



## AIRE project TOOLS

21<sup>st</sup> April WindEurope 2026- Madrid  
Beatriz Méndez - CENER



# 4 tools

- Blade erosion risk atlas
- Wind farm control tool
- AEP tools
- Erosion safe mode tool



# Designed by the industry

AIRE Project's Industrial partners have participated actively in:

- tool's requirements definition
- tools evaluation

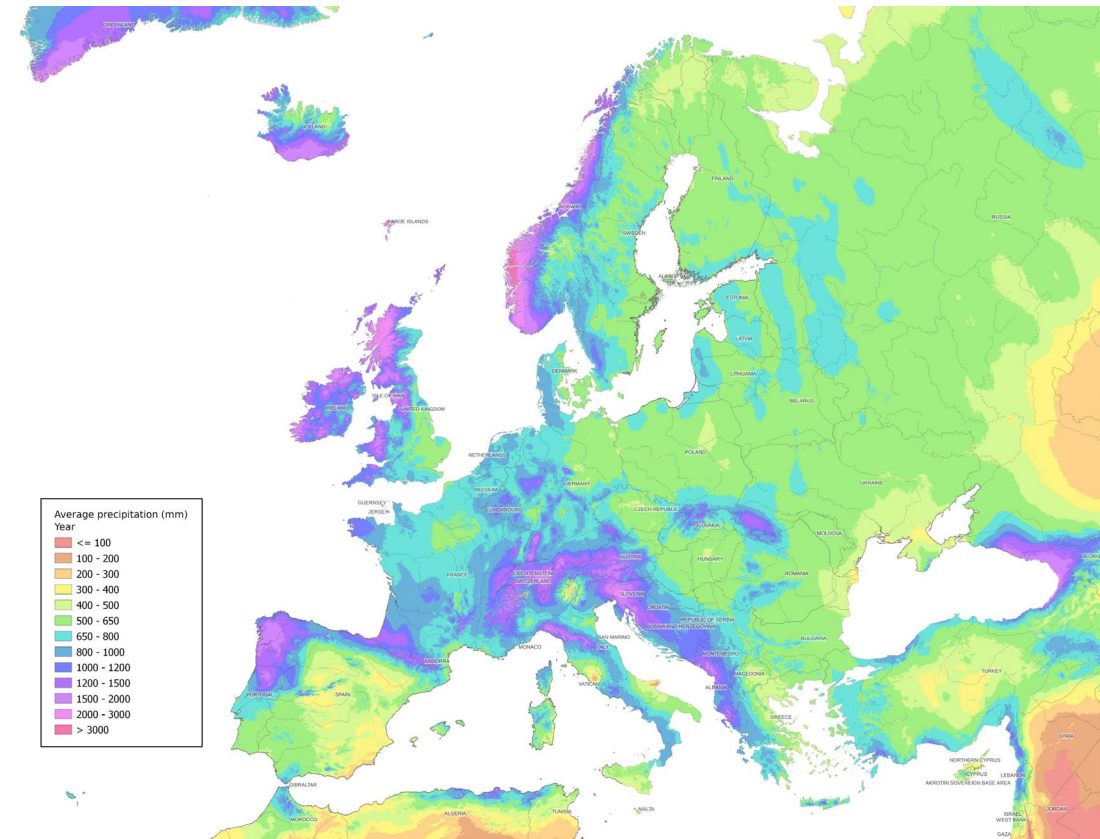


# Erosion risk atlas



# Erosion Risk Atlas Overview

- Decision making tool to understand associated **erosion risk to the environmental conditions** (wind and precipitation).
- Qualitative tool



Data source: worldclim.org

# Erosion Risk Atlas Method and Capabilities

## Period

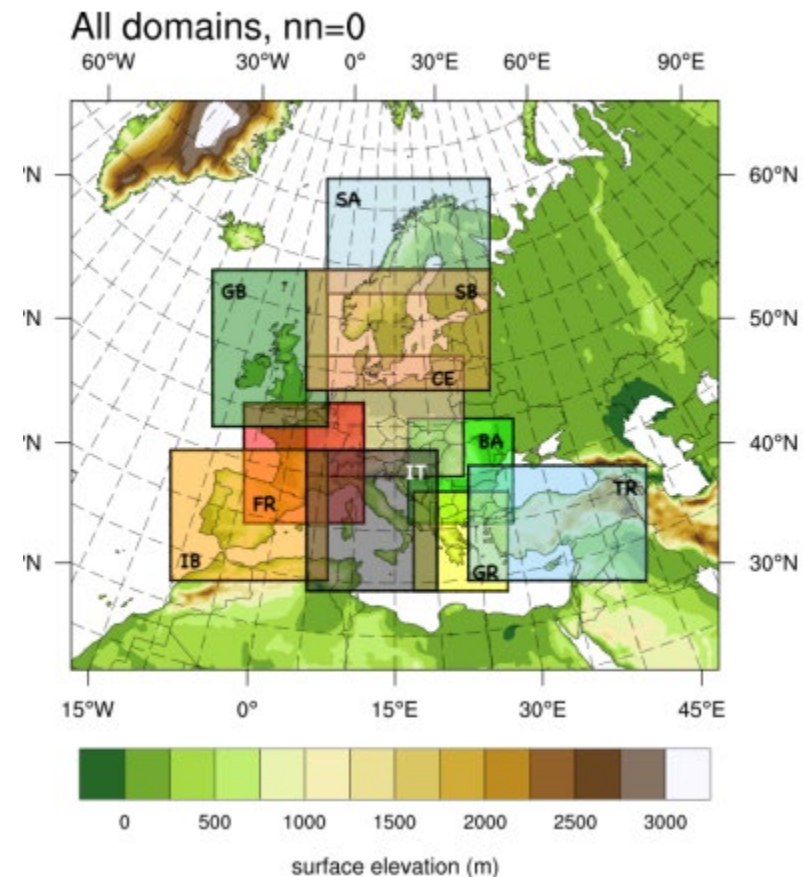
- Representative year 2021
- Based on yearly damage of 9 locations over Europe

