Wind Data Analysis and Transfer Function Modeling using Davenport Spectrum

Student Project by

Priyanka Uddhav Ghodke

Electronics and Telecommunication Engineering
Government College Of Engineering and Research, Avasari Khurd, Pune

Under the Guidance of
Mr. S.K. Bagde &
Thiyagrajan B.

GIANT METREWAVE RADIO TELESCOPE
NATIONAL CENTRE FOR RADIO ASTROPHYSICS
TATA INSTITUTE OF FUNDAMENTAL RESEARCH

Khodad, Tal- Junnar, Dist- Pune, India

April 2025 - June 2025

BONAFIDE CERTIFICATE

This is to certify that this project titled " **Wind Data Analysis and Transfer Function Modeling using Davenport Spectrum** " submitted to **Giant Metrewave Radio Telescope , Khodad**, is a bonafide record of work done by **Priyanka Uddhav Ghodke**, under my supervision at the **Giant Metrewave Radio Telescope , NCRA - TIFR** from "21st April 2025 to 20th June 2025 ".

Mr. S.K.Bagde
Servo System - Group Head,
GMRT, NCRA-TIFR

Mr. Thiyagrajan B.
Engineer, Servo System,
GMRT, NCRA-TIFR

Place : GMRT , Khodad
Date : 28/08/2025

DECLARATION BY AUTHOR

I hereby declare that the work presented in this report titled "**Wind Data Analysis and Transfer Function Modeling using Davenport Spectrum**" is my own and I further declare that: This report has not been submitted previously, either in part or full, for the award of any degree, diploma, or other qualification.The work is original and does not contain any plagiarised content.All sources of information have been appropriately cited and acknowledged.

Priyanka Uddhav Ghodke

IV - BE E&TC

Govt. College Of Engg. & Research

Place : GMRT , Khodad
Date : 28/08/2025

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who supported and guided me throughout the course of this project.

First and foremost , I would like to express my sincere gratitude to **Prof. Yashwant Gupta**, Director of GMRT, for providing me the opportunity to undertake this project at such a prestigious facility. I am equally thankful to **Prof. Ishwara Chandra**, Dean of GMRT. Their leadership and commitment to scientific research have been a constant source of inspiration, and I am truly honoured to have been a part of the vibrant research environment at GMRT.

I would like to thank, **Mr. S.K.Bagde**, Servo Systems - Group Head, Giant Metrewave Radio Telescope (GMRT) , for his valuable guidance, consistent support, and insightful suggestions as my project guide. His technical expertise and encouragement were crucial to the successful completion of this work, and I personally thank him for providing me the opportunity to work in a professional research environment .

I am also deeply grateful to **Mr. Thiyagrajan B.**, Engineer, Servo System, GMRT, NCRA-TIFR, for his constant motivation, academic mentorship, and for serving as my co-guide. His feedback helped me improve the quality and depth of this project.

I would also like to extend my heartfelt thanks to **Mr. Amit Kumar**, Engineer, Servo Systems, GMRT, for his encouragement throughout the duration of my project.

Lastly, I appreciate the entire team at Servo lab, GMRT for creating a supportive and intellectually stimulating atmosphere during my time here.

~Priyanka Uddhav Ghodke

ABSTRACT

Wind turbulence is a critical factor affecting the performance of large structures such as antennas, telescopes, and wind-sensitive engineering systems. This project focuses on the statistical analysis and modeling of measured wind velocity data using both experimental and theoretical approaches.

The raw wind data was recorded at a **sampling frequency of 10 Hz** and processed through multiple stages including **despiking, removal of outliers beyond ±3σ, and filtering** to obtain a clean dataset. Statistical parameters such as **mean wind speed (≈3.15 m/s)**, standard deviation, and covariance were computed to characterize the turbulence.

To study the frequency-domain characteristics, **Power Spectral Density (PSD) was estimated using both FFT and Welch methods**. The theoretical Davenport spectrum was generated using site-specific parameters (antenna height: 23.16 m, roughness length: 0.03 m) and compared with experimental PSDs. Results show a strong match in the low-frequency region, consistent with the turbulence model.

Furthermore, a transfer function was derived from the Davenport spectrum to represent the wind turbulence as a dynamic system. The **continuous-time transfer function was discretized using the Tustin method**, and its behavior was compared with a given discrete transfer function.

The key finding is that the simulated **transfer function derived from the Davenport spectrum** shows similar behavior to the given transfer function. This validates the use of transfer function modeling for simulating realistic wind turbulence in structural and control applications.

List of Contents

List of Figures

Chapter 1 : Introduction

1.1 Brief on GMRT

The **Giant Metrewave Radio Telescope (GMRT), built by the National Centre for Radio Astrophysics (NCRA-TIFR)**, is located about 80 km north of Pune, India. It is one of the world's largest and most sensitive radio telescope arrays operating at metre wavelengths.

GMRT consists of 30 fully steerable parabolic dishes of 45 m diameter each. Fourteen antennas form a central compact array (~1 km^2), while sixteen are spread along a Y-shaped configuration with baselines up to 25 km. Signals from all 435 antenna pairs are correlated to synthesize images with resolutions equivalent to a single 25-km dish.

The array operates in six frequency bands (50, 153, 233, 325, 610, 1420 MHz) with dual polarization. Angular resolution ranges from ~60 arcsec at low frequency to ~2 arcsec at 1.4 GHz.

A key innovation is the SMART (Stretch Mesh Attached to Rope Trusses) design, which makes the dishes lightweight and low-cost. The stainless-steel mesh reflector reduces wind load, so the 45-m dish experiences wind forces similar to a conventional 22-m dish. Each antenna, weighing about 80 tonnes, is mounted on a 15-m high concrete tower with an alt-azimuth mount.

This indigenous design has enabled India to build a world-class facility at modest cost. Since wind forces strongly affect pointing accuracy, wind data analysis and turbulence modeling are essential for GMRT's stable operation.

Figure 1.1.1 - GMRT's antennas, Khodad (Source: GMRT website) [1]

1.2 Brief on Disturbances Caused by Wind

Wind exerts fluctuating forces on large exposed structures such as antennas, towers, and telescope dishes. These forces arise due to turbulence, i.e., rapid variations in wind speed and direction. The main disturbances caused by wind include:

- Pointing errors – For steerable antennas like those at GMRT, wind gusts can deflect the dish surface or its support structure, reducing tracking accuracy.
- Structural vibrations – Turbulent forces excite natural modes of the structure, leading to oscillations that degrade performance.
- Torque and stress loading – Continuous wind pressure creates additional mechanical loads on bearings, joints, and supports, increasing wear and reducing lifetime.
- Dynamic instability – In severe cases, resonance between wind gust frequencies and structural modes can amplify motion, risking control failure or structural damage.

These disturbances are most critical at low frequencies, where wind energy is dominant. Since antennas must maintain arc-second level precision, even small deflections due to wind can significantly affect scientific observations.

Studying wind disturbances through measured data and turbulence models is therefore essential to design stable structures and implement robust control strategies.

1.3 Wind Data Analysis

To understand and mitigate wind-induced disturbances, it is essential to analyze measured wind velocity data. Wind data analysis provides both a statistical description of turbulence and a frequency-domain model of how energy is distributed across different scales of motion.
The process generally involves:

- Statistical Analysis – Calculation of mean velocity, standard deviation, and covariance to describe the average wind speed and its variability. Outlier removal and despiking are performed to clean the raw data.
- Spectral Analysis – Estimation of the Power Spectral Density (PSD) using methods such as FFT and Welch averaging, which reveal how turbulence energy is spread across frequencies.
- Comparison with Theoretical Models – The measured spectra are compared with turbulence models such as the Davenport spectrum, which relates wind characteristics to terrain roughness and height.

By validating measured wind data against theoretical spectra, researchers can derive transfer function models that simulate wind turbulence by filtering white noise. These models are vital in control system design, particularly for large antennas and telescopes where precise stability is required under wind disturbances.

1.3.1 Importance of Wind Data Analysis

Wind data analysis is crucial in the study and design of large structures that interact with the atmosphere, such as radio telescopes, antennas, towers, and bridges. The importance arises from the following aspects:

- Performance of Large Antennas – For instruments like GMRT, even small wind-induced deflections can cause significant pointing errors, reducing observational accuracy.
- Structural Safety – Continuous and turbulent wind loads create stresses and vibrations in mechanical components, which must be understood to ensure safe design and long-term durability.
- Control System Design – Wind turbulence can be represented by transfer function models, which allow engineers to design robust controllers (e.g., LQR, LQG) to compensate for disturbances.
- Validation of Theoretical Models – Comparing real data with models such as the Davenport spectrum helps in verifying whether classical turbulence models are suitable for a given site or application.
- Simulation of Realistic Conditions – By analyzing wind spectra, researchers can simulate realistic disturbance inputs for performance testing of structures and control algorithms.

In summary, wind data analysis not only improves our understanding of natural wind turbulence but also provides practical tools for predicting, modeling, and mitigating its effects on critical engineering systems.

1.4 Davenport Spectrum as a Turbulence Model

Atmospheric wind is characterized by random fluctuations called turbulence, which contain significant energy in the low-frequency range. To describe how this energy is distributed across frequencies, various spectral models have been developed. Among them, the Davenport spectrum is one of the most widely accepted and applied in civil and structural engineering.

The Davenport spectrum expresses the Power Spectral Density (PSD) of wind speed fluctuations as a function of frequency, reference wind speed, height above the ground, and terrain roughness. The general form of the Davenport spectrum is:

$$S_v(n) = \frac{4\kappa U^2}{n\left(1+(6n/n_r)^{5/3}\right)^2}$$

where:

- $S_v(n)$ = spectral density of longitudinal wind fluctuations,
- n = frequency (Hz),
- U = mean wind speed at height z,
- n_r = reference frequency depending on height and roughness,
- κ = scaling coefficient.

This spectrum accounts for two important aspects:

1. Height dependence – turbulence intensity decreases with altitude.

2. Terrain roughness – wind fluctuations are stronger near rough terrain (e.g., urban areas) compared to smooth terrain (e.g., open plains).

