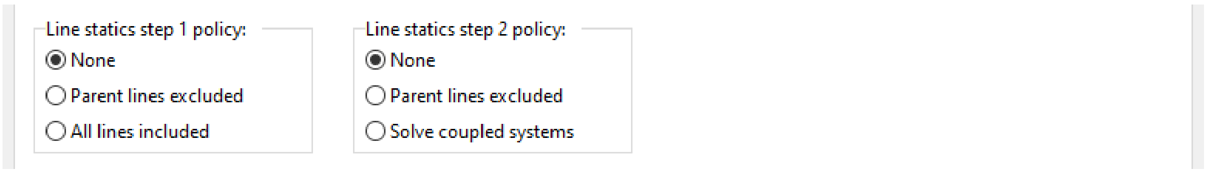


Tower and monopile modelling

The turbine support structure comprises a *Tower* and *Monopile*, modelled using line objects. The corresponding line types utilise the *homogeneous pipe* category, which allows for modelling of the variable outer and inner diameter profiles of these items as well as their physical properties.

Although it is possible to model the support structure using a single line, with multiple sections, we have intentionally chosen to model these items as separate line objects, connected using line-to-line connections. This provides the option to check interface loads, between the tower and monopile, using the line *end load* results available in OrcaFlex.

As the *Tower* and *Monopile* lines are very stiff, and already close to their static equilibrium positions in the reset state, we have set the *line statics step 1* & …*step 2* policies to 'None' on the *statics* page of the general data form. This bypasses the line statics stage of the statics analysis process, meaning OrcaFlex will move straight to solving whole system statics. This helps to improve the efficiency of the statics solve.



Line statics step 1 policy: [✓ None] Parent lines excluded / All lines included

Line statics step 2 policy: [✓ None] Parent lines excluded / Solve coupled systems

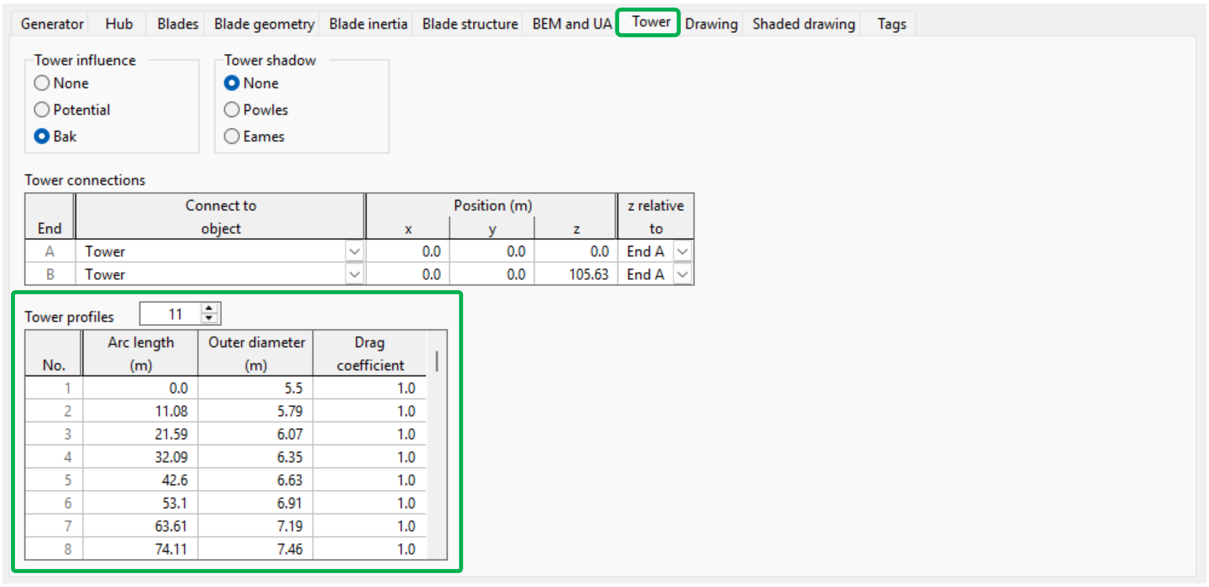
Another consequence of the support structure being very stiff is that it can give rise to results precision problems in dynamics. It is possible that *single precision* logging, which is the default setting specified on the *dynamics* page of the general data form, will not log the node position results with enough significant figures to allow certain results to be accurately derived from them.

For this reason, we have decided to log the dynamic results using *double precision*. Further details about this can be found on the [Modelling, data and results | General data | Logging] help page.

We have also implemented the *Bak* tower influence model, which is set on the *tower* page of the turbine data form (pictured below). This makes use of Bak's model which incorporates a correction to the classic *potential* theory model.

When using this model, the *tower connections* must be specified along with the *tower profiles*. The profiles are defined at several discrete unstretched arc lengths, starting at End A of the *tower* line. The *outer diameter* for each tower profile is also specified here. The data are assumed to vary linearly between the arc lengths at which they are given.

Further details about the available tower influence models can be found on the [Modelling, data and results | Turbines | Turbine data | Tower] help page.



Tower influence: None / Potential / [✓ Bak]

Tower shadow: [✓ None] Powles / Eames

Tower connections

End	Connect to object	Position (m) x	y	z	z relative to
A	Tower	0.0	0.0	0.0	End A
B	Tower	0.0	0.0	105.63	End A

Tower profiles 11

No.	Arc length (m)	Outer diameter (m)	Drag coefficient
1	0.0	5.5	1.0
2	11.08	5.79	1.0
3	21.59	6.07	1.0
4	32.09	6.35	1.0
5	42.6	6.63	1.0
6	53.1	6.91	1.0
7	63.61	7.19	1.0
8	74.11	7.46	1.0

Soil modelling

The interaction between the seabed and the monopile foundation has been modelled using the 'elastic halfspace model'. Here, the interaction is represented by linear springs which resist the translational and rotational movement of the embedded monopile in 6 degrees of freedom (DOFs).