## Domain: NEWA domains

- Model configuration from preliminary sensitivity test
- 3 km resolution
- Merge at country borders

## Layers:

- Accumulated annual **precipitation**
- **Wind speed** at 100m & 150m agl
- Accumulated **annual damage / time to erosion onset**
  - blade damage models
  - turbine types
  - coating material



Source: Dörenkämper et al. (2020)

# Wind farm control tool

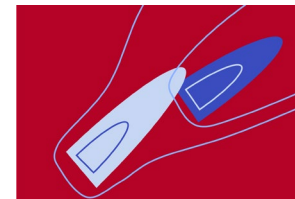
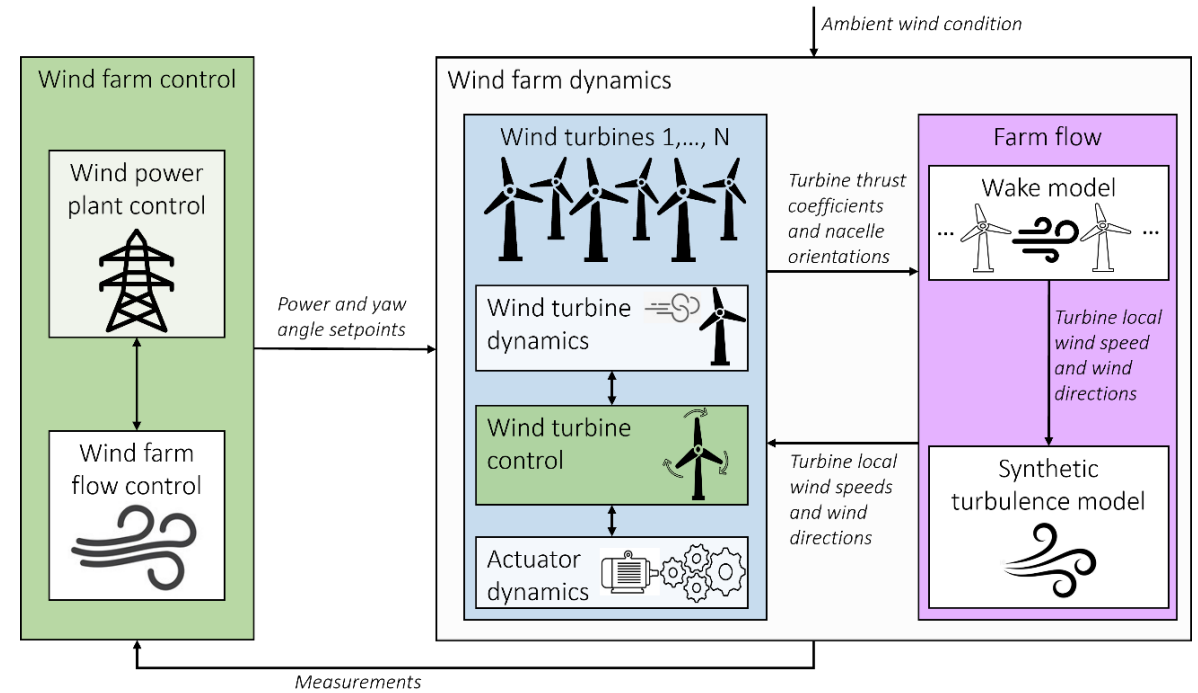


# Wind farm control tool - Overview

Fast dynamic simulation of wind farm response

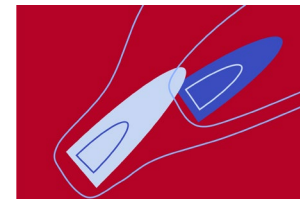
Specific capabilities:

- Impact of erosion
- Impact of erosion safe mode
- Impact of yaw misalignment control
- Different wake models included



