



# VelantisWind

## Energy / AEP Module

Practical guide to inputs, wind resource, PyWake, AEP calculation and output review

User manual for the energy module: input preparation, resource loading, physical model selection, calculation execution and results review.

User guide

VelantisWind · Energy / AEP Module

# Contents

Follow this order the first time. Afterwards, use each block as a validation checklist.

|          |                                                        |           |
|----------|--------------------------------------------------------|-----------|
| <b>1</b> | Basic concepts                                         | <b>3</b>  |
| 1.1      | Wind resource: what it is and where it comes from      | 3         |
| 1.2      | How AEP is calculated                                  | 4         |
| 1.3      | Physical models used by PyWake                         | 5         |
| <b>2</b> | Step 1 - Wind turbine model and power/Ct curve         | <b>6</b>  |
| <b>3</b> | Step 2 - Layout, X,Y coordinates and CRS               | <b>7</b>  |
| <b>4</b> | Step 3 - Wind resource: loading and understanding it   | <b>8</b>  |
| 4.1      | WRG/ZIP: real file columns and node line               | 9         |
| 4.2      | WRG: sectors, scaling and multiple heights             | 10        |
| 4.3      | WASP / Surfer grids: how they are loaded in the plugin | 11        |
| 4.4      | Surfer grids: minimum files and variables              | 12        |
| 4.5      | How to read a Surfer Grid DSAA                         | 13        |
| <b>5</b> | Step 4 - Ambient turbulence TI                         | <b>14</b> |
| <b>6</b> | Step 5 - PyWake engine and physical models             | <b>15</b> |
| 6.1      | Plugin PyWake selectors                                | 16        |
| 6.2      | Recommended settings and links                         | 17        |
| <b>7</b> | Step 6 - Calculation, maps and results                 | <b>18</b> |
| <b>8</b> | Final checklist and common errors                      | <b>19</b> |
| <b>9</b> | Full calculation example with screenshots              | <b>20</b> |
| 9.1      | Wind resource selection                                | 21        |
| 9.2      | Loaded resource and ambient turbulence TI              | 22        |
| 9.3      | Physical models and calculation execution              | 23        |

## 1.1 What the wind resource is and where it comes from

The wind resource describes how wind blows at the site: speeds, directions, frequencies, heights and spatial variation.

In VelantisWind, the wind resource is an input. The plugin does not calculate it from scratch: it reads prepared files (WRG/ZIP or WAsP/Surfer grids) and combines them with the power curve, layout and PyWake models to estimate AEP.

### What it contains

Speed and direction distributions, sector frequencies and Weibull A/k parameters by sector, height and position.

### Where it comes from

It should come from measurements, resource campaigns, consultants or specialized software/providers: Vortex, UL/Windnavigator, WAsP, WindPRO, Meteodyn or ContinuumWind as a free/open-source alternative.

### What VelantisWind does

It checks that the file loads, covers the layout and that heights, sectors and variables are consistent before the calculation.

## Resource formats that the module can read

### WRG / ZIP

Grid file with resource nodes. Each node includes coordinates, height, total A/k and data by directional sector. It is commonly supplied by external tools or providers.

### WAsP / Surfer grids

Folder with .grd grids in DSAA format. Each grid represents a spatial variable: Weibull-A, Weibull-k, sector frequency, speed-up, turn, elevation, etc.

### Important idea

AEP is only as defensible as the starting resource. WRG/ZIP and WAsP/Surfer grids are alternative routes: for each calculation choose one or the other based on the resource available; both are not required.

## 1.2 How AEP is calculated

AEP combines resource, power curve, layout and wake models to estimate expected annual energy.

### Base formula

$$AEP_i = 8760 \int_0^{\infty} P(v_{\text{eff}}) f(v) dv$$

For each turbine, power  $P_i$  is integrated at the effective speed  $v_{\text{eff}}$  and weighted by the wind distribution  $f(v)$ .  
The 8760 factor converts that annual probability into hours per year.

### Wind farm AEP

$$AEP_{\text{parque}} = \sum_{i=1}^N AEP_i$$

The total AEP is obtained by summing the AEP of all turbines in the layout.

### Discrete version in the plugin

$$AEP_i \approx 8760 \sum_s p_s \sum_b P(v_{b, \text{eff}}) p(v_b | A_s, k_s) \Delta v$$

In practice, the calculation is discretized by sectors  $s$  and speed bins  $b$ . For each sector, frequency  $p_s$  and Weibull parameters  $A_s$  and  $k_s$  are used. The power curve is evaluated using the effective speed after wake losses.

#### 1. Free-flow resource

Probability of speeds and directions before accounting for the presence of other turbines.

#### 2. $P(v)$ curve

The power curve converts each wind speed into instantaneous electrical power.

#### 3. Wakes and $v_{\text{eff}}$

PyWake modifies the incoming wind speed downstream or upstream blockage depending on the active model.

#### 4. Result

AEP per turbine and total wind farm AEP are obtained. From there, wake loss can also be calculated.

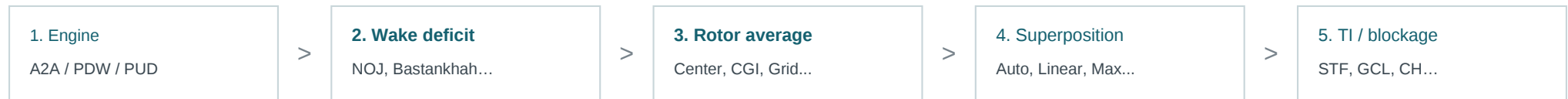
$$\text{Wake loss(\%)} = \frac{AEP_{\text{free}} - AEP_{\text{wake}}}{AEP_{\text{free}}} \times 100$$

## 1.3 Physical models used by PyWake

There is no single global model: the final calculation is a combination of compatible families.

In the plugin you choose a calculation engine, a wake deficit model, a rotor-average model, a superposition rule and, depending on the case, turbulence or blockage families. Each selector changes a specific part of the calculation physics.

### Calculation chain in the plugin



### What each family controls

#### Motor PyWake

Controls how turbine interactions are propagated. PDW is simpler; A2A and PUD allow more complete interactions.

#### Wake deficit

Defines the mathematical wake shape and velocity deficit: NOJ, Bastankhah, Niayifar, etc.

#### Rotor average

Decides whether speed is evaluated at the rotor center or averaged over several disk points.

#### Superposition

Specifies how multiple wakes are combined on the same rotor point. The available option depends on the rest of the model.