The *P-y models*, available in OrcaFlex, cannot account for these spring constants in 6 DOFs so we have chained four *calculated DOFs* constraints together to model the translational and rotational spring stiffnesses at each embedded *Monopile* line node: **(1)** horizontal (*x*, *y*), **(2)** vertical (*z*), **(3)** rocking (*Rx*, *Ry*), **(4)** twisting (*Rz*).

Each line node is then connected to its respective 'chain' of constraints, to create the following connection sequence:

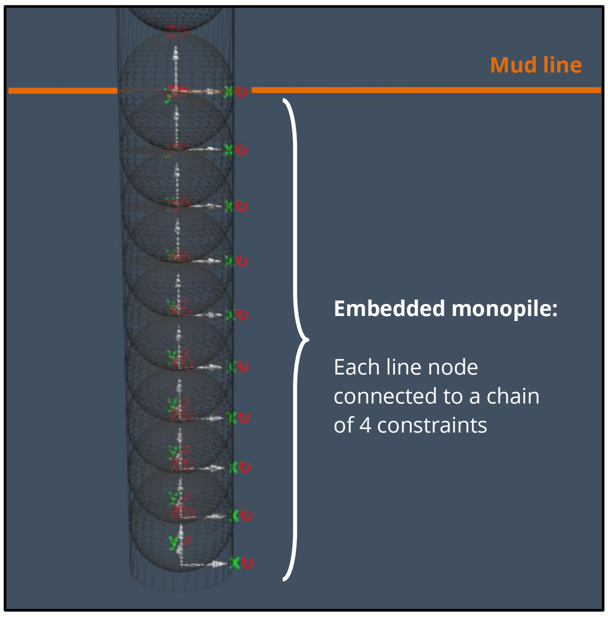
Node *→ torsional constraint (Rz) → rocking constraint (Ry & Rx) → vertical constraint (z) → horizontal constraint (x & y) →* ***Global*** *(anchored)*

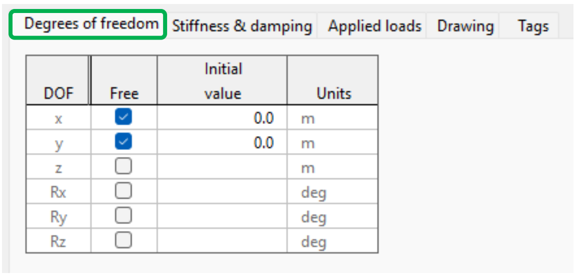
The stiffnesses calculated for each constraint account for the embedded depth (*h*) of each monopile line node and the constraint chains work to resist the loads applied to the embedded *monopile* in all 6 DOFs. Note that the sequence terminates at the *horizontal constraint (x & y)* constraint, which is connected (*anchored*) to the global axis system.

For each constraint, the corresponding DOFs are set to *free*, on the *degrees of freedom* page of the *constraint data* form. The necessary translational/rotational stiffnesses are calculated and assigned to each constraint, through the *stiffness & damping* page.

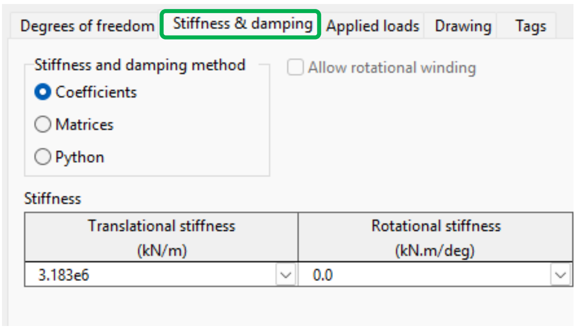
The stiffnesses required for the horizontal (*x*, *y*) and vertical (*z*) constraints are input using the *translational stiffness* data item. The stiffnesses required for the rocking (*Rx*, *Ry*) and twisting (*Rz*) constraints are input through the *rotational stiffness* data item.

As an example, the right-hand images (below) show the constraint settings required to model the horizontal spring stiffness (x & y) at the line node which sits level with the mud line.




Degrees of freedom | Stiffness & damping | Applied loads | Drawing | Tags

DOF	Free	Initial value	Units
x	✓	0.0	m
y	✓	0.0	m
z	☐		m
Rx	☐		deg
Ry	☐		deg
Rz	☐		deg


