Hochschule Flensburg University of Applied Sciences

Wind Energy Technology Institute

Blade Element Momentum Theory

Exercise 02 : BEM code for the NREL 5MW Wind Turbine

Presented by:	Matriculation No.
Hassan Pour Saeid	730257
Mozafary Mostafa	750247
Patel Manthan	750301
Patil Rahul Bhatubhai	750532
Soni Karan	760153

Overview

1. Flow charts
2. Highlight of the Code
3. Results
4. Problems and Solutions
5. Summary
6. Contribution
7. References

1. Flow Chart

Flow chart of BEM code:

Start

Define Parameters

Calculate $V_{tilt},$ V_{tilt}=vcos ϕ

Import Airfoil-data

Calculate local Tip-Speed ratio, $\lambda_r = \frac{w*r}{v_t}$

Use three Formula to Find first assumption of flow angle, a & a’

$$\beta = 90° - tan^{-1}\left(\frac{1}{\lambda_r}\right)$$

$$\frac{a}{1-a} = \frac{\sigma_r\ (c_L\ \ cos\ \phi)}{4\ \sin^2(\phi)}$$

$$\frac{a\prime}{1-a} = \frac{\sigma_r\ c_L}{4\lambda_r \sin\phi}$$

[3]

Use original equation to find flow angle, a & a’

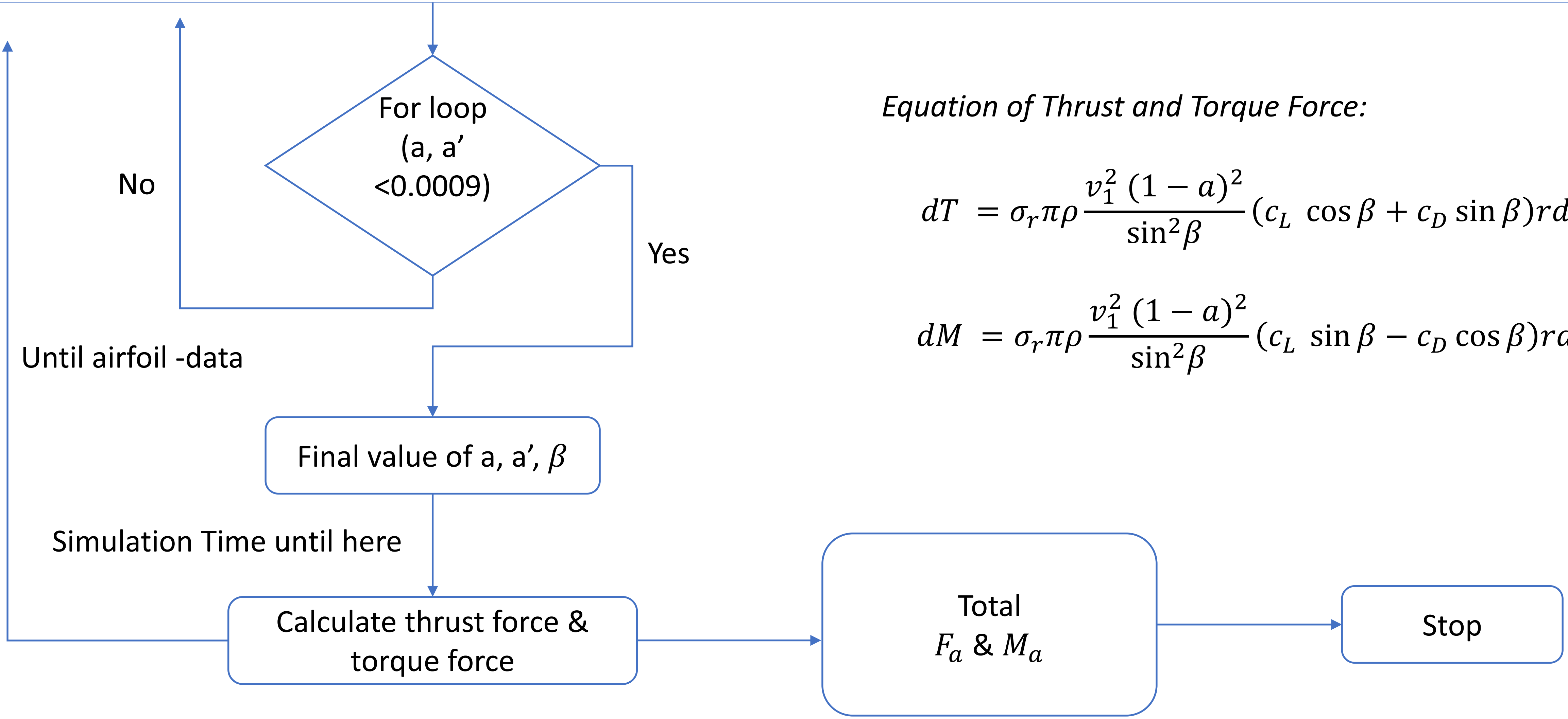
$$\beta = tan^{-1}\left(\frac{\lambda_r(1+a')}{1-a}\right)$$

