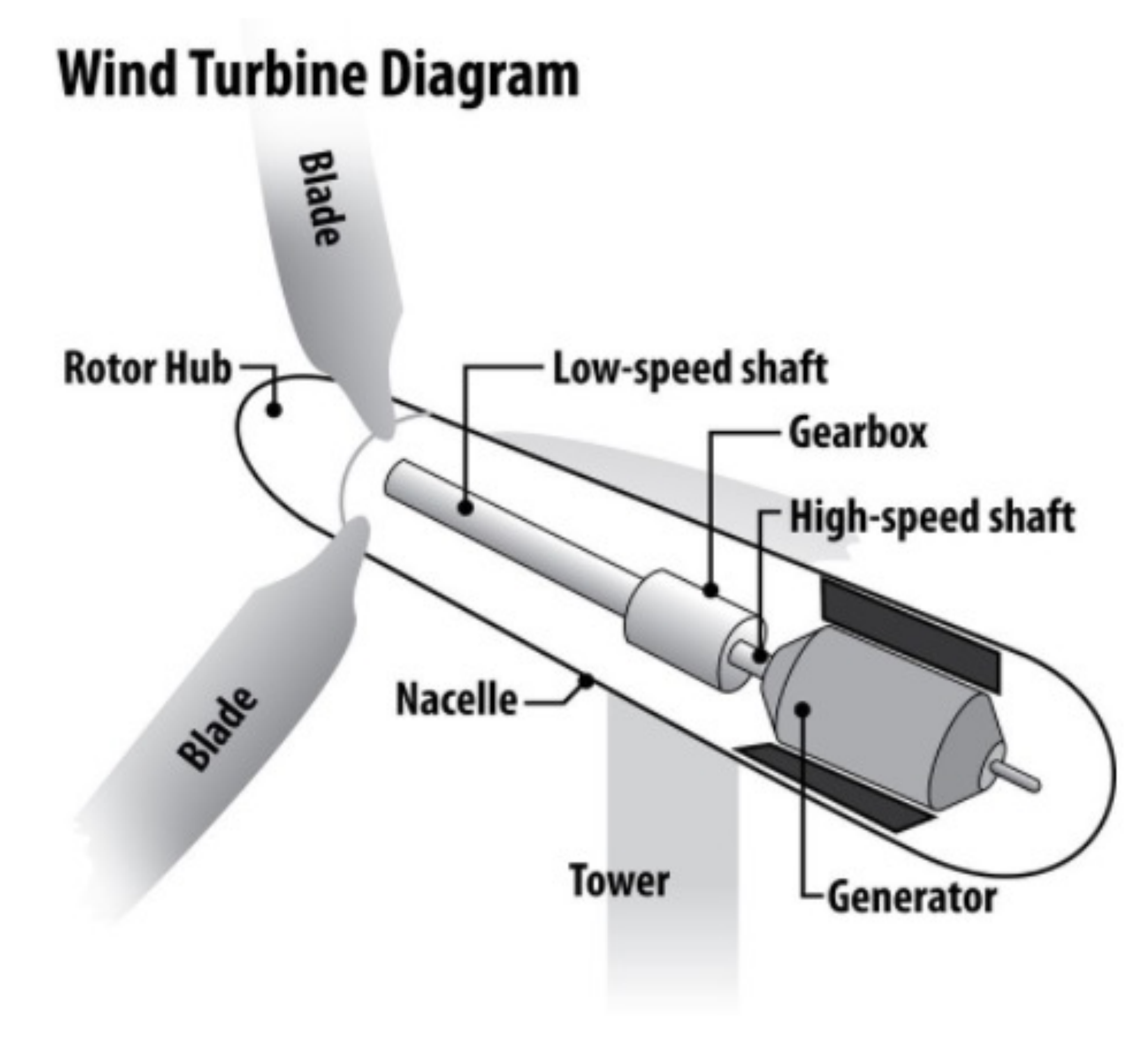
The role of computer programming in wind energy