#### Added turbulence

Adds wake-induced turbulence to the ambient turbulence from the resource and affects wake recovery.

#### Blockage / induction

Represents upstream effects of the rotor and is applied only with compatible engines or more complete configurations.

#### Compatibility note

The plugin only allows compatible combinations of engine, wake deficit, superposition, turbulence and blockage.

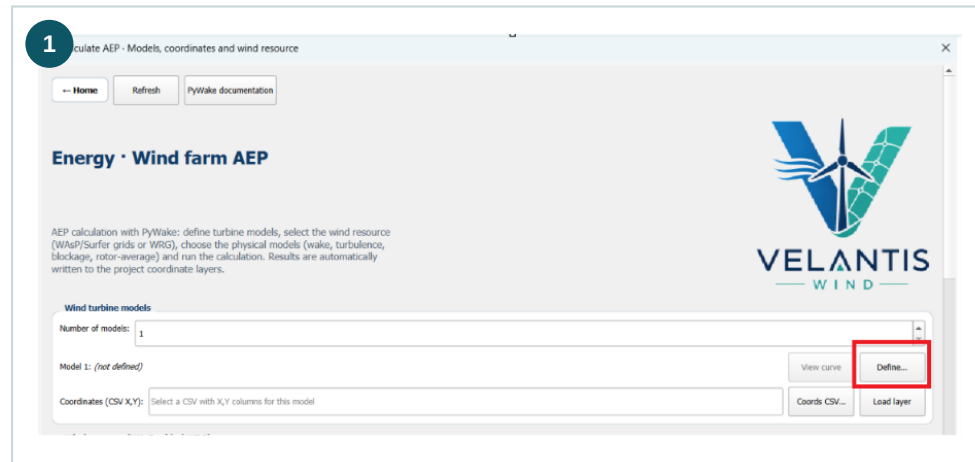
## 2. Step 1 - Wind turbine model

Define the turbine and, if you have a CSV/TXT file, load its power curve from the TXT/CSV tab.

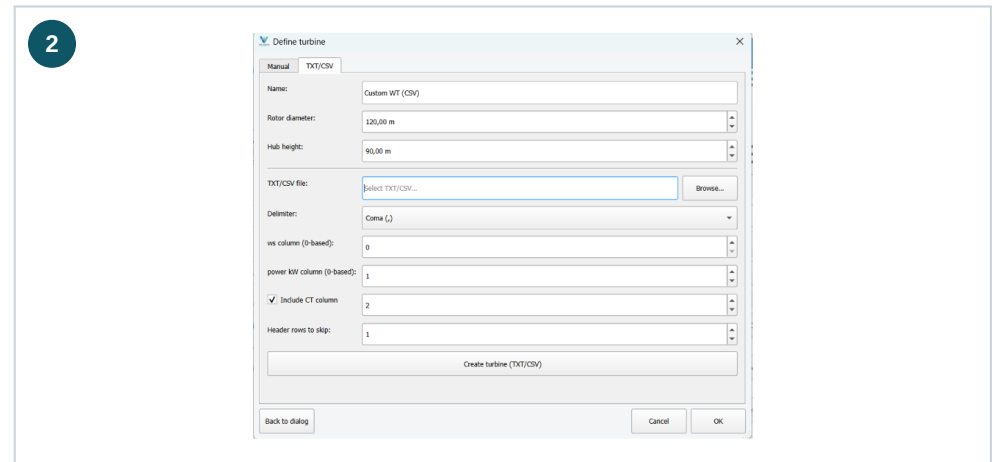
1. Click "Define..." in Wind turbine models

2. Open the TXT/CSV tab

3. Load the file and create the turbine



In the main window, click Define... under Wind turbine models.



In Define turbine, use the TXT/CSV tab if you have a curve file.

### Accepted format

|    | A      | B        | C   |
|----|--------|----------|-----|
| 1  | ws_m_s | power_kW | Ct  |
| 2  | 3.0    | 15.0     |     |
| 3  | 4.0    | 69.0     |     |
| 4  | 5.0    | 174.0    |     |
| 5  | 6.0    | 327.0    |     |
| 6  | 7.0    | 539.0    | 837 |
| 7  | 8.0    | 813.0    | 825 |
| 8  | 9.0    | 1131.0   | 766 |
| 9  | 10.0   | 1456.0   | 646 |
| 10 | 11.0   | 1741.0   | 503 |
| 11 | 12.0   | 1931.0   | 381 |
| 12 | 13.0   | 1979.0   | 292 |
| 13 | 14.0   | 1992.0   | 229 |
| 14 | 15.0   | 1998.0   | 185 |
| 15 | 16.0   | 1999.0   | 152 |
| 16 | 17.0   | 2000.0   | 127 |
| 17 | 18.0   | 2000.0   | 109 |
| 18 | 19.0   | 2000.0   | 95  |

The file may include the columns ws\_m\_s, power\_kW and Ct. The example power\_curve.csv follows this format.

### Recommended setup

- TXT/CSV tab · Delimiter: Comma (,)
- ws column: 0 · power kW column: 1
- Include Ct: enabled, column 2
- Header rows to skip: 1

Then click Create turbine (TXT/CSV) and finally OK.

### What to check next

Use View curve to check power and Ct. Power should rise to P\_nom and then stabilize. Ct should not show odd jumps or impossible values.

### 3. Step 2 - Layout, coordinates and CRS

Load the layout from a CSV with X,Y or from a point layer and check that the CRS is consistent.

1. Prepare the CSV or point layer

2. Use "Coords CSV..." or "Load layer"

3. Check CRS, scale and coverage

1

|    | A        | B         |
|----|----------|-----------|
| 1  | X        | Y         |
| 2  | 349209.0 | 4637646.0 |
| 3  | 348936.0 | 4637327.0 |
| 4  | 348824.0 | 4636980.0 |
| 5  | 349508.0 | 4636705.0 |
| 6  | 349620.0 | 4636349.0 |
| 7  | 349912.0 | 4635999.0 |
| 8  | 350054.0 | 4635724.0 |
| 9  | 350207.0 | 4635452.0 |
| 10 | 350614.0 | 4635060.0 |
| 11 | 350991.0 | 4634726.0 |
| 12 | 351254.0 | 4634339.0 |
| 13 | 350134.0 | 4634771.0 |
| 14 | 350376.0 | 4634243.0 |
| 15 | 350484.0 | 4633971.0 |
| 16 | 351088.0 | 4638921.0 |

Example CSV with X and Y columns in metric coordinates.