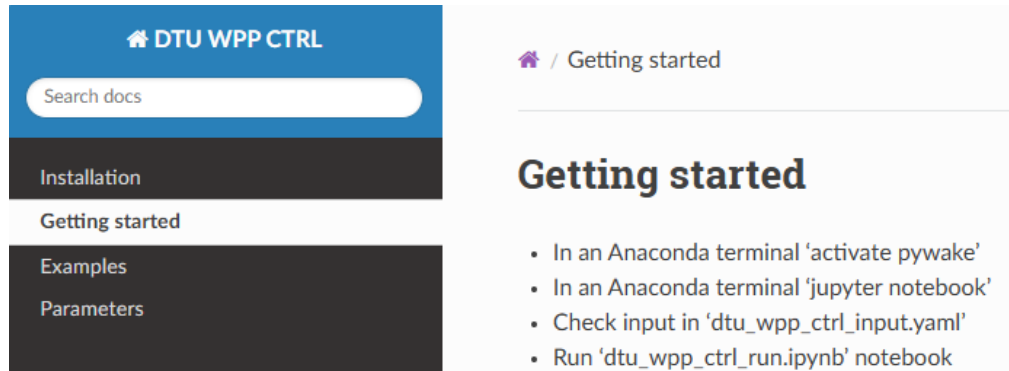
# Wind farm control tool - Capabilities

- Control framework for Wind Power Plant Control and Wind Farm (Flow) Control
- Dynamic vs static evaluation of Erosion, Erosion Safe Operation, Yaw Misalignment Control
- Impact of wake model choice
- **Erosion** impact through Cp/Ct data
- **Wake models**: Python implementation (with quasi-static PyWake/FLORIS/FOXES)



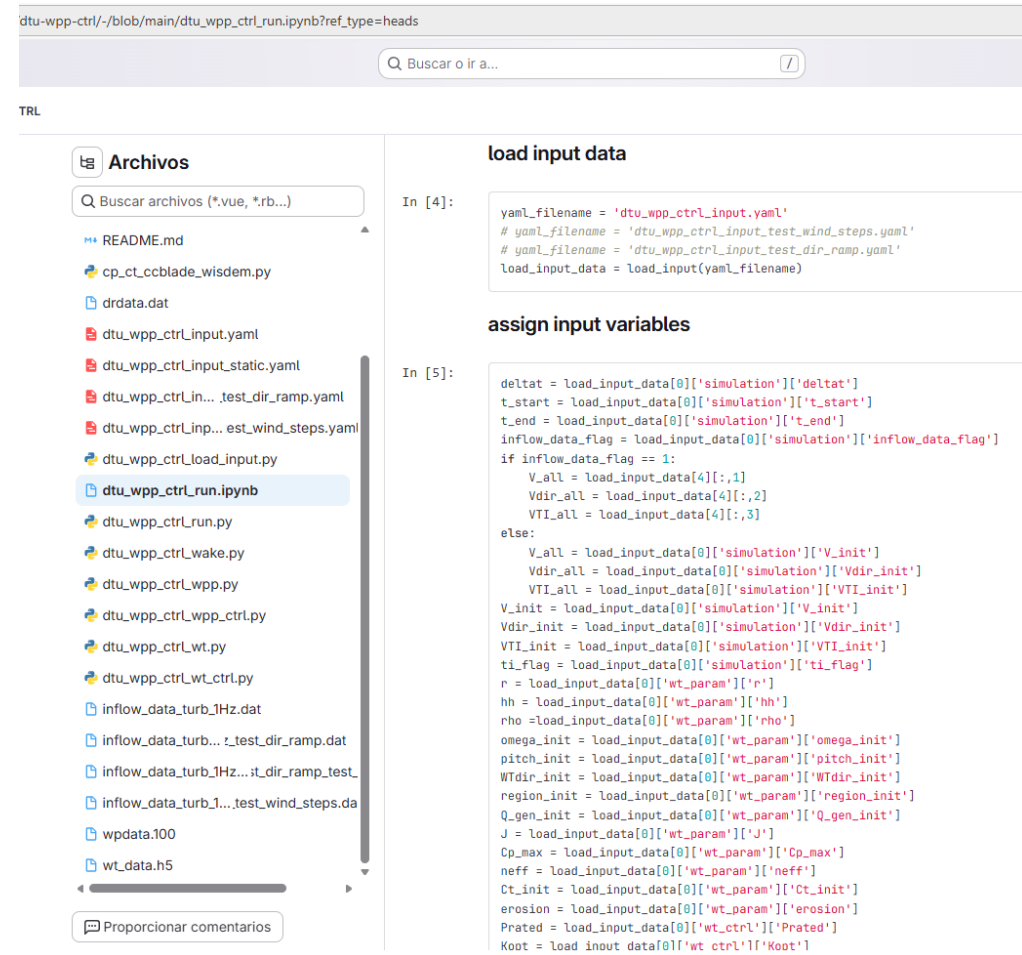
# Wind farm control tool - Usability

- Web site with the documentation
- Examples
- Run using an editable jupyter notebook