Degrees of freedom | Stiffness & damping | Applied loads | Drawing | Tags

Stiffness and damping method: [✓ Coefficients] Matrices Python

☐ Allow rotational winding

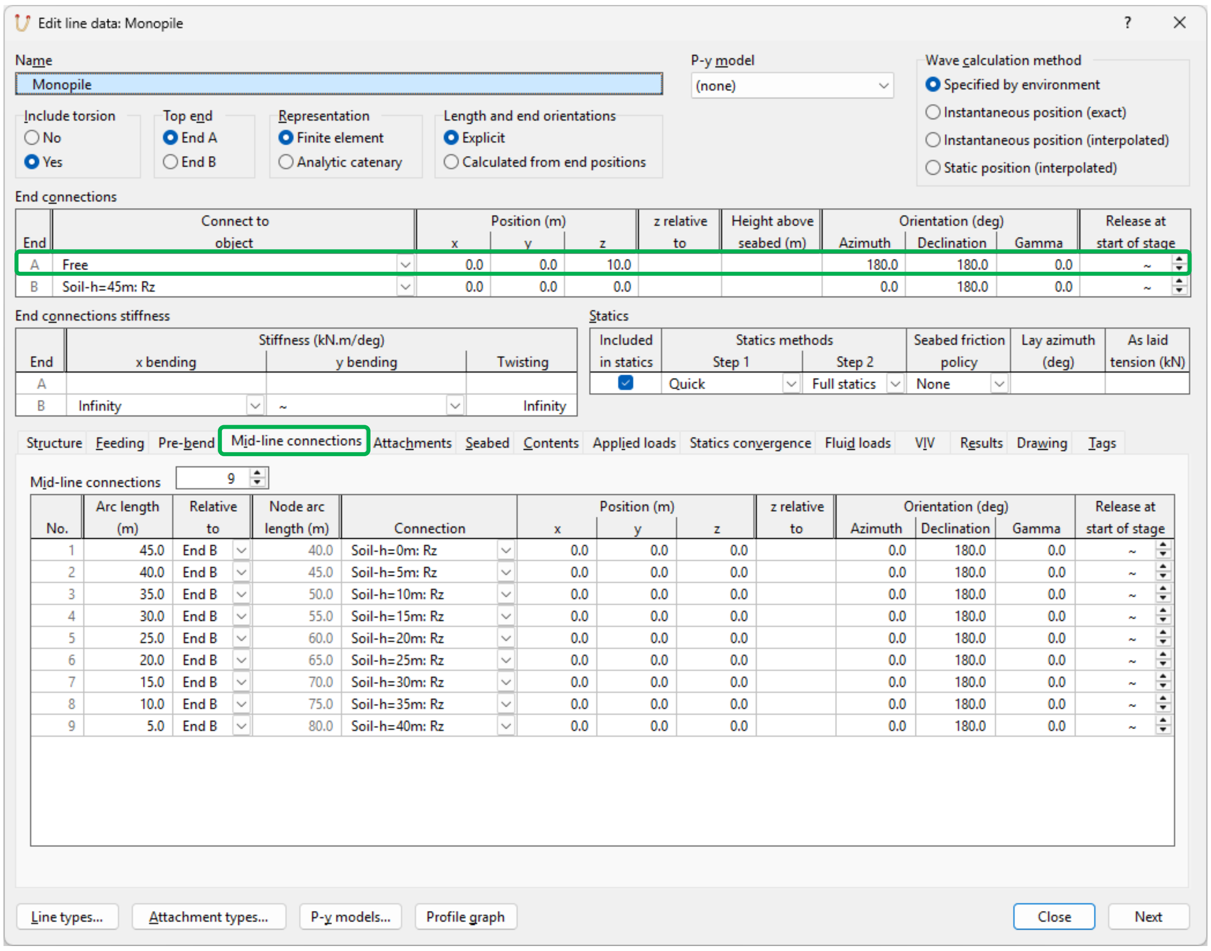
Stiffness

Translational stiffness (kN/m)	Rotational stiffness (kN.m/deg)
3.183e6	0.0

To ensure each of the *Monopile* lines nodes are connected to their respective constraint chains, *End B* of the line is connected to its constraint chain through the monopile's *end connection*. The other embedded nodes are connected using *mid-line connections*, as shown by the image below.

As these constraints effectively take over from the seabed model in OrcaFlex, the seabed *normal* and *shear stiffness* – specified on the *seabed* page of the *environment data* form – are both set to 0.

Note: this method of 'chaining' constraints is strictly only valid for small angular displacements. In this case, the soil stiffnesses are very high and so the displacement of each embedded monopile node is expected to be small. However, chaining constraints in this way may not be appropriate in cases where the constraint out-frame rotates significantly relative to the in-frame.


Edit line data: Monopile

Name: Monopile

P-y model: (none)

Wave calculation method: [✓ Specified by environment] Instantaneous position (exact) Instantaneous position (interpolated) Static position (interpolated)

Include torsion: No [✓ Yes]

Top end: [✓ End A] End B

Representation: [✓ Finite element] Analytic catenary

Length and end orientations: [✓ Explicit] Calculated from end positions

End connections

End	Connect to object	Position (m) x	y	z	z relative to	Height above seabed (m)	Orientation (deg) Azimuth	Declination	Gamma	Release at start of stage
A	Free	0.0	0.0	10.0			180.0	180.0	0.0	~
B	Soil-h=45m: Rz	0.0	0.0	0.0			0.0	180.0	0.0	~

End connections stiffness

End	Stiffness (kN.m/deg) x bending	y bending	Twisting
A			
B	Infinity	~	Infinity

Statics

Included in statics	Statics methods Step 1	Step 2	Seabed friction policy	Lay azimuth (deg)	As laid tension (kN)
✓	Quick	Full statics	None		

