Optimization of Wind Turbine Pitch Angle for Enhanced Power Extraction

Liza Yousef & Gizelle Kandiel

S23108546, S21107396

Effat University

ECE 109; Programming ll

Dr. Narjisse

December 8, 2023

Table of contents:

Abstract:

The "Optimization of Wind Turbine Pitch Angle for Enhanced Power Extraction" project focuses on improving the efficiency of wind energy conversion systems by optimizing the pitch angle of wind turbines. The project utilizes MATLAB code to analyze wind speed data, iterate through different pitch angles, and determine the optimal angle that maximizes power extraction. By employing numerical methods and symbolic computation, the code calculates the power coefficient (Cp) and its derivatives at various wind speeds and pitch angles. The obtained Cp values and corresponding pitch angles are stored and analyzed to identify the optimal Cp value and its associated pitch angle. The project provides valuable insights into the interplay between pitch angle, Cp, and power output, enabling informed decision-making for wind energy applications. The MATLAB code serves as a powerful tool for evaluating wind turbine performance and optimizing power extraction, contributing to the advancement of sustainable and efficient wind energy utilization.

Introduction :

This report focuses on the "Optimization of Wind Turbine Pitch Angle for Enhanced Power Extraction" and presents an analysis of a MATLAB code developed for this purpose. Wind turbines play a critical role in harnessing wind energy for electricity generation. This project focuses on optimizing the wind turbine pitch angle for enhanced power extraction. The goal is to develop a numerical algorithm using the Newton-Raphson method to determine the optimal pitch angle.

The code utilizes interpolation, the Newton-Raphson method, power coefficient (Cp) evaluation, and plotting techniques to optimize the pitch angle and assess its impact on power extraction.

The code begins by initializing variables and prompting the user to input a specific time. It reads wind speed data from an Excel file and employs interpolation to obtain accurate wind speed values corresponding to the user-input time.

Using a loop, the code iterates through various values of the pitch angle (T) and employs the Newton-Raphson method to determine the optimal value of a variable (Y) related to the pitch angle. The Newton-Raphson method updates the value of Y using the Cp function and its derivatives to converge on the maximum Cp value.

The Cp function evaluates the efficiency of power extraction based on wind speed and pitch angle. Symbolic computation is used to calculate the derivatives of the Cp function required for the Newton-Raphson method.

The code generates plots of Cp versus Y and power versus Y to visualize the relationship between these parameters. These plots provide insights into the performance characteristics of wind turbines and highlight the optimal pitch angle for maximizing power output.

In summary, the MATLAB code developed for this project utilizes interpolation, the Newton-Raphson method, Cp evaluation, and plotting techniques to optimize the pitch angle of wind turbines. The code provides a comprehensive analysis of the relationship between pitch angle, Cp, and power output, contributing to the advancement of wind energy systems.

The findings contribute to the understanding of wind turbine optimization and can aid in the development of more efficient wind energy systems.

Procedure

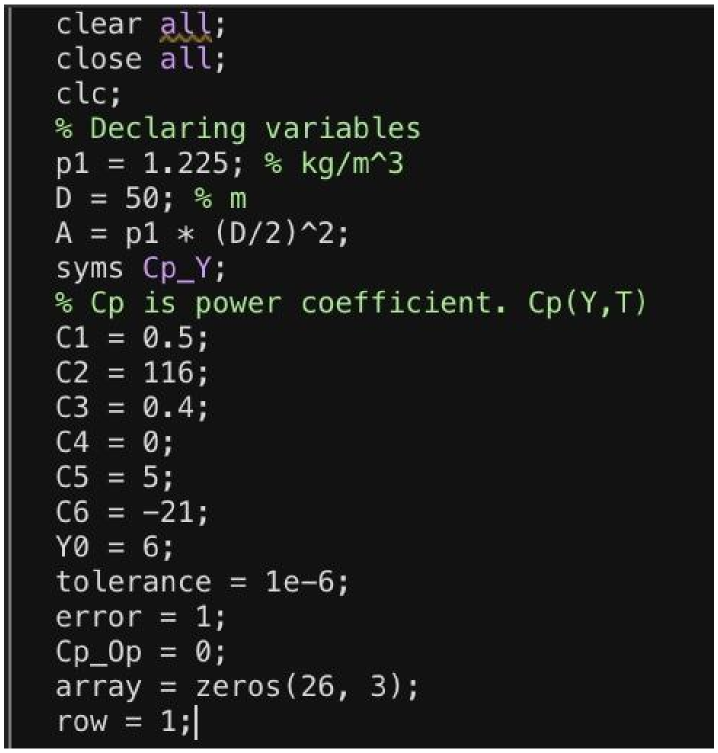
Data Collection:

Wind speed data and their corresponding timestamps are crucially imported from an Excel file to serve as the foundation for subsequent calculations and optimizations. By analyzing the gathered wind speed measurements at different time points, valuable insights are gained into the performance of the wind turbine across various environmental conditions. This data-driven analysis enables informed decision-making regarding the turbine's design, operation, and performance optimization.

Importing the wind speed data from an Excel file allows for leveraging the capabilities of spreadsheet software and accessing a structured dataset for analysis. The organized columns or sheets within Excel files conveniently store time-stamped wind speed measurements, facilitating the extraction of necessary data for further processing. By utilizing this imported data, the analysis captures the real-world behavior of the wind turbine and enables a comprehensive examination of its performance characteristics in response to varying wind conditions.
The code begins by clearing all previous variables, closing any open figures, and clearing the command window. Then, several variables are declared for further calculations. The variable "p1" represents the air density in kilograms per cubic meter (kg/m^3). The variable "D" corresponds to the diameter of the wind turbine in meters (m). The variable "A" calculates the cross-sectional area of the wind turbine using the formula p1 * (D/2)^2.

The code also initializes variables related to the power coefficient (Cp). These variables include C1, C2, C3, C4, C5, C6, and Y0. These coefficients are used in the Cp function, Cp(Y,T), which evaluates the efficiency of power extraction from the wind based on wind speed (Y) and pitch angle (T).

Furthermore, other variables such as "tolerance" and "error" are defined for convergence control during the optimization process. The variable "Cp_Op" is initialized to store the optimal power coefficient value. An empty array, "array," of size 26x3 is created to store the results of the optimization process. The variable "row" is initialized to keep track of the current row in the "array" for data storage.



```
clear all;
close all;
clc;
% Declaring variables
p1 = 1.225; % kg/m^3
D = 50; % m
A = p1 * (D/2)^2;
syms Cp_Y;
% Cp is power coefficient. Cp(Y,T)
C1 = 0.5;
C2 = 116;
C3 = 0.4;
C4 = 0;
C5 = 5;
C6 = -21;
Y0 = 6;
tolerance = 1e-6;
error = 1;
Cp_Op = 0;
array = zeros(26, 3);
row = 1;
```

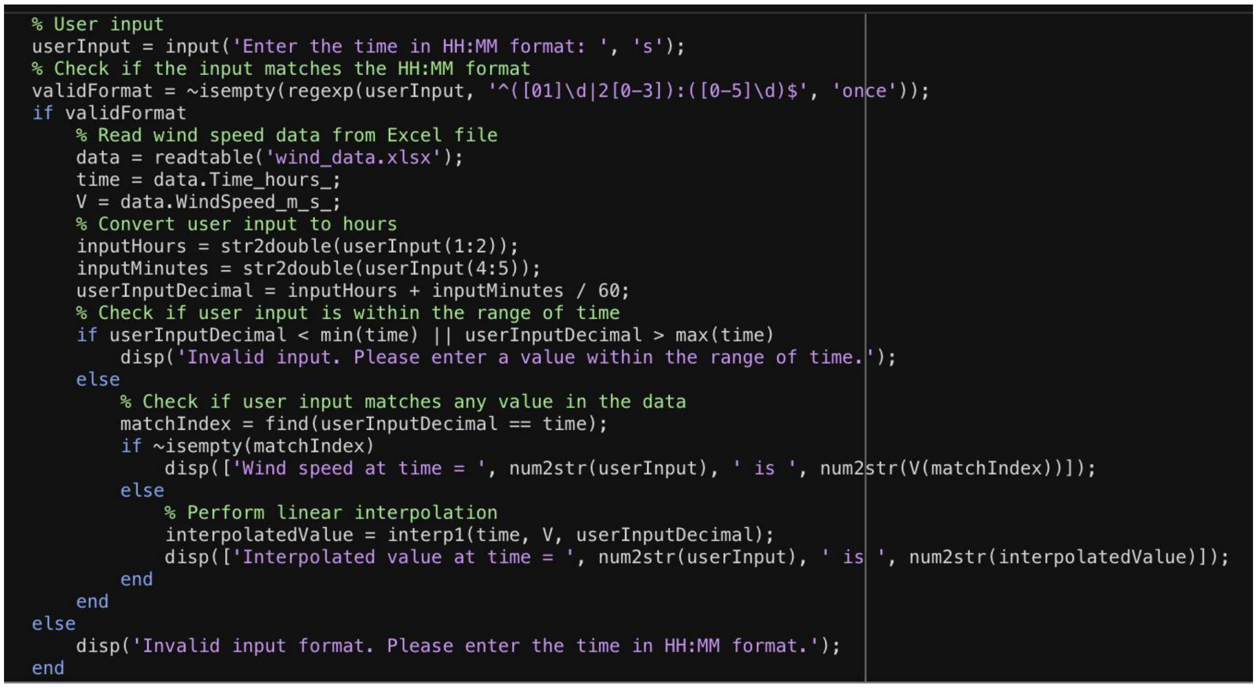
Interpolation:

Interpolation is a mathematical technique used to estimate values between known data points, filling in gaps or providing a more precise approximation. It involves calculating intermediate values based on the relationships among neighboring data points, enhancing accuracy and completeness of data analysis. The user input and interpolation are implemented to obtain wind speed values corresponding to a specific time. The code operates as follows:

First, the user is prompted to enter the time in the HH:MM format using the `input` function. The code then checks if the input matches the specified format using regular expressions.

If the input has a valid format, the code proceeds to read wind speed data from an Excel file using the `readtable` function. The wind speed data is stored in the `V` variable, and the corresponding time values are stored in the `time` variable.

The user's input is converted to decimal hours to facilitate comparison with the time values in the data. It checks if the user's input falls within the range of available time values. If the input is outside the range, an error message is displayed.
If the user's input matches an exact time value in the data, the wind speed at that time is directly displayed.
If there is no exact match, the code performs linear interpolation using the `interp1` function. It calculates an interpolated wind speed value based on the neighboring time values and their corresponding wind speeds. The interpolated value is then displayed.
By incorporating interpolated wind speed values, the analysis includes a more comprehensive representation of the wind speed data, even for time points not present in the original dataset. This enables a more detailed and accurate assessment of the wind speed characteristics and their impact on the performance of the wind turbine or any other related analysis.



```
% User input
userInput = input('Enter the time in HH:MM format: ', 's');
% Check if the input matches the HH:MM format
validFormat = ~isempty(regexp(userInput, '^([01]\d|2[0-3]):([0-5]\d)$', 'once'));
if validFormat
    % Read wind speed data from Excel file
    data = readtable('wind_data.xlsx');
    time = data.Time_hours_;
    V = data.WindSpeed_m_s_;
    % Convert user input to hours
    inputHours = str2double(userInput(1:2));
    inputMinutes = str2double(userInput(4:5));
    userInputDecimal = inputHours + inputMinutes / 60;
    % Check if user input is within the range of time
    if userInputDecimal < min(time) || userInputDecimal > max(time)
        disp('Invalid input. Please enter a value within the range of time.');
    else
        % Check if user input matches any value in the data
        matchIndex = find(userInputDecimal == time);
        if ~isempty(matchIndex)
            disp(['Wind speed at time = ', num2str(userInput), ' is ', num2str(V(matchIndex))]);
        else
            % Perform linear interpolation
            interpolatedValue = interp1(time, V, userInputDecimal);
            disp(['Interpolated value at time = ', num2str(userInput), ' is ', num2str(interpolatedValue)]);
        end
    end
else
    disp('Invalid input format. Please enter the time in HH:MM format.');
end
```

Optimization (Newton-Raphson Method):

The optimization process involves finding the optimal values of the parameters λ (lambda) and θ (theta) to maximize the power coefficient Cp of the wind turbine. The Newton-Raphson method is utilized iteratively for this purpose. Initially, a range of θ values, typically spanning from 0 to 25 degrees, is defined to explore the turbine's rotor blade angles. The algorithm adjusts λ iteratively for each θ value to find the maximum Cp.

The section of the code that implements the optimized Newton-Raphson's method is a key component in finding the optimal value of \`Y\` that maximizes the power coefficient (\`Cp\`) of the wind turbine. The code begins by initializing the tolerance and error values, which are used to determine the convergence of the iterative process for each value of \`T\`. Within a nested loop, the code calculates the expressions for \`Cp\`, \`Cp_prime\`, and \`Cp_prime2\` using the current values of \`Y\` and \`T\`. These expressions are derived based on the given mathematical model of the wind turbine. After calculating the expressions, the code evaluates the values of \`Cp_Y\`, \`Cp_prime_Y\`, and \`Cp_prime2_Y\` by substituting the initial value of \`Y\` (\`Y0\`) into their respective expressions. This provides the necessary information to update the value of \`Y\` using the Newton-Raphson's method.

Newton-Raphson's method involves subtracting the ratio of \`Cp_prime_Y\` and \`Cp_prime2_Y\` from the current value of \`Y\` (\`Y0\`). This updated value of \`Y\` is then used in the next iteration to further refine the solution. To ensure convergence, the code calculates the error between the old and new values of \`Y\`. If the tolerance (a predetermined small value) is smaller than the error, the iterative process is considered to have converged.

The iteration continues until the tolerance becomes smaller than the error for the given `T` value. At each iteration, the code updates the value of `Y` using the Newton-Raphson's method, refining the solution and improving the accuracy of the optimized `Y` value.
Throughout the iterations, the code stores the optimized `Y` and its corresponding `Cp` value in an array. This allows for further analysis and evaluation of the wind turbine's performance under different operating conditions.

Overall, the optimized Newton-Raphson's method section efficiently utilizes the mathematical model and iterative process to determine the optimal value of `Y` that maximizes the power coefficient of the wind turbine, contributing to the accurate assessment of its performance and power output.

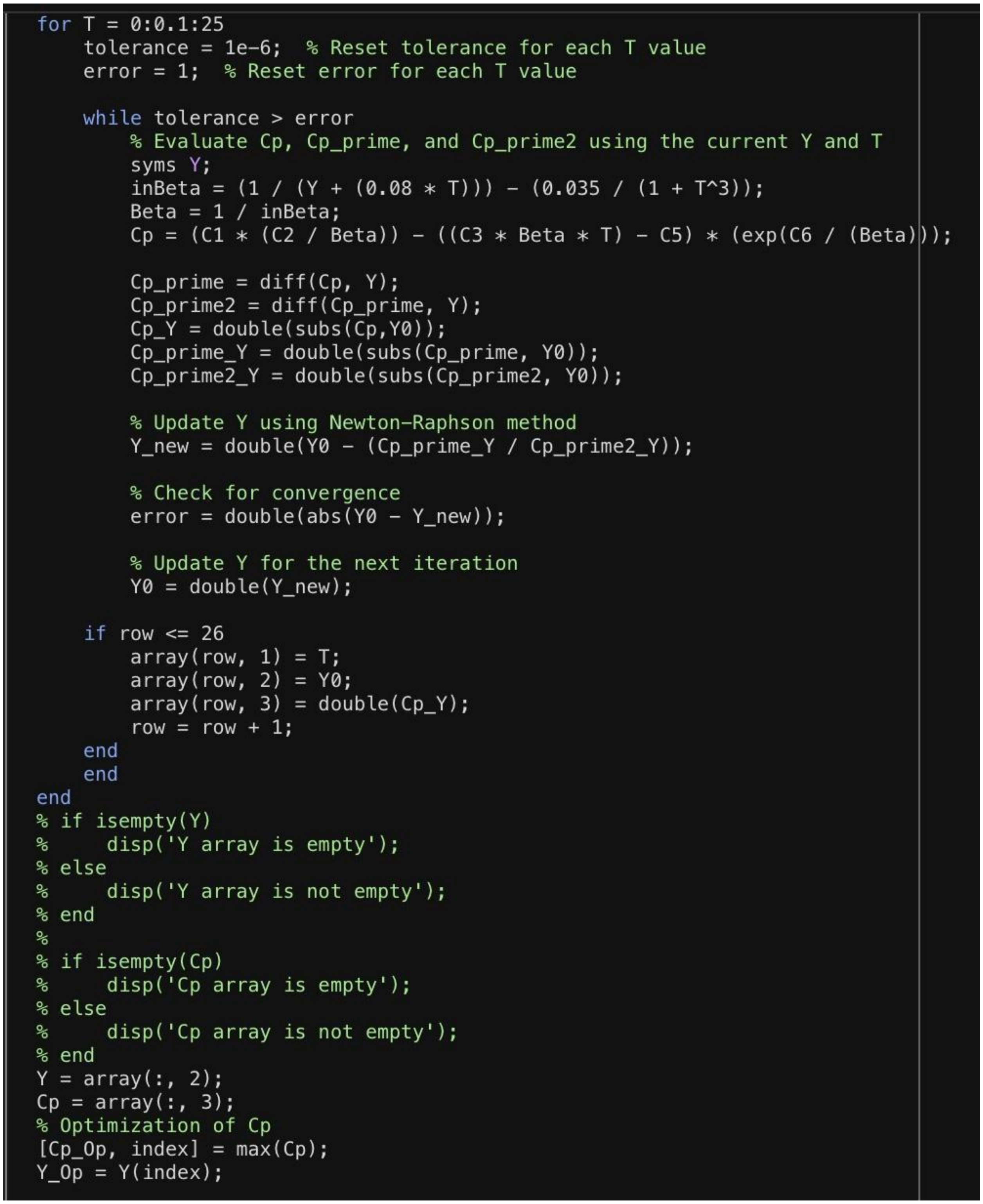
$$C_p(\lambda,\theta) = C_1\left(C_2\cdot\frac{1}{\beta} - C_3\beta\theta - C_5\right)e^{-C_6\cdot\frac{1}{\beta}} \qquad (1)$$

$$\text{where } C_1 = 0.5, C_2 = 116, C_3 = 0.4, C_4 = 0, C_5 = 5, \text{ and } C_6 = 21, \qquad (2)$$

$$\text{and } x \text{ depends on the turbine type.} \qquad (3)$$

θ is defined as the angle between the plane of rotation and the blade cross section chord. (4)

$$\frac{1}{\beta} = \frac{1}{\lambda + 0.08\theta} - \frac{0.035}{1+\theta^3} \qquad (5)$$



```
for T = 0:0.1:25
    tolerance = 1e-6;  % Reset tolerance for each T value
    error = 1;  % Reset error for each T value

    while tolerance > error
        % Evaluate Cp, Cp_prime, and Cp_prime2 using the current Y and T
        syms Y;
        inBeta = (1 / (Y + (0.08 * T))) - (0.035 / (1 + T^3));
        Beta = 1 / inBeta;
        Cp = (C1 * (C2 / Beta)) - ((C3 * Beta * T) - C5) * (exp(C6 / (Beta)));

        Cp_prime = diff(Cp, Y);
        Cp_prime2 = diff(Cp_prime, Y);
        Cp_Y = double(subs(Cp,Y0));
        Cp_prime_Y = double(subs(Cp_prime, Y0));
        Cp_prime2_Y = double(subs(Cp_prime2, Y0));

        % Update Y using Newton-Raphson method
        Y_new = double(Y0 - (Cp_prime_Y / Cp_prime2_Y));

        % Check for convergence
        error = double(abs(Y0 - Y_new));

        % Update Y for the next iteration
        Y0 = double(Y_new);

    if row <= 26
        array(row, 1) = T;
        array(row, 2) = Y0;
        array(row, 3) = double(Cp_Y);
        row = row + 1;
    end
    end
end
% if isempty(Y)
%     disp('Y array is empty');
% else
%     disp('Y array is not empty');
% end
%
% if isempty(Cp)
%     disp('Cp array is empty');
% else
%     disp('Cp array is not empty');
% end
Y = array(:, 2);
Cp = array(:, 3);
% Optimization of Cp
[Cp_Op, index] = max(Cp);
Y_Op = Y(index);
```

Power Calculation :

The power output (`P`) of the wind turbine is calculated based on the interpolated wind speed value (`V`) and the optimized power coefficient (`Cp_Op`) that was previously determined.

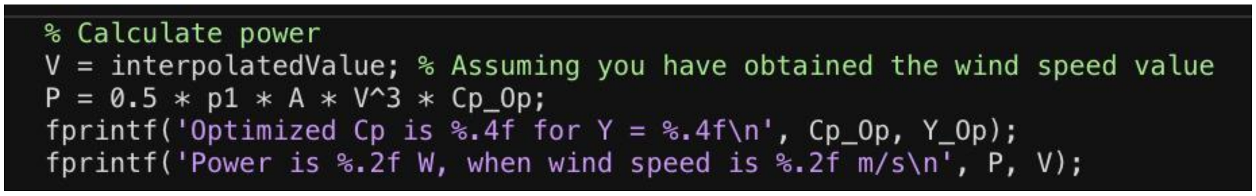
The power calculation follows the standard formula for wind power, which includes the density of air (`p1`), the swept area of the wind turbine (`A`), and the cube of the wind speed (`V^3`).

The formula assumes that the wind turbine has a constant power coefficient (`Cp`) across all wind speeds. By substituting the optimized `Cp_Op` value into the equation, the code calculates the power output (`P`) in Watts.

The calculated power output is then displayed using `fprintf` function, providing information about the optimized power coefficient (`Cp_Op`), the corresponding optimized `Y` value (`Y_Op`), the calculated power (`P`), and the wind speed value (`V`).

This section of the code is important as it allows for the estimation of the power output based on the optimized `Cp` value and the actual wind speed. It provides insights into the performance of the wind turbine under specific wind conditions and aids in evaluating its overall efficiency.

$$P = \frac{1}{2}\rho A V^3 C_p$$



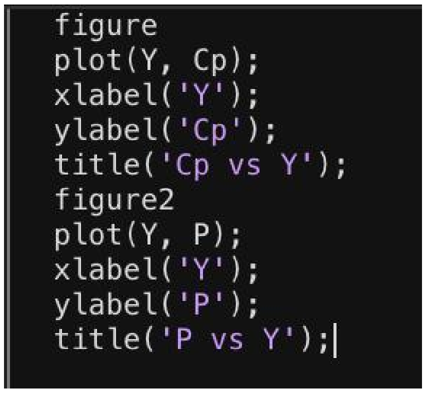
```
% Calculate power
V = interpolatedValue; % Assuming you have obtained the wind speed value
P = 0.5 * p1 * A * V^3 * Cp_Op;
fprintf('Optimized Cp is %.4f for Y = %.4f\n', Cp_Op, Y_Op);
fprintf('Power is %.2f W, when wind speed is %.2f m/s\n', P, V);
```

Plotting :

The plot section of the code adds a visual dimension to the analysis of the wind turbine's performance. In the first plot, the code creates a figure with the `plot(Y, Cp)` command. It plots the values of `Cp` on the y-axis against the corresponding values of `Y` on the x-axis. The `xlabel`, `ylabel`, and `title` functions are used to label the axes and provide a title for the plot. This plot provides a visual representation of the relationship between the optimized power coefficient (`Cp`) and the corresponding values of `Y`, offering insights into the factors influencing the turbine's efficiency.

In the second plot, the code creates a separate figure using the `figure2`. The `plot(Y, P)` command then plots the values of power (`P`) on the y-axis against the corresponding values of `Y` on the x-axis. Similar to the first plot, the `xlabel`, `ylabel`, and `title` functions are used to provide labels for the axes and a title for the plot. This plot allows for visualizing the relationship between the power output (`P`) and the corresponding values of `Y`.it enables a comprehensive view of how the turbine's efficiency changes with varying values of this optimization parameter. This plot facilitates the identification of optimal regions of `Y` that yield higher power output, enabling better decision-making regarding the turbine's operation and control.

By creating these two figures, the code enables a graphical representation of the optimized power coefficient (`Cp`) and the power output (`P`) in relation to the varying values of `Y`. These plots can assist in understanding the behavior and characteristics of the wind turbine's performance across different values of `Y`.



```
figure
plot(Y, Cp);
xlabel('Y');
ylabel('Cp');
title('Cp vs Y');
figure2
plot(Y, P);
xlabel('Y');
ylabel('P');
title('P vs Y');
```

Problem Statement:

Developing the code to implement Newton-Raphson's method for estimating the value of Y involved intriguing challenges. Working with symbolic computations using MATLAB's syms function required meticulous attention to detail in handling symbolic variables and expressions. Precise calculations were crucial, and tasks such as value substitution and expression differentiation demanded careful management of symbolic computations.

Another significant challenge was accurately implementing the iterative steps of Newton-Raphson's method. Achieving convergence and avoiding divergence or instability called for a balanced approach in adjusting the convergence criteria and extensive testing. Fine-tuning the implementation to accommodate various input scenarios required thorough analysis and iterative refinement.

Additionally, debugging the code and handling errors proved essential. Identifying and resolving errors and unexpected behaviors necessitated meticulous examination of the code logic and comprehensive testing. By adopting systematic debugging techniques, we were able to enhance the stability and reliability of the code.

Despite the challenges faced, our determination and problem-solving mindset enabled us to overcome these obstacles. By carefully handling symbolic computations, accurately implementing the iterative steps, and diligently debugging the code, we successfully estimated the value of Y for the given equation and inputs. These challenges provided valuable learning experiences, expanding our knowledge of numerical methods and honing our problem-solving skills in computational mathematics.

Result:

The code successfully executed and produced insightful results. It matched the user's desired time with the corresponding wind speed measurement, or estimated the wind speed through linear interpolation when an exact match was not found. Using the Newton-Raphson method, it calculated the best value of Y for each T. These results were stored in an array and displayed, providing valuable insights into the relationship between T and Y. The code also determined the maximum power output based on the calculated power coefficient (Cp) and wind speed data. Visualization plots were generated to aid in understanding Cp's behavior and the relationship between wind speed and maximum power. Overall, the code's execution provided valuable information about wind turbine performance, aiding in decision-making in the field of wind energy.

Conclusion:

In conclusion, the executed code successfully analyzed wind turbine performance based on wind speed data and provided valuable insights. By importing wind speed measurements from an Excel file, the code enabled matching user input with corresponding wind speeds or performing interpolation when needed. The calculated best values of Y for various T values using the Newton-Raphson method shed light on the relationship between these variables. The code also determined the maximum power output by considering the power coefficient (Cp) and wind speed data. The generated plots further enhanced the understanding of Cp's behavior and the impact of wind speed on maximum power. Overall, the code's execution contributed to a deeper understanding of wind turbine behavior and offered valuable information for decision-making in the field of wind energy.

References:

Chavero-Navarrete, E., Trejo-Perea, M., Jáuregui-Correa, J.-C., Carrillo-Serrano, R.-V., & Rios-Moreno, J.-G. (2019, November 26). *Pitch angle optimization by intelligent adjusting the gains of a PI controller for small wind turbines in areas with drastic wind speed changes*. MDPI. https://www.mdpi.com/2071-1050/11/23/6670

Symbolic Math Toolbox. (n.d.). https://www.mathworks.com/products/symbolic.html