First updated: 12 September 2025

Last updated: 27 October 2025

Last updated by: Oliver STEPHENSON

ERA5 COPERNICUS DATA SOURCE

DATA SOURCE LOCATION

https://cds.climate.copernicus.eu/datasets/reanalysis-era5-single-levels?tab=download

TECHNICAL INSTRUCTIONS

- API location: https://cds.climate.copernicus.eu/how-to-api

Need a token

Log in ID is: tech@energyexe.com

P/w is vJpsqVW5DKEP6

Instructions:

Log in: https://cds.climate.copernicus.eu/how-to-api#

1. Once logged in, copy the code displayed below to the file **$HOME/.cdsapirc** (in your Unix/Linux environment)

   url: https://cds.climate.copernicus.eu/api key: fea40a33-7b81-4f9a-a145-a1556b25c940

2. Install the CDS API client

The CDS API client is a Python based library. It provides support for Python 3.

You can install the CDS API client via the package management system pip, by running on Unix/Linux the command below.

```
$ pip install "cdsapi>=0.7.7"
```

Full details are here: https://cds.climate.copernicus.eu/how-to-api

Note comments: **New API package for advanced users**

- These 2 links provide technical information on how to access the data.

https://confluence.ecmwf.int/display/CKB/How+to+install+and+use+CDS+API+on+Windows

https://confluence.ecmwf.int/display/CKB/How+to+download+ERA5

See sections 3 and 4:

3. Downloading online ERA5 family data through the CDS web interface

4. Download ERA5 family data through the CDS API

- The files are in GRIB format:

https://confluence.ecmwf.int/display/CKB/What+are+GRIB+files+and+how+can+I+read+them

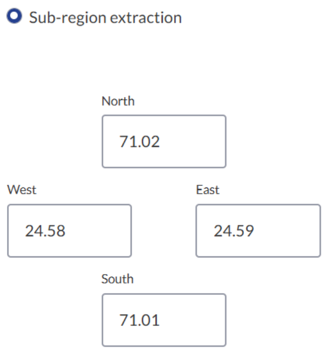
The GRIB file format is designed for storing and distributing weather data. GRIB files are widely used in meteorological applications.

KEY POINTS

- ERA5 is updated daily with a latency of about 5 days

- hourly estimates for a large number of atmospheric, ocean-wave and land-surface
- Data has been regridded to a regular lat-lon grid of 0.25 degrees for the reanalysis
- Data is in GRIB format. Requires a GRIB reader to interpret the files
- All data is UTC time zone
- Data goes back to 1940. We only want from 2010 to the most recent data available. And then downloaded as a cron job.
- Each record provides data for a single parameter at a single point in time and level.
- ** Important** – limit by Geographical Area to reduce the data collection. Only collect tiles that match wind farm lat/lon data.
    - It looks like the API query will:
        - limit the lat / lon data input to 2 decimal places
        - ask for a range for latitude (1x North, 1x South) and a range for longitude (1x West, 1x East).
            - If this is the case, then need to 1) roundup AND rounddown our latitude attribute and 2) roundup AND rounddown our longitude attribute.
            - For instance,
                - Latitude =71.011940 -> rounddown for North to 71.10 and roundup for South to 71.02
                - Longitude = 24.589210 -> rounddown for East to 24.58 and roundup for West to 24.59

Latitude
71.011940° N
Longitude
24.589210° E


Sub-region extraction

North: 71.02
West: 24.58
East: 24.59
South: 71.01

ATTRIBUTES

We want to collect the following attributes from the data

1) **100m u-component of wind** [m s-1]

the *eastward* component of the 100 m wind. It is the horizontal speed of air moving towards the east, at a height of 100 metres above the surface of the Earth, in metres per second. This parameter can be combined with the northward component to give the speed and direction of the horizontal 100 m wind.

2) **100m v-component of wind** [m s-1]

This parameter is the *northward* component of the 100 m wind. It is the horizontal speed of air moving towards the north, at a height of 100 metres above the surface of the Earth, in metres per second. This parameter can be combined with the eastward component to give the speed and direction of the horizontal 100 m wind.

3) 2m temperature

This parameter should be stored with no further calculations required

CALCULATIONS

For each location and time stamp we want to calculate and store the following data:

1) Store:Wind speed at 100m (m/s)
2) Wind direction (degrees)
3) Wind speed at hub height (m/s)

Calculation 1

To calculate **wind speed**:

wind_speed = sqrt($u^2 + v^2$)

Calculation 2

To calculate **wind direction**:

Wind direction = atan2(v northward, u eastward)

1. function atan2(v, u) returns an angle in radians that represents the direction the wind is blowing (the vector's polar angle), often in the range of -π to +π or 0 to 2π.
2. Convert to Meteorological Direction (degrees)

3. Wind direction is conventionally reported as the direction the wind is coming from. Meteorological direction is typically measured clockwise from true north.
4. Subtract the angle from 90 degrees, and then add 360 degrees (if negative) to bring it into a 0-360 range.
5. Convert radians to degrees by multiplying by 180/π.
6. Add this angle to 270 degrees.
7. Take the result modulo 360 to get the final meteorological wind direction.

<u>Calculation 3</u>

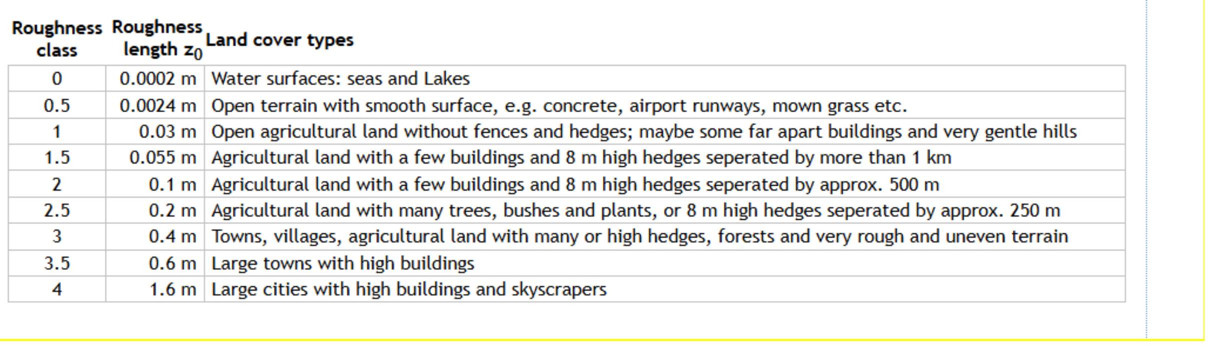
To calculate **<u>estimated wind speed at hub height:</u>**

Use a logarithmic calculation.

This method requires knowing the **roughness length** (z_0) of the terrain.

Add roughness length attribute to each wind farm (use centroid coordinates?). Two step process. 1) Add to DB structure and data input tool; 2) add hard data for each wind farm. Eg all offshore = 0; etc, etc

On front end tool, display our estimated Roughness factor. But also provide user with the option to change our estimated roughness factor. They can overwrite with their own assumption and see the sensitivity.



Roughness class	Roughness length z_0	Land cover types
0	0.0002 m	Water surfaces: seas and Lakes
0.5	0.0024 m	Open terrain with smooth surface, e.g. concrete, airport runways, mown grass etc.
1	0.03 m	Open agricultural land without fences and hedges; maybe some far apart buildings and very gentle hills
1.5	0.055 m	Agricultural land with a few buildings and 8 m high hedges seperated by more than 1 km
2	0.1 m	Agricultural land with a few buildings and 8 m high hedges seperated by approx. 500 m
2.5	0.2 m	Agricultural land with many trees, bushes and plants, or 8 m high hedges seperated by approx. 250 m
3	0.4 m	Towns, villages, agricultural land with many or high hedges, forests and very rough and uneven terrain
3.5	0.6 m	Large towns with high buildings
4	1.6 m	Large cities with high buildings and skyscrapers

This is the formula:

$$v_2 = v_1 \frac{\ln(h_2/z_0)}{\ln(h_1/z_0)}$$

- V_2 = wind speed at hub height
- $V_2 = V_1 * (\ln(h_2/Z_0) / \ln(h_1/Z_0))$
- The reference wind speed v_1 is measured at height h_1. This is calculated in Calculation 1 (see above for details)

- Height h_1 = 100m, which is the height that we get wind speed data from Copernicus
- h_2 is the hub height of the generation unit
- z_0 is the roughness length

POTENTIAL ISSUES AND (SOLUTIONS)

- **Chunked requests / splitting by time periods**
  Because the CDS API or the system may reject large requests, many of the scripts split downloads into manageable chunks — e.g. month-by-month or two-month blocks.
  Example: the ERA5_hourly_data_request repo suggests that you can't request very long months in one go, so the code downloads two months at a time. GitHub
  The era5cli tool also supports splitting outputs (by years / months). GitHub
  In the Climate-CAFE/era5-daily-heat-aggregation-python repo, they query data monthly due to CDS limits. GitHub
- **Reading GRIB / NetCDF formats**
  Many repos rely on xarray + cfgrib or other engines to open ERA5 GRIB/NetCDF files.
  E.g. sayantanonfire uses xarray with the cfgrib engine. GitHub
  Care must be taken with variable names, coordinate conflicts, and unit conversion.
- **Aggregation / downsampling**
  If your goal is to go from hourly to daily or monthly summaries, many projects include functions to do so — e.g., downsample_hourly_to_daily, compute daily min/mean/max, merging of variables, etc.
  E.g. jashvina-devadoss' module. GitHub
  Brianvgarcia's script computes annual averages / maxima / minima from hourly files. GitHub
  The era5-daily-heat-aggregation-python repo shows deriving daily metrics from hourly data (e.g. for heat index). GitHub

GITHUB ASSISTANCE

Here are some notable GitHub projects that include instructions, scripts, or helper functions for dealing with ERA5 hourly data:

Repo	What it does / instructions	Relevant parts you can use
nbusired/ERA5_hourly_data_request	Python script to download ERA5 hourly single-level data from CDS. GitHub	Uses cdsapi, prompts user for start/end years, variable selection, etc.
jashvina-devadoss/ERA5_time_downsample	Download hourly data + functions to downsample (e.g. hourly → daily or yearly) GitHub	The downsample_hourly_to_daily.py, download_ERA5_hourly.py, etc.
MRPHarris/ERA5handlers	R package / functions for downloading, importing, parsing and summarising ERA5 netCDF data (via CDS) GitHub	The get_CDS_era5 function, collate_era5, etc.
sayantanonfire/ERA5_data_extraction_and_post_processing	Python pipeline for extracting variables, converting units, merging, and exporting (e.g. to Zarr) GitHub	Good for larger workflows: GRIB loading, filtering, merging, efficient storage (Zarr)
JuanCrls17/ERA5-Land-Data-Downloader	Tool in Python to download ERA5-Land hourly data with error handling and logging GitHub	Can help manage crashes, retries, chunked downloads
era5cli (eWaterCycle/era5cli)	Command-line interface for retrieving ERA5 / ERA5-Land via CDS API. GitHub	Good if you prefer scripting / CLI usage, selecting variables, splitting by year, etc.
BioDT/general-copernicus-weather-data	Utilities to fetch weather data from Copernicus (ERA5-Land) at hourly resolution, then optionally convert / aggregate to daily, etc. GitHub	Has an API for “final_time_resolution” (hourly vs daily) and downloading per location
loicduffar/ERA5-tools	Jupyter notebooks showing how to download and extract	Good as tutorial / worked examples

	time series, map, etc. (hourly → monthly etc.) GitHub+1	
Brianvgarcia/Transform_ERA5_hourly_to_average_max_min_daily	Python script to compute annual / daily stats from hourly netCDF files (e.g. max, min) GitHub	Useful for aggregating from hourly to derived metrics
JuanCrls17/AtmosForge-ERA5toWRF	Notebooks to download ERA5 single-level and pressure-level data (hourly) for WRF boundary/initial conditions GitHub	Good for boundary-condition workflows
GeoRegionsEcosystem/ERA5Reanalysis.jl	Julia package supporting hourly ERA5 data (0.25° resolution) from CDS, etc. GitHub	If you ever use Julia

OTHER STUFF

Might be worth looking at https://weatherlayers.com/open-source.html

Weibull calculations

This is a way of showing wind speed distribution.

Wind speed values can be divided into wind speed classes, each comprising 1 m/s. So, there are ‘bins’ for 0-1m/s, 1-2 m/s, 2-3 m/s, etc, etc. The distribution shows the percentage of data points that are included in each ‘bin’.

Calculate (and display) 3 key data points: v, c, k

1) mean wind speed
   ‘v’
   unit for measurement is m/s

2) ‘c’
   Confusingly, ‘c’ can also sometimes be denoted as A
   unit for measurement is m/s
   c' is the scale parameter that determines the wind speed distribution's scale or location. It is related to the average wind speed but is not the average itself.

3) ‘k’.
   there are no units for measurement
   'k' is the shape parameter that describes the distribution of wind speeds. A low 'k' value indicates highly variable, gusty winds, while a high 'k' value signifies steadier winds with less variation. It determines the shape of the distribution curve, showing how concentrated the wind speeds are around the average.

Quantity	**Formula**	**Excel equivalent**
PDF (Weibull Probability Density Function)	((k/c)(v/c)^{k-1} e^{-(v/c)^k})	=($F$3/$F$4)*POWER(B2/$F$4,$F$3-1)*EXP(-POWER(B2/$F$4,$F$3))
Weibull Cumulative Distribution Function (CDF)	(1 - e^{-(v/c)^k})	=1-EXP(-POWER(B2/$F$4,$F$3))

Quantity	Formula	Excel equivalent
Mean Wind Speed	(cΓ(1+1/k))	=F4*EXP(GAMMALN(1+1/F3))

Python code and method

Fit directly to the data (maximum likelihood / curve fitting)

Use scipy.stats.weibull_min.fit()

— it finds the best-fitting k and c automatically.

CODE:

```
from scipy.stats import weibull_min


# Fit Weibull distribution to the data (forcing location = 0)

params = weibull_min.fit(wind_speeds, floc=0)


k_fit, loc, c_fit = params

print(f"Fitted shape (k): {k_fit:.3f}")

print(f"Fitted scale (c): {c_fit:.3f}")
```

OUTPUT EXAMPLE

```
Fitted shape (k): 2.016

Fitted scale (c): 5.990
```

1) **Weibull Probability Density Function (PDF)**

This gives the **shape of the curve** — the value plotted in your column **E (Fitted Weibull PDF)** in the Excel calculator.

$$f(v) = \frac{k}{c}(\frac{v}{c})^{k-1}e^{-(\frac{v}{c})^k}$$

Symbol	Meaning
v	wind speed (m/s)
k	shape parameter
c	scale parameter

2) **Weibull Cumulative Distribution Function (CDF)**

This gives the **probability that wind speed ≤ v**, i.e., the cumulative proportion.

$$F(v) = 1 - e^{-(\frac{v}{c})^k}$$

3. Mean Wind Speed (theoretical)

Once $k$and $c$are known, you can compute the mean from the fitted curve:

$$\bar{v} = c\,\Gamma(1 + \frac{1}{k})$$

where $\Gamma$is the gamma function.

See excel sheet called 'Weibull calculator' to test any calculated results

See Charts document for examples of charts