2

Calculate AEP · Models, coordinates and wind resource

Home Refresh PyWake documentation

**Energy · Wind farm AEP**

AEP calculation with PyWake: define turbine models, select the wind resource (WAsP/Surfer grids or WRG), choose the physical models (wake, turbulence, blockage, rotor-average) and run the calculation. Results are automatically written to the project coordinate layers.

Wind turbine models

Number of models: 1

Model 1: (not defined)

Coordinates (CSV X,Y): Select a CSV with X,Y columns for this model

View curve Define...

Coords CSV... Load layer

Click Coords CSV... to import the CSV, or Load layer if you already have a QGIS layer.

| Check    | Sign that it is correct                                   |
|----------|-----------------------------------------------------------|
| Columns  | Clean X and Y values in meters.                           |
| CRS      | Same CRS as the resource, or an intentional reprojection. |
| Scale    | Reasonable distances; avoid decimal degrees.              |
| Coverage | All turbines fall inside the resource.                    |

#### Common error

Loading degree coordinates as if they were meters. It may look right visually, but the physical calculation will be wrong.

#### Recommended use

If the layer is already in QGIS, Load layer is usually the quickest route.

#### Practical summary

Use X and Y in meters, check CRS, scale and coverage, and make sure all turbines are inside the resource.

## 4. Step 3 - Wind resource: loading and understanding it

Choose one main resource route: WRG/ZIP or WAsP/Surfer grids. If several Surfer tiles are contiguous, merge them first.

Step 3

| Input               | When to use it                                           | What is selected                                    | Key check                                                          |
|---------------------|----------------------------------------------------------|-----------------------------------------------------|--------------------------------------------------------------------|
| WRG / ZIP           | When the resource comes as a WRG or ZIP file.            | One or more WRG/ZIP files.                          | Height, CRS, extent, sectors and layout coverage.                  |
| WAsP / Surfer grids | When the resource comes as a folder of .grd files.       | The folder containing the prepared Surfer grids.    | Minimum variables, Sector All, DSAA format and recognizable names. |
| Merge resource      | When the Surfer resource is split into contiguous tiles. | Several tile folders and one unified output folder. | Same CRS, compatible cell size, sectors and heights consistent.    |
| TI raster(s)        | Optional ambient turbulence or external to the resource. | One or more TI rasters and their heights.           | Unit %, fraction, NODATA, height and coverage.                     |

### Practical reading of the Wind resource block

1. Choose WRG/ZIP or WAsP/Surfer grids as the main source.
2. For Surfer grids, select a folder, not individual .grd files.
3. If you use Surfer tiles, run Merge resource first and then load the output folder.
4. If you do not use tiles, place the required Surfer grids in one folder and select it.
5. TI is independent and can be added separately.

### Minimum Surfer input folder

For each sector and height: Weibull-A, Weibull-k, Sector frequency, Orographic speed, Orographic turn and Mean speed.  
 For Sector All: at least Elevation for each height.  
 TI is optional if you want to read turbulence from GRD.

Important: use Merge resource only for Surfer folders split into tiles; otherwise load the single prepared Surfer folder directly.

## 4.1 WRG/ZIP: real file columns

Quick reference to understand the grid header and the fixed part of each WRG node.

Before reading WRG columns, check in the Wind resource block which resource route you are using. This page applies only to WRG/ZIP; if you work with Surfer grids, continue with section 4.3.

### Line 1 - grid header

| Col. | Example | Data      | Meaning                                   |
|------|---------|-----------|-------------------------------------------|
| 1    | 355     | Nx        | Number of grid points in the X direction. |
| 2    | 354     | Ny        | Number of grid points in the Y direction. |
| 3    | 337923  | Xmin      | X coordinate of the lower-left node.      |
| 4    | 4617911 | Ymin      | Y coordinate of the lower-left node.      |
| 5    | 100     | cell_size | Grid cell size, in meters.                |

### Node line - columns before the sectors

| Col. | 80 m    | 120 m   | Data          | Practical reading                                                                   |
|------|---------|---------|---------------|-------------------------------------------------------------------------------------|
| 1    | 360823  | 360823  | X             | X coordinate of the resource node. Same CRS as the layout.                          |
| 2    | 4635311 | 4635311 | Y             | Y coordinate of the resource node.                                                  |
| 3    | 831     | 831     | Z / elevation | Ground level or elevation, in meters.                                               |
| 4    | 80.0    | 120.0   | Height        | Height above ground to which the wind climate applies.                              |
| 5    | 6.7     | 7.4     | A total       | Total equivalent Weibull A of the node, in m/s.                                     |
| 6    | 1.90    | 1.88    | k total       | Total equivalent Weibull k of the node.                                             |
| 7    | 233     | 318     | Power density | Power density, usually W/m <sup>2</sup> ; depending on export it may be production. |
| 8    | 16      | 16      | Nsec          | Number of directional sectors. Here there are 16.                                   |

Note: if the WRG includes an initial text field (for example GridPoint), it may be omitted when splitting by spaces if it is empty; this practical table therefore starts at X.

## 4.2 WRG: sectors, scaling and multiple heights

How to read sector triplets, how they are scaled and how several WRGs at different heights are used.

### Key idea

A WRG has a grid header and, for each node, 8 fixed columns. Then each sector adds three values: frequency, Weibull A and Weibull k. With 16 sectors, one node line contains  $8 + 16 \times 3 = 56$  values.

| Sector | Columns    | What it represents                               | Typical scaling                |
|--------|------------|--------------------------------------------------|--------------------------------|
| 1      | 9, 10, 11  | Frequency, Weibull A and Weibull k of sector 1.  | freq/10 = %, A/10 = m/s, k/100 |
| 2      | 12, 13, 14 | Frequency, Weibull A and Weibull k of sector 2.  | Same scaling.                  |
| 3      | 15, 16, 17 | Frequency, Weibull A and Weibull k of sector 3.  | Same scaling.                  |
| ...    | ...        | The pattern is repeated for all sectors.         | ...                            |
| 16     | 54, 55, 56 | Frequency, Weibull A and Weibull k of sector 16. | Same scaling.                  |

### Quick WRG validation

- 1) Check the first 8 columns: X, Y, elevation, height, total A, total k, power density and number of sectors.
- 2) From column 9 onward, each sector uses three values: frequency, Weibull A and Weibull k.
- 3) With 16 sectors, the line must have 56 values.

### Multiple-height loading in the plugin

If you have resource at several heights, select the corresponding WRGs together, for example 80 m and 120 m. They must share area, CRS, sector structure and consistent extent. The plugin will use the height corresponding to the hub when possible.