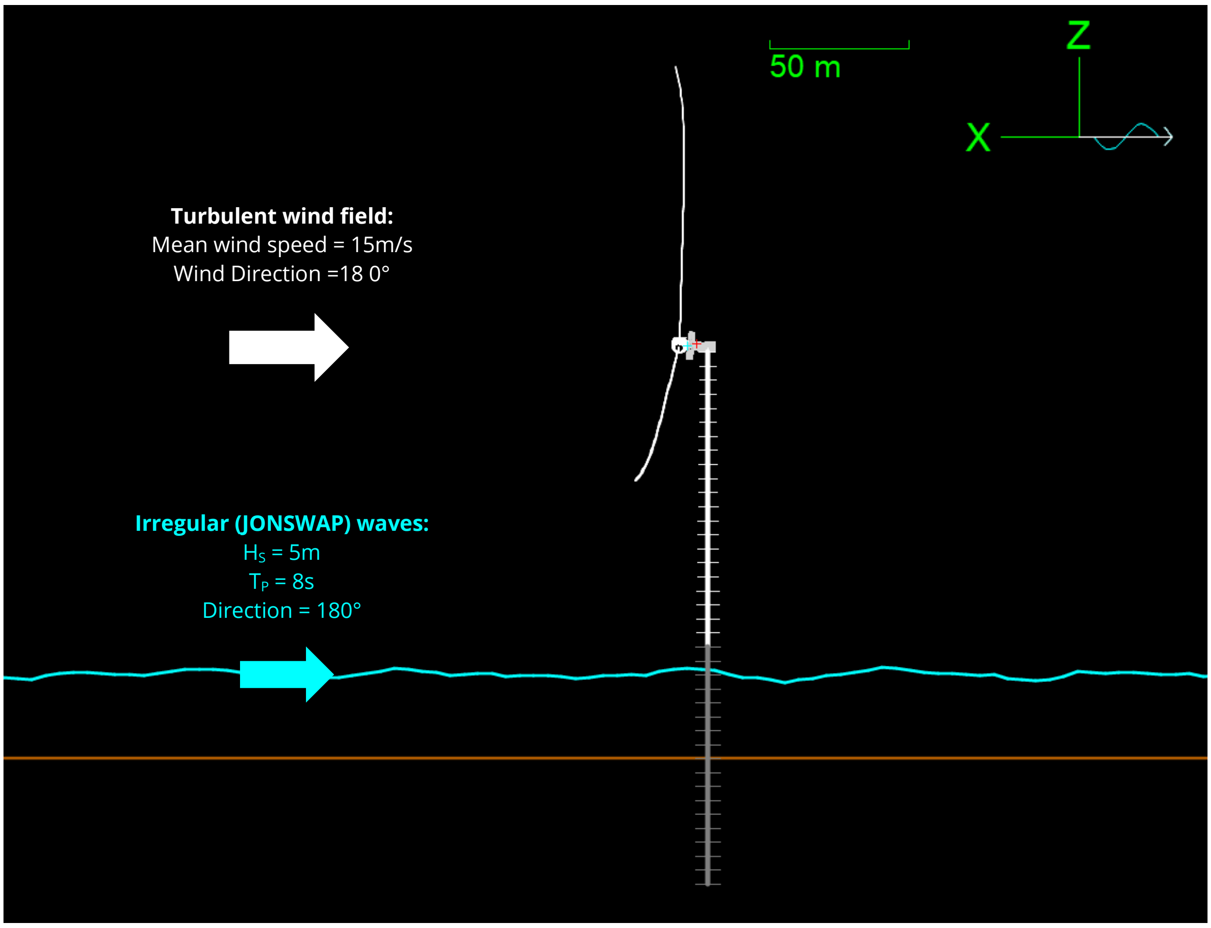
Structure | Feeding | Pre-bend | Mid-line connections | Attachments | Seabed | Contents | Applied loads | Statics convergence | Fluid loads | VIV | Results | Drawing | Tags

Mid-line connections: 9

No.	Arc length (m)	Relative to	Node arc length (m)	Connection	Position (m) x	y	z	z relative to	Orientation (deg) Azimuth	Declination	Gamma	Release at start of stage
1	45.0	End B	40.0	Soil-h=0m: Rz	0.0	0.0	0.0		0.0	180.0	0.0	~
2	40.0	End B	45.0	Soil-h=5m: Rz	0.0	0.0	0.0		0.0	180.0	0.0	~
3	35.0	End B	50.0	Soil-h=10m: Rz	0.0	0.0	0.0		0.0	180.0	0.0	~
4	30.0	End B	55.0	Soil-h=15m: Rz	0.0	0.0	0.0		0.0	180.0	0.0	~
5	25.0	End B	60.0	Soil-h=20m: Rz	0.0	0.0	0.0		0.0	180.0	0.0	~
6	20.0	End B	65.0	Soil-h=25m: Rz	0.0	0.0	0.0		0.0	180.0	0.0	~
7	15.0	End B	70.0	Soil-h=30m: Rz	0.0	0.0	0.0		0.0	180.0	0.0	~
8	10.0	End B	75.0	Soil-h=35m: Rz	0.0	0.0	0.0		0.0	180.0	0.0	~
9	5.0	End B	80.0	Soil-h=40m: Rz	0.0	0.0	0.0		0.0	180.0	0.0	~

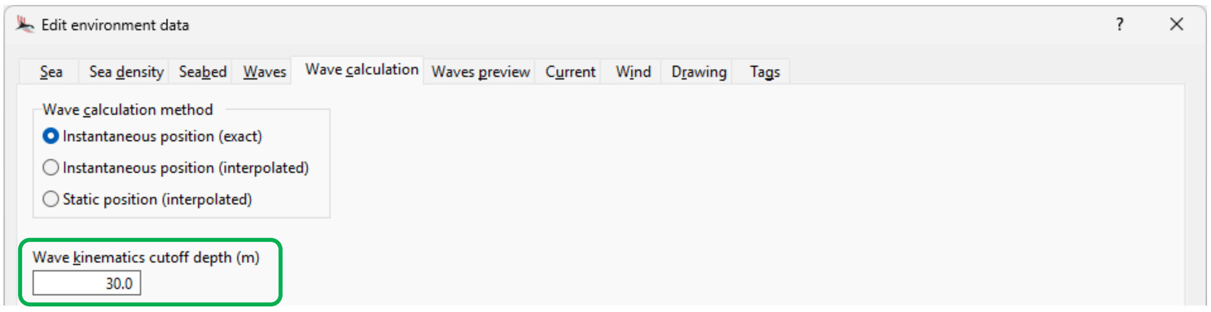
Environment

The simulation considers a build-up (*stage 0*) duration of 100s followed by a *stage 1* duration of 100s. Irregular (JONSWAP) waves, with a *significant wave height* (H_S) of 5m and *spectral peak wave period* (T_P) of 8s are also considered.



