**1.**BladeGeometry**Class**

**Purpose**: Store and manage blade geometry data (chord, twist, airfoil data along the span).

**Attributes**:

* radius (float): Total blade radius.
* hub\_radius (float): Hub radius.
* r\_sections (list[float]): Radial positions along the blade.
* chord (list[float]): Chord length at each section.
* twist (list[float]): Twist angle (degrees/radians) at each section.
* airfoil\_data (list[dict]): Airfoil polar data (Cl, Cd vs. alpha) for each section.

**Methods**:

* \_\_init\_\_(self, radius, hub\_radius, r\_sections, chord, twist, airfoil\_data)
* get\_section\_properties(self, i) → Returns (r, chord, twist, airfoil\_data) for section i.

**2.**OperationalConditions**Class**

**Purpose**: Store operational parameters (wind speed, rotor speed, pitch angle).

**Attributes**:

* wind\_speed (float): Freestream wind speed (m/s).
* rotor\_speed (float): Rotational speed (rad/s or RPM).
* pitch\_angle (float): Blade pitch angle (degrees/radians).
* air\_density (float): Air density (kg/m³).

**Methods**:

* \_\_init\_\_(self, wind\_speed, rotor\_speed, pitch\_angle, air\_density=1.225)
* get\_tip\_speed\_ratio(self) → Computes TSR (rotor\_speed \* radius / wind\_speed).

**3.**MomentumTheory**Class**

**Purpose**: Compute axial (a) and tangential (a\_prime) induction factors.

**Methods**:

* compute\_induction\_factors(self, a\_guess, a\_prime\_guess, phi, solidity, Cn, Ct) → Solves for a and a\_prime iteratively (e.g., using fixed-point iteration).

**4.**BladeElementTheory**Class**

**Purpose**: Compute lift and drag forces at each blade section.

**Methods**:

* compute\_local\_angle\_of\_attack(self, phi, twist, pitch\_angle) → Returns local AoA (alpha = phi - twist - pitch).
* compute\_forces(self, alpha, chord, V\_rel, airfoil\_data) → Returns (Cl, Cd, Cn, Ct) using airfoil polars.

**5.**BEMSolver**Class (Main Iterative Loop)**

**Purpose**: Balance momentum and blade element equations iteratively.

**Attributes**:

* blade (BladeGeometry object).
* ops (OperationalConditions object).
* tolerance (float): Convergence threshold.
* max\_iterations (int): Max iterations allowed.

**Methods**:

* \_\_init\_\_(self, blade, ops, tolerance=1e-5, max\_iterations=100)
* solve\_section(self, i) → Solves BEM equations for a single blade section.
* solve(self) → Runs BEM for all sections and returns performance metrics.

**6.**PerformanceMetrics**Class**

**Purpose**: Integrate sectional forces to compute power, thrust, and torque.

**Methods**:

* compute\_thrust(self, Fn\_list) → Integrates normal forces.
* compute\_torque(self, Ft\_list, r\_list) → Integrates tangential forces.
* compute\_power(self, torque, rotor\_speed) → Computes power (P = torque \* rotor\_speed).