### Practical criterion

To validate a WRG, you do not need to interpret every sector value. Check height, number of sectors, typical scaling and that the layout falls inside the grid.

## 4.3 Merge resource: WAsP / Surfer tile mosaicking

This utility merges contiguous resource tiles so they can later be simulated as a single Surfer folder.

Definition: Merge resource mosaics contiguous WAsP/Surfer tiles. The mosaic is built by variable + sector + height. It works with one height or several heights (90 m, 120 m, etc.). Heights are kept separate; they are not mixed.

### Workflow



### Example

#### Case A · one height

You have Tile\_West\_90m and Tile\_East\_90m. Both contain the same variables and sectors. Merge resource generates a continuous mosaic at 90 m. Then load the output folder as WAsP/Surfer grids.

#### Case B · several heights

You have Tile\_West and Tile\_East with grids at 90 m and 120 m. Merge resource creates mosaics separately for 90 m and 120 m within the same output. The plugin will use the height that corresponds when it calculates.

Quick rules: same CRS, compatible cell size and consistent variables/sectors. If a variable or height is missing in a tile, that part may remain incomplete. pywake\_compat can be the clean output of this utility, but it is not a general rule: what matters is selecting the folder containing the valid Surfer grids.

## 4.4 Surfer grids: minimum files and variables

Summary of the minimum files required by the resource and what to check before calculating.

|                                                           |                  |             |          |
|-----------------------------------------------------------|------------------|-------------|----------|
| Resource grid 1 Sector 1 Height 70m Mean speed.grd        | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 70m Orographic speed.grd  | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 70m Orographic turn.grd   | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 70m Sector frequency.grd  | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 70m Weibull-A.grd         | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 70m Weibull-k.grd         | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 90m Mean speed.grd        | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 90m Orographic speed.grd  | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 90m Orographic turn.grd   | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 90m Sector frequency.grd  | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 90m Weibull-A.grd         | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 90m Weibull-k.grd         | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 125m Mean speed.grd       | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 125m Orographic speed.grd | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 125m Orographic turn.grd  | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 125m Sector frequency.grd | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 125m Weibull-A.grd        | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |
| Resource grid 1 Sector 1 Height 125m Weibull-k.grd        | 28/06/2026 17:01 | Archivo GRD | 1,501 KB |

Typical structure: sector, height and variable.

| Group                       | Minimum files / variables                                                                 |
|-----------------------------|-------------------------------------------------------------------------------------------|
| Each sector and each height | Weibull-A, Weibull-k, Sector frequency, Orographic speed, Orographic turn and Mean speed. |
| Sector All                  | Elevation at minimum for each height.                                                     |
| Optional                    | Turbulence intensity / TI, if TI should be read from GRD.                                 |

### Minimum check

- The .grd files must be prepared and recognizable.
- Minimum by sector and height: Weibull-A, Weibull-k, Sector frequency, Orographic speed, Orographic turn and Mean speed.
- Sector All must contain at least Elevation.

### What to remember in the plugin

- No tiles: select the folder that directly contains the required Surfer grids.
- Tiles: first run Merge resource, then load the output folder.
- Folder name is not important: pywake\_compat is only an example.
- If the resource does not load, first check variables, names and DSAA format.

Note: pywake\_compat is only a folder name; use any folder containing prepared Surfer grids.

If the input is tiled, use Merge resource first and load the output folder.

## 4.5 How to read a Surfer Grid DSAA

Basic reading of the DSAA header to verify variable, dimensions and spatial extent.

```
DSAA
284 361
346850 361000
4632000 4650000
2.650438 4.271874
3.242175 3.241101 3.265683 ...
```

| Line        | What it represents                                |
|-------------|---------------------------------------------------|
| 1           | DSAA signature: identifies the Surfer ASCII Grid. |
| 2           | Number of columns and rows.                       |
| 3           | Grid X range.                                     |
| 4           | Grid Y range.                                     |
| 5           | Z range or value range.                           |
| From line 6 | Matrix values, row by row.                        |

| Variable .grd    | Physical meaning        | Practical handling                                            |
|------------------|-------------------------|---------------------------------------------------------------|
| Weibull-A        | Sector A parameter.     | Use as provided if it is in physical units.                   |
| Weibull-k        | Sector k parameter.     | Use as provided.                                              |
| Sector frequency | Frequency by sector.    | If provided in %, convert to a 0-1 fraction when appropriate. |
| Orographic speed | Orographic speed-up.    | Wind multiplier or correction.                                |
| Orographic turn  | Direction turn.         | Angular wind correction.                                      |
| Mean speed       | Mean speed.             | Diagnostic variable and resource support.                     |
| Elevation        | Topography / elevation. | Must be in Sector All.                                        |

## 5. Step 4 - Ambient turbulence TI

Load the TI raster or define a reasonable fallback, and make sure the unit and height are clear.

```
ncols 319
nrows 313
xllcorner -4.948558706907
yllcorner 41.702800414501
NODATA_value -999
9.94 9.93 9.92 10.00 ...
```

| Case                     | What to do                                                   |
|--------------------------|--------------------------------------------------------------|
| I have a TI raster       | Load it in Choose TI raster(s).                              |
| I have several raster(s) | Enter heights for vertical interpolation.                    |
| I do not have TI         | Use Ambient TI fallback.                                     |
| AEP barely changes       | This may be normal if the wake model is not sensitive to TI. |

Practical reading: if the raster shows 9.94, it is interpreted as approximately 9.94%, i.e. 0.0994.  
The fallback should only be used as a uniform assumption.

## 6. Step 5 - PyWake engine and physical models

Choose the calculation engine and physical models using an initial simple and consistent configuration.

| Motor                    | Wakes | Blockage | Speed  | Typical use                      |
|--------------------------|-------|----------|--------|----------------------------------|
| PropagateDownwind        | Yes   | No       | Fast   | First tests and sweeps.          |
| PropagateUpDownIterative | Yes   | Yes      | Medium | Balance between detail and cost. |
| All2AllIterative         | Yes   | Yes      | Slow   | More complete calculation.       |

| Selector        | What it decides                 | Practical rule                                                        |
|-----------------|---------------------------------|-----------------------------------------------------------------------|
| Wake model      | Wake velocity deficit.          | Start with default; use a TI-sensitive model if you want to study TI. |
| Rotor averaging | How the rotor disk is averaged. | More points = more realism and more time.                             |
| Superposition   | How several wakes are combined. | Automatic or SquaredSum is usually a good start.                      |

Quick rule: first calculate with a stable configuration. Then change only one thing at a time to compare results.