The screenshot shows the DTU WPP CTRL website. On the left is a navigation menu with links for 'Installation', 'Getting started', 'Examples', and 'Parameters'. The main content area is titled 'Getting started' and contains a list of instructions:

- In an Anaconda terminal 'activate pywake'
- In an Anaconda terminal 'jupyter notebook'
- Check input in 'dtu\_wpp\_ctrl\_input.yaml'
- Run 'dtu\_wpp\_ctrl\_run.ipynb' notebook



The screenshot shows a Jupyter Notebook interface. The left sidebar displays a file explorer with a list of files, including 'dtu\_wpp\_ctrl\_run.ipynb' which is selected. The main area shows two code cells:

```
In [4]:  
yaml_filename = 'dtu_wpp_ctrl_input.yaml'  
# yaml_filename = 'dtu_wpp_ctrl_input_test_wind_steps.yaml'  
# yaml_filename = 'dtu_wpp_ctrl_input_test_dir_ramp.yaml'  
load_input_data = load_input(yaml_filename)
```

```
In [5]:  
deltat = load_input_data[0]['simulation']['deltat']  
t_start = load_input_data[0]['simulation']['t_start']  
t_end = load_input_data[0]['simulation']['t_end']  
inflow_data_flag = load_input_data[0]['simulation']['inflow_data_flag']  
if inflow_data_flag == 1:  
    V_all = load_input_data[4][:,1]  
    Vdir_all = load_input_data[4][:,2]  
    VTI_all = load_input_data[4][:,3]  
else:  
    V_all = load_input_data[0]['simulation']['V_init']  
    Vdir_all = load_input_data[0]['simulation']['Vdir_init']  
    VTI_all = load_input_data[0]['simulation']['VTI_init']  
V_init = load_input_data[0]['simulation']['V_init']  
Vdir_init = load_input_data[0]['simulation']['Vdir_init']  
VTI_init = load_input_data[0]['simulation']['VTI_init']  
ti_flag = load_input_data[0]['simulation']['ti_flag']  
r = load_input_data[0]['wt_param']['r']  
hh = load_input_data[0]['wt_param']['hh']  
rho = load_input_data[0]['wt_param']['rho']  
omega_init = load_input_data[0]['wt_param']['omega_init']  
pitch_init = load_input_data[0]['wt_param']['pitch_init']  
Wtdir_init = load_input_data[0]['wt_param']['Wtdir_init']  
region_init = load_input_data[0]['wt_param']['region_init']  
Q_gen_init = load_input_data[0]['wt_param']['Q_gen_init']  
J = load_input_data[0]['wt_param']['J']  
Cp_max = load_input_data[0]['wt_param']['Cp_max']  
neff = load_input_data[0]['wt_param']['neff']  
Ct_init = load_input_data[0]['wt_param']['Ct_init']  
erosion = load_input_data[0]['wt_param']['erosion']  
Prated = load_input_data[0]['wt_ctrl']['Prated']  
Koot = load_input_data[0]['wt_ctrl']['Koot']
```

# AEP prediction tools



# ATENEA

(Advance Tool for prediction of  
annual Energy under Erosion and  
calima)



**CENER**



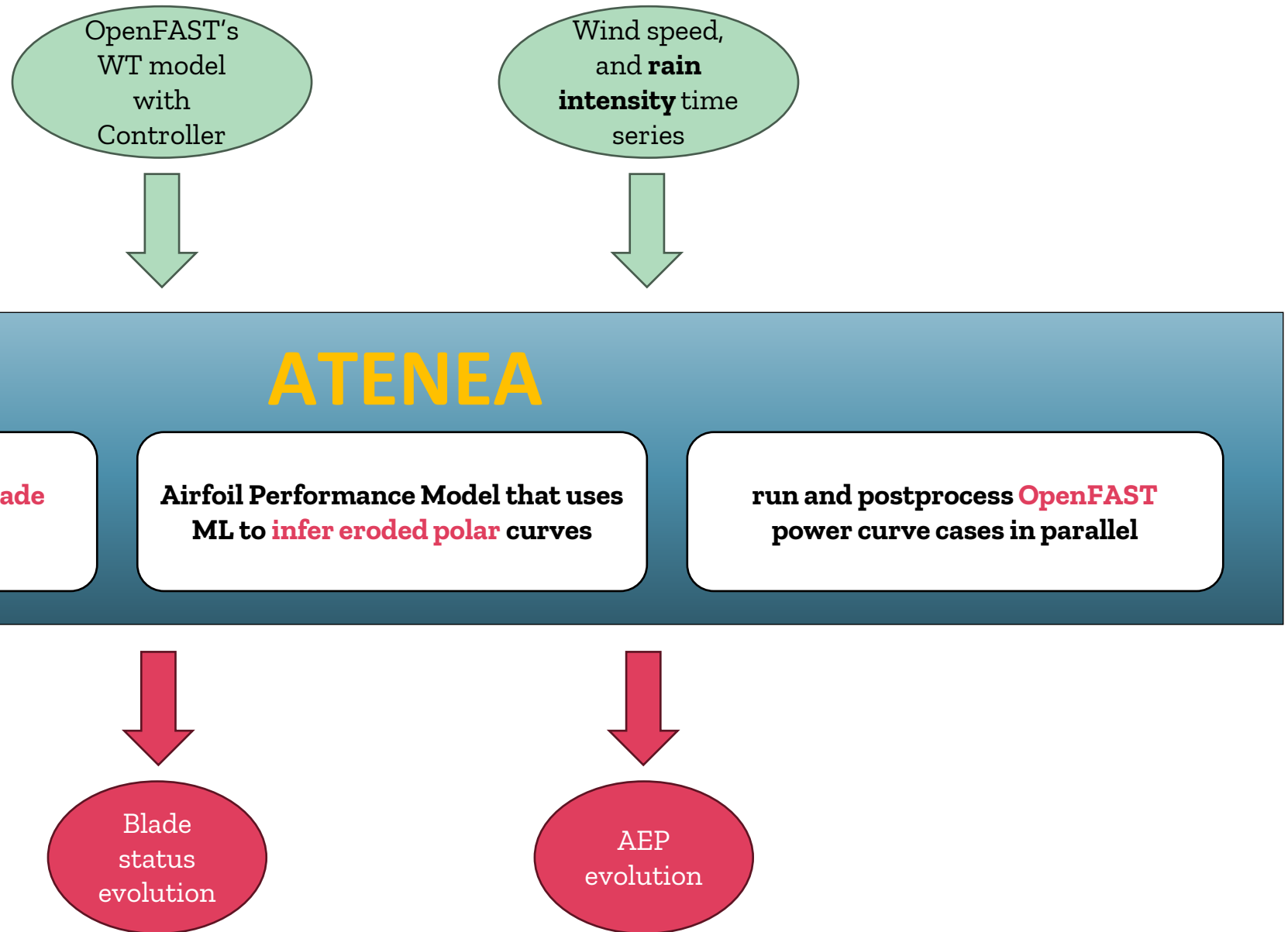
# ATENEA overview

ATENEA combines the aeroelastic tool **OpenFAST** with an **airfoil performance model** developed with machine learning to obtain AEP depending on the blade erosion progression due to rain and the wind turbine operational conditions.

ATENEA uses:

- A series of a **site-specific wind speed** and **rain intensity** data
- **OpenFAST** simulations to obtain the needed power curves
- Springer Model to predict **blade erosion damage**
- **Airfoil Performance Model (IA)** to infer eroded polars for each year and blade section

# ATENEA overview



\*APM:

# ATENEA usability

atenea\_input.yml

```
site:
  meteo_series_file: 'MeteoData.txt' # path to the file containing the wind speed time series
  separator: ";" # Separator used in the meteo series file. Default value ','
  shear: 0.2 # power law exponent of wind profile [-]
  upflow: 8.0 # wind upflow angle [deg]
  date_format: "%d/%m/%Y %H:%M"

wind_turbine:
  fst_file: 'OpenFAST/model.fst' # path to the wind turbine (WT) model file in OpenFAST v4.0.1
  v_in: 4.0 # cut in wind speed of the WT [m/s]
  v_out: 25.0 # cut out wind speed of the WT [m/s]
  v_rated: 11.5 # rated wind speed of the WT [m/s]
  omega_min: 6.0 # minimum rotational speed (Low-Speed-Shaft) of the WT [rpm]
  omega_max: 9.6 # maximum rotational speed (Low-Speed-Shaft) of the WT [rpm]
  taped_blades: True # Boolean indicating whether the blades are taped. If they are not, paint protection is assumed
```

```
Timestamp;10_m WdSpd_Avg [m/s];10_m RnFlt_Avg [mm]
01/01/2015 0:00;6.785;0
01/01/2015 1:00;8.075;0
01/01/2015 2:00;8.445;0
01/01/2015 3:00;9.257;0
01/01/2015 4:00;10.758;0
01/01/2015 5:00;13.36;0
01/01/2015 6:00;12.558;0
```

## Running the code

```
import atenea
results = atenea.run('inputExamples/example_atenea_inputFile.yml', max_workers=4)
```

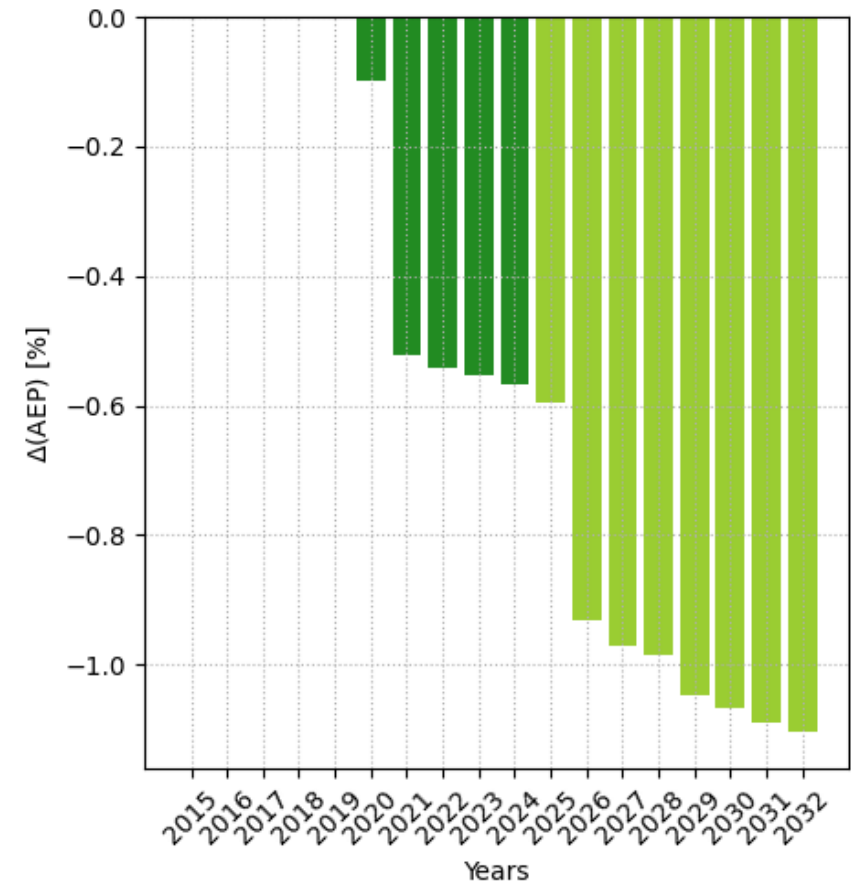
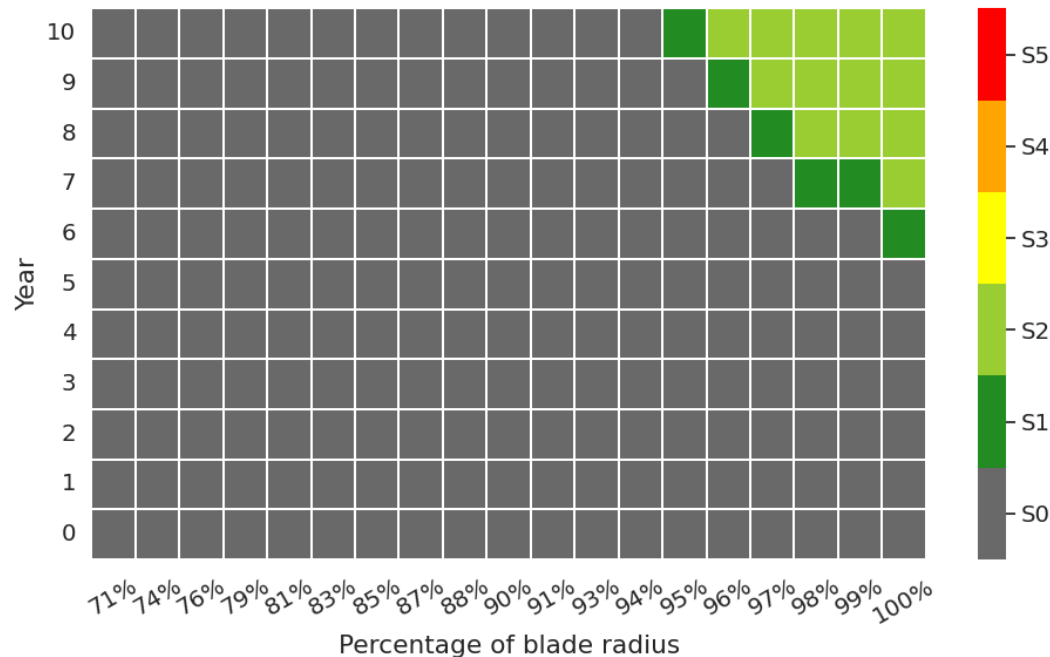


ATENEA User's Manual



# ATENEA capabilities

- **AEP** of a wind turbine over different years based on a **meteo series**
- It estimates the evolution of **differentiated erosion for each blade section** depending on the WT's operation (**controller-dependent**)
- **AEP losses** with respect to the first year of the meteo series
- The **IEA-Task46 erosion status categories (different for blades and power)**
- **Cp/Ct**
- **Power curves**



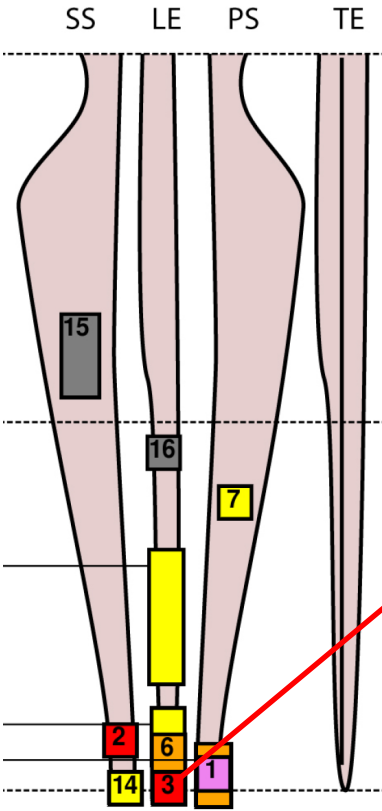
# SALT

(Simplified Aerodynamic Loss Tool)



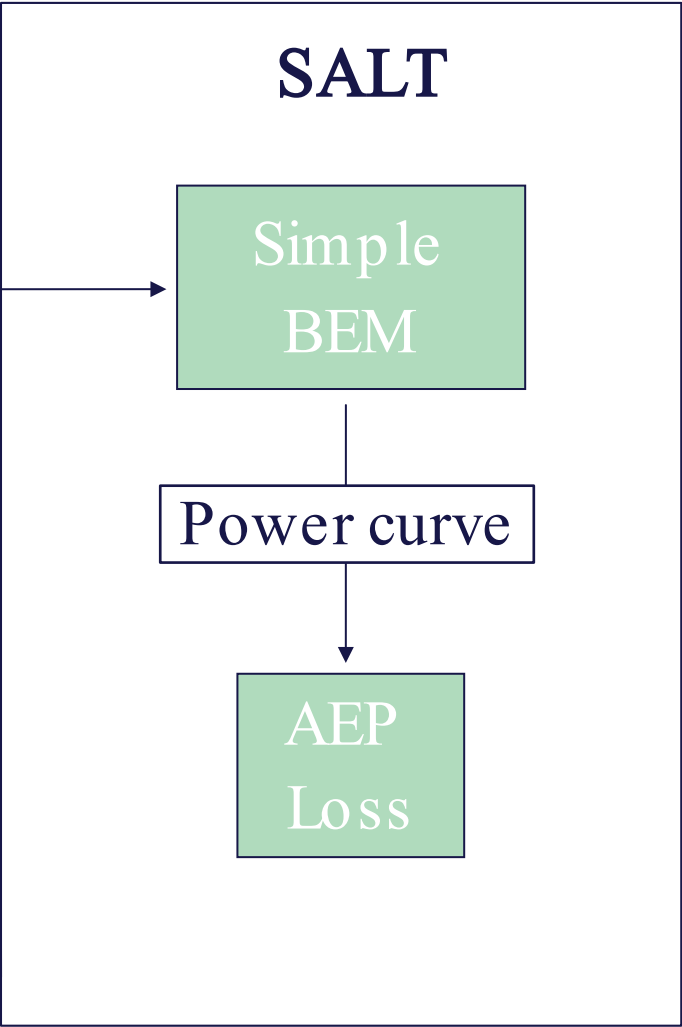
# SALT – Simplified Aerodynamic Loss Tool

Blade inspection



Aerodynamic damage categorisation

LERCat: **D**



Fast tool to compute aerodynamic losses from leading edge roughness and erosion inspection data

- Simplified BEM (Bak, 2022)
- Excel & Python versions
- Runs on any laptop
- Instant results
- Validated & Accurate
- Openly available

Open repositories

SALT

LERCat

# Erosion safe mode operation



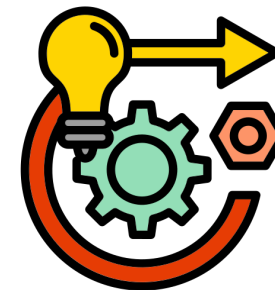
# Erosion-safe operation Overview

The tool enables a **proactive control strategy** that optimizes the trade-off between power production and blade coating life. By reducing rotor speeds during high-intensity precipitation events, the tool derives **optimal erosion-safe curtailment strategies**.

**Design ESO**

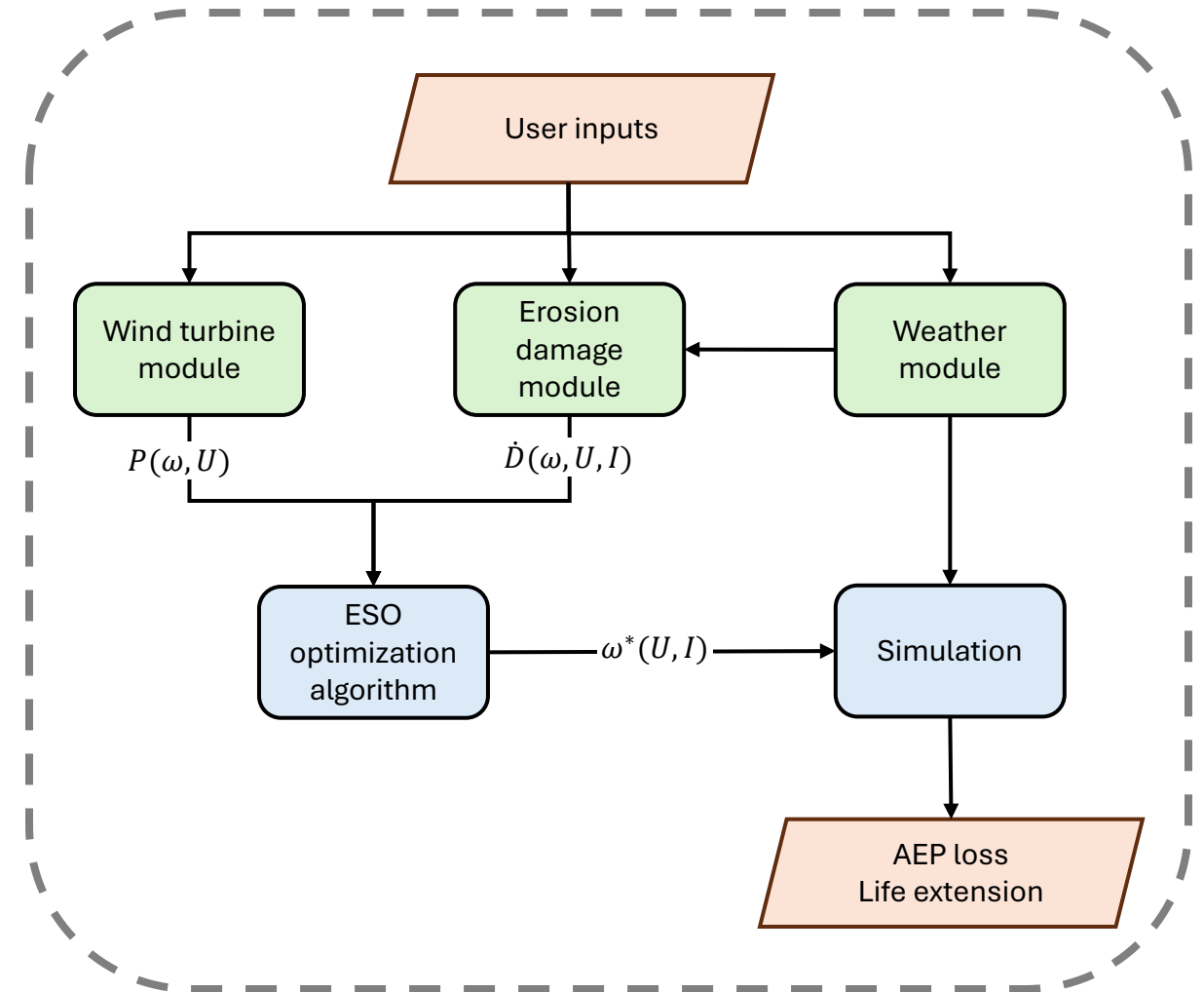


**Operational ESO**



# ESO Tool Overview

- Design tool used for calculating optimal turbine- and site-specific **rotor speed schedules** used for erosion-safe operation
- Modular, lightweight and flexible
- Basic user inputs and access to pre-defined models.  
Hi-fi inputs → hi-fi outputs



# ESO Usability

- Web site with the documentation
- Examples
- Run using an editable jupyter notebook

## Usage in Notebooks

Because the project is installed in "editable" mode, you can import the core logic directly into your notebooks without worrying about file paths:

```
from eso import *

# Load an existing turbine from the YAML
wt = erosion_safe_operation(turbine_name="IEA 15MW")

# call the DTU impingement model and create the damage surface
damage_model = DTURETImpingementDamageModel(c=3.45e20, m=9.57)
wt.create_damage_surface(damage_model=damage_model)

# Create ESO tip speed schemes
wt.create_ESO_tip_speed_schemes()
```

README.md 2,47 KiB

Código

Vista previa



## Erosion-safe operation (ESO) design tool

### Overview

Leading edge erosion (LEE) is the gradual degradation of a wind turbine blade's front surface, primarily caused by high-velocity impacts with environmental particles such as rain, hail, dust, and insects. As modern turbines grow larger, their blade tips can reach speeds exceeding 300 km/h, turning even soft raindrops into abrasive projectiles that cause microscopic pits and cracks in the protective coating. Over time, these small imperfections evolve into severe gouges that expose the underlying composite laminate, significantly increasing aerodynamic drag and disrupting the smooth, laminar airflow required for efficient power generation. This surface roughening typically results in a loss in annual energy production (AEP) and, if left unaddressed, can compromise the structural integrity of the blade, leading to expensive repairs and premature failure.

This repository provides a tool for probabilistic modelling of LEE on wind turbine blades. The tool also allows for designing and optimizing tip speed curtailment schemes used for erosion-safe operation (ESO).




### Quick Start

Follow these steps to set up the environment and run the examples.

#### 1. Clone the Repository

```
git clone https://gitlab.windenergy.dtu.dk/aire/ESO-strategy-design.git
cd ESO-strategy-design
```



# More details about AIRE tools at 17h today!



## Thank you.

 @ProjectAire  @Aire Project



Funded by  
the European Union

