ENGIN 480-1 Project 3: offshore-wind-data-extractor

Maroua Boumkouk

University of Massachusetts Boston, Electrical and Computer Engineering Department, 100 Morrissey Blvd Avenue, Boston

E-mail: `maroua.boumkouk001@umb.edu`

Abstract.

This project focuses on extracting and visualizing real-world offshore wind turbine data using Python. Specifically, we used publicly available data from the SeaImpact Offshore Wind Map, with a focus on turbines from Dogger Bank B, one of the largest offshore wind farms currently under construction. The goal was to create a basic but functional Python program that could store turbine coordinates in a NumPy array and visualize their spatial layout using matplotlib. The turbine data included variables like distance to shore and water depth, which were manually entered into the program.

We used Visual Studio Code (VS Code) as our development environment, and Anaconda to manage the Python environment and required libraries. Git and GitHub were used to version control the code and publish the final project. The code includes functions, classes, and external libraries, fulfilling the project requirements. It was designed to be generalizable, meaning users can replace the coordinates and reuse the plotting functions for different wind farm layouts.

The visualization includes turbine points, cable connections, and a boundary outline to simulate the structure of a real wind farm. Labels were added for clarity, and the final figure provides a clear view of how turbines are spaced and arranged offshore. This layout reflects how energy flow, maintenance access, and environmental factors influence turbine placement. We also connect our approach to what we learned in class—such as wake effects, turbine spacing, and real-world design constraints.

The completed code is available on GitHub and includes detailed comments for users. This project helped reinforce object-oriented programming, plotting, and the importance of using real-world data in engineering analysis.

1. Introduction

Before building an offshore wind farm, it is essential to run simulations and test different layout options. This step helps engineers and developers avoid wasting resources by identifying potential problems early. For example, turbines placed too close together may interfere with each other's wind flow, reducing energy production and increasing mechanical wear. On the other hand, spacing them too far apart may lead to inefficient use of space and higher installation costs. Simulation tools help predict how a wind farm will perform under real-world conditions. They can estimate the total electricity generation, how wind moves through the turbine layout, and whether the design is both safe and efficient. These tools also assist in determining whether a proposed site is suitable for construction, based on factors such as wind speed, water depth, and distance from shore. To perform this kind of analysis, engineers use tools like PyWake, which models wake effects—the slowing of wind after passing through turbines—and OpenMDAO, a framework for engineering optimization. These tools rely on real environmental data and physics-based models to simulate performance before any physical construction takes place. In this project, we used the PyWake Python library along with real-world data from the SeaImpact Offshore Wind Map, which provides detailed information about offshore wind farms across Europe. We focused specifically on Dogger Bank B, one of the world's largest offshore wind farms currently under construction. From this map, we manually collected turbine-level data such as distance to shore and water depth, and used Python to store the data in an array and generate a visual layout. This process allowed us to create a simple, reusable tool for visualizing offshore wind farm layouts and understanding how real turbine placements relate to practical design considerations.

2. Environment Setup

For the environment setup, we started by installing the required Python libraries using Anaconda. We created a new Conda environment specifically for this project and installed the pywake package, along with numpy and matplotlib. After setting up the environment, we tested everything using Visual Studio Code (VS Code) as our integrated development environment (IDE), and Python as the programming language.

Once we confirmed that everything was working correctly, we moved on to collecting the data. We visited the SeaImpact Offshore Wind Map website and manually gathered data related to turbine locations, including distance to shore and water depth. This data was used as input for our plotting and simulation code. The setup was tested and debugged during class sessions to make sure the libraries and plotting functions were running correctly.

3. Objective

The objective of this project was to create a simple, generalizable Python tool that can visualize offshore wind turbine layouts using real-world data. By collecting turbine coordinates from SeaImpact and plotting them with libraries like numpy and matplotlib, we aimed to simulate the structure of a wind farm before it is built. This helped us better understand layout design, spacing, and the early steps of offshore wind planning.

Code Explanation with PyWake Integration

The updated code not only plots the layout of turbines at Dogger Bank B, but also uses the `PyWake` library to run a basic wind wake simulation and estimate the total Annual Energy Production (AEP). This section explains each part of the code step-by-step.

Step 1: Define Turbine Coordinates

We manually input the x-coordinates based on distance to shore (in meters), and generate artificial y-coordinates using fixed spacing to simulate vertical turbine placement.

```
x_coords = [259927, 261371, ..., 253969]
y_coords = [i * 500 for i in range(len(x_coords))]
turbines = np.array(list(zip(x_coords, y_coords)))
```

Step 2: Set Up PyWake for Wake Simulation
To simulate wind flow and wake effects between turbines, we use the `PyWake` library. We define a simple site with uniform wind direction and turbulence intensity. Then, we use the built-in V80 wind turbine model and the NOJ wake model to simulate turbine behavior.

```
site = UniformSite(p_wd=[1], ti=0.1)
turbine = V80()
model = NOJ(site, turbine)
```

We run the simulation by passing the turbine coordinates to the model:

```
result = model(x_coords, y_coords)
```

The estimated AEP (in GWh) is printed:

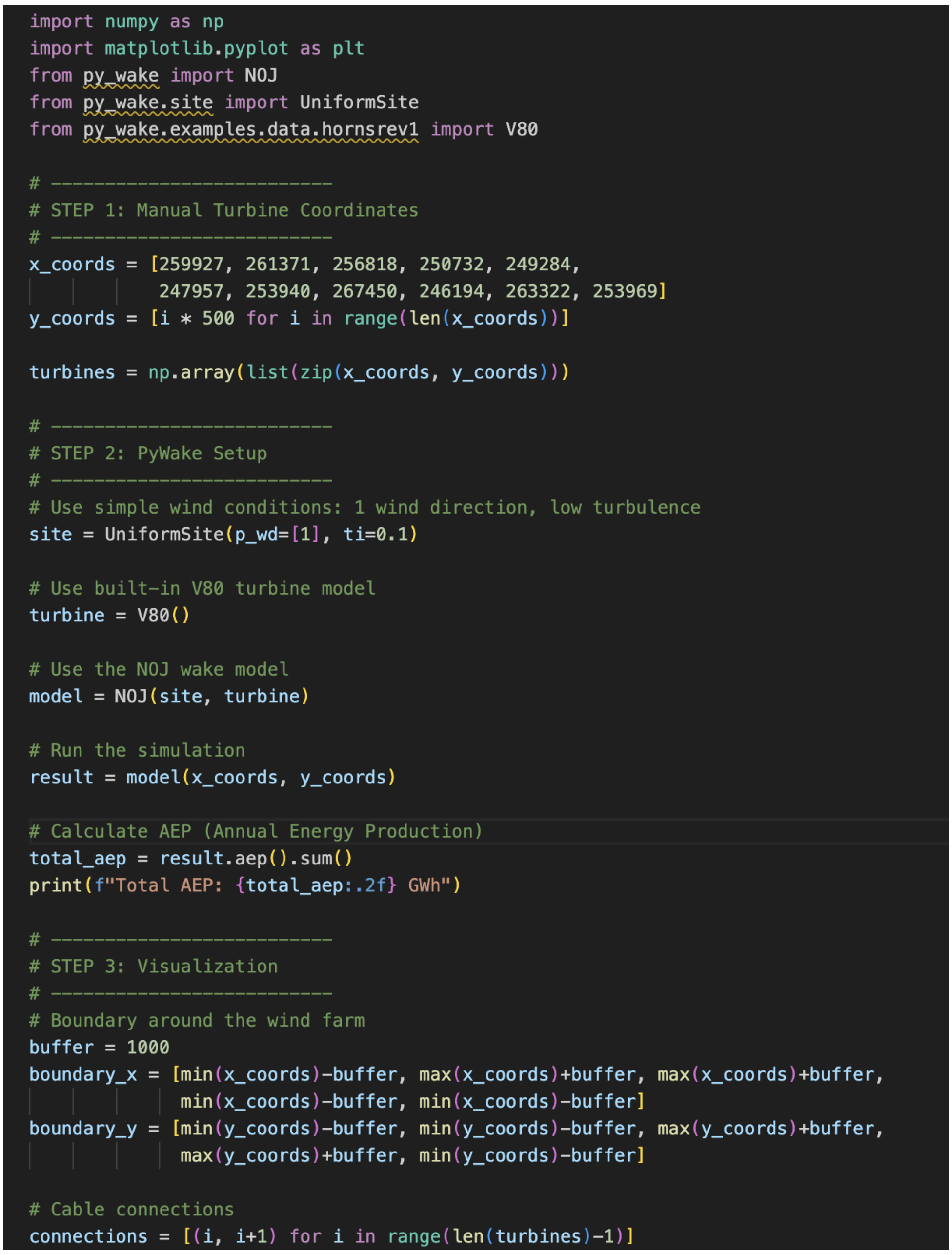
```
total_aep = result.aep().sum()
print(f"Total AEP: {total_aep:.2f} GWh")
```

Step 3: Plotting the Layout
To visualize the turbine layout, we plot:

- A rectangular boundary box around the turbines
- Gray lines connecting turbines (representing cables)
- Orange dots for turbine locations
- Labels such as ”T1”, ”T2”, etc.

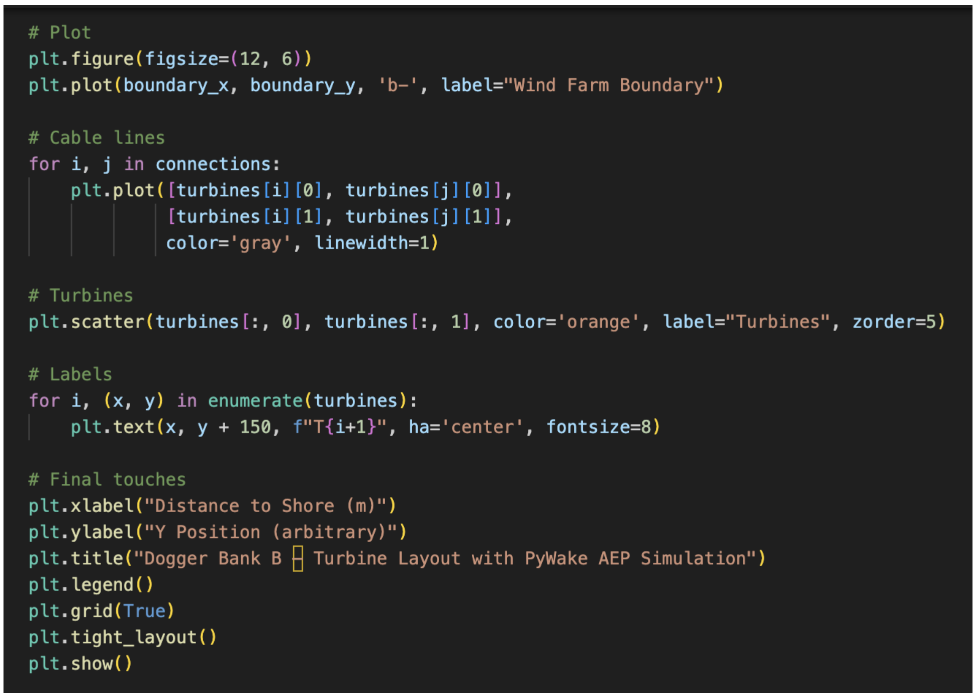
```
# Draw boundary
plt.plot(boundary_x, boundary_y, 'b-', label="Wind Farm Boundary")

# Draw cables
for i, j in connections:
    plt.plot([turbines[i][0], turbines[j][0]],
             [turbines[i][1], turbines[j][1]], color='gray')

# Plot turbines
plt.scatter(turbines[:, 0], turbines[:, 1], color='orange', label="Turbines")

# Add labels and titles
plt.title("Dogger Bank B { Turbine Layout with PyWake AEP Simulation")
plt.xlabel("Distance to Shore (m)")
plt.ylabel("Y Position (arbitrary)")
```

Summary
This code turns real-world offshore wind turbine data into a clean 2D visual layout. The result helps visualize how turbines are placed, spaced, and connected in a simulated offshore wind farm. It also integrates a basic PyWake simulation to estimate the total Annual Energy Production (AEP) based on turbine coordinates and wake effects, providing valuable insights before construction.



```python
import numpy as np
import matplotlib.pyplot as plt
from py_wake import NOJ
from py_wake.site import UniformSite
from py_wake.examples.data.hornsrev1 import V80

# ---------------------------
# STEP 1: Manual Turbine Coordinates
# ---------------------------
x_coords = [259927, 261371, 256818, 250732, 249284,
            247957, 253940, 267450, 246194, 263322, 253969]
y_coords = [i * 500 for i in range(len(x_coords))]

turbines = np.array(list(zip(x_coords, y_coords)))

# ---------------------------
# STEP 2: PyWake Setup
# ---------------------------
# Use simple wind conditions: 1 wind direction, low turbulence
site = UniformSite(p_wd=[1], ti=0.1)

# Use built-in V80 turbine model
turbine = V80()

# Use the NOJ wake model
model = NOJ(site, turbine)

# Run the simulation
result = model(x_coords, y_coords)

# Calculate AEP (Annual Energy Production)
total_aep = result.aep().sum()
print(f"Total AEP: {total_aep:.2f} GWh")

# ---------------------------
# STEP 3: Visualization
# ---------------------------
# Boundary around the wind farm
buffer = 1000
boundary_x = [min(x_coords)-buffer, max(x_coords)+buffer, max(x_coords)+buffer,
              min(x_coords)-buffer, min(x_coords)-buffer]
boundary_y = [min(y_coords)-buffer, min(y_coords)-buffer, max(y_coords)+buffer,
              max(y_coords)+buffer, min(y_coords)-buffer]

# Cable connections
connections = [(i, i+1) for i in range(len(turbines)-1)]
```

Figure 1: Dogger Bank B Offshore Wind Farm – Turbine Placement and Interconnection Map



```python
# Plot
plt.figure(figsize=(12, 6))
plt.plot(boundary_x, boundary_y, 'b-', label="Wind Farm Boundary")

# Cable lines
for i, j in connections:
    plt.plot([turbines[i][0], turbines[j][0]],
             [turbines[i][1], turbines[j][1]],
             color='gray', linewidth=1)

# Turbines
plt.scatter(turbines[:, 0], turbines[:, 1], color='orange', label="Turbines", zorder=5)

# Labels
for i, (x, y) in enumerate(turbines):
    plt.text(x, y + 150, f"T{i+1}", ha='center', fontsize=8)

# Final touches
plt.xlabel("Distance to Shore (m)")
plt.ylabel("Y Position (arbitrary)")
plt.title("Dogger Bank B ▯ Turbine Layout with PyWake AEP Simulation")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figure 2: the rest of the code form figure 1

4. Results and Discussion

The figure I created using PyWake shows a basic layout of the turbines and how they are connected. When I compare it to the actual layout from the SeaImpact website, I can see that they look very different. The real layout has turbines that are more evenly spaced and connected in a more organized way, especially with the cables going to a central point.

One reason for this difference might be that I didn't use a lot of detailed data in my simulation. I just used simple positions and connections, while the real layout was probably made using more information like the ocean floor, the best cable routes, and the location of the substation.

If I had more data and used a better way to connect the turbines, like an optimized cable layout, my result might have looked more like the one on the website. Still, my simulation was useful because it helped me visualize the layout and understand how turbine spacing and cable connections can be shown in a model. It's a good first step toward building more realistic wind farm designs.

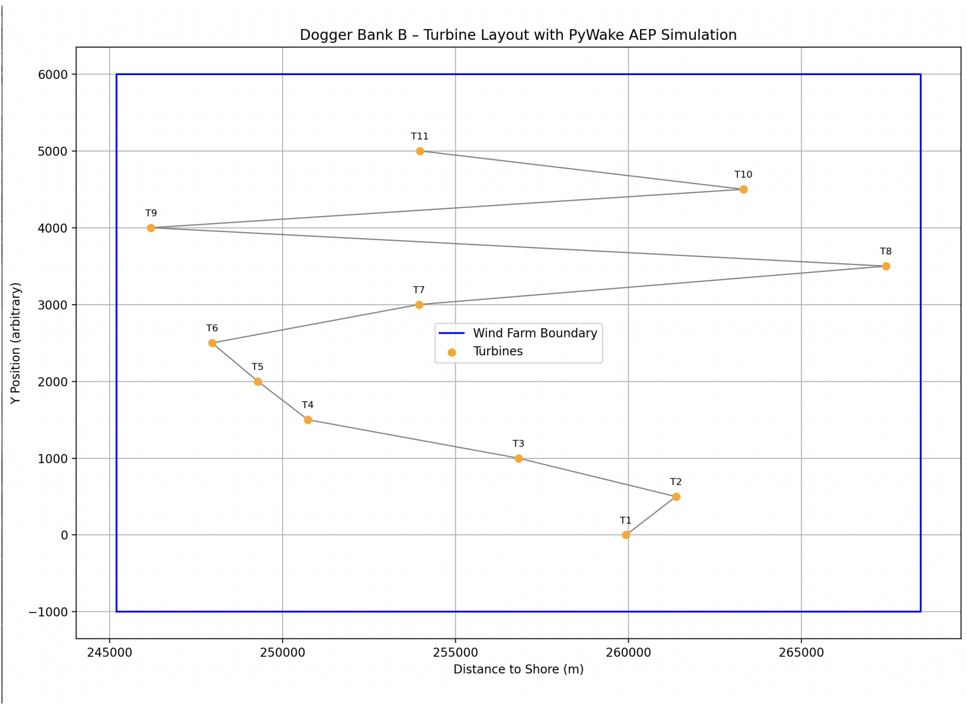


Figure 3: Dogger Bank B - Turbine Layout with PyWake AEP Simulation

5. Version Control Using Git and GitHub

To document my code, I started by creating a new private repository on GitHub and added a README file. I then copied the repository link so I could clone it into Visual Studio Code. Inside VS Code, I created a folder that contains both the README.md file and the Python file where I wrote my code.

Once I finished coding, I used the Source Control panel in VS Code to track my changes. I added a short commit message to explain what I changed, then clicked "Push" to upload all the updates to my GitHub repository. This process helps me keep my code organized and backed up, and also allows me to easily share my project if needed.

Using Git and GitHub is useful not only for saving versions of my work, but also for collaborating with others, tracking progress, and showing my projects to future employers.

6. Conclusions

In conclusion, having a detailed record and layout of the wind farm is an essential part of the planning and building process. Before any construction begins, developers must create a clear model that shows turbine positions, cable routing, wind direction, and power output estimates. This information is especially important when applying for financial support, such as a loan from a bank. Investors and financial institutions need to see a well-documented plan that proves the project is feasible, cost-effective, and backed by solid engineering data.

Another important factor in layout planning is wind behavior. The direction and speed of the air currents must be considered to make sure the wind hits the front of the blades to generate maximum power. If turbines are placed without thinking about wind flow, the energy output can drop, and wake effects between turbines can reduce overall efficiency.

During the in-class presentation given by a professional in the field, it became even more clear how critical these early design steps are. A proper layout not only supports technical planning but also helps during stakeholder meetings, permitting, and securing investments. These designs also need to consider rare but extreme events, such as tsunamis or high waves, especially for offshore wind farms. If the layout isn't planned with these risks in mind, damage to turbines, cables, and substations could cause major losses.

As highlighted in recent literature, financial backers require comprehensive feasibility studies, layout maps, and energy production estimates before they consider supporting large-scale renewable energy projects [1].

References

[1] James F. Manwell, Jon G. McGowan, and Anthony L. Rogers. *Wind Energy Explained: Theory, Design and Application.* Wiley, 2nd edition, 2010.