[More information: official PyWake documentation \(DTU\)](#)

## 6.1 Plugin PyWake selectors

Summary of the real interface options and the practical rule for choosing them.

| Block               | Main options                                                                                           | Practical rule                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| Motor PyWake        | PDW / PropagateDownwind; A2A / All2AllIterative; PUD / PropagateUpDownIterative.                       | PDW for speed. A2A/PUD when you want iteration, upstream effects or blockage.                                         |
| Wake model          | NOJ, TurboNOJ, BastankhahGaussian, NiayifarGaussian, TurboGaussian, ZongGaussian.                      | NOJ/TurboNOJ for screening. Bastankhah as a robust baseline. Niayifar/Zong/TurboGaussian if you want TI sensitivity.  |
| Rotor-average       | None, RotorCenter, CGIRotorAvg(7/9/21), EqGridRotorAvg.                                                | More points = better rotor disk representation, but more time. CGI7 is usually a good balance.                        |
| Superposition       | Automatic, LinearSum, SquaredSum, MaxSum, WeightedSum.                                                 | Automatic to start. SquaredSum is usually balanced; LinearSum is more conservative; MaxSum is useful for diagnostics. |
| Blockage            | None, SelfSimilarityDeficit2020, SelfSimilarityDeficit, VortexCylinder, VortexDipole, HybridInduction. | Do not use with PDW. To test blockage, use A2A/PUD and start with SelfSimilarityDeficit2020 if available.             |
| Added turbulence    | None, STF2005, STF2017, GCLTurbulence, CrespoHernandez.                                                | STF2017 as a general option. It does not replace the ambient TI raster/fallback.                                      |
| Advanced parameters | Internal wake model constants.                                                                         | Leave Default PyWake unless calibrating, sensitivity testing or doing controlled technical comparison.                |

### Key idea

do not change five selectors at once. First get a stable calculation; then compare A/B scenarios by changing only one thing.

## 6.2 Recommended settings and links

Starting points for users who do not want to choose every model from scratch.

| Objective                   | Starting configuration                                                          | Comment                                                                      |
|-----------------------------|---------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| Check that everything loads | PDW + default wake model or Bastankhah + automatic superposition + no blockage. | Prioritizes stability and speed. Useful to validate input, CRS and resource. |
| General calculation         | PDW or PUD + Bastankhah/Niayifar + CGI7 + Auto/SquaredSum.                      | Good starting point to compare layouts without making the model too complex. |
| TI sensitivity              | Niayifar, Zong or TurboGaussian + TI raster or documented fallback + STF2017.   | Ambient TI is only clearly visible if the wake model uses it.                |
| Blockage / induction        | A2A or PUD + SelfSimilarityDeficit2020 if available.                            | Do not use PDW for blockage: that engine propagates effects downstream.      |
| Many scenarios              | PDW + fast and stable model + PyWake defaults.                                  | Ideal for sweeps, tests and quick comparison.                                |

### Before comparing

- keep the same curve, layout and resource
- save scenario A/B
- note TI raster or fallback
- document engine, wake model and superposition

### Useful link

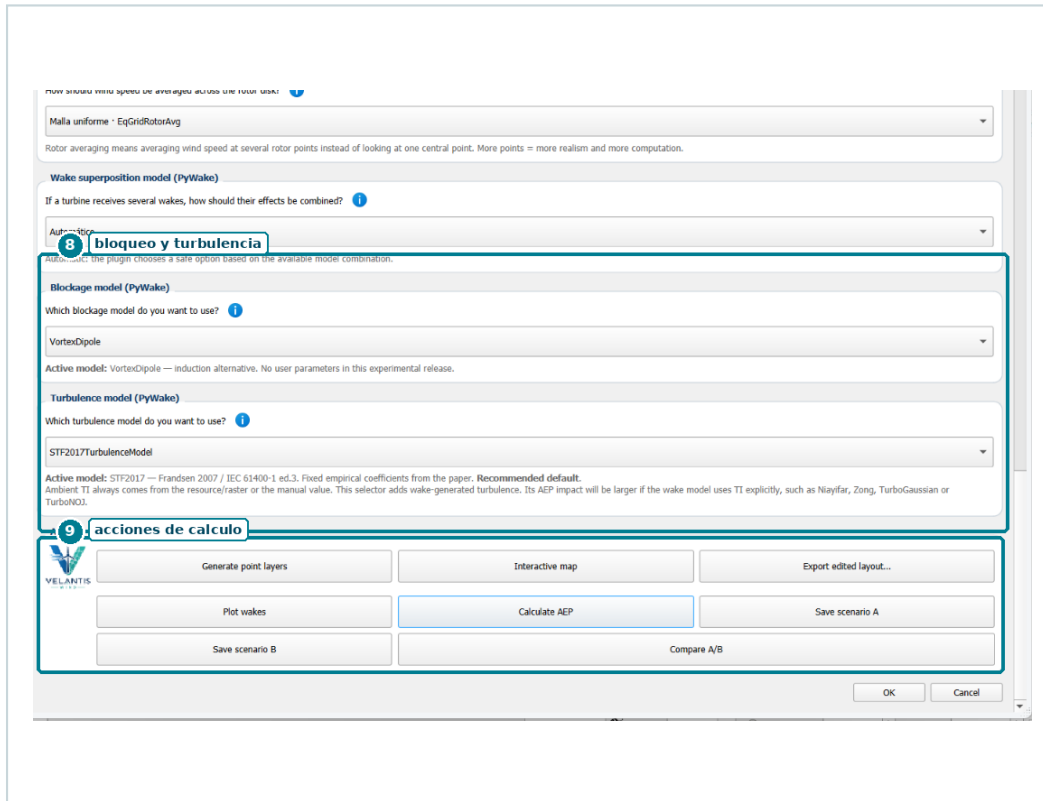
The plugin PyWake Documentation button opens the official PyWake documentation. Use it to check model availability in the installed version.

Official PyWake documentation: <https://topfarm.pages.windenergy.dtu.dk/PyWake/>

Recommendation: unless you are calibrating a model, leave advanced parameters at Default PyWake and change only high-level selectors.

## 7. Step 6 - Calculation, maps and results

Run Calculate AEP and check that the calculation finishes and generates an interpretable output.



