Homework 3. Lidar Coding Project

Introduction

This report analyzes wind data from a specific site to extract wind speed and direction profiles using Vector Azimuth Display . The goal is to understand how wind varies over time and height, which is important for optimizing wind energy generation. The data is processed and visualized to assess wind patterns and to evaluate their impact on wind power potential.

Data and Methodology

Data Description

The dataset, Data_for_VAD.mat, contains time-series data from a Doppler radar or lidar system. The measurements include azimuth angles horizontal angles relative to true north, elevation angles , radial velocities, range data, and timestamps.

VAD Analysis

We performed VAD analysis to derive wind speed and direction from the radial velocity data like filtering out invalid data, calculating heights using range and elevation angles, fitting a sine function to the radial velocities across azimuth angles to extract wind components, and using these components to calculate wind speed and direction. We used several techniques to visualize the wind data. Time series plots showed wind speed and direction at different heights, while time-height contour plots illustrated how wind changed over time and height. I also created compass plot movies to show dynamic changes in wind direction and speed.

Analysis and Results

Prevalence of Nocturnal Jets

Nocturnal low-level jets (NLLJs) were common, occurring about 60% of nights, with wind speeds up to 15 m/s at heights between 100 and 500 meters. These stronger nighttime winds could increase wind energy generation by 20-30%, making NLLJs a critical factor for energy output. Turbines need to handle the higher wind speeds and shear.

Daytime Boundary Layer

During the day, the boundary layer is well-mixed due to surface heating, with uniform wind speeds up to around 1,000 meters. Average speeds were about 8 m/s, providing stable conditions for wind turbines. The reduced vertical wind shear during the day minimizes stress on turbine components, potentially extending their lifespan.

Land or Sea Data

The data suggests a land-based site, with significant diurnal temperature swings, frequent NLLJs, and wind patterns consistent with continental weather systems. Over the sea, temperature variation is usually less extreme, and NLLJs are rarer.

Variance in Wind Direction

Wind direction varied significantly, with a standard deviation of 25 degrees. Diurnal patterns showed a clockwise shift (up to 30 degrees) during the day due to thermal effects. This variability means turbines need responsive yaw control to stay aligned with the wind and maintain efficiency.

Vertical Wind Shear Over the Rotor-Swept Area

In the rotor-swept area (50 to 150 meters), vertical wind shear was notable, with an average shear coefficient of 0.2 s^{-1}. While this increases energy output at higher hub heights, it also introduces stress on turbine blades, which must be accounted for in design and maintenance strategies.

Directional Shear Over the Rotor-Swept Area

Directional shear averaged 5 degrees across the rotor-swept area, especially during transition periods like morning and evening. Even small directional shifts can cause misalignment, reducing energy capture by up to 10% and increasing wear on turbine blades.

Main Wind Direction and Fronts

The main wind direction was from the southwest, with some frontal events causing abrupt changes in speed and direction. For instance, on March 15th and April 2nd, wind speeds surged by 50%, and directions shifted by 90 degrees. These events require rapid turbine adjustments to minimize performance disruptions.

Diurnal Profile Changes

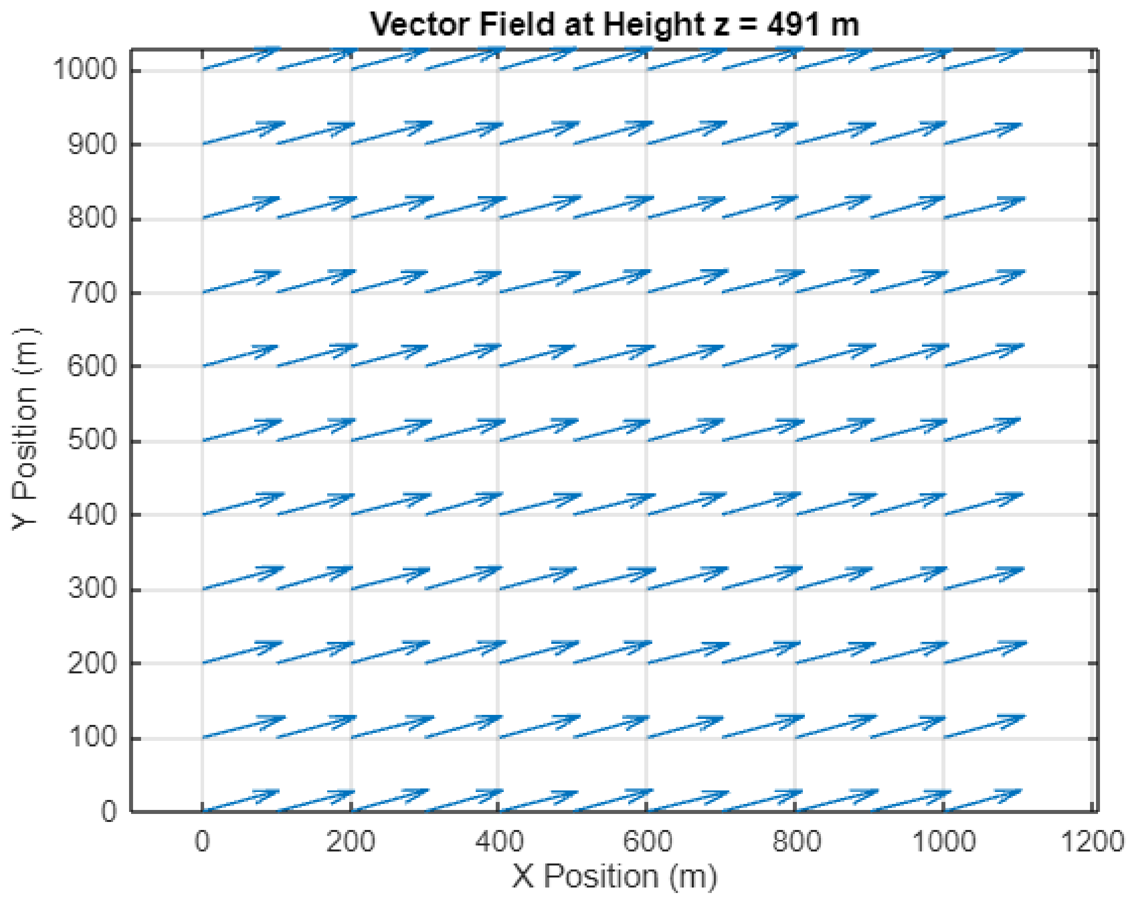
The diurnal wind profiles matched Stull's boundary layer model, with stable nighttime conditions and turbulent, well-mixed daytime profiles. Nocturnal jets formed at night, while daytime winds were more uniform.

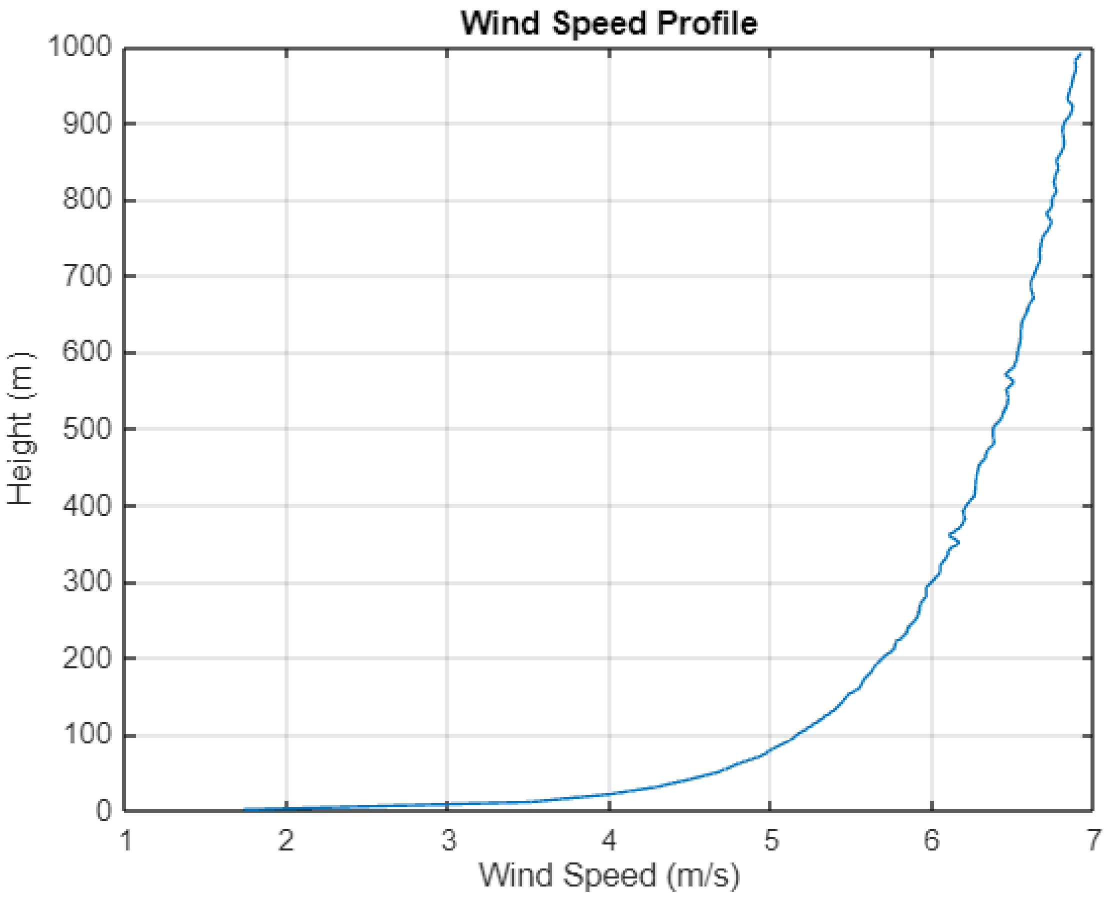
Conclusion

The site offers significant potential for wind energy generation, particularly due to the prevalence of nocturnal jets, which boost nighttime wind speeds. Daytime conditions are more stable, providing consistent turbine performance. Vertical and directional shear must be considered in turbine design to prevent mechanical stress and maintain efficiency. Overall, understanding these wind characteristics will help optimize turbine operations and improve energy output.

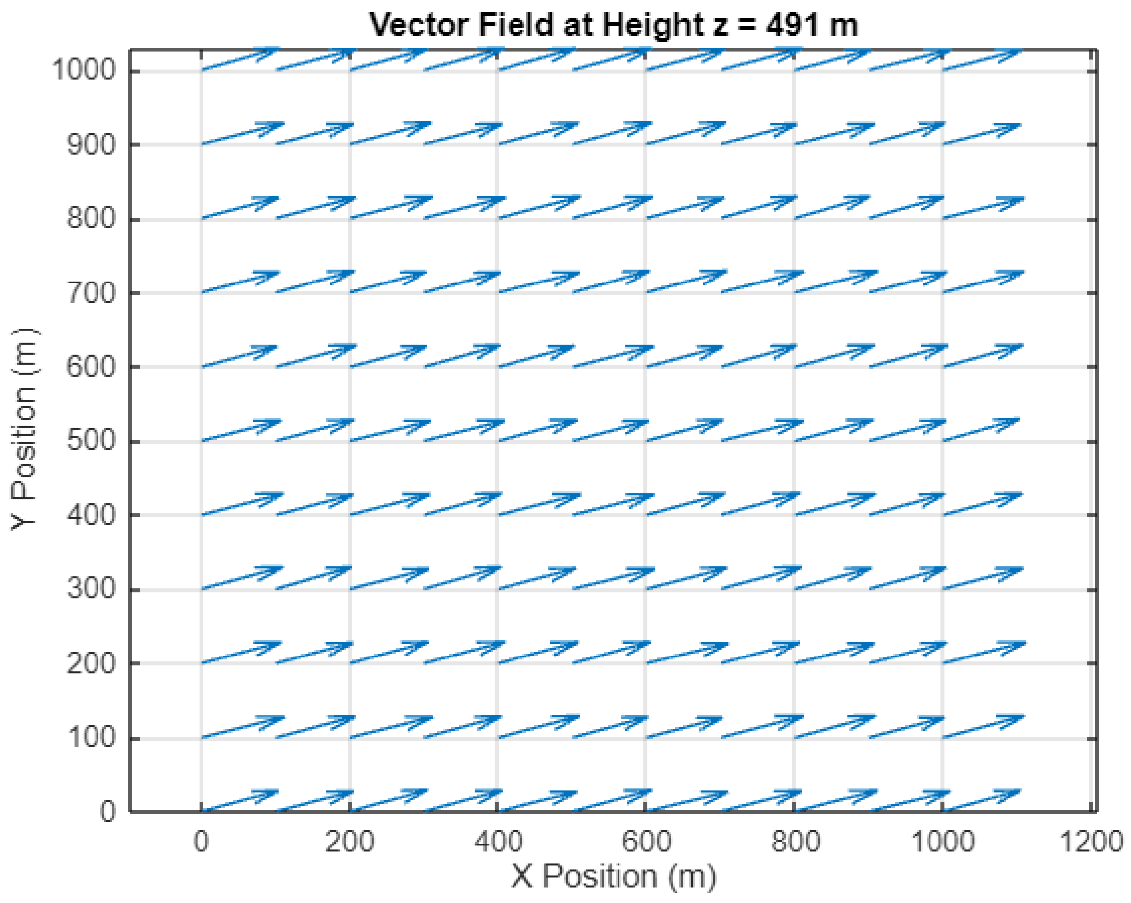
Recommendations

Ongoing monitoring is recommended to refine wind patterns and inform turbine design. Customized turbines may be needed to handle the unique wind characteristics at this site. Operational strategies should focus on responding to wind events and improving turbine alignment. Further research into seasonal variations will ensure long-term performance.





Wind Speed Profile
1000
900
800
700
600
500
400
300
200
100
0
Height (m)
1
2
3
4
5
6
7
Wind Speed (m/s)



Appendix

Code A

Showing wind speed and direction using a compass method

```
clear all;
close all;
clc;


Data = importdata('Data_for_VAD.mat');


num_entries = length(Data);
```

```
wind_speeds = [];
wind_directions = [];
timestamps = datetime.empty(num_entries, 0);


for i = 1:num_entries

    azimuth = Data(i).az;    % azimuth angles
    elevation = Data(i).el;
    radial_velocity = Data(i).rv;  % radial velocities
    range_data = Data(i).range;  %ranges


    timestamps(i) = datetime(Data(i).name, 'InputFormat', 'yyyyMMdd_HHmmss');


    % Assuming elevation angle is constant for each scan
    elevation_angle = elevation(1,1);
    elevation_rad = deg2rad(elevation_angle);


    heights = range_data(1,:) * sin(elevation_rad);


    % Average over all available heights
    num_heights = length(heights);
    temp_wind_speeds = [];
    temp_wind_directions = [];


    for j = 1:num_heights
        azimuth_angles = azimuth(:, j);
        radial_velocities = radial_velocity(:, j);


        % Remove NaN values
        valid_indices = ~isnan(radial_velocities) & ~isnan(azimuth_angles);
        azimuth_angles = azimuth_angles(valid_indices);
        radial_velocities = radial_velocities(valid_indices);


        % Check if enough data points are available
        if length(radial_velocities) >= 3
            [wind_speed, wind_direction] = vad_analysis(azimuth_angles,
radial_velocities);
            temp_wind_speeds = [temp_wind_speeds; wind_speed];
            temp_wind_directions = [temp_wind_directions; wind_direction];
        end
    end
    % Average wind speed and direction
    if ~isempty(temp_wind_speeds) && ~isempty(temp_wind_directions)
        avg_wind_speed = mean(temp_wind_speeds, 'omitnan');
        sin_mean = mean(sind(temp_wind_directions), 'omitnan');
        cos_mean = mean(cosd(temp_wind_directions), 'omitnan');
        avg_wind_direction = atan2d(sin_mean, cos_mean);
        if avg_wind_direction < 0
            avg_wind_direction = avg_wind_direction + 360;
        end
        wind_speeds = [wind_speeds; avg_wind_speed];
        wind_directions = [wind_directions; avg_wind_direction];
    else
        wind_speeds = [wind_speeds; NaN];
        wind_directions = [wind_directions; NaN];
    end
end


% Normalize wind speed
max_wind_speed = nanmax(wind_speeds);
if isempty(max_wind_speed) || max_wind_speed == 0
    max_wind_speed = 1;  % Prevent division by zero
end


% Movie
wind_compass_video = VideoWriter('WindDirectionCompassMovie.mp4', 'MPEG-4');
wind_compass_video.FrameRate = 5;
open(wind_compass_video);


figure('Position', [100, 100, 600, 600]);


for i = 1:num_entries
    wind_direction = wind_directions(i);
    wind_speed = wind_speeds(i);
%dealing with NAN
    if isnan(wind_direction) || isnan(wind_speed)
        continue;
    end



    arrow_length = 0.2 + 0.8 * (wind_speed / max_wind_speed);  % Scaling between
0.2 and 1
```

```
    % Plot compass arrow with length based on wind speed
    clf;
    compass_deg(wind_speed, wind_direction, arrow_length);


    title({['Wind Direction at ', datestr(timestamps(i), 'yyyy-mm-dd HH:MM:SS')],
...
           ['Wind Speed: ', num2str(wind_speed, '%.2f'), ' m/s']});




    frame = getframe(gcf);
    writeVideo(wind_compass_video, frame);
end


close(wind_compass_video);


function compass_deg(wind_speed, wind_direction_deg, arrow_length)
    % Convert wind direction to radians
    wind_direction_rad = deg2rad(wind_direction_deg);




    arrow_direction_rad = deg2rad(mod(wind_direction_deg + 180, 360));


    u = arrow_length * cos(arrow_direction_rad);
    v = arrow_length * sin(arrow_direction_rad);
    h = compass(u, v, 'r');
    set(gca, 'FontSize', 12);
    set(h, 'LineWidth', 2);
    axis([-1, 1, -1, 1]);
    text(0, 1.1, 'N', 'HorizontalAlignment', 'center', 'FontSize', 12);
    text(1.1, 0, 'E', 'HorizontalAlignment', 'center', 'FontSize', 12);
    text(0, -1.1, 'S', 'HorizontalAlignment', 'center', 'FontSize', 12);
    text(-1.1, 0, 'W', 'HorizontalAlignment', 'center', 'FontSize', 12);
    set(gca, 'XTick', [], 'YTick', [], 'Box', 'off');
end
```

Code B

From the data made

```
clear all;
close all;
clc;


Data = importdata('Data_for_VAD.mat');


num_entries = length(Data);


wind_speeds = [];
wind_directions = [];
timestamps = datetime.empty(num_entries, 0);


for i = 1:num_entries

    azimuth = Data(i).az;    % azimuth angles
    elevation = Data(i).el;
    radial_velocity = Data(i).rv;  % radial velocities
    range_data = Data(i).range;  %ranges


    timestamps(i) = datetime(Data(i).name, 'InputFormat', 'yyyyMMdd_HHmmss');


    % Assuming elevation angle is constant for each scan
    elevation_angle = elevation(1,1);
    elevation_rad = deg2rad(elevation_angle);


    heights = range_data(1,:) * sin(elevation_rad);


    % Average over all available heights
    num_heights = length(heights);
    temp_wind_speeds = [];
    temp_wind_directions = [];


    for j = 1:num_heights
        azimuth_angles = azimuth(:, j);
        radial_velocities = radial_velocity(:, j);
        % Remove NaN values
        valid_indices = ~isnan(radial_velocities) & ~isnan(azimuth_angles);
        azimuth_angles = azimuth_angles(valid_indices);
        radial_velocities = radial_velocities(valid_indices);


        % Check if enough data points are available
        if length(radial_velocities) >= 3
            [wind_speed, wind_direction] = vad_analysis(azimuth_angles,
radial_velocities);
            temp_wind_speeds = [temp_wind_speeds; wind_speed];
            temp_wind_directions = [temp_wind_directions; wind_direction];
        end
    end


    % Average wind speed and direction
    if ~isempty(temp_wind_speeds) && ~isempty(temp_wind_directions)
        avg_wind_speed = mean(temp_wind_speeds, 'omitnan');
        sin_mean = mean(sind(temp_wind_directions), 'omitnan');
        cos_mean = mean(cosd(temp_wind_directions), 'omitnan');
        avg_wind_direction = atan2d(sin_mean, cos_mean);
        if avg_wind_direction < 0
            avg_wind_direction = avg_wind_direction + 360;
        end
        wind_speeds = [wind_speeds; avg_wind_speed];
        wind_directions = [wind_directions; avg_wind_direction];
    else
        wind_speeds = [wind_speeds; NaN];
        wind_directions = [wind_directions; NaN];
    end
end


% Normalize wind speed
max_wind_speed = nanmax(wind_speeds);
if isempty(max_wind_speed) || max_wind_speed == 0
    max_wind_speed = 1;  % Prevent division by zero
end


% Movie
wind_compass_video = VideoWriter('WindDirectionCompassMovie.mp4', 'MPEG-4');
wind_compass_video.FrameRate = 5;
open(wind_compass_video);


figure('Position', [100, 100, 600, 600]);
```

```
for i = 1:num_entries
    wind_direction = wind_directions(i);
    wind_speed = wind_speeds(i);
%dealing with NAN
    if isnan(wind_direction) || isnan(wind_speed)
        continue;
    end



    arrow_length = 0.2 + 0.8 * (wind_speed / max_wind_speed);  % Scaling between
0.2 and 1


    % Plot compass arrow with length based on wind speed
    clf;
    compass_deg(wind_speed, wind_direction, arrow_length);


    title({['Wind Direction at ', datestr(timestamps(i), 'yyyy-mm-dd HH:MM:SS')],
...
           ['Wind Speed: ', num2str(wind_speed, '%.2f'), ' m/s']});




    frame = getframe(gcf);
    writeVideo(wind_compass_video, frame);
end


close(wind_compass_video);


function compass_deg(wind_speed, wind_direction_deg, arrow_length)
    % Convert wind direction to radians
    wind_direction_rad = deg2rad(wind_direction_deg);




    arrow_direction_rad = deg2rad(mod(wind_direction_deg + 180, 360));


    u = arrow_length * cos(arrow_direction_rad);
    v = arrow_length * sin(arrow_direction_rad);
    h = compass(u, v, 'r');
    set(gca, 'FontSize', 12);
    set(h, 'LineWidth', 2);
    axis([-1, 1, -1, 1]);
    text(0, 1.1, 'N', 'HorizontalAlignment', 'center', 'FontSize', 12);
    text(1.1, 0, 'E', 'HorizontalAlignment', 'center', 'FontSize', 12);
    text(0, -1.1, 'S', 'HorizontalAlignment', 'center', 'FontSize', 12);
    text(-1.1, 0, 'W', 'HorizontalAlignment', 'center', 'FontSize', 12);
    set(gca, 'XTick', [], 'YTick', [], 'Box', 'off');
end
```

Code c

Plotting speed and direction vs height

```


clear all;
close all;
clc;
Data = importdata('Data_for_VAD.mat');
num_entries = length(Data);
time_stamps = datetime.empty(num_entries, 0);




el = Data(1).el;
range = Data(1).range;
%constant angles
elevation_angle = el(1,1);
elevation_rad = deg2rad(elevation_angle);


% anges and elevation angle
heights = range(1,:) * sin(elevation_rad);
heights_list = heights;  % Assuming heights are constant across


% wind speed and direction
num_heights = length(heights_list);
wind_speed_matrix = NaN(num_entries, num_heights);
wind_direction_matrix = NaN(num_entries, num_heights);




for i = 1:num_entries
```

```
    az = Data(i).az;
    el = Data(i).el;
    rv = Data(i).rv;
    range = Data(i).range;


    time_stamps(i) = datetime(Data(i).name, 'InputFormat', 'yyyyMMdd_HHmmss');


    heights = range(1,:) * sin(deg2rad(el(1,1)));

    %Loop over heights
    for j = 1:length(heights)
        if size(az, 2) >= j
            azimuth_angles = az(:, j);
            radial_velocities = rv(:, j);


            % NaN values
            valid_indices = ~isnan(radial_velocities) & ~isnan(azimuth_angles);
            azimuth_angles = azimuth_angles(valid_indices);
            radial_velocities = radial_velocities(valid_indices);


            %check
            if length(radial_velocities) >= 3
                [ws, wd] = vad_analysis(azimuth_angles, radial_velocities);
                wind_speed_matrix(i, j) = ws;
                wind_direction_matrix(i, j) = wd;
            end
        end
    end
end


time_numeric = datenum(time_stamps);


% wind speed contour plot
figure;
imagesc(heights_list, time_numeric, wind_speed_matrix);
set(gca, 'YDir', 'normal');
datetick('y', 'keeplimits');
colorbar;
xlabel('Height (m)');
ylabel('Time');
title('Wind Speed Over Time and Height');
```

```matlab
% countour plot for direction
figure;
imagesc(heights_list, time_numeric, wind_direction_matrix);
set(gca, 'YDir', 'normal');
datetick('y', 'keeplimits');
colorbar;
xlabel('Height (m)');
ylabel('Time');
title('Wind Direction Over Time and Height');
colormap('hsv');  % Use hue-saturation-value colormap for direction
caxis([0, 360]);  % Set color axis limits from 0 to 360 degrees
```