$$\frac{a}{1-a} = \frac{\sigma_r\ (c_L\ \ sin\ \beta + c_D\ cos\ \beta)}{4Qcos^2\beta}$$

$$\frac{a\prime}{1-a} = \frac{\sigma_r\ (c_L\ \ cos\ \beta - c_D\ sin\ \beta)}{4Q\lambda_r cos^2\beta}$$

[3]

1. Flow Chart



Equation of Thrust and Torque Force:

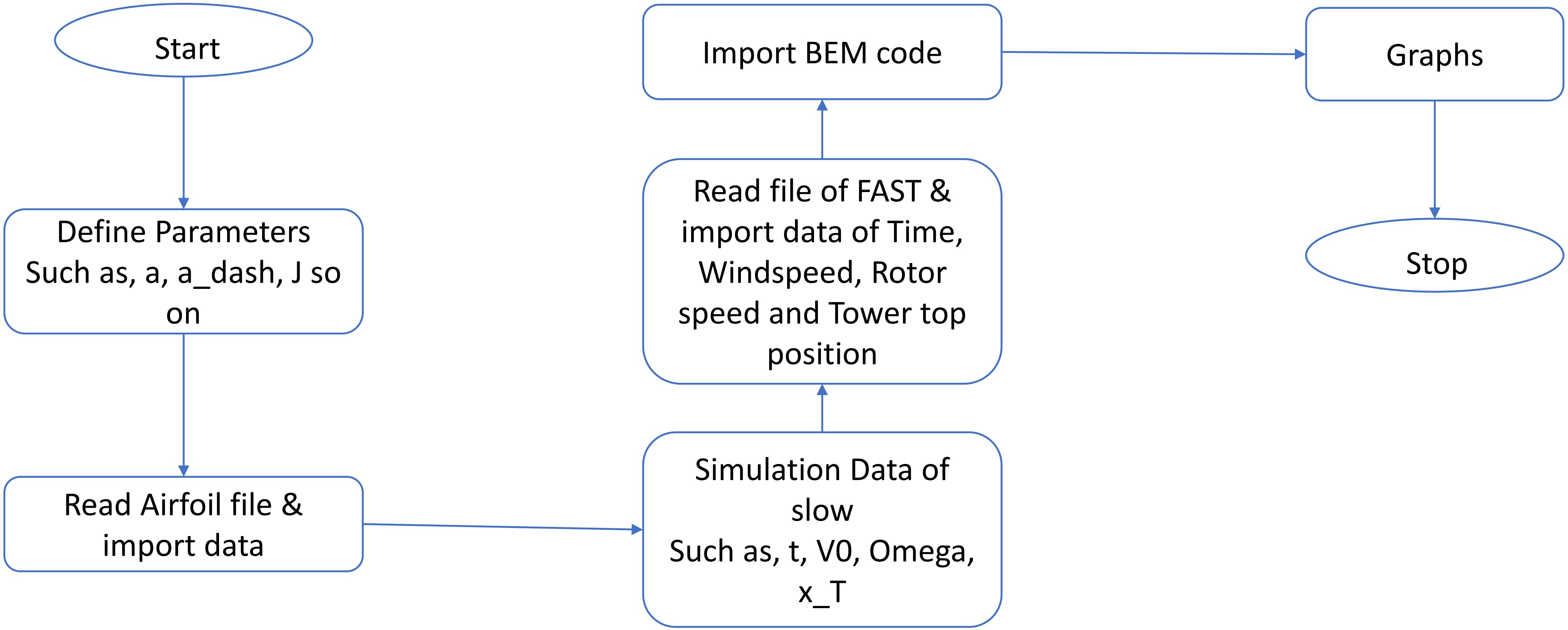
$$dT = \sigma_r \pi \rho \frac{v_1^2 (1-a)^2}{\sin^2\beta}(c_L \cos\beta + c_D \sin\beta) r dr$$

$$dM = \sigma_r \pi \rho \frac{v_1^2 (1-a)^2}{\sin^2\beta}(c_L \sin\beta - c_D \cos\beta) r dr$$