Final calculation actions block.

| Button                | Recommended use            |
|-----------------------|----------------------------|
| Generate point layers | Create turbine layers.     |
| Plot wakes            | View wakes.                |
| Calculate AEP         | Run the main calculation.  |
| Interactive map       | Review results on the map. |
| Compare A/B           | Compare scenarios.         |

Recommended order: generate layers, check boundary, view wakes if useful and only then calculate AEP.

### Input loading sequence before Calculate AEP

1. Turbine: Define... > TXT/CSV > Create turbine, then check View curve.
2. Layout: Coords CSV... or Load layer; use metric X,Y coordinates and correct CRS.
3. Resource: choose one route only - WRG/ZIP or WAsP/Surfer grids.
4. Surfer: no tiles -> select the prepared folder; tiles -> Merge resource first, then output folder.
5. TI: load TI raster(s) with heights, or use and document Ambient TI fallback.
6. Then Generate point layers, check boundary/coverage and run Calculate AEP.

**Do not move to PyWake model tuning until turbine, layout, resource and TI/fallback are coherent.**

## 8. Final checklist and common errors

Use this page as a quick validation checklist before repeating calculations or comparing scenarios.

| Before calculating | Check                                                                              |
|--------------------|------------------------------------------------------------------------------------|
| Curve              | The curve looks consistent in View curve.                                          |
| Layout and CRS     | CSV X,Y or point layer in the correct CRS.                                         |
| Resource           | The boundary covers all turbines.                                                  |
| WRG/Surfer         | Heights, sectors, DSAA and minimum Surfer variables checked; tiles merged if used. |
| TI and PyWake      | Clear raster(s) and heights, or assumed fallback; compatible engine and models.    |
| Output             | Layers, maps or exports generated without strange warnings.                        |

### Common errors

- Folder selector appears empty: check the path shown at the bottom of the widget.
- Surfer input fails: select the folder, not an individual .grd file.
- AEP looks odd: check curve units, CRS, hub height and resource coverage.
- Files in \_bad\_for\_pywake: check naming, DSAA and minimum variables.

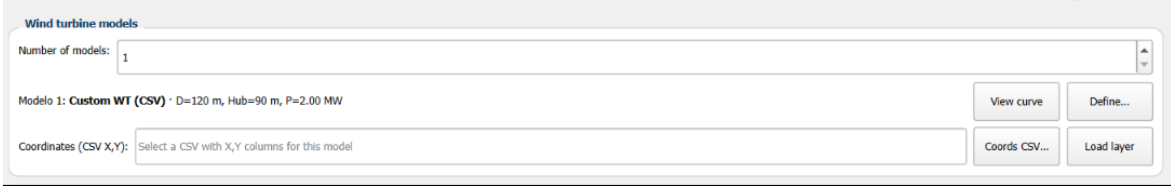
### Defensible result

A defensible calculation is not just an AEP number: it must be repeatable with the same inputs, plugin version, resource, turbine, layout, TI and PyWake configuration.

Summary: load turbine/curve, layout, resource folder or WRG, and TI/fallback first; only then choose physics and calculate.

## 9. Full calculation example - inputs

From here, the full workflow is repeated using real plugin screenshots.



Wind turbine models

Number of models: 1

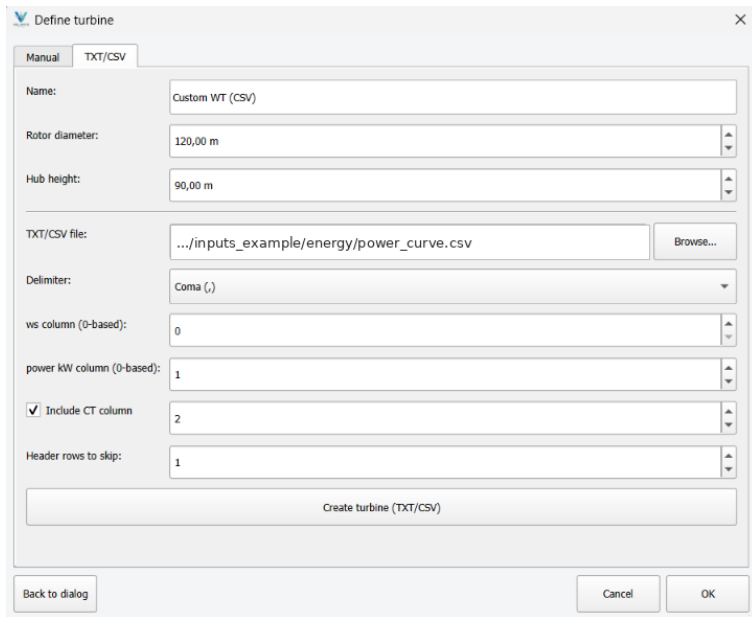
Modelo 1: Custom WT (CSV) · D=120 m, Hub=90 m, P=2.00 MW

Coordinates (CSV X,Y): Select a CSV with X,Y columns for this model

View curve Define...

Coords CSV... Load layer

Model loaded in the main panel: diameter, hub height and power are summarized.



Define turbine

Manual TXT/CSV

Name: Custom WT (CSV)

Rotor diameter: 120,00 m

Hub height: 90,00 m

TXT/CSV file: ../inputs\_example/energy/power\_curve.csv Browse...

Delimiter: Coma (,)

ws column (0-based): 0

power kW column (0-based): 1

☒ Include CT column 2

Header rows to skip: 1

Create turbine (TXT/CSV)

Back to dialog Cancel OK

Define turbine window, TXT/CSV tab.

### What the user should check

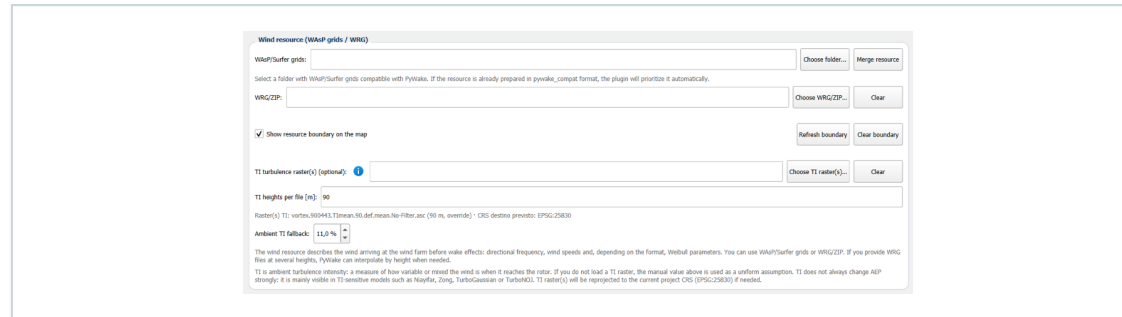
- Clear turbine name.
- Correct rotor diameter and hub height.
- Power curve CSV file selected.
- ws, power\_kW and Ct columns correctly assigned.
- Header rows to skip consistent with the CSV.

