

Wind Speed Estimation and Conversion Jupyter notebook

Overview

The purpose of the Jupyter notebook presented in this project is to assign wind speeds to property locations in Florida that were damaged during hurricanes. The property locations can come from insurance data or from reconnaissance data. The notebook can process wind field input data on any grid format and for different exposure types (like marine conditions, open terrain, or urban terrain), different time averages (including 3-second gusts, 1-minute sustained winds, and 10-minute average), and different heights. The notebook performs three key functions: (1) linear interpolation from source grid points to any target location in Florida; (2) conversion of wind speeds between exposure types and time averages; and (3) conversion of wind speeds from one height to another, all using established wind engineering formulas.

Since 2017, Applied Research Associates (ARA), under contract with the National Institute of Standards and Technology (NIST), has provided wind field data on a grid, with both 10 m 3-second gusts and 1-minute sustained winds, both for open terrain, for hurricanes impacting the USA. These datasets are published in the DesignSafe reconnaissance portal. They are a possible source of input for the Jupyter notebook. Other sources of wind field data can be used if available.

The other input data needed for the Jupyter notebook are the files with the lat/long of the properties, either from insurance data or from reconnaissance data (available in the DS reconnaissance portal). In the case of insurance data where issues of confidentiality might be at play, it is left at the discretion of the user to deal with these issues.

The final piece of input data that the Jupyter notebook needs is the surface roughness coefficient to characterize the terrain as urban, suburban, coastal, etc. They are currently contained in a GeoTIFF data file provided by Dr. Steve Cocke from Florida State University. This current GeoTIFF file covers only the state of Florida, but provided that this data is available for other states, the notebook could be extended to other states. It needs to be updated as needed since terrain exposure varies over time and development.

The Jupyter notebook reads the wind field data and first interpolates wind speeds to exact property lat/lon coordinates using linear interpolation, then applies terrain adjustment formulas to get actual terrain wind speeds at each property location. If needed, a final adjustment is performed to bring the winds to the desired height. For example, given any hurricane wind field data from ARA (e.g., hurricane Michael), the notebook interpolates the ARA 10m 3-second gusts or 1-minute sustained winds to any given location, then adjusts these values based on terrain roughness (e.g., urban, coastal, or forested), to get actual terrain 10m wind speeds at that location.

Methodology

The methodology addresses the user's need for location-specific wind estimation. The model begins by preparing and validating input data, accepting wind field information in any grid format while ensuring coordinate accuracy and proper unit conversion. The core of the analysis lies in the spatial interpolation stage, where the notebook calculates wind speed estimates at any specified location. The input data and the corresponding interpolated data can be for different time averages (e.g., 3-second gust or 1-minute sustained wind). The model then applies validated terrain adjustments. These modifications account for real-world environmental factors, including urban drag effects, coastal wind enhancements, and surface roughness variations derived from high-resolution GeoTIFF data. Each adjustment follows established meteorological principles. This data was provided by Dr. Steven Cocke (Florida State University) through a dedicated Jupyter notebook, which is incorporated in this model. The imported dataset—preserving its original formulas, databases, and methodology—accounts for:

- Urban Drag: Reduces wind speeds by 15% in cities (buildings create friction)
- Coastal Effects: Boosts winds near shorelines (+10-15%)
- Surface Roughness: Adjusts for terrain exposure like forests, fields, and water (using roughness coefficients z_0 values from GeoTIFF files).

The final stage is a map visualization. The methodology comprises the following key stages.

1- Data Preparation & Configuration

The methodology begins with data preparation to ensure correct inputs. The model requires two primary datasets: wind field data in CSV format from any given location with different exposure types (marine, open terrain, urban) and time - averages (3-second gusts, 1-minute sustained, 10-minute averages); and property location data with policy IDs and geographic coordinates (longitude, latitude).