[3]

1. Flow Chart

Flow chart of Simulation and comparison of FAST and SLOW code:



2. Highlight of the Code

```
% Iterate to find axial and tangential induction factors
for iter = 1:30
phi = atan((1 - a) * v_0 / ((1 + ap) * Omega * r)); % Angle of relative flow
AOA = phi * 180 / pi - TA; % Angle of Attack
 N_airfoil = BEM.NFoil(i); % Number of Airfoils
L_airfoil = BEM.DRNodes(i); % length of airfoil for each node

% Compute Cl and Cd directly from the BEM
Cl = interp1(BEM.AoA{N_airfoil}, BEM.Cl{N_airfoil}, AOA, 'linear', 'extrap’);
Cd = interp1(BEM.AoA{N_airfoil}, BEM.Cd{N_airfoil}, AOA, 'linear', 'extrap’);

% Calculation of axial and tangential induction factor
Sigma_i = Sigma(i);
lambda_r_i = lambda_r(i);

eq1 = @(a_new) a_new / (1 - a_new) - (Sigma_i * Cl * cos(phi)) / (4 * sin(phi)^2);

 eq2 = @(ap_new, a_new) ap_new / (1 - a_new) - (Sigma_i * Cl) / (4 * lambda_r_i * sin(phi));

a_new = fzero(eq1, a);
ap_new = fzero(@(ap_new) eq2(ap_new, a_new), ap);
```

MATLAB
a & a’

2. Highlight of the Code


```
# Function to calculate the new flow angle (new_beta)
def new_Beta(local_tsr, a, a_dash):
return math.degrees(math.atan((local_tsr * (1 + a_dash)) / (1 - a)))
# Function to calculate the axial induction factor (a)
def axial_induction_factor(B, c, r, new_beta, Cl):
Q = ((2 / math.pi) * math.acos(math.exp(-((B / 2) * (1 - r / R)) / (r / R * math.cos(math.radians(new_beta))))))
Solidity = (B * c / (2 * math.pi * r))
numerator = Solidity * Cl * math.sin(math.radians(new_beta))
denominator = 4 * Q * (math.cos(math.radians(new_beta))**2) + Solidity * Cl * math.sin(math.radians(new_beta))
return numerator / denominator
# Function to calculate the tangential induction factor (a_dash)
def tangential_induction_factor(B, c, r, new_beta, a, Cl, Cd):
Q = ((2 / math.pi) * math.acos(math.exp(-((B / 2) * (1 - r / R)) / (r / R * math.cos(math.radians(new_beta))))))
Solidity = (B * c / (2 * math.pi * r))
numerator = Solidity * (1 - a) * Cl
denominator = 4 * Q * ((2 * math.pi * blade_rotational_speed * r) / (Vt * 60)) * (math.cos(math.radians(new_beta)))
return numerator / denominator
```

PYTHON
a & a’

Q = Tip loss correction factor

Cl & Cd

Vt = Tilt Velocity

Hochschule Flensburg University of Applied Sciences

2. Highlight of the Code

Wind Energy Technology Institute

MATLAB
Thrust & Torque

```
% Thrust force and moment calculation for each airfoil

Thrust_in_points(i) = Sigma_i * pi * rho * v_0^2 * (1 - a)^2 * (Cl * cos(phi) + Cd * sin(phi)) * r * L_airfoil / sin(phi)^2;
Moment_in_points(i) = Sigma_i * pi * rho * v_0^2 * (1 - a)^2 * (Cl * sin(phi) - Cd * cos(phi)) * r^2 * L_airfoil / sin(phi)^2;
end

% Total of Thrust force and moment
F_a = sum(Thrust_in_points);
M_a = sum(Moment_in_points);
end
```

2. Highlight of the Code

PYTHON

Thrust & Torque

```
# Function to calculate aerodynamic thrust force for each section
def aerodynamic_thrust_force(B, Beta, a, Density, Vt, r):
Solidity = (B * c / (2 * math.pi * r))
thrust = (Solidity * math.pi * Density * Vt**2 * (1 - a)**2 * (Cl* math.sin(math.radians(Beta)) + Cd * math.cos(math.radians(Beta))) * r * DRNodes) /
            math.cos(math.radians(Beta))**2
return thrust
# Function to calculate aerodynamic torque force for each section
def aerodynamic_torque(B, R, Beta, a, a_dash, Density, Vt, r, blade_rotational_speed):
Solidity = (B * c / (2 * math.pi * r))
torque = (Solidity * math.pi * Density * Vt**2 * (1 - a)**2 * (Cl* math.cos(math.radians(Beta)) - Cd * math.cos(math.radians(Beta))) * r**2 * DRNodes) /
            math.cos(math.radians(Beta))**2
return torque
```

2. Highlight of the Code

MATLAB
RK4

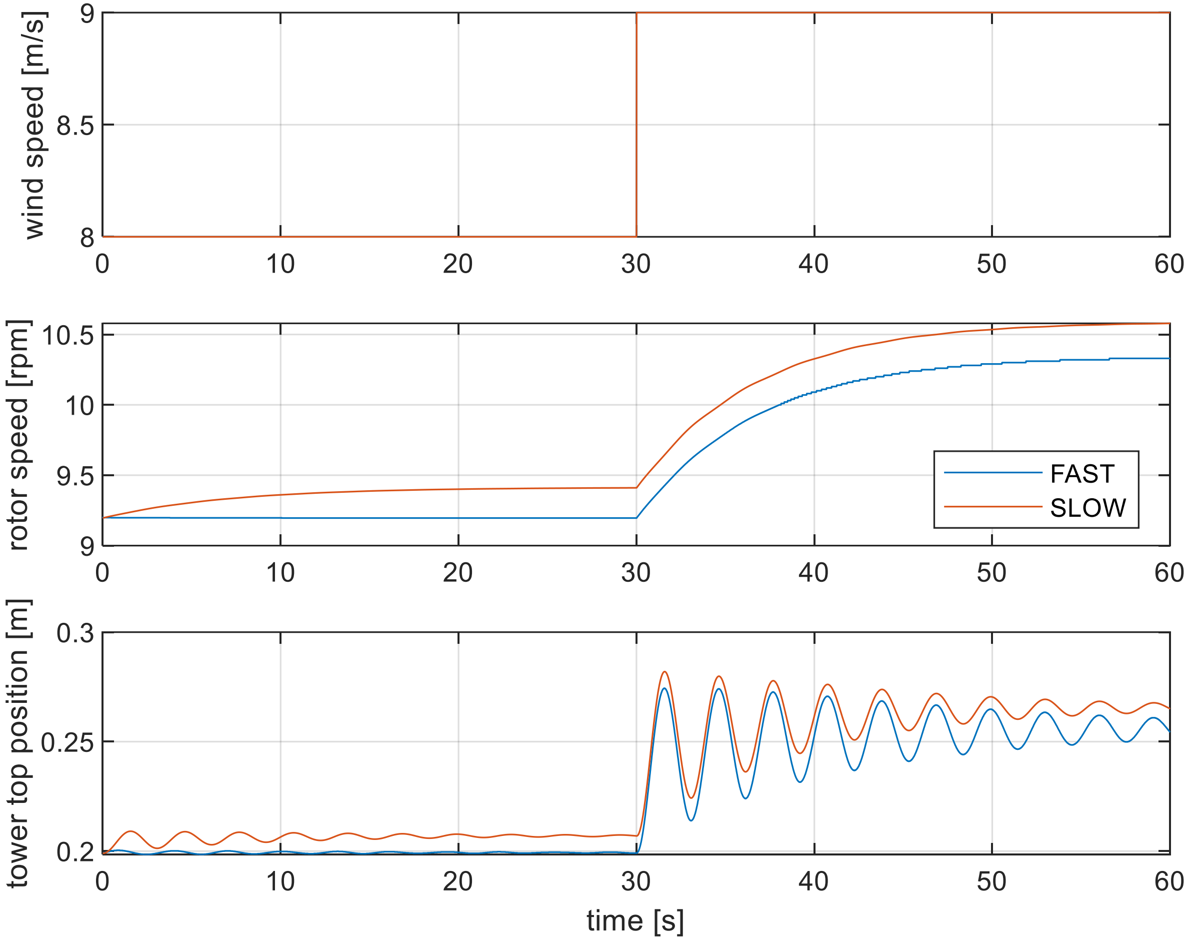
```
% simulation using Runge kutta 4
Tic
for k = 1:n-1
k1 = dt * SLOW(x(k,:), v_0(k), Parameter);
k2 = dt * SLOW(x(k,:) + 0.5 * k1, v_0(k) + 0.5 * dt, Parameter);
k3 = dt * SLOW(x(k,:) + 0.5 * k2, v_0(k) + 0.5 * dt, Parameter);
k4 = dt * SLOW(x(k,:) + k3, v_0(k) + dt, Parameter);x(k+1,:) = x(k,:) + (1/6) * (k1 + 2*k2 + 2*k3 + k4);

End
TimeRatio   = TMax/toc;
```

2. Highlight of the Code

Python
RK4

```
# Simulation using Runge Kutta 4
start_time = time.time()
for k in range(n - 1):
k1 = dt * self.slow(x[k, :], v_0[k], parameters)
k2 = dt * self.slow(x[k, :] + 0.5 * k1, v_0[k] + 0.5 * dt, parameters)
k3 = dt * self.slow(x[k, :] + 0.5 * k2, v_0[k] + 0.5 * dt, parameters)
k4 = dt * self.slow(x[k, :] + k3, v_0[k] + dt, parameters)
x[k + 1, :] = x[k, :] + (1 / 6) * (k1 + 2 * k2 + 2 * k3 + k4)
end_time = time.time()
TimeRatio = TMax / (end_time - start_time)
print(f"Time Ratio (Sim/CPU): = {TimeRatio:.4f} MN")
```

3. Results



Graphs &
Compare with FAST

Time Ratio SLOW (Sim/CPU): 0.998007

4. Problems and Solutions

1. Calculation of thrust force and torque for each blade section as we were taking fixed wind speed instead of Wind vector.
2. No supporting files for python as provided in MATLAB, so implementation of supporting files in main code were main issue.
3. Facing problems in generating graph in python.

5. Summary

We implemented the BEM code using both Python and MATLAB.

Python Implementation:
In our Python implementation, we followed the formula provided by G. Ingram [3]. However, due to time constraints and difficulties in converting all supporting files from MATLAB to Python, we were unable to create graphs. Although we started, but it is not yet complete.

MATLAB Implementation:
In MATLAB, we followed to the notes from Professor's Lecture 02 [7]. Using MATLAB, we were able to obtain results and compare them with OPENFAST.

6. Contribution

Topic	Contribution person name
Aerodynamic concepts	Karan Soni , Rahul Patil, Mostafa Mozafary
BEM code in python	Karan Soni, Rahul Patil
BEM code in MATLAB	Mostafa Mozafary, Rahul Patil
Presentation	Manthan Patel, Hassan Pour Saeid
Review and changes in Presentation	Karan Soni , Rahul Patil, Mostafa Mozafary

References

1. J. Jonkman, S. Butterfield, W. Musial, and G. Scott. Definition of a 5-MW Reference Wind Turbine for Offshore System Development. Tech. rep. TP-500-38060. NREL, 2009. DOI: 10. 2172/947422.
2. J. Jonkman and M. L. Buhl. FAST User's Guide. Tech. rep. EL-500-38230. NREL, 2005. URL: https://www.nrel.gov/docs/fy06osti/38230.pdf.
3. G. Ingram. Wind Turbine Blade Analysis using the Blade Element Momentum Method. 2011. URL: www.dur.ac.uk/g.l.ingram.
4. D. Schlipf. "Lidar-Assisted Control Concepts for Wind Turbines". PhD thesis. University of Stuttgart, 2015. DOI: 10.18419/opus-8796
5. Python Software Foundation. The Python Standard Library. From https://docs.python.org .
6. Prof. Ragunathan Rengasamy. " Python for Data science, IIT Madras " [Online Lecture]. https://nptel.ac.in.
7. Prof. Dr.-Ing. David Schlipf. "Lecture #1 Introduction to Wind Turbine Aerodynamics" [Lecture notes]. 25.03.2024.