On the *wave calculation* page of the environment data form, we have specified a *wave kinematics cutoff depth* of 30m. This is measured downwards from the mean sea surface, and, below this depth, the wave kinematic fluid velocity and acceleration are taken to be zero. The default value here is *infinity*, which is equivalent to not having a cutoff depth at all.

As OrcaFlex does not automatically exclude wave kinematics below the seabed (mud line), specifying a nominal depth ensures the embedded monopile does not experience any wave loading. Note, the *wave kinematics cutoff depth* has no effect on fluid velocity due to current.



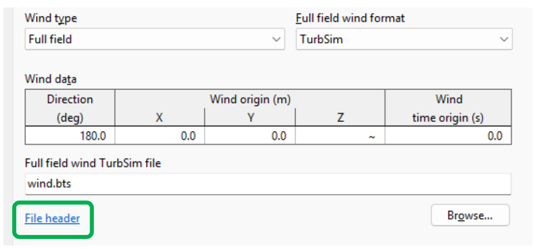
Wave calculation method
- [✓ Instantaneous position (exact)]
- Instantaneous position (interpolated)
- Static position (interpolated)

Wave kinematics cutoff depth (m): 30.0

Wind modelling

In this example, we have chosen to apply a turbulent wind flow in the simulation environment. This is applied using the *full field* wind model, which models the variation of wind velocity, in both space and time, with data specified in an external file. In this case, a TurbSim file (*wind.bts*) has been imported. The file contains a time series of wind velocity, in 3-dimensions, at points on an evenly spaced 2-dimensional grid (in the vertical yz-plane).

On the *wind* page of the *environment data* form, the name of the .bts file is specified along with the *wind direction* and *origin*; which determines how the .bts file's coordinate system is mapped on to the OrcaFlex coordinate system. Clicking *file header* provides some useful information about the imported wind data, including the size, position, and resolution of the 2D grid.

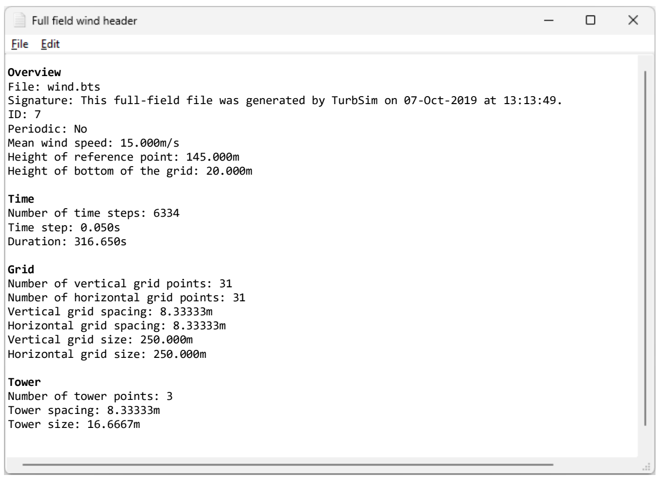

Wind type: Full field

Full field wind format: TurbSim

Wind data

Direction (deg)	Wind origin (m) X	Wind origin (m) Y	Wind origin (m) Z	Wind time origin (s)
180.0	0.0	0.0	~	0.0

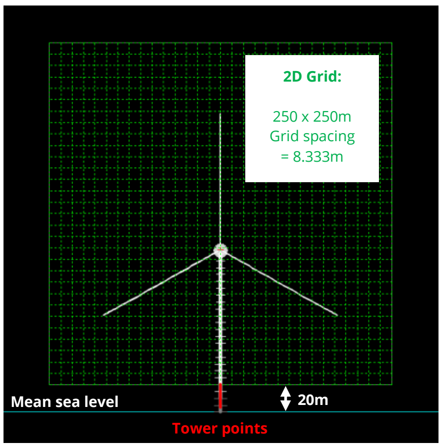
Full field wind TurbSim file: wind.bts

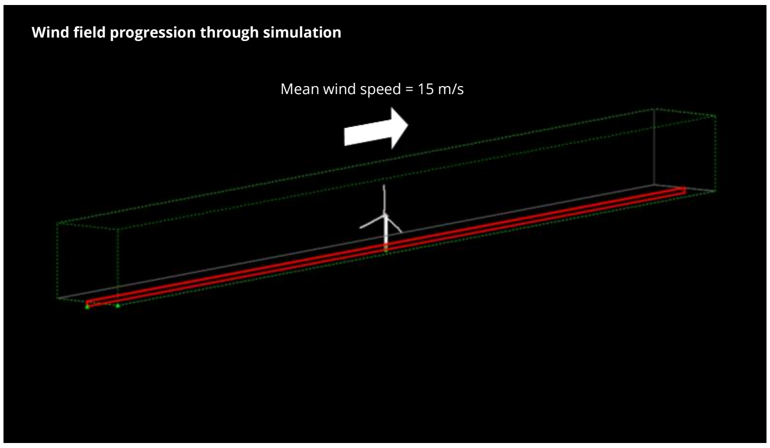


```
Overview
File: wind.bts
Signature: This full-field file was generated by TurbSim on 07-Oct-2019 at 13:13:49.
ID: 7
Periodic: No
Mean wind speed: 15.000m/s
Height of reference point: 145.000m
Height of bottom of the grid: 20.000m

Time
Number of time steps: 6334
Time step: 0.050s
Duration: 316.650s

Grid
Number of vertical grid points: 31
Number of horizontal grid points: 31
Vertical grid spacing: 8.33333m
Horizontal grid spacing: 8.33333m
Vertical grid size: 250.000m
Horizontal grid size: 250.000m

Tower
Number of tower points: 3
Tower spacing: 8.33333m
Tower size: 16.6667m
```

During the simulation, the grid moves forward at a mean speed of 15 m/s. In doing so, it sweeps through a volume in the model space within which the turbine is positioned; thus, imparting aerodynamic loading on the system. The images below provide a visual representation of this.

Here, the grid has been configured to pass the global origin (X=0, Y=0) at a simulation time of zero. This means that, at the beginning of stage 0, the full field wind starts some distance away from the turbine.



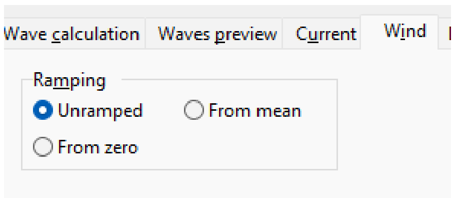


The *file header* also reports some information about the tower points included as part of the *.bts* file. As the bottom of the 2D grid is positioned 20m above the sea surface, the tower points help to ensure wind loads are received by the length of tower between the grid and the sea surface.

Based on the reported *file header* information, we have specified a *Z wind origin* value of ~, which means that the vertical origin is at the mean water level.

A *wind direction* of 180° is considered, which is the direction in which the 2D grid will propagate through the model space.

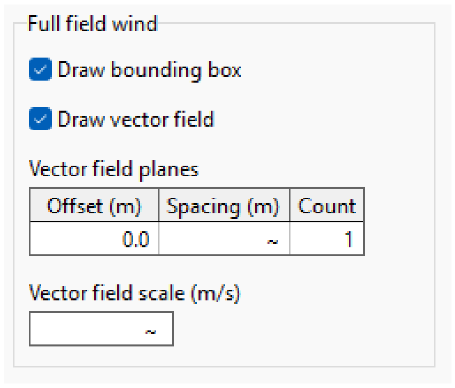
It is also possible to select a wind *ramping* option by selecting the appropriate radio button. We have not implemented any ramping, for this example, which means the wind velocity at the simulation start time is applied in statics.



Ramping

[✓ Unramped] From mean From zero

To help visualise the full field wind, a wire frame 3D box and arrows representing the wind vector field can be drawn using the options available on the *environment* data form, *drawing* page. You can switch these options on or off using the available check boxes. It is also possible to draw the vector field in the shaded graphics mode. Further details about this can be found on the help page, Modelling, data and results | Environment | Drawing.



Full field wind

✓ Draw bounding box

✓ Draw vector field

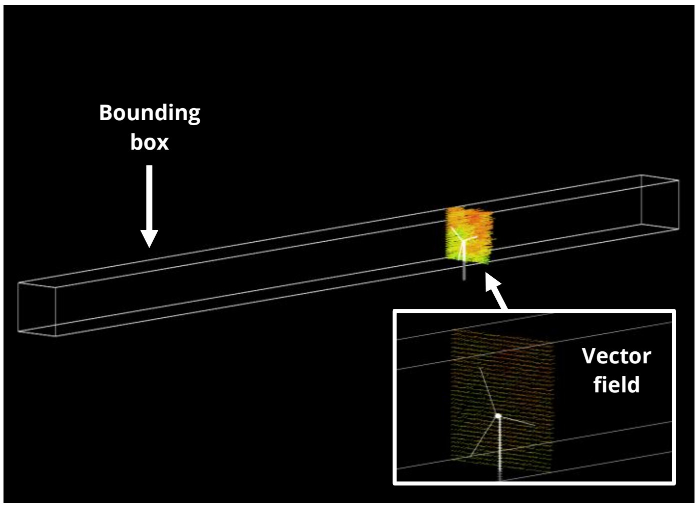
Vector field planes

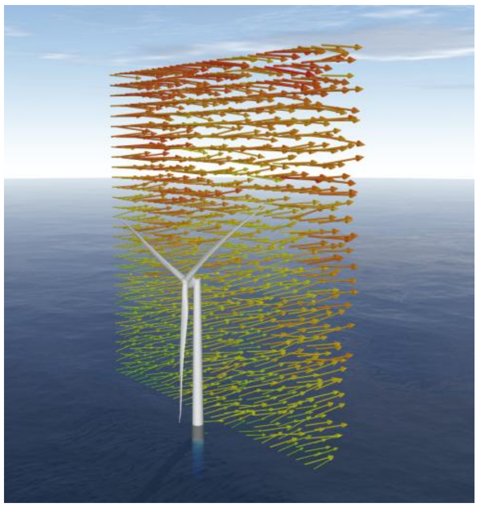
Offset (m)	Spacing (m)	Count
0.0	~	1

Vector field scale (m/s)

~

Further details related to general wind modelling in OrcaFlex can be found on the help page, Modelling, data and results | Environment | Wind data.





Results

Wind

Opening the workspace file named *K02 Full field wind.wrk* provides some visibility on the applied wind conditions.

The turbulent nature of the wind field is made evident by the speed and direction profiles shown over *stage 1* at the rotor hub position (*0*, *0*, *118.5* m). Note that the wind speed and direction, at the hub, both remain constant during the build-up phase (*stage 0*), until the grid eventually reaches the modelled system (at a simulation time of t = 0s). Again, it is possible to visualise this by ensuring the *draw bounding box* option is ticked on the *environment* data form, *drawing* page.

Rotor and generator response

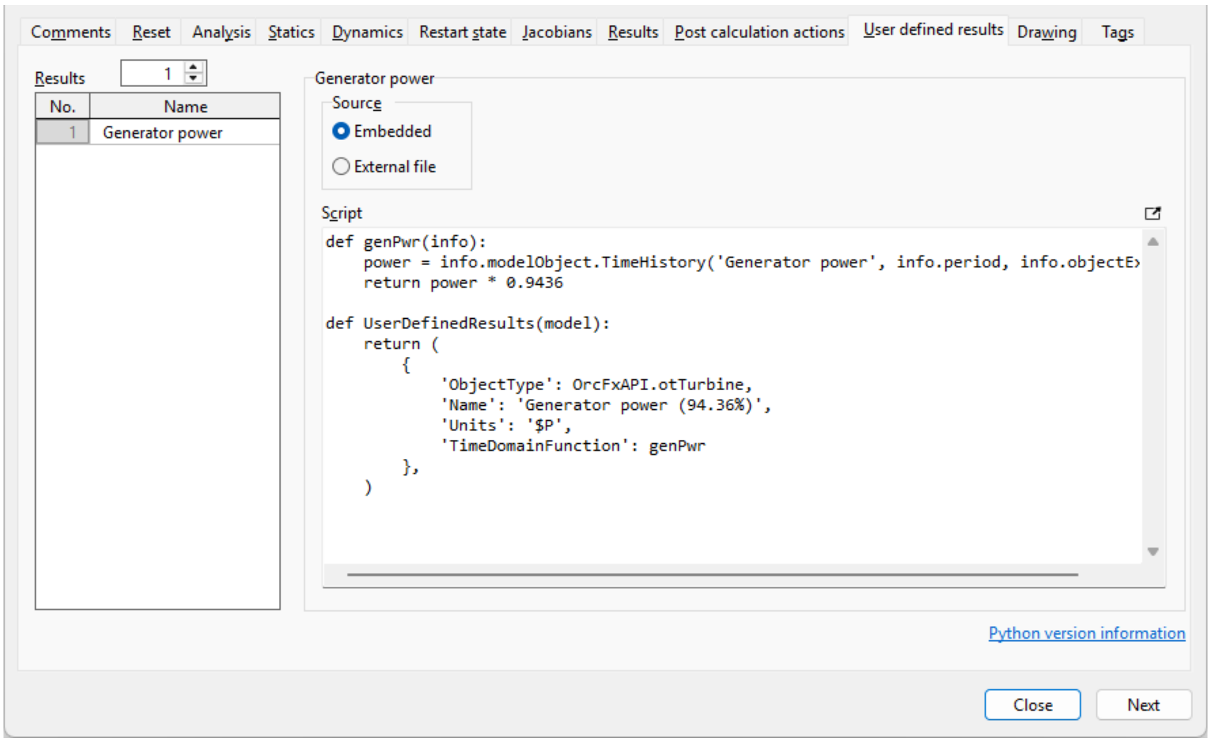
The workspace file, named *K02 Rotor and generator response.wrk*, opens some time history results for the *generator power (94.36%)* and *blade pitch* response over *stage 1*.

The top right-hand graph shows that the applied wind is, in the most part, above the 'rated' wind speed for the turbine (11 m/s), meaning the pitch controller is active (see the bottom right-hand graph). This helps to regulate power output from the turbine near its rated power (10MW).

Note, the generator power results account for the efficiency of the generator (94.36%). To achieve this, we have implemented a *user defined result*, specified on the corresponding page of the *general* data form. Here we have selected the *embedded* source option (pictured below), meaning the user defined results script can be written directly on the data form, rather than in an *external file*.

The *script* itself makes use of an existing results variable (*generator power*) to which a factor of 0.9436 is applied to calculate a new result named *generator power (94.36%)*.

For further details and examples related to user defined results, please see the Modelling, data and results | General data | User defined results help page.



Results 1

No.	Name
1	Generator power

Generator power

Source: [✓ Embedded] External file

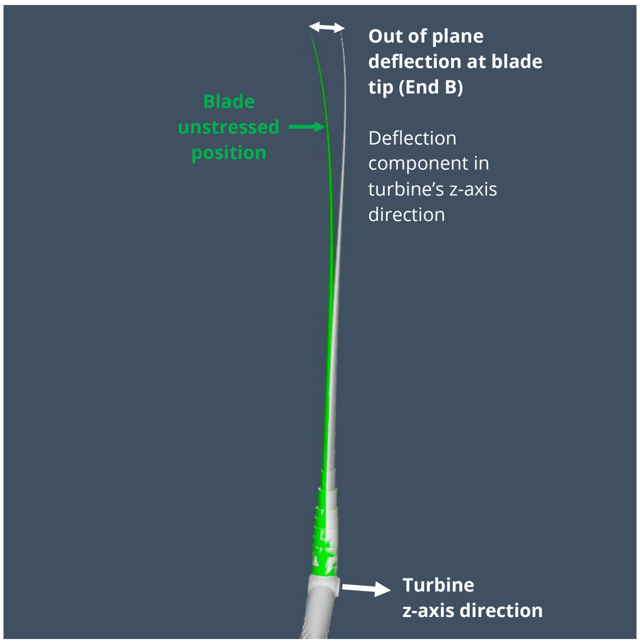
Script

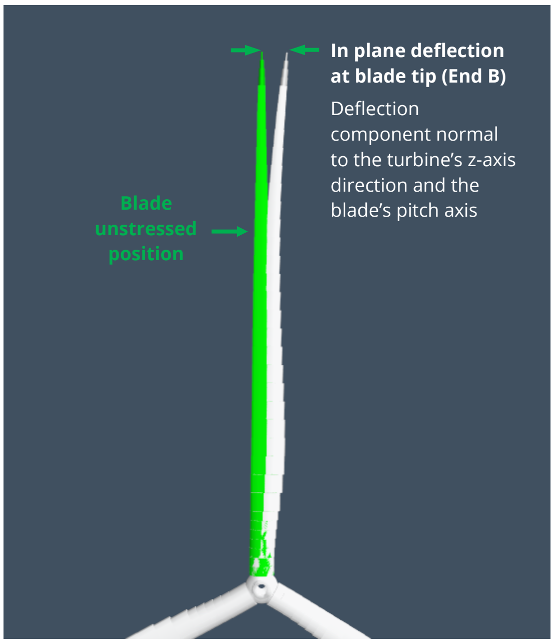
```
def genPwr(info):
    power = info.modelObject.TimeHistory('Generator power', info.period, info.objectE›
    return power * 0.9436

def UserDefinedResults(model):
    return (
        {
            'ObjectType': OrcFxAPI.otTurbine,
            'Name': 'Generator power (94.36%)',
            'Units': '$P',
            'TimeDomainFunction': genPwr
        },
    )
```

Blade tip response

Opening the workspace file named *K02 Blade tip response.wrk* produces time history results, over *stage 1*, showing the *out of plane deflection* and *in plane deflection* at the tip of *blade 1* (*End B*).