### Added visual validation

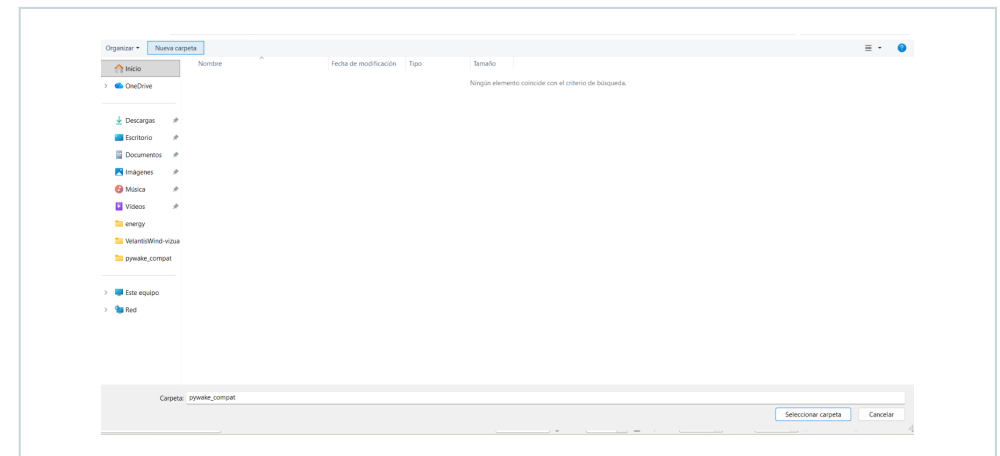
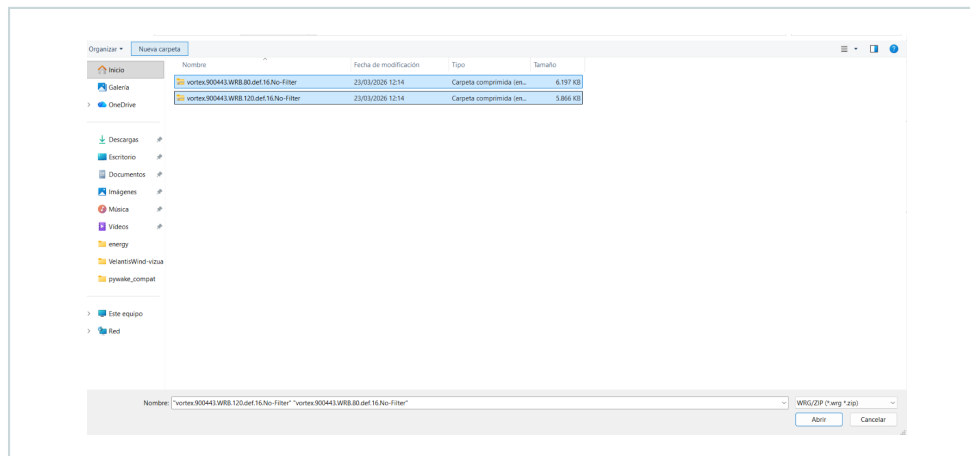
After creating the turbine, click View curve and check the power curve and Ct before continuing with resource, TI and physical models.

## 9.1 Wind resource selection

Practical example of loading the resource through the WRG/ZIP route and the Surfer grids route.



Resource block before loading files.



### Route A - WRG/ZIP

- Use Choose WRG/ZIP.
- You can select several WRGs if they represent different heights.
- Then check that the boundary covers the layout.

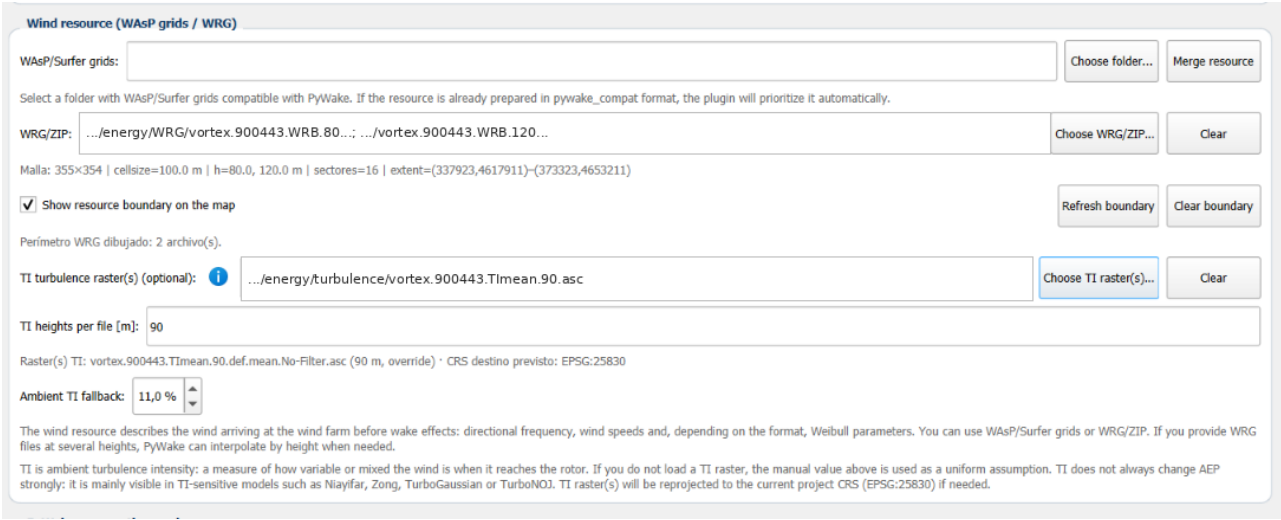
### Route B - Surfer grids

- Use Choose folder.
- Select the folder containing the .grd files, normally pywake\_compat.
- Even if the window seems empty, confirm the lower folder path.

What matters for the user: WRG is loaded as file(s); Surfer grids are loaded as a folder. This distinction avoids many user errors.

## 9.2 Loaded resource and ambient turbulence TI

Once the resource is loaded, TI is added and both inputs are checked for consistency.



**Wind resource (WASP grids / WRG)**

WASP/Surfer grids:  Choose folder... Merge resource

Select a folder with WASP/Surfer grids compatible with PyWake. If the resource is already prepared in pywake\_compat format, the plugin will prioritize it automatically.