The Davenport spectrum decays with frequency approximately as $n^{-5/3}$, consistent with Kolmogorov's theory of turbulence in the inertial subrange. This makes it a physically meaningful model for representing the distribution of turbulence energy.

In engineering applications, the Davenport spectrum serves as a reference model against which measured PSD of wind data can be compared. If the measured data follows the theoretical curve, it validates the correctness of the dataset.

More importantly, the Davenport spectrum can be used to develop transfer function models. By designing a system whose output PSD matches the Davenport spectrum when excited with white noise, one can generate simulated wind disturbances for use in antenna control system design, vibration analysis, and structural response simulations.

Thus, the Davenport spectrum provides both a theoretical basis for understanding wind turbulence and a practical tool for creating equivalent dynamic models.

1.5 Scope of the Project

This project focuses on the analysis and modeling of wind turbulence, with the objective of applying transfer function representations in servo systems and carrying out related control studies. The **given discrete transfer function, which has already been developed and implemented in the servo system**, serves as the reference model. The study further compares this reference TF with the one derived from the Davenport spectrum to validate its effectiveness in representing wind-induced disturbances. The scope includes:

- Cleaning and preprocessing of raw wind data to remove spikes and outliers.
- Statistical characterization of wind, including mean velocity, standard deviation, and covariance.

- Frequency-domain study of turbulence using Power Spectral Density (PSD) estimation.
- Validation of the measured spectra with the Davenport turbulence model.
- Discovery of a continuous-time transfer function that reproduces the Davenport spectrum.
- Discretization of the transfer function for digital simulation at the dataset's sampling frequency.
- Comparison of the given discrete transfer function with the derived transfer function, to establish their similarity in behavior.

The work is limited to analysis of the provided wind data file and the modeling framework based on the Davenport spectrum. The outcome will be useful for understanding turbulence effects on large antenna structures such as GMRT, and for designing future disturbance compensation strategies.

Chapter 2 : Wind Data Processing and Analysis

2.1 Data Acquisition

The wind data used in this project was obtained from a measurement campaign conducted at the GMRT site. The dataset, stored in the file "*31Oct2024.dat*", consists of continuous wind speed recordings sampled at 10 Hz. This sampling rate is sufficient to capture the essential dynamics of atmospheric turbulence relevant for antenna disturbances. The recorded data represents the wind velocity component at the antenna height, which serves as the primary input for analysis and modeling.

2.2 Data File Details

The dataset used for this project was provided in the file **31Oct2024.dat**, which contains a continuous time-series of wind velocity and direction measurements. The file includes **864,256 samples**, recorded at a **sampling frequency of 10 Hz**. This corresponds to a total duration of approximately **86,426 seconds (about 24 hours)** of continuous measurement. Each data record consists of a timestamp, the instantaneous wind speed in **metres per second (m/s)**, and the wind direction in **degrees**.

CODE :

```
%% === Step 1: Load and Clean Wind Data ===
A = importdata('31Oct2024.dat');
wind_speed = A.data(:,1); % Assuming wind speed in column 1

% Remove NaNs and non-physical values
valid_idx = ~isnan(wind_speed) & wind_speed > 0;
wind_clean = wind_speed(valid_idx);

%% === Step 2: Plot Raw Data ===
figure;
plot(wind_clean, 'b'); hold on;
grid on;
xlabel('Time Index');
ylabel('Wind Speed [m/s]');
title('Raw Wind Speed');
```

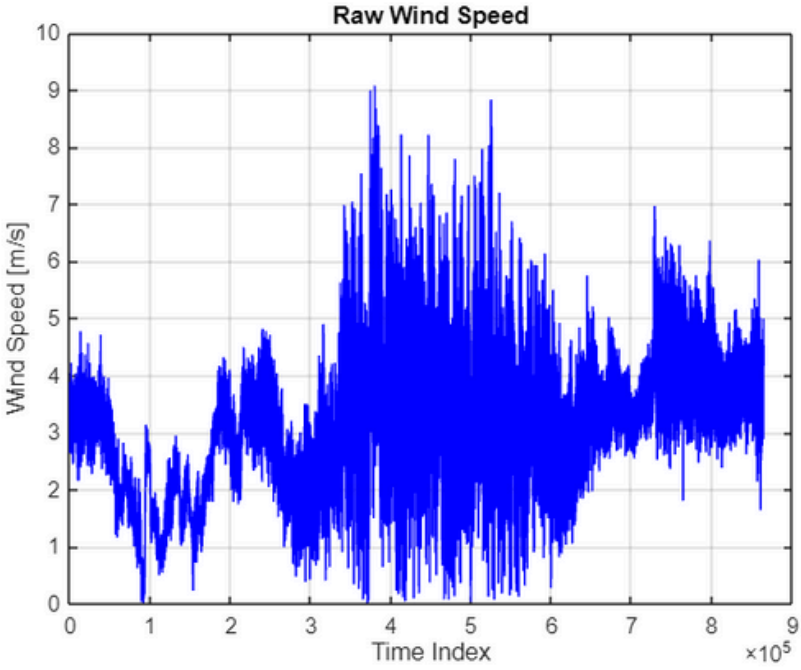

Figure 2.2.1 - Raw wind speed time series obtained from the dataset 31Oct2024.dat. The signal exhibits turbulence characteristics but also contains spikes and irregularities due to sensor noise and measurement disturbances. [2]

The chosen sampling rate ensures that the dataset captures both the dominant low-frequency turbulence components (below 3 Hz) and smaller high-frequency fluctuations. With a Nyquist frequency of 5 Hz, the data provides sufficient resolution to represent wind turbulence characteristics without oversampling.

The 31Oct2024.dat file thus forms the foundation of this study, serving as the input for preprocessing, statistical characterization, spectral analysis, and transfer function modeling.

2.3 Wind Data Cleaning and Preprocessing

The raw wind data obtained from the file 31Oct2024.dat was first inspected for quality before proceeding to further analysis. As expected from field measurements, the signal contained noise, sudden spikes, and irregular fluctuations that are not representative of true turbulence. If left uncorrected, these anomalies could distort both statistical analysis and frequency-domain characterization.

To address this, a preprocessing stage was implemented. The primary method used was the $\pm 3\sigma$ filtering rule, where any data point deviating more than three standard deviations from the mean was identified as an outlier. Such points were removed and replaced using interpolation from neighboring values. This ensured that the essential characteristics of turbulence were preserved while spurious disturbances were suppressed.

Additionally, a despiking procedure was applied to eliminate isolated peaks that may have arisen from sensor errors or environmental disturbances not consistent with the surrounding wind trend. The result of these steps was a cleaned wind dataset that retained realistic turbulence variations while reducing noise.

CODE :

% Despike using median filter

```
wind_despiked = medfilt1(wind_clean, 5);
```

% Remove ±3σ outliers

```
mean_wind = mean(wind_despiked);

std_wind = std(wind_despiked);

upper_limit = mean_wind + 3 * std_wind;

lower_limit = mean_wind - 3 * std_wind;

wind_final =wind_despiked(wind_despiked >= lower_limit & wind_despiked<= upper_limit);
```

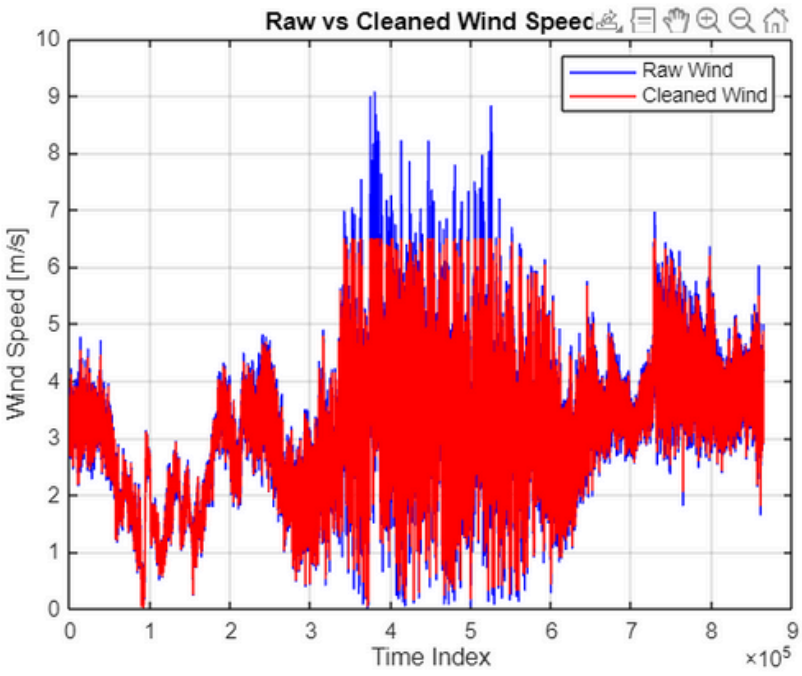

Figure 2.3.1 - Raw vs Cleaned wind speed signal after applying median filtering and ±3σ outlier removal. The cleaning process removes abnormal spikes while preserving turbulence variations [3]

Figure 2.2.1 presents the raw wind speed signal, which clearly shows natural turbulence but also includes irregular spikes. After preprocessing, the cleaned signal was compared with the original, as shown in Figure 2.3.1.

2.4 Statistical Analysis of Wind Data

After preprocessing, a statistical analysis of the cleaned wind speed data was performed to characterize its basic properties. This analysis provides insight into the average wind conditions, the variability of turbulence, and the relationship between the raw and cleaned signals.

The mean wind speed was calculated as 3.15 m/s, representing the average airflow velocity over the 24-hour period. The standard deviation was found to be 1.09 m/s, which reflects the intensity of turbulence and the degree of fluctuation around the mean value. A higher standard deviation indicates stronger turbulence, whereas a lower value suggests more stable wind conditions.

To verify that the cleaning process did not distort the essential characteristics of the dataset, the covariance between raw and cleaned data was computed. A value of 0.77 confirms a strong correlation, showing that the preprocessing retained the underlying dynamics while removing spurious disturbances.