- **Wind Field CSV**
 - Arbitrary grid; any height or exposure
 - Columns: Longitude, Latitude, plus one or more wind-speed fields
 - config1.txt must supply:
 - wind_data, wind_speed_columns
 - input_exposure_type (open_terrain|marine|urban)
 - input_time_interval (gust_3s|sustain_1min|avg_10min)
 - reference_height, target_height

- **Property Locations CSV**
 - Required: ID, Lon, Lat
- **Terrain Rasters**
 - rough_2022.tif → surface roughness z_0
 - distance_2022.tif → distance to coastline (km)
- **Gust-Factor Binaries**
 - gfac_3sec.dat, gfac2_3sec.dat, gfacm.dat, gfaco.dat

2- Wind Interpolation

The wind speed interpolation process employs a deterministic linear interpolation approach using `scipy.interpolate.griddata` with the linear method. This algorithm estimates both open-terrain 3-second gust wind speeds (mph) and 1-minute sustained wind speeds (mph) at each property location by analyzing the surrounding wind field data points. The mathematical foundation uses inverse distance weighting, where the interpolated wind speed value is calculated as the weighted sum of nearby wind speeds. The implementation handles edge cases systematically; for coordinates falling outside the wind field grid boundaries, it applies nearest-neighbor extrapolation to provide reasonable estimates, while explicitly maintaining null values (NaN) for locations where reliable interpolation cannot be performed. The results are output in a structured CSV format containing property coordinates alongside their corresponding interpolated wind values, ensuring the data remains organized for subsequent terrain adjustment processing.

The wind speed at any given location is estimated by calculating a weighted average of observed wind speeds from nearby measurement points or stations. This process accounts for spatial variation by giving more influence to closer stations.

3- Wind Speeds Conversion

This stage transforms the interpolated “reference” wind (10 m, marine-10 min basis) into actual-terrain 3s gusts by adjusting it according to local terrain characteristics, resulting in an actual, site-specific wind speed estimate. The process involves several phases:

- Terrain-Specific Wind Adjustment

The model uses the rough_2022.tif GeoTIFF raster, which is provided by Dr. Steven Cocke (Florida State University) through a dedicated Jupyter notebook, which maps terrain roughness lengths (z_0) across the study area. Terrain is classified into four main categories, each with a characteristic roughness range:

Water surfaces: $z_0 = 0.0002\text{--}0.002$ m;

Open terrain: $z_0 = 0.002\text{--}0.03$ m;

Forested areas: $z_0 = 0.03\text{--}0.4$ m;

Urban environments: $z_0 = 0.4\text{--}1.0$ m;

The roughness length z_0 influences how wind speed varies with height above the ground.

- Wind Profile Adjustment Using Logarithmic Scaling

The wind speed at height z is calculated from the reference wind speed at 10 m height over open terrain by applying the logarithmic wind profile formula.

- Coastal Proximity Effects

Coastal zones experience different wind dynamics due to the transition from water to land surfaces. The model uses the distance_2022.tif raster, which is provided by Dr. Steven Cocke (Florida State University) through a dedicated Jupyter notebook, to compute the distance of each location from the coastline. Within 50 km of the shore, specialized gust factor tables (gfac2_3sec.dat) are applied to adjust wind gusts, capturing the marine-to-terrain transition effects on wind behavior.

Coastal Transition:

- ≤ 50 km: use gfac2_3sec.dat
- 50 km: use gfac_3sec.dat

Urban Drag:

- If $z_0 > 0.4$ m, reduce by 15 %

Time-Basis Conversion:

- 1 min \rightarrow 10 min via gfacm.dat / gfaco.dat
- Gust \leftrightarrow 10 min invert via gfac_3sec.dat

4- Mapping & Visualization

This report presents two complementary approaches to visualize actual terrain-adjusted wind speed data, each with distinct strengths and suited for different purposes:

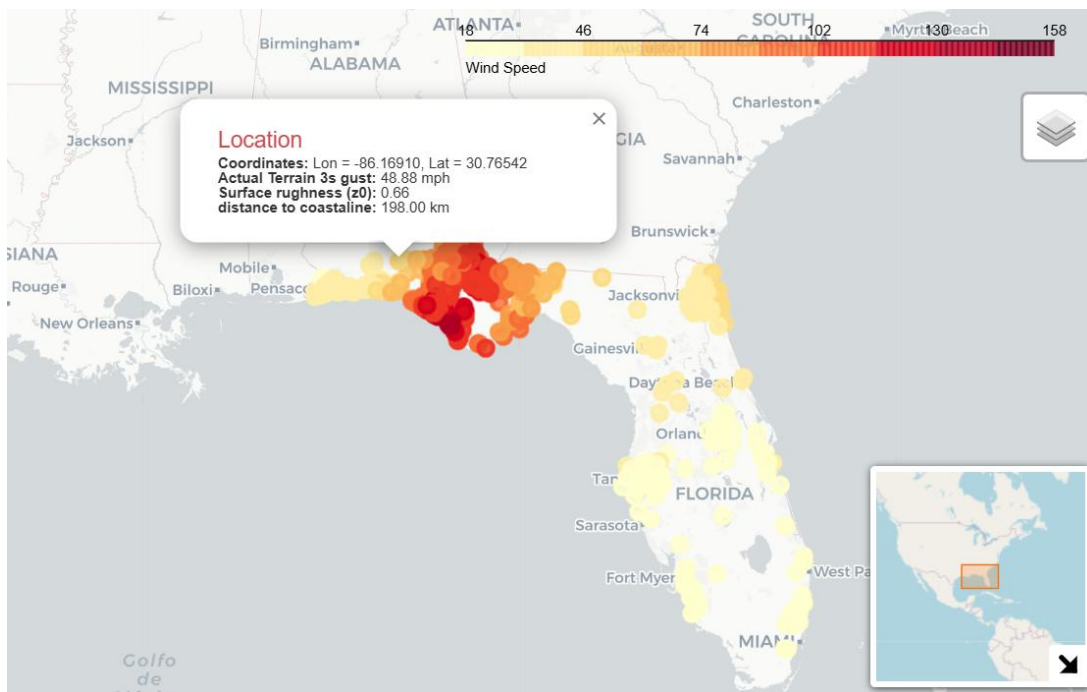
I. Static Geospatial Map with Matplotlib

This method generates a static map where data points are displayed as scatter points colored according to the actual 10 m height, 3-second gust wind speed (ActualT_3sg), utilizing a reversed Inferno colormap to enhance contrast. The Florida state boundaries are approximated by limiting the map extents, and a high-resolution satellite basemap from Esri

World Imagery is incorporated to provide geographic context. The map features a vertical colorbar, a title, and a clean layout with axes hidden for clarity.

II. Interactive Web Map with Folium

This method creates an interactive web map that can be embedded in Jupyter notebooks or saved as a standalone HTML file. The map is centered on the mean latitude and longitude of the filtered dataset, adapting dynamically to the geographic extent of the data—Florida in this case, but applicable to other regions or states. Wind speed data points are represented as color-coded circle markers using a sequential YlOrRd colormap scaled to the range of wind speeds. Each marker features a clickable pop-up providing detailed, location-specific information, including geographic coordinates, ActualT_3sg, z0_raster, and dist_raster, aiding in the interpretation of local wind conditions. Additional interactive features, such as a color legend, minimap overview, and layer control widgets, enhance usability. The map can be saved as an HTML file for viewing in any web browser or displayed directly within a Jupyter Notebook.



Interactive map, actual terrain, 3-second gust wind speed (ActualT_3sg),

Exports:

- interpolated.csv
- wind speeds conversion.csv
- Map file

Project Directory Structure

The project folder is organized into several subfolders and key files, each serving a specific role in the wind speed estimation and conversion workflow:

1. Configuration/

This configuration file (config1.txt) defines all the settings for the three-part Jupyter Notebook workflow that estimates and converts wind speeds at property locations. It includes:

- Input/output file paths
- Selected wind-speed columns
- Exposure and time-interval metadata
- Reference and target heights
- Terrain raster paths
- Output CSV paths

2. Gust-Factor Binaries/

Contains binary lookup tables used for time-basis and coastal/terrain conversions:

- gfac_3sec.dat, gfac2_3sec.dat — Coastal/terrain gust factors
- gfacm.dat, gfaco.dat — Time-basis conversions (e.g., 1-min ↔ 10-min, gust ↔ sustained)

3. Interpolation&ConversionOutput/

Stores notebook-generated output files:

- interpolated.csv — Raw interpolated wind speeds at property locations
- Wind speeds conversion.csv — Final terrain-adjusted 3-second gusts including roughness (z0) and distance-to-coast columns

4. Property Locations/

Contains CSV files with property exposures, including:

- Required columns: ID, Longitude, Latitude
- Used by the notebook to map wind values to specific locations

5. Terrain Rasters/

GeoTIFF files used for terrain adjustments:

- rough_2022.tif — Surface roughness (z0) raster
- distance_2022.tif — Distance-to-coast raster

6. Wind-field Data/

Gridded wind datasets for hurricane or regional events:

- Each CSV should include Longitude, Latitude, and the wind-speed columns specified in config1.txt (e.g., WindSpeed (3 gust_mph), WindSpeed (1 sust_mph))

7. Key Files

- **wind_map.html** — Interactive Folium map visualizing interpolated wind speeds and nearest nodes
- **Wind_Speed_Estimation & Conversion_Notebook.ipynb** — Main Jupyter notebook executing the full workflow

Typical run (in notebook)

1. Edit config1.txt if needed (paths, columns).
2. Run notebook cells in order.
3. Provide longitude/latitude when prompted for single-point checks, or run batch processing to process all properties.

How to run

1. Load config1.txt and input datasets.
2. Interpolate wind data from the gridded wind field to the property Lon/Lat
3. Convert interpolated winds (apply terrain/gust-factor corrections using binary tables and rasters)
4. Save intermediate and final CSV outputs; create interactive maps.

Common issues & tips

- Column name mismatches: ensure config1.txt wind_speed_columns matches CSV headers (case and spaces matter).
- CRS: Rasters are assumed to align with lon/lat. If not, reproject query coordinates to raster CRS before indexing.
- Missing output column: If ActualT_3sg or similar is absent, check Interpolation&ConversionOutput for the correct final file and column names.

Dependencies

Before running the notebook, make sure the following Python libraries are installed:

Category	Package	Description
Core scientific computing	numpy	Numerical operations and array handling
	pandas	Data manipulation and CSV file management
	scipy	Interpolation and spatial analysis
Geospatial processing	rasterio	Reading and querying GeoTIFF raster datasets
	pyproj	Coordinate transformations (e.g., Lon/Lat → Web Mercator)
Mapping & visualization	folium	Interactive web maps (Leaflet.js wrapper)
	branca	Pop-up and HTML templating for Folium
	contextily	Static basemaps (tile layers for matplotlib)
	matplotlib	Static plotting and visualization

License

This notebook is released under the BSD 3-Clause License.

Data Sources and Confidentiality

The Jupyter notebook also requires input files containing the latitude and longitude of the analyzed properties. These data can be obtained either from insurance claim records or from publicly available reconnaissance datasets (such as those hosted on the DesignSafe Reconnaissance Portal).

For insurance-related data, confidentiality restrictions may apply. Users are responsible for ensuring that any proprietary or sensitive information is handled in compliance with data privacy agreements and institutional policies. The dataset used in this notebook does not contain any company identifiers or personally identifiable information, ensuring confidentiality is maintained.

Acknowledgements

This research was supported by the National Science Foundation (NSF) through the following awards: DesignSafe Cyberinfrastructure, Award No. 2022469. The opinions, conclusions, and results presented in this document are solely those of the authors and do not necessarily represent the views of the NSF.