The blade deflection results, available at blade nodes, report a node's position relative to its unstressed position. The deflection component in the turbine's z-axis direction represents the *out of plane deflection* and its component normal to both the turbine's z-axis direction & the blade's pitch axis, is reported as the *in plane deflection* (pictured below).

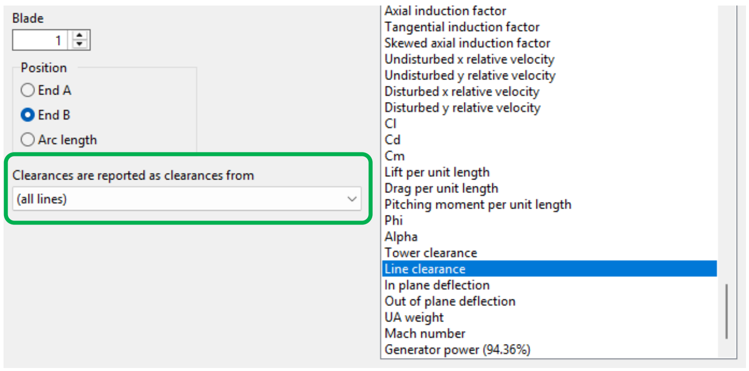

Out of plane deflection at blade tip (End B)
Deflection component in turbine's z-axis direction
Blade unstressed position
Turbine z-axis direction


In plane deflection at blade tip (End B)
Deflection component normal to the turbine's z-axis direction and the blade's pitch axis
Blade unstressed position

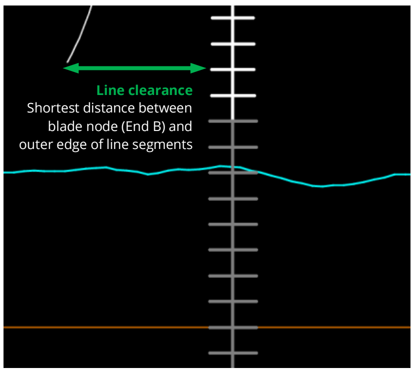
The bottom left-hand graph shows a time history of the *line clearance* results for *blade 1*. With reference to the left-hand image below, the results selection form lets you choose how to report clearances: (i) from the blade to all other lines or (ii) from the blade to a specified line. In this case, we have chosen to report the blade tip clearance from all lines, so the *line clearance* graph shows the shortest distance between the node at *End B* of blade 1 and the outer edges of the line segments in the model.

Note: the *line clearance* result does not account for the cross-sectional extent of the blade.

For further details relating to the turbine results, available in OrcaFlex, please refer to the Modelling, data and results | Turbines | Turbine results help page.


Blade: 1
Position: ○ End A [✓ End B] ○ Arc length
Clearances are reported as clearances from: (all lines)

Axial induction factor
Tangential induction factor
Skewed axial induction factor
Undisturbed x relative velocity
Undisturbed y relative velocity
Disturbed x relative velocity
Disturbed y relative velocity
Cl
Cd
Cm
Lift per unit length
Drag per unit length
Pitching moment per unit length
Phi
Alpha
Tower clearance
Line clearance
In plane deflection
Out of plane deflection
UA weight
Mach number
Generator power (94.36%)


Line clearance
Shortest distance between blade node (End B) and outer edge of line segments

Monopile response

Opening the workspace file named *K02 Monopile response.wrk* displays some *instantaneous value* range graphs showing the monopile response.

The upper right-hand graph shows the *x Morison force* experienced by the monopile line. This represents the x-component of the total force defined by Morison's equation, which is the sum of the *x drag force* and the *x fluid inertia force* (also shown by the bottom two graphs).

When playing the simulation replay (*Ctrl+R*), the graphs provide live feedback showing how the fluid loads vary during the simulation. Note that the embedded monopile – arc length range of 40 to 85m – does not experience any wave loading because the *wave kinematics cutoff depth* is set to 30m.

Soil response

Finally, the workspace file named *K02 Soil response.wrk*, displays some results showing how the constraint chain, at the mud line (*h=0m*), works to represent the interaction between the monopile and the soil.

For the constraint representing the horizontal soil stiffness (*Soil-h=0m: y-x*), the lower left-hand graph shows a time history of the *out-frame X*; which is the global X position of the out-frame. For the constraint representing the rotational spring stiffness of the soil (*Soil-h=0m: Rz*), the lower right-hand graph shows a time history of the *out-frame dynamic Rz*. This represents the orientation of the out-frame relative to its static orientation.

In this case, the soil stiffnesses assigned to each degree of freedom are very high. Consequently, the calculated out-frame displacements are small.

The out-frame rotations are also small. This is important when considering the small-angle approximation which accompanies this method of chaining constraints together.

Lastly, the upper right-hand graph shows the *in-frame connection force* results for the horizontal constraint (x & y). With this constraint being connected to the global axis system (*anchored*), the chain effectively terminates here and so the reported force is the resultant load the line node imparts on the constraint chain.

Further details about the available results for constraints can be found on the Modelling, data and results | Constraints | Results help page.