WRG/ZIP:  Choose WRG/ZIP... Clear

Malla: 355x354 | cellsize=100.0 m | h=80.0, 120.0 m | sectores=16 | extent=(337923,4617911)-(373323,4653211)

☒ Show resource boundary on the map Refresh boundary Clear boundary

Perímetro WRG dibujado: 2 archivo(s).

TI turbulence raster(s) (optional):  Choose TI raster(s)... Clear

TI heights per file (m):

Raster(s) TI: vortex.900443.TImean.90.def.mean.No-Filter.asc (90 m, override) · CRS destino previsto: EPSG:25830

Ambient TI fallback:  11,0 %

The wind resource describes the wind arriving at the wind farm before wake effects: directional frequency, wind speeds and, depending on the format, Weibull parameters. You can use WASP/Surfer grids or WRG/ZIP. If you provide WRG files at several heights, PyWake can interpolate by height when needed.

TI is ambient turbulence intensity: a measure of how variable or mixed the wind is when it reaches the rotor. If you do not load a TI raster, the manual value above is used as a uniform assumption. TI does not always change AEP strongly: it is mainly visible in TI-sensitive models such as Niyafar, Zong, TurboGaussian or TurboNO3. TI raster(s) will be reprojected to the current project CRS (EPSG:25830) if needed.

Example with two WRGs loaded and one TI raster selected.

### TI height

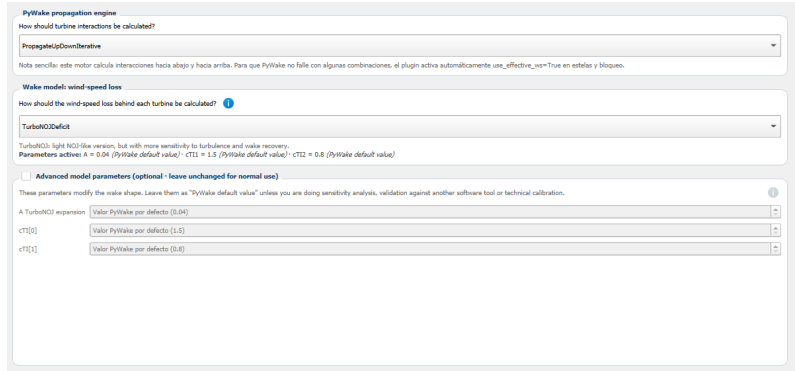
- If a TI raster is loaded, enter the corresponding height.
- With several raster(s), enter the heights in the same order.
- Example: one raster at 90 m -> TI heights per file = 90.

### What should be explained

TI does not always change AEP much: it depends on whether the selected wake model uses TI explicitly. If there is no raster, the fallback is a uniform assumption.

## 9.3 Physical models and calculation execution

Final step of the example: selecting physical models and running the calculation before reviewing the output.



**PyWake propagation engine**

How should turbine interactions be calculated?

**PropagateInteractive**

Nota: sencilla: este motor calcula interacciones hacia abajo y hacia arriba. Para que PyWake no falle con algunas combinaciones, el plugin activa automáticamente use\_effective\_wake=True en estado y bloquea.

**Wake model: wind-speed loss**

How should the wind-speed loss behind each turbine be calculated?

**TurbulenceDefault**

Turbulence: light NO3-like version, but with more sensitivity to turbulence and wake recovery.  
Parameters active: A = 0.04 (PyWake default value) · CT1 = 1.5 (PyWake default value) · CT2 = 0.8 (PyWake default value)

☐ **Advanced model parameters (optional - leave unchanged for normal use)**

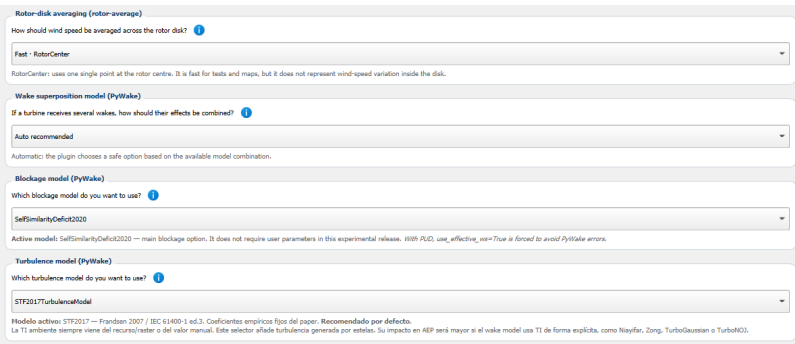
These parameters modify the wake shape. Leave them as "PyWake default value" unless you are doing sensitivity analysis, validation against another software tool or technical calibration.

A TurbNO3 expression: Valor PyWake por defecto (0.04)

CT0: Valor PyWake por defecto (1.5)

CT1: Valor PyWake por defecto (0.8)

PyWake engine, wake model and advanced parameters.



**Rotor-disk averaging (rotor-average)**

How should wind speed be averaged across the rotor disk?

**Fast - RotorCenter**

RotorCenter: uses one single point at the rotor centre. It is fast for tests and maps, but it does not represent wind-speed variation inside the disk.

**Wake superposition model (PyWake)**

If a turbine receives several wakes, how should their effects be combined?

**Auto recommended**

Automatic: the plugin chooses a safe option based on the available model combination.

**Blockage model (PyWake)**

Which blockage model do you want to use?

**SelfSimilarityDeficit2020**

Active model: SelfSimilarityDeficit2020 — main blockage option. It does not require user parameters in this experimental release. With PLO use\_effective\_wake=True is forced to avoid PyWake errors.

**Turbulence model (PyWake)**

Which turbulence model do you want to use?

**STF2017TurbulenceModel**

Modelo activo: STF2017 — Frandsen 2007 / IEC 61400-1 ed.3. Coeficientes empíricos fijos del paper. Recomendado por defecto.  
La TI ambiente siempre viene del recurso/refer o del valor manual. Este selector añade turbulencia generada por estela. Su impacto en AEP será mayor si el wake model usa TI de forma explícita, como Naysfar, Zong, TurboGaussian o TurbNO3.

Rotor-average, superposition, blockage and added turbulence.

### Closing criterion

For a first calculation, use a simple configuration and save the scenario. For sensitivity, change only one selector at a time.

### Expected result

After Calculate AEP, review total AEP, wake loss, AEP per turbine and layers/export. If something does not add up, go back to curve, CRS, resource, TI and height before touching models.