By: Kara White



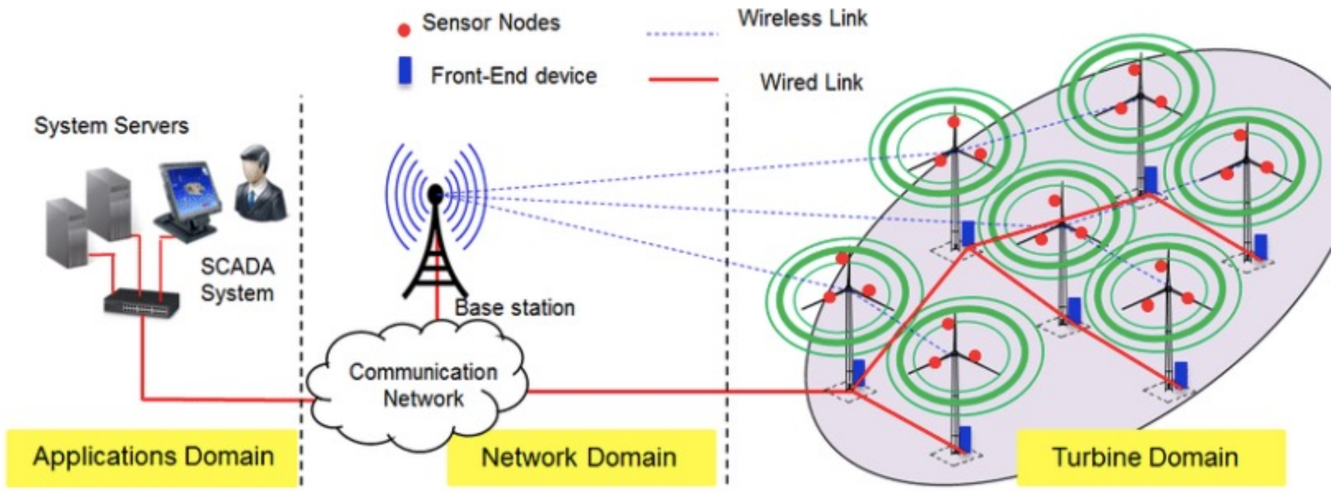
How does a wind turbine work?

- Wind turns blades around a rotor
- Rotor spins a generator and creates kinetic energy
- DC electricity is generated
- Electricity is converted to AC via inverter before entering the grid (or a home)

How is programming used for wind energy?

- Design turbine blades for maximum aerodynamic performance
- Prevent the wind turbine from exceeding safe max speed
- To determine Rated Power Capacity

Supervisory Control & Data Acquisition (SCADA)

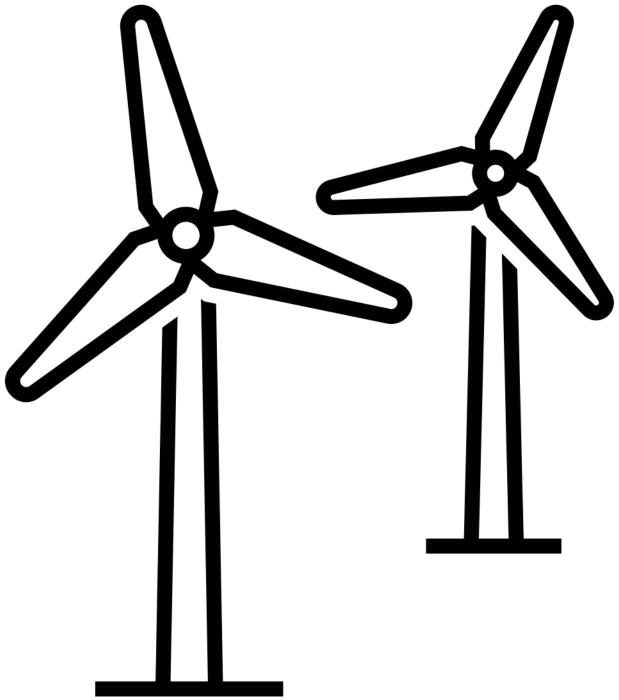


Using programming to build smarter wind energy

- "The National Wind Technology Center (NWTC) at NREL develops advanced computer-aided engineering (CAE) tools to support the wind power industries with state-of-the-art design and analysis capabilities." (NREL)

Airfoil data for aerodynamic performance of blades

- An airfoil is **the foundation of wind turbine blade design**
- AirfoilPrep (NREL)
  - Correcting the angles and direction of the blades for maximum efficiency
  - Uses characteristics of a turbine blade as parameters (weight, pitch (blade angle), drag)
  - Generates airfoil data files from 2-D data
  - Adjusts 2-D data for rotational augmentation (3-D effects)
  - PYTHON CODE (Source: NREL): https://github.com/WISDEM/AirfoilPreppy/blob/master/airfoilprep/airfoilprep.py



```
airfoilprep.py

Created by Andrew Ning on 2012-04-16.
Copyright (c) NREL. All rights reserved.

```

```python

class Polar(object):
    """
    Defines section lift, drag, and pitching moment coefficients as a
    function of angle of attack at a particular Reynolds number.

    """

    def __init__(self, Re, alpha, cl, cd, cm):
        """Constructor

        Parameters
        ----------
        Re : float
            Reynolds number
        alpha : ndarray (deg)
            angle of attack
        cl : ndarray
            lift coefficient
        cd : ndarray
            drag coefficient
        cm : ndarray
            moment coefficient
        """

        self.Re = Re
        self.alpha = np.array(alpha)
        self.cl = np.array(cl)
        self.cd = np.array(cd)
        self.cm = np.array(cm)

```

```
airfoilprep.py

Created by Andrew Ning on 2012-04-16.
Copyright (c) NREL. All rights reserved.

```

```
        import matplotlib.pyplot as plt

        p = self

        figs = []

        # plot cl
        fig = plt.figure()
        figs.append(fig)
        ax = fig.add_subplot(111)
        plt.plot(p.alpha, p.cl, label='Re = ' + str(p.Re/1e6) + ' million')
        ax.set_xlabel('angle of attack (deg)')
        ax.set_ylabel('lift coefficient')
        ax.legend(loc='best')

        # plot cd
        fig = plt.figure()
        figs.append(fig)
        ax = fig.add_subplot(111)
        ax.plot(p.alpha, p.cd, label='Re = ' + str(p.Re/1e6) + ' million')
        ax.set_xlabel('angle of attack (deg)')
        ax.set_ylabel('drag coefficient')
        ax.legend(loc='best')

        # plot cm
        fig = plt.figure()
        figs.append(fig)
        ax = fig.add_subplot(111)
        ax.plot(p.alpha, p.cm, label='Re = ' + str(p.Re/1e6) + ' million')
        ax.set_xlabel('angle of attack (deg)')
        ax.set_ylabel('moment coefficient')
        ax.legend(loc='best')

        return figs
```

```
airfoilprep.py

Created by Andrew Ning on 2012-04-16.
Copyright (c) NREL. All rights reserved.
```

```python
class Airfoil(object):
    """A collection of Polar objects at different Reynolds numbers

    """

    def __init__(self, polars):
        """Constructor

        Parameters
        ----------
        polars : list(Polar)
            list of Polar objects

        """

        # sort by Reynolds number
        self.polars = sorted(polars, key=lambda p: p.Re)

        # save type of polar we are using
        self.polar_type = polars[0].__class__
```

```
airfoilprep.py

Created by Andrew Ning on 2012-04-16.
Copyright (c) NREL. All rights reserved.

```

```python

    def correction3D(self, r_over_R, chord_over_r, tsr, alpha_max_corr=30,
                     alpha_linear_min=-5, alpha_linear_max=5):
        """Applies 3-D corrections for rotating sections from the 2-D data.

        Parameters
        ----------
        r_over_R : float
            local radial position / rotor radius
        chord_over_r : float
            local chord length / local radial location
        tsr : float
            tip-speed ratio
        alpha_max_corr : float, optional (deg)
            maximum angle of attack to apply full correction
        alpha_linear_min : float, optional (deg)
            angle of attack where linear portion of lift curve slope begins
        alpha_linear_max : float, optional (deg)
            angle of attack where linear portion of lift curve slope ends

        Returns
        -------
        polar : Polar
            A new Polar object corrected for 3-D effects
```

What aspects of this code have we already worked with?

Loops and conditions

2-D arrays and Numpy

Data analysis

Functions

Classes and Objects

Works Cited

https://www.earthtechproducts.com/faq-wind-power.html

https://www.ni.com/en-us/innovations/white-papers/08/wind-turbine-control-methods.html

https://www.intechopen.com/chapters/59756

https://www.axiomtek.com/Default.aspx?MenuId=Solutions&FunctionId=SolutionView&ItemId=45&Title=Wind+Power+Automationhttps://sites.psu.edu/sustainabilitycis/c-project-wind-power/

https://www.forbes.com/sites/quora/2017/12/15/what-role-can-computer-scientists-play-in-reducing-the-effects-of-climate-change/?sh=467714616758

https://www.nrel.gov/wind/computer-aided-engineering-tools.html

https://phys.org/news/2011-07-optimizes-farm.html

https://www.sciencedaily.com/releases/2020/04/200414095737.htm

https://www.hindawi.com/journals/sv/2019/6405153/