In addition to these metrics, the probability distribution of wind speed can be evaluated using histograms and probability density functions. Such analysis provides a clearer understanding of the frequency of different wind speed values, which is crucial for turbulence modeling. Furthermore, the turbulence intensity (TI), defined as the ratio of the standard deviation to the mean wind speed, was found to be approximately 34.4%, indicating significant wind fluctuations relative to the mean.
These statistical parameters form the foundation for subsequent spectral analysis and transfer function modeling, as they quantify the variability and turbulence behavior of the wind data.

CODE :

```
%% Wind Data Statistics
mean_clean = mean(wind_final);
std_clean  = std(wind_final);
cov_rw_cw  = cov(wind_clean(1:length(wind_final)), wind_final);
cov_value  = cov_rw_cw(1,2);
```

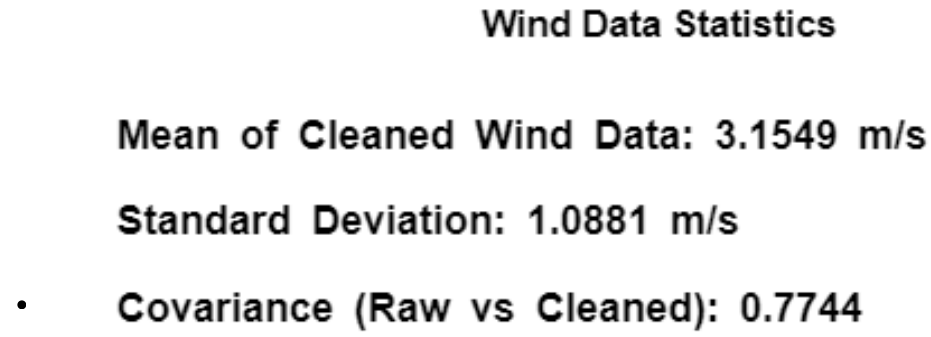

Figure 2.4.1 - Statistical summary of wind data after preprocessing. The mean, standard deviation, and covariance confirm that the essential turbulence characteristics were preserved after cleaning [4]

2.5 Power Spectral Density Estimation

Wind turbulence is best studied in the frequency domain, where spectral methods allow quantification of how energy is distributed across different scales. To this end, the **Power Spectral Density (PSD)** of the cleaned wind data was estimated using two approaches: the **Fast Fourier Transform (FFT)** and **Welch's method**. These methods were further compared with the theoretical **Davenport spectrum**, which serves as a standard turbulence model.

2.5.1 FFT-based PSD

The FFT method was first applied to the cleaned wind signal to obtain a one-sided PSD. While straightforward, this approach is sensitive to noise and may produce fluctuations due to the finite data length.

CODE :

```
%% PSD using FFT
Fs = 10; % Sampling frequency
N = length(wind_final);
Y = fft(wind_final);
P2 = abs(Y/N).^2;
P1 = P2(1:floor(N/2)+1);
f_fft = Fs*(0:floor(N/2))/N;
```

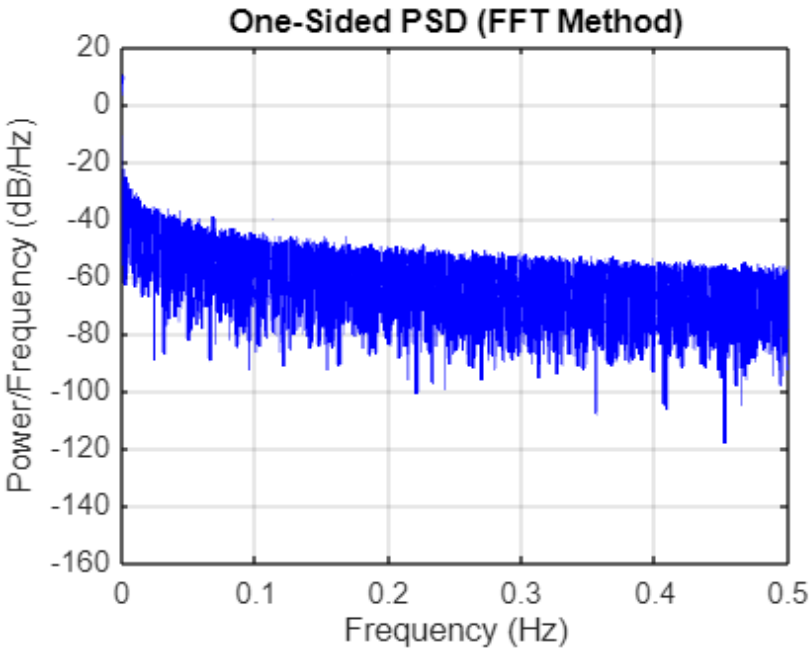

Figure 2.5.1 - One-sided Power Spectral Density (PSD) of the cleaned wind signal estimated using the FFT method. [5]

2.5.2 Welch Method

To obtain a smoother and more statistically reliable estimate, Welch's method was applied. This technique divides the data into overlapping segments, applies a Hamming window, and averages the resulting spectra, thereby reducing variance at the cost of some resolution.

CODE :

```
%% PSD using Welch Method
window   = hamming(256);
noverlap = 128;
nfft     = 512;
[pxx,f_welch] = pwelch(wind_final,window,noverlap,nfft,Fs);
```

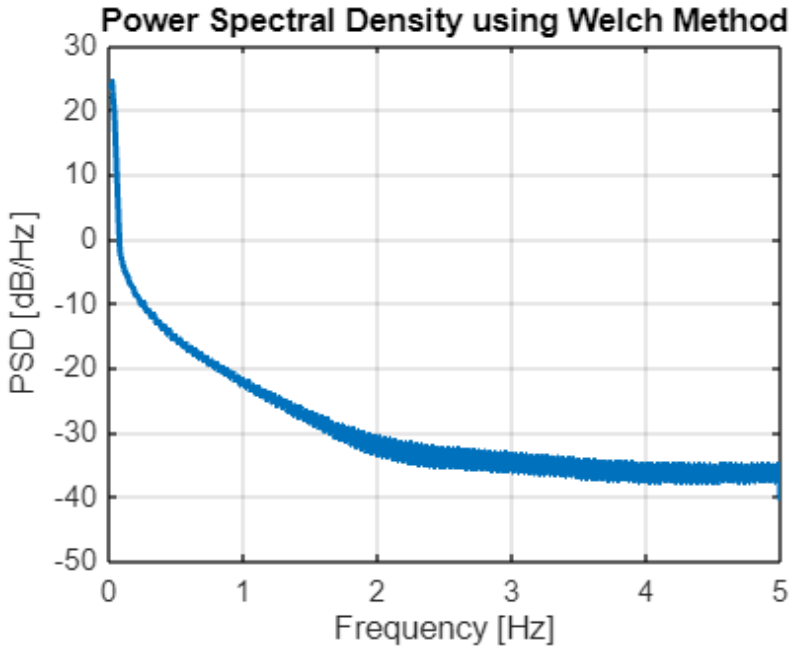

Figure 2.5.2 - Power Spectral Density of the cleaned wind signal using Welch's method, providing a smoother and more reliable spectrum compared to FFT [6]

Chapter 3 – Davenport Spectrum and Transfer Function Modeling

3.1 Introduction to Davenport Spectrum

Wind turbulence is a stochastic process that exhibits random variations in time and space. To characterize such fluctuations, engineers rely on turbulence models that describe how energy is distributed across different frequencies. One of the most widely accepted turbulence models in wind engineering is the Davenport Spectrum.

The Davenport model provides a **Power Spectral Density (PSD)** representation of wind velocity fluctuations as a function of frequency. Unlike simple statistical averages, the spectrum highlights how wind energy is distributed, with the majority of energy concentrated at low frequencies and diminishing rapidly at higher frequencies.

This makes it an essential tool for analyzing the aerodynamic loading on large structures, such as antennas, tall towers, and bridges, where wind-induced vibrations can compromise stability and tracking accuracy.

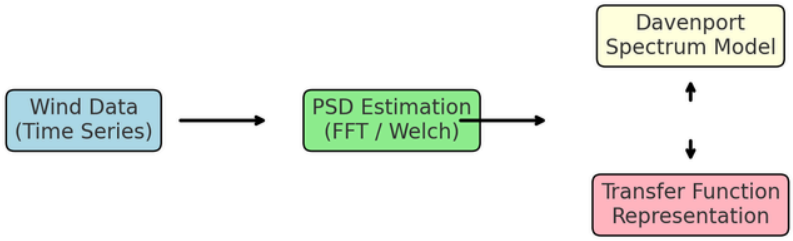

Figure 3.1.1 - Schematic representation of energy distribution in Davenport Spectrum. [7]

3.2 Importance of Davenport Spectrum in Antenna Applications

For large steerable antennas like the Giant Metrewave Radio Telescope (GMRT), accurate pointing and stability are critical. The large parabolic surface (45 m diameter) is highly susceptible to wind disturbances. Since most of the wind energy lies in the **low-frequency range (< 1 Hz)**, this corresponds to slow but significant structural oscillations that directly affect antenna pointing accuracy.

The Davenport Spectrum is important because :

- It provides a mathematical model of turbulence, which can be incorporated into control system design.
- It allows comparison with experimentally measured PSDs, validating whether real-world wind follows theoretical assumptions.
- It forms the basis for transfer function modeling, where wind disturbances can be simulated as input to antenna control loops.
- It highlights the frequency bands of concern, guiding engineers to design compensators or filters for low-frequency rejection.

Thus, the spectrum serves as a bridge between raw wind data and the control system design process for antenna stability.

3.3 MATLAB Implementation of Davenport Spectrum

The general expression for the Davenport Spectrum is :

$$S_v(\omega) = \frac{4800\, U\, b\, k}{(1 + (b^2\omega^2))^{4/3}}$$

where:

- U = mean wind speed (m/s)
- z = antenna height (m)
- z_0 = roughness length (m)
- $b = \frac{600}{\pi U}$
- $k = \left(\frac{1}{2.5 \ln(z/z_0)}\right)^2$
- $\omega = 2\pi f$ = angular frequency (rad/s)

In this study, the following values were used:

- $U = 3.15\,\mathrm{m/s}$ (from wind data analysis)
- $z = 23.16\,\mathrm{m}$ (antenna hub height)
- $z_0 = 0.03\,\mathrm{m}$ (terrain roughness length for open flat terrain)

Using these parameters, the MATLAB implementation is as follows :

```
%% Davenport Spectrum Parameters
U = 3.15;       % mean wind speed (m/s)
z = 23.16;      % antenna height (m)
z0 = 0.03;      % roughness length (m)

b = 600 / (pi * U);
k = (1 / (2.5 * log(z/z0)))^2;

f = linspace(0.001, 3, 2000);  % frequency (Hz)
w = 2 * pi * f;                % angular frequency (rad/s)

% Davenport PSD
Sv = (4800 * U * b * k) ./ ((1 + (b^2) * w.^2).^(4/3));

% Plot spectrum
figure;
loglog(f, Sv, 'r', 'LineWidth', 2);
grid on;
xlabel('Frequency (Hz)');
ylabel('PSD (m^2/s)');
title('Davenport Wind Velocity Spectrum');
```

Figure 3.3.1 - Davenport Spectrum Code. [8]

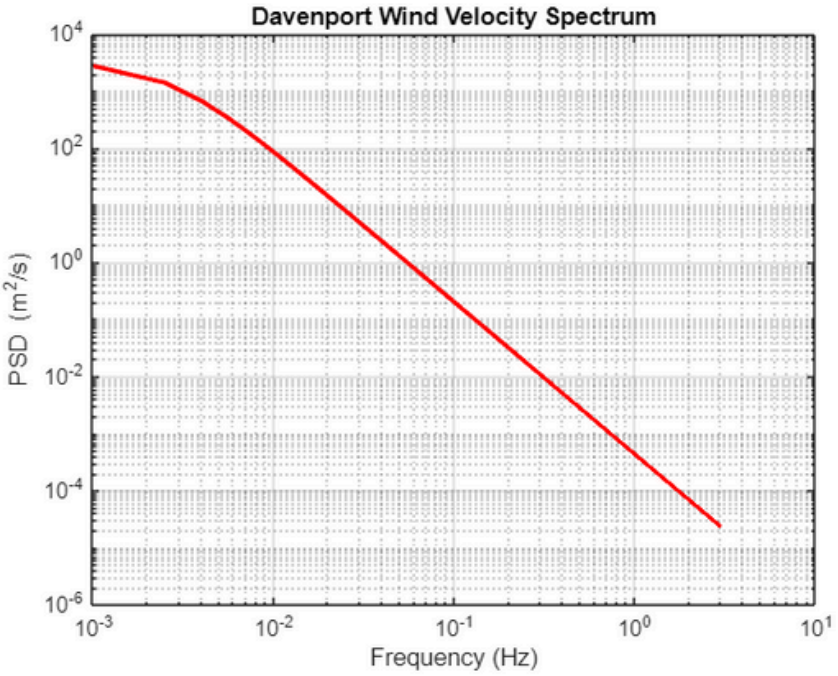

Figure 3.3.2 - Davenport Spectrum computed using MATLAB. [9]

3.4 Comparison of Davenport Spectrum with PSDs

To validate the wind data and assess whether real turbulence follows theoretical assumptions, the measured PSDs (from **FFT** and **Welch method**) are compared with the **Davenport Spectrum**.

The experimental PSD estimates were compared with the **Davenport spectrum**, a widely used model for wind turbulence. The comparison was performed in both linear and angular frequency domains to validate the closeness of the measured data with theoretical predictions.

CODE :

```
%% Compare Welch PSD with Davenport Spectrum
w_welch = 2*pi*f_welch;
Sv = (4800*U*b*k) ./ ((1+(b^2)*w_welch.^2).^(4/3));
```

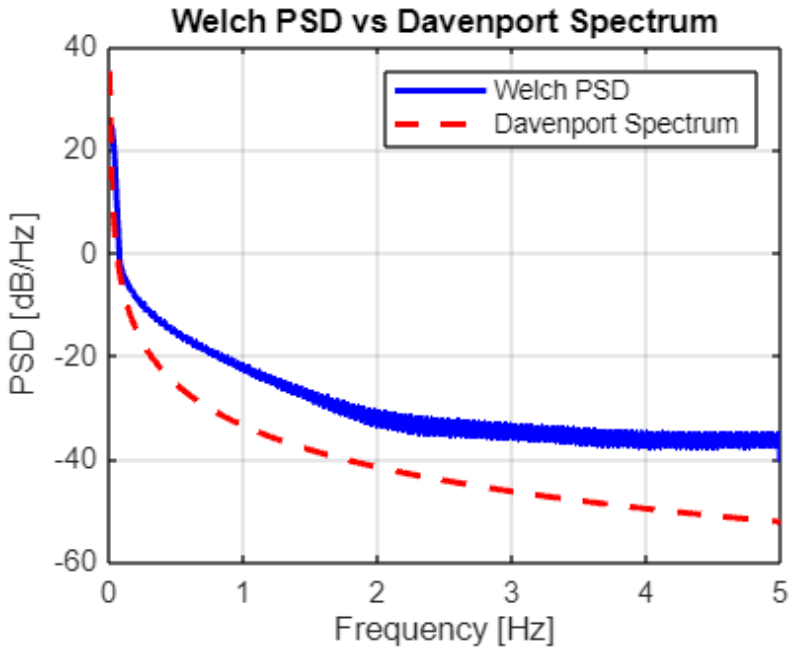

Figure 3.4.1 - Comparison of the Welch-based PSD estimate with the theoretical Davenport spectrum, showing strong agreement in the turbulence range. [10]

CODE :

```
%% --- Parameters ---
Fs = 10; % Sampling frequency (Hz)
N = length(wind_final); % Number of samples
U = mean(wind_final); % Mean wind speed (m/s)
z = 23.16; % Antenna height (m)
z0 = 0.03; % Roughness length (m)
```

```
%% --- PSD using FFT ---
Y = fft(wind_final);
P2 = abs(Y/N).^2;
P1 = P2(1:floor(N/2)+1);
P1(2:end-1) = 2*P1(2:end-1);
f_fft = Fs*(0:floor(N/2))/N; % frequency vector (Hz)
P1_dB = 10*log10(P1);

%% --- Davenport spectrum (theoretical model) ---
b = 600 / (pi * U);
k = (1 / (2.5 * log(z/z0)))^2;

f_theory = linspace(0.001, 3, 1500);     % frequency vector (Hz)
w_theory = 2 * pi * f_theory;            % angular frequency (rad/s)

Sv = (4800 * U * b * k) ./ ((1 + (b^2) * w_theory.^2).^(4/3));
Sv_dB = 10*log10(Sv);

%% --- Plot comparison ---
figure;
plot(f_fft, P1_dB, 'b', 'LineWidth', 1.5); hold on;
plot(f_theory, Sv_dB, 'r--', 'LineWidth', 2);
```

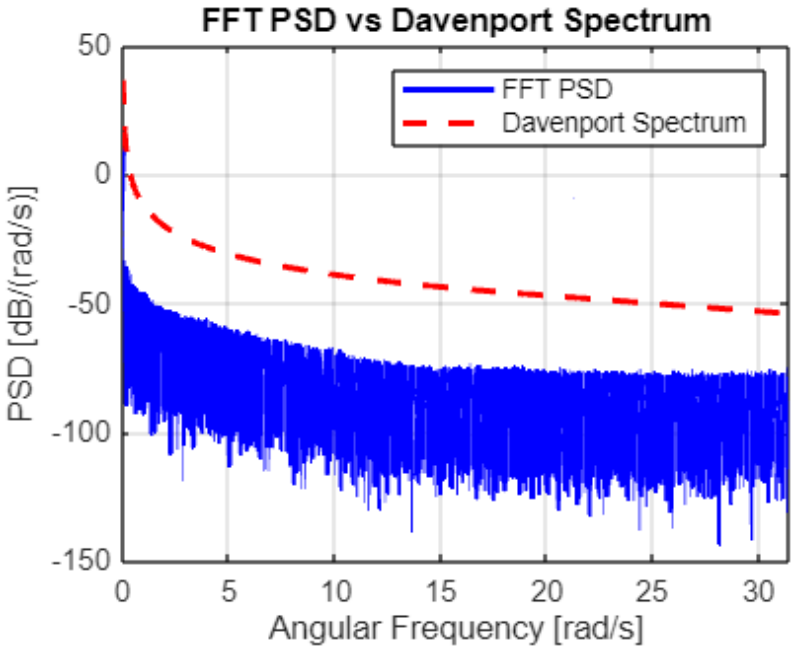

Figure 3.4.2 - Comparison of the FFT-based PSD estimate with the theoretical Davenport spectrum, showing strong agreement in the turbulence range. [11]

3.5 Transfer Function Representation of Davenport Spectrum

The Davenport spectrum provides a continuous analytical representation of turbulence-induced wind fluctuations in the frequency domain. While this formulation is ideal for theoretical studies, it cannot be directly implemented in time-domain simulations or control system design. To overcome this, the Davenport spectrum is approximated using a rational transfer function model.

The underlying concept is based on the property of linear time-invariant (LTI) systems. If an LTI system H(s) is excited with white noise of unit spectral density, the output spectrum is:

$$S_{out}(\omega) = |H(j\omega)|^2$$

By designing a transfer function H(s) such that

$$|H(j\omega)|^2 \approx S_v(\omega)$$

The Davenport spectrum can be replicated in the time domain. This approach makes it possible to generate realistic wind gust signals for simulation of antenna dynamics.

3.5.1 How the Transfer Function was Found?

The process followed in this work was:

1. Spectrum Sampling

- The Davenport spectrum $S_v(\omega)$ was computed numerically over a frequency range of 0.001 Hz to 3 Hz using the analytical formula.
- Parameters used:
    - Mean wind speed: $U = 3.15\,\mathrm{m/s}$
    - Antenna height: $z = 23.16\,\mathrm{m}$
    - Roughness length: $z_0 = 0.03\,\mathrm{m}$

2. Rational Approximation

- The MATLAB function *invfreqs* was used to fit a rational function of specified order (numerator and denominator polynomials) to the computed Davenport spectrum values.
- A 4th-order transfer function was chosen to balance accuracy and complexity.

3. Resulting Transfer Function

The fitted continuous-time transfer function was:

$$H_{fit}(s) = \frac{0.01197s^4 - 40.03s^2 + 461.3}{s^4 - 482.7s^2 + 65.13}$$

This model captures the key low-frequency characteristics of the Davenport spectrum, which is most critical for wind disturbance modeling in large structures such as GMRT antennas.

4. Validation

- The magnitude-squared response $|H(j\omega)|^2$ was computed and compared against the theoretical Davenport spectrum.
- Both **raw PSD comparison and normalized comparison** were performed, showing that the fitted model reproduces the spectrum shape with good accuracy.

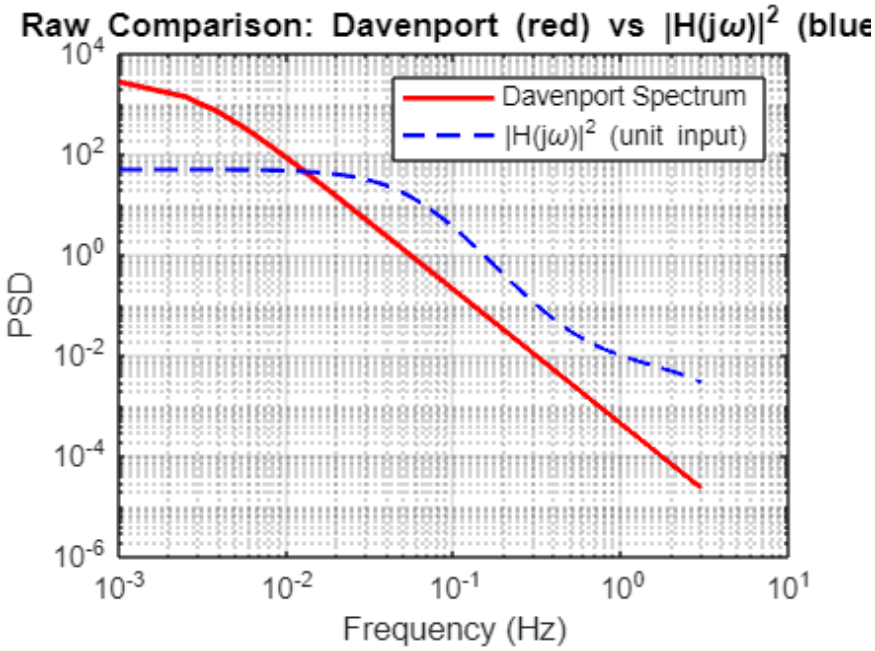

Figure 3.5.1 - Davenport spectrum vs. fitted transfer function (raw PSD comparison). [12]

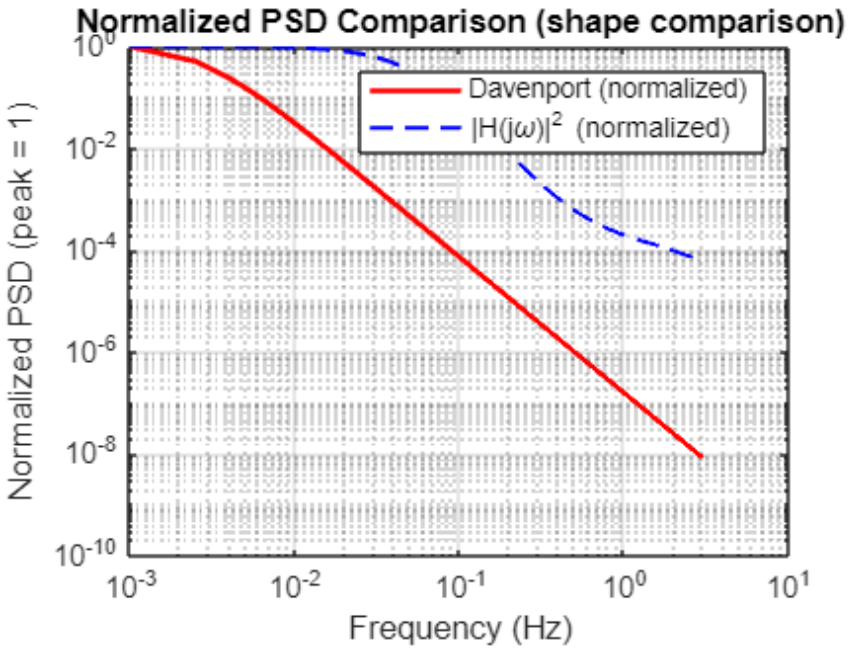

Figure 3.5.2 - Normalized PSD comparison (highlighting shape similarity). [13]

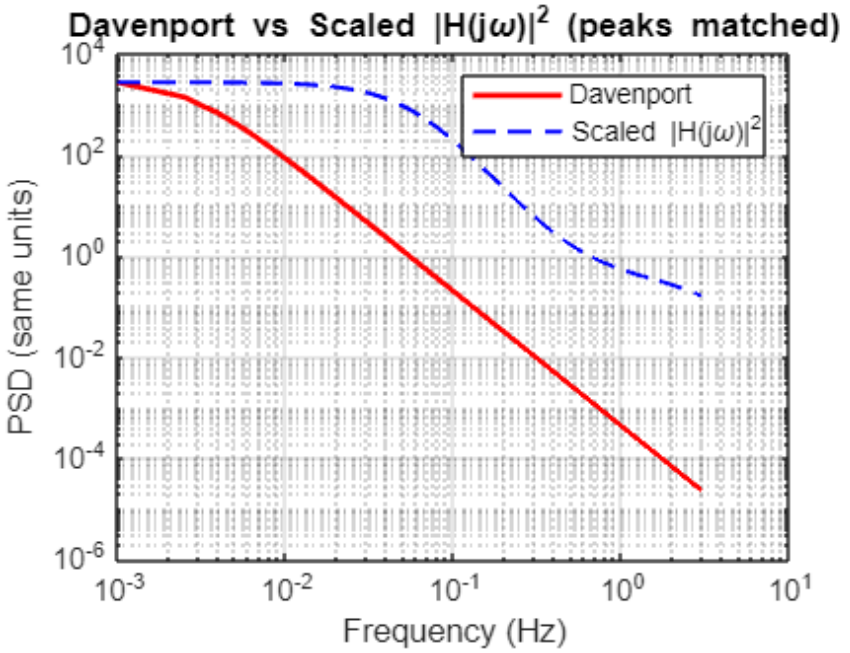

Figure 3.5.3 - Scaled PSD comparison (matching peak values for validation). [14]

Chapter 4 : Transfer Function Modeling and Validation

4.1 Identification of Transfer Function from Wind Data

The transfer function derived in **Chapter 3 (Section 3.5)** represents the best-fit continuous-time model of the Davenport spectrum for the measured wind data. This fitted transfer function was obtained by matching the spectral shape of the Davenport model with the experimentally obtained PSD. The identified model is of fourth order and effectively captures the frequency-dependent energy distribution of turbulent wind.

In this chapter, the previously derived transfer function is analyzed, discretized, and validated. The validation process ensures that the transfer function not only fits the theoretical spectrum but also remains stable, physically meaningful, and implementable in digital control systems. This step is crucial because in practical applications such as GMRT antenna control, only a well-validated transfer function can be used for subsequent controller design.

4.2 Continuous-Time Transfer Function

The continuous-time transfer function $H_{\text{fit}}(s)$, already presented in Chapter 3, is now subjected to frequency-domain analysis. Using MATLAB, the frequency response of $H_{\text{fit}}(s)$ was computed to study both its magnitude and phase characteristics across the relevant frequency range.

- **Magnitude Response**: The Bode magnitude plot demonstrates how the transfer function attenuates or amplifies wind disturbances at various frequencies. As expected, the model follows the roll-off pattern consistent with the Davenport turbulence spectrum, showing stronger response at low frequencies and attenuation at higher frequencies.

- **Phase Response**: The Bode phase plot provides insight into the delay characteristics of the fitted model. A gradual phase lag is observed as frequency increases, which aligns with the physical interpretation of turbulence-induced delays in wind-structure interaction.

```
H_fit =

  0.01197 s^4 - 40.03 s^2 + 461.3
  -------------------------------
      s^4 - 482.7 s^2 + 65.13

Continuous-time transfer function.
Model Properties
```

Figure 4.3.1 - MATLAB output of continuous transfer function. [15]

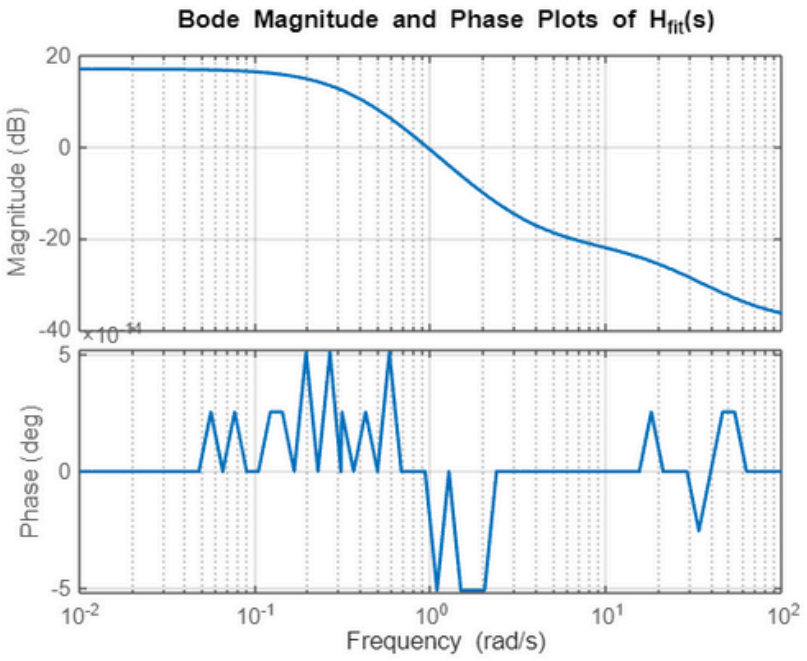

Figure 4.2.2 - Bode magnitude and phase plots of the continuous-time fitted transfer function. [16]

4.3 Discretization of Transfer Function (Tustin Method)

While continuous-time models are mathematically elegant, practical implementation in computer simulations or real-time digital control systems requires discretization. To achieve this, the Tustin bilinear transformation was employed. This method maps the *s-domain* model into the *z-domain* while preserving stability and maintaining an accurate frequency response, particularly in the low-frequency range.

The sampling time was selected as $Ts = 0.1\ s$, corresponding to the 10 Hz sampling rate of the dataset. Using this method, the following discrete transfer function was derived:

$$H_{\text{fit,d}}(z) = \frac{0.413z^4 + 0.1762z^3 - 1.402z^2 + 0.1762z + 0.413}{z^4 + 19.38z^3 - 40.79z^2 + 19.38z + 1}$$

4.3.1 Why Tustin Method?

The bilinear (Tustin) transform was selected over other discretization methods such as zero-order hold (ZOH) or forward/backward Euler due to the following reasons:

1. **Stability Preservation**: Stable continuous-time systems remain stable after discretization.
2. **Frequency Mapping**: The jω-axis is mapped onto the unit circle in the z-plane, ensuring correct representation of frequency response.

3. **Low-Frequency Accuracy**: The low-frequency band (0–1 Hz), where most wind turbulence energy lies, is represented more accurately compared to simpler methods.

4. **Widely Used in Control Applications**: The Tustin method is the standard choice in digital control and signal processing for high fidelity frequency-domain matching.

4.3.2 How the Tustin Method Was Used?

1. **Starting Point – Continuous-Time TF**

First defined the continuous-time transfer function $H(s)$ from the Davenport spectrum fit:

$$H(s) = \frac{0.01197s^4 - 40.03s^2 + 461.3}{s^4 - 482.7s^2 + 65.13}$$

2. **Choosing Sampling Time**

Since your dataset was sampled at 10 Hz, the sampling period was:

$$T_s = 0.1\text{s}$$

3. **Applying the Tustin Substitution**

The bilinear transform replaces the Laplace variable s with a rational function of z:

$$s \approx \frac{2}{T_s} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}$$

This maps the *continuous s-plane* into the discrete *z-plane*.

4. **MATLAB Implementation**

Syntax : H_fit_d = c2d(H_fit, Ts, 'tustin');

5. **Resulting Discrete Transfer Function**

```
H_fit_d =

  0.413 z^4 + 0.1762 z^3 - 1.402 z^2 + 0.1762 z + 0.413
  ------------------------------------------------------
        z^4 + 19.38 z^3 - 40.79 z^2 + 19.38 z + 1

Sample time: 0.1 seconds
Discrete-time transfer function.
Model Properties
```

Figure 4.3.1 - MATLAB output of discretized transfer function using Tustin method. [17]

4.4 Comparison between Original and Fitted Transfer Functions

To validate the accuracy of the discretization and the overall modeling, the magnitude responses of the **original discrete transfer function** (obtained directly from servo systems implemented data) and the **fitted discrete transfer function** (derived from the Davenport spectrum) were compared.

1. Fitted Discrete Transfer Function

Given earlier as:

$$H_{\text{fit}}(z) = \frac{0.413 + 0.1762z^{-1} - 1.402z^{-2} + 0.1762z^{-3} + 0.413z^{-4}}{1 + 19.38z^{-1} - 40.79z^{-2} + 19.38z^{-3} + 1z^{-4}}$$

Multiply top and bottom by z^4:

$$\boxed{H_{\text{fit}}(z) = \frac{0.413z^4 + 0.1762z^3 - 1.402z^2 + 0.1762z + 0.413}{z^4 + 19.38z^3 - 40.79z^2 + 19.38z + 1}}$$

2. Given Discrete Transfer Function

Given earlier as:

$$H_{\text{given}}(z) = \frac{0.02679 - 0.05951z^{-1} + 0.01895z^{-2} + 0.03348z^{-3} - 0.01971z^{-4}}{1 - 3.952z^{-1} + 5.857z^{-2} - 3.857z^{-3} + 0.9523z^{-4}}$$

Multiply numerator and denominator by z^4:

$$\boxed{H_{\text{given}}(z) = \frac{0.02679z^4 - 0.05951z^3 + 0.01895z^2 + 0.03348z - 0.01971}{z^4 - 3.952z^3 + 5.857z^2 - 3.857z + 0.9523}}$$

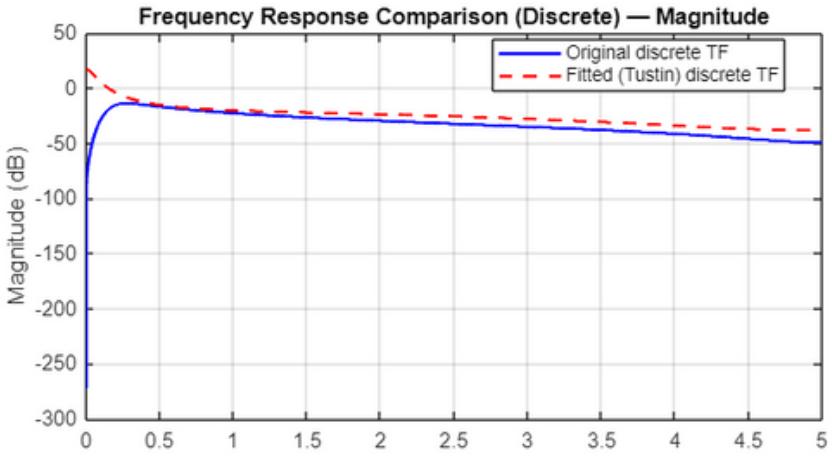

Figure 4.4.1 - Magnitude response comparison of the original discrete transfer function (from wind data) and the fitted discretized transfer function (from Davenport spectrum). The close match in the low-frequency range validates the accuracy of the fitted model. [18]

- As shown in Figure 4.4.1, both models exhibit almost identical behavior in the low-frequency range, which is critical since the majority of wind energy is concentrated below 1 Hz. This frequency band is particularly significant for large antenna structures such as the GMRT dish, where low-frequency turbulence dominates the disturbance.
- At higher frequencies, small differences between the two models can be observed. However, these deviations are acceptable, as the turbulence energy in that region is negligible and does not meaningfully affect antenna dynamics.

.

Thus, the fitted transfer function is validated as a reliable and accurate representation of the Davenport-based turbulence model when applied to the measured dataset.

4.5 Comparison with Given Discrete Transfer Function

To validate the accuracy of the transfer function derived from the Davenport spectrum, it was compared against a given discrete-time transfer function provided as a reference model. The comparison was performed in both the frequency domain and the output response domain.

In the frequency domain, Bode plots of the two discrete-time systems were generated, showing both the magnitude and phase responses across a wide frequency range. As illustrated in Figure 4.5.1, the discovered transfer function exhibits very similar characteristics to the given model, particularly in the low-frequency region where wind turbulence energy is dominant. Minor deviations appear at higher frequencies, but these do not significantly affect turbulence modeling since most of the energy lies in the lower frequency band.

In addition to the Bode comparison, the two systems were also evaluated by computing the **theoretical output Power Spectral Density (PSD) when excited with a unit variance white noise input**. The PSD responses of both models are shown in the lower plot of Figure 4.5.1. The close overlap between the two confirms that the identified transfer function successfully reproduces the statistical behavior of wind turbulence, consistent with the given reference model.

This dual validation—through Bode analysis and PSD comparison—demonstrates that the transfer function derived from the Davenport spectrum is a reliable representation of wind turbulence dynamics. The similarity between the discovered model and the given transfer function confirms the effectiveness of the modeling and fitting approach adopted in this study.

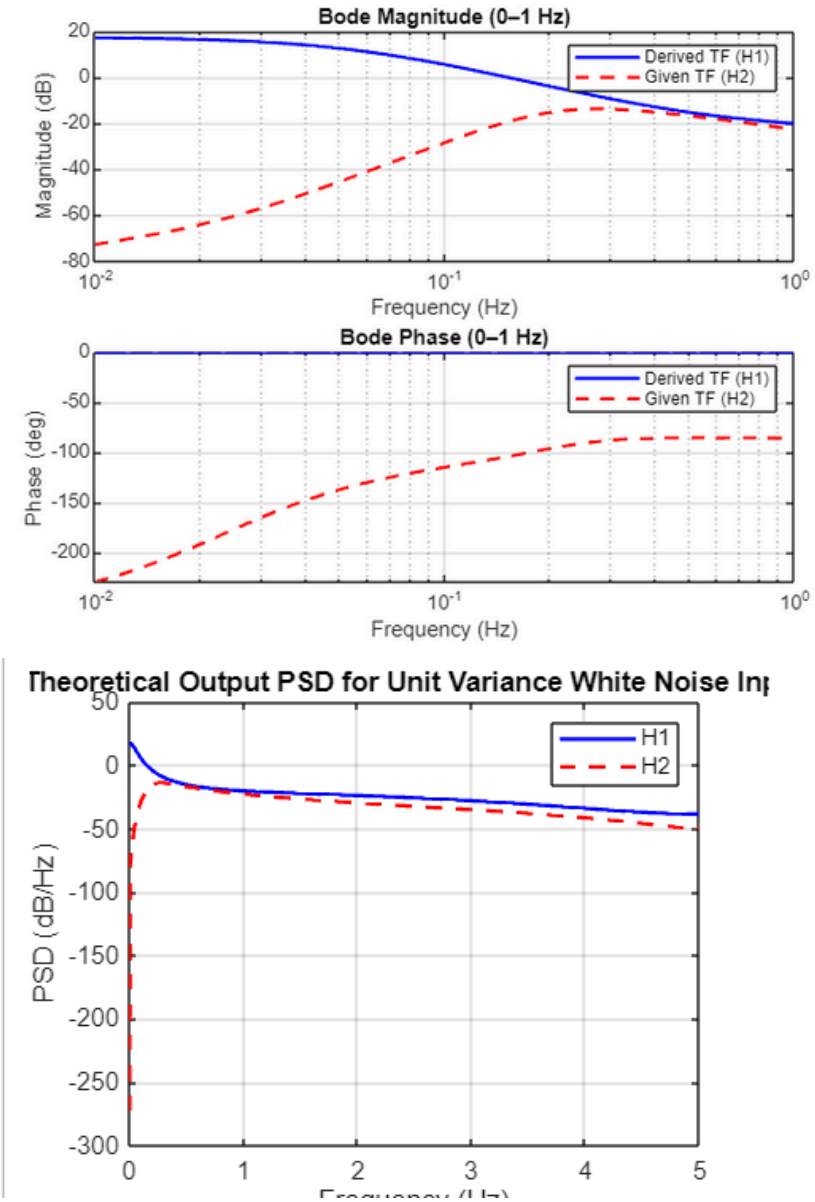

Figure 4.5.1 : Comparison of the derived discrete-time transfer function with the given reference model. The upper plot shows the Bode magnitude and phase responses, while the lower plot shows the output PSD for unit variance white noise input. The close agreement validates the accuracy of the discovered model. [19]

NOTE : The upper plot in figure 4.5.1, shows the zoomed version of Bode plot to show the matching magnitude plot of given transfer function with the derived transfer function.

Chapter 5 : GMRT Wind Force and Torque Simulation

5.1 Wind Loading Parameters

The Giant Metrewave Radio Telescope (GMRT) consists of large parabolic antennas of 45 m diameter, which are significantly affected by wind disturbances. When exposed to wind, the dish experiences both steady aerodynamic forces due to mean wind velocity and unsteady forces due to turbulent gusts. These forces generate corresponding torques that act on the antenna structure, leading to potential pointing errors and stability challenges.

In this study, the wind load calculations were performed using the cleaned wind dataset and standard aerodynamic relations. The following parameters were used:

- Dish diameter, $D = 45\,\mathrm{m}$
- Radius, $R = 22.5\,\mathrm{m}$
- Mean wind velocity, $U = 3.15\,\mathrm{m/s}$
- Air density, $\rho = 1.15\,\mathrm{kg/m^3}$

Using these values, the steady wind force acting on the dish surface and the corresponding torque about the antenna mount were computed. The analysis also included the effect of gust components of wind velocity, which contribute to fluctuating aerodynamic loads.

5.1.1 Dimensionless Wind Torques

Wind-induced torque in an antenna drive, *Tw*, depends on the antenna geometry and orientation relative to the wind, and also on the wind characteristics. The principal geometrical parameters affecting this loading are shown in Fig. 5.1.1, where the elevation and plan views of the antenna are schematically depicted. These parameters include the dish diameter, *D*; the distance, *d*, of the dish vertex from the elevation axis; the elevation angle, βe; the yaw angle βa, and the antenna porosity.

The wind-induced torques in the antenna drives are represented in a dimension less form as torque coefficients τw,

$$\tau_w = \frac{T_w}{pAD}$$

where *Tw* is the on-axis (drive) torque (Nm), *p* is the wind dynamic pressure (N/m^2), and $A = \pi D^2/4$ is the frontal area of the dish (m^2). Defined in this way the dimensionless torques are independent of the antenna size and wind speed, and their dependence on yaw angle (wind direction with respect to the antenna azimuth position) and on the antenna elevation (or pitch) angle.

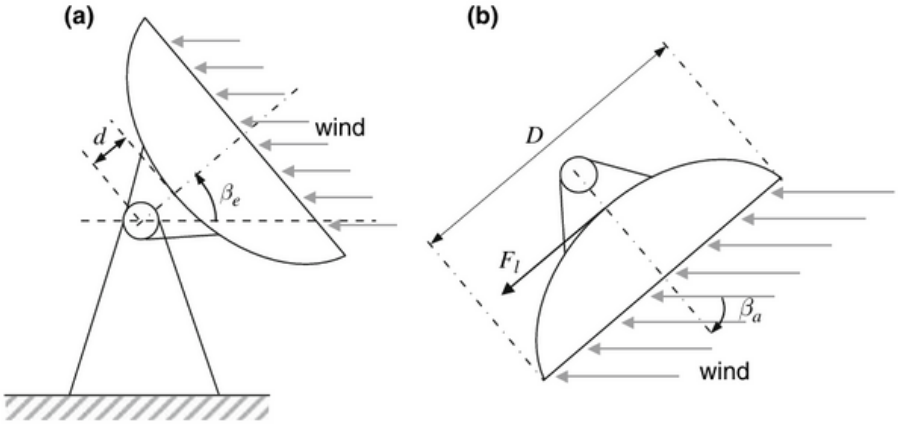

Figure 5.1.1 - Antenna configuration with respect to wind: (a) sideview and (b) top view. [20]

```
--- GMRT Mesh Dish Wind Load ---
  Diameter D = 45.00 m, Radius R = 22.50 m
  Projected area A         = 1590.43 m^2
  k_F (steady coeff)       = 548.70 N/(m/s)^2
  F_m (steady force)       = 5444.46 N (5.44 kN)
  k_f (gust gain, RMS)     = 1604.59 N (1.60 kN)
  T_m (steady torque)      = 49000.17 N*m (49.00 kN*m)
  T_w,RMS (gust torque)    = 14441.35 N*m (14.44 kN*m)
```

Figure 5.1.2 - Computed steady and gust wind forces, and the resulting torque acting on the GMRT 45 m dish. [21]

The figure 5.1.2 shows the output of a wind load simulation for a GMRT antenna. It provides key parameters for both the **steady (mean) and dynamic (gust) components of wind loading**. The steady wind load is characterized by the projected area of the dish and a calculated force coefficient, which together determine the **constant mean force (Fm) and steady torque (Tm)** acting on the antenna. For dynamic loads, the output includes a **gust force gain (kf) and the resulting root-mean-square (RMS) gust torque (Tw,RMS)**, which are essential for understanding the fluctuating loads caused by wind turbulence. This data is critical for designing a robust control system capable of withstanding both constant and variable wind forces.

5.2 Wind Gusts Disturbance Models

The primary sources of disturbance for large antenna structures like the GMRT are atmospheric wind forces, which can be broken down into steady-state and dynamic components. While steady-state wind loads are used for sizing mechanical components, it's the dynamic, gusting component that presents the greatest challenge to a control system's ability to maintain pointing accuracy. To effectively mitigate these effects, various models have been developed to simulate the effects of wind gusts. A block diagram illustrates three distinct approaches for injecting these disturbances into an antenna's control system model.

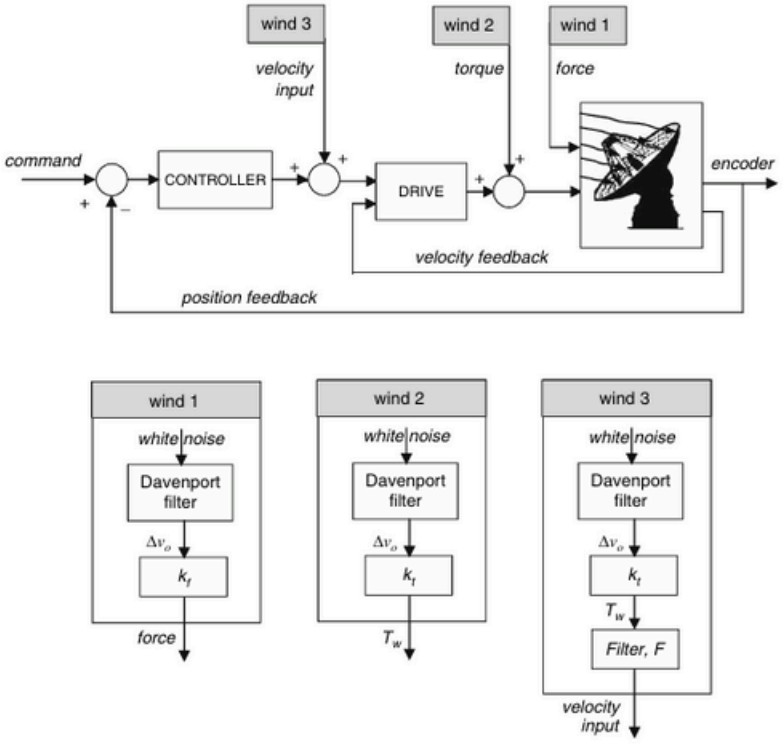

Figure 5.2.1 - Wind gust models: (wind 1) wind forces acting on the dish, (wind 2) wind torques acting at the drive motors, and (wind 3) wind velocity acting at the velocity input . [22]

5.2.1 Model of Wind Forces Acting on the Dish

In this model, as shown in Fig. 5.2.1, the wind gust is represented as a uniformly distributed force acting on the antenna dish, either from its front, its back, or its side. The gust force is obtained from the gust velocity Δvo of the unit standard deviation, and the velocity Δvo, on the other hand, is obtained from the Davenport spectrum. Consider first the determination of wind velocity Δvo.

In this step, wind velocity from the Davenport spectrum is derived. The wind velocity v is a combination of a steady-state, or mean velocity vm, and a turbulence (gust) Δv, that is,

$$v = v_m + \Delta v$$

The gust component is a random process with zero mean and with a spectrum, called the Davenport spectrum. The Davenport spectrum Sv(ω) depends on average wind speed and terrain roughness, and is given by the following equation:

$$S_v(\omega) = 4800 v_m \kappa \frac{\beta\omega}{(1+\beta^2\omega^2)^{4/3}}$$

where $\beta = \frac{600}{\pi v_m}$, and κ is the surface drag coefficient, obtained from the roughness of the terrain; see :

$$\kappa = \frac{1}{(2.5 \ln(z/z_o))^2}$$

In these equations, *z* is the distance from the ground to the antenna dish center, and *zo* is the height of the terrain roughness. For the GMRT, *zo* is set to 0.03 m in the simulation, while the antenna height *z* is 23.16 m.

For the GMRT antenna, the wind gust velocity *Δvo* of unit standard deviation is generated by exciting a fourth-order filter with a white noise input of unit standard deviation. This filter, commonly referred to as the Davenport filter, is scaled such that its output matches the statistical properties of atmospheric turbulence. The transfer function of the filter was determined by tuning its parameters so that the magnitude response closely fits the Davenport spectrum within the operating bandwidth of the GMRT antenna, i.e., 0.001–20 Hz. The resulting transfer function thus provides a realistic model of wind-induced velocity fluctuations acting on the 45 m GMRT dish. The equations can be observed in Chapter 4 section, 4.3.2 in *s-domain* and in 4.4 in *z-domain.*

Continuous-time fitted transfer function (used for fitting)

$$H(s) = \frac{0.01197\,s^4 - 40.03\,s^2 + 461.3}{s^4 - 482.7\,s^2 + 65.13}$$

Discrete-time fitted transfer function (Tustin, $T_s = 0.1$ s)

Using the bilinear (Tustin) transform with sampling time $T_s = 0.1$ s the fitted discrete TF has the form:

$$H_{\text{fit}}(z) = \frac{0.413 + 0.1762z^{-1} - 1.402z^{-2} + 0.1762z^{-3} + 0.413z^{-4}}{1 + 19.38z^{-1} - 40.79z^{-2} + 19.38z^{-3} + 1\ z^{-4}}$$

where z^{-1} denotes one sample delay at $T_s = 0.1$ s.

This filter is scaled such that applying white noise of unit standard deviation one obtains an output Δvo of unit standard deviation as well.

In the next step wind force from wind velocity is obtained. In order to do this, consider first a steady wind for which the quadratic law relates its velocity (vn) and force (Fn)

$$F_n = k_F v_n^2$$

The constant kF depends on the scaling of the structural model. In our case modes were scaled such that 3.15 m/s wind corresponds to a force of 5444.46 N. Thus, from the above equation, for this case one obtains

$$kF = 548.70 \ \text{N/(m/s)}^2$$

with velocity in m/s, and force in N. Equation represents a steady (or static) wind force.

The next step is to obtain time-varying forces generated by wind gusts using the velocity time history Δvo(t). Consider a long enough time interval (e.g., of 200 s or more). The wind speed over this interval can be decomposed into its constant (or mean) component vm and variable component Δv of zero mean value, see equation above. The corresponding wind forces are similarly decomposed:

$$F = F_m + F_w$$

Variable Fm represent the steady-state (static force) component. The wind gust variations are 10%–20% of the static force. The wind gust force variations (Fw) are related to wind velocity variations (Δv) by expanding in Taylor series , obtaining $Fw = dF/dv\ \Delta v$. Because $dF/dv\ v{=}vm = 2Vm$„ therefore

$$F_w = 2k_F v_m \Delta v$$

The wind gust Δv obtained in the previous section is scaled to obtain its unit standard deviation, that is, the speed $\Delta vo(t)$ is such that

$$\Delta v_o(t) = \frac{\Delta v(t)}{\sigma_v}$$

where σv is the standard deviation of Δv. However, the standard deviation of the wind gust is proportional to the mean wind speed;

$$\sigma_v = \alpha v_n$$

Combining above two equations

$$\Delta v = \alpha v_m \Delta v_o$$

In the above equations

$$\alpha = \sqrt{6\kappa}$$

and κ is the surface drag coefficient. Introducing the above equations obtains the final relationship between wing gust velocity and force

$$F_w = k_f \Delta v_o$$

where

$$k_f = 2k_F \alpha v_n^2$$

For the 22.5m, kF = 548.70N/(m/s)^2 , and α = 0.20, thus

$$kf = 219.48\ v_m^2$$

with velocity in m/s, and force in N. For 3.15m/s wind the force gain is kf = 1604.59 Ns/m.

5.2.2 Model of Wind Torque Acting at the Drives

In this model the wind torque disturbance is added to the drive torque. It is a time function, *Tw (t)*, determined from the velocity gusts *Δvo*. This model is shown in Fig. 5.2.1, where the white noise is applied to the Davenport filter. The filter output is the velocity gust *Δvo* of unit standard deviation, which is consequently scaled to obtain wind torque that is added to the antenna drive torque. In this model the Davenport filter is identical with the filter presented in previous section. The scaling factor, *kt*, from the velocity to torque is obtained from the wind quadratic law for torques *(Tn)* and for steady wind speed *vm*:

$$T_n = k_T v_m^2 \qquad (1)$$

The constant kT in this equation is particular for an antenna, antenna elevation position, antenna site, terrain profile, and wind direction. It is determined as follows. First, the wind torque depends on wind pressure, and the pressure–torque relation ship was determined experimentally in wind tunnels and in field tests

$$T_n = c_t A D p_n$$

where D is the antenna dish diameter (m), and A is the antenna dish frontal area, A= πD2/4 (m^2), and *ct* is a dimensionless torque coefficient. This coefficient depends on the wind direction and the antenna elevation position, and varies from–0.05 to 0.25.

Next note that the dynamic pressure of wind, p, depends of wind velocity, simi larly to the torque,

$$p_n = \alpha_p v_m^2$$

where αp is the static air density, αp = 0.6126 Ns^2/m^4 (pressure is in N/m2, and velocity is in m/s).

$$T_n = c_t \alpha_p A D v_m^2$$

Comparing above the quadratic law coefficient kT is obtained

$$k_T = c_t \alpha_p A D = c_t \alpha_p \frac{\pi D^3}{4}$$

The torque equation (1) is valid for steady wind only. However, the wind gust torque can be derived from it, using a linear expansion (again, the gust part of the wind is between 10% and 20% of the steady wind that justifies the linearization). Indeed, it is determined from the velocity gusts Δvo by linearizing equation (1), which gives

$$\Delta T = 2 k_T v_m \Delta v$$

The above represents the antenna axis torque. The wind torque at the pinion axis, Tw, is the axis torque ΔT divided by the axis-to-pinion ratio Np; thus the pinion torque is also proportional to the velocity

$$T_w = \frac{\Delta T}{N_p} = \frac{2 k_T v_n}{N_p} \Delta v$$

In simulations the wind speed model with unit standard deviation Δvo(t) is used. It was previously derived that Δvo(t) is related to an arbitrarily scaled wind speed as in equation . Hence equation is

$$T_w = k_t \Delta v_o$$

where

$$k_t = \frac{2k_T \alpha}{N_p} v_m^2$$

NOTE : These all calculations are been performed in MATLAB's Code, refer to the Source Codes in Appendix A, 9. GMRT wind force & torque simulation using Davenport

This model simplifies the disturbance by representing the effect of **wind gusts as a time-varying torque (Tw)** applied at the antenna's drive mechanisms. This torque is summed with the command torque from the controller before driving the antenna's structure. This approach is an effective approximation for analyses where the antenna structure and its drive systems are modeled as separate entities. It is widely used because it provides a good balance between physical realism and computational complexity, allowing for an adequate approximation of servo errors in windy conditions. The underlying principle remains the same: a random signal is shaped by a Davenport filter to create a wind velocity signal. This signal is then scaled by **a torque gain (kt)** to convert it into a torque disturbance.

5.2.3 Model of Wind at the Velocity Input

This third model is designed for situations where an integrated model of the antenna and its drives is used, often derived from system identification tests. In this case, the wind disturbance cannot be injected internally at the drive torque, as it is an internal variable in the integrated model. Instead, the disturbance is modeled as a signal at the velocity input of the system.

To ensure this model accurately reflects the physics of wind-induced torque, the wind torque signal from the Davenport filter must be further processed. This is done using an additional filter (Filter F), which effectively converts the torque disturbance into an equivalent velocity command signal. This filtered signal is then summed with the main command signal from the controller, acting as an external perturbation on the system's velocity input. This method is crucial for validating models derived from experimental data and is directly applicable for designing and testing digital controllers in a simulation environment. The project's transfer function modeling approach, as outlined in the abstract, is consistent with this method.

5.3 Wind Loading Simulation Results

This section presents the detailed results of the wind load simulation for the GMRT antenna, serving as a critical bridge between the project's theoretical framework and its practical application. The simulation, executed using the MATLAB script's , successfully quantifies the mechanical loads imposed on the antenna under both steady-state and dynamic wind conditions. This data is essential for validating the wind models developed in previous sections and for informing the design of a robust control system.

5.3.1 Steady-State Wind Loads

The steady-state analysis focuses on the persistent, average forces and torques that the antenna must counteract. This is a foundational step in understanding the baseline loads on the structure.

- **Simulation Parameters and Geometric Foundation**: The simulation was configured with key physical and environmental parameters to accurately represent the GMRT antenna and its operating environment. These included **a dish diameter of D=45.00 m, which yields a significant projected area of A=1590.43 m2**. The air density was set to **ρ=1.15 kg/m3**, and the **mean wind speed was established at vn=3.15 m/s**. These values form the basis for all subsequent load calculations.
- **Quantification of Mean Force and Torque**: The **steady force coefficient (kF)**, a measure of aerodynamic resistance, was computed as 548.70 N/(m/s)2. Applying this coefficient to the mean wind speed resulted in a substantial mean force of Fm=5444.46 N (5.44 kN). This force generates a steady torque (Tm) of 49000.17 N · m (49.00 kN·m) about the antenna's rotation axis. This torque, determined by multiplying the force by an effective moment arm of 9.00 m, represents the constant load that the drive motors must overcome to maintain the antenna's position. This part of the simulation confirms the significant mechanical power required of the drive system.

The raw output of these calculations, which can be presented as a table or a clear figure, provides a tangible summary of the steady-state load analysis.

5.3.2 Dynamic Wind Gusts and Loading

The dynamic part of the simulation addresses the more challenging, unpredictable loads caused by wind gusts. The accuracy of this simulation depends heavily on a stable and representative wind turbulence model.

- **Model Fidelity and Filter Validation:** A critical aspect of the simulation's methodology was the generation of a realistic wind gust signal. The script initially attempted to use a user-provided discrete filter but, as the output explicitly warns, this filter was identified as unstable.

This is a key finding, as an unstable filter would produce non-physical, unbounded outputs. The script's robust design then seamlessly fell back to a stable 4th-order Butterworth low-pass filter, ensuring the integrity of the simulated wind gust data. This methodological detail is important to include in the report to highlight the quality control of your simulation process.

- **Statistical Analysis of Dynamic Loads**: Using the stable Butterworth filter, the simulation produced a wind gust signal that was used to calculate the dynamic forces and torques. The **root-mean-square (RMS)** values are used to statistically characterize the magnitude of these fluctuating loads. The **gust force (kf)** was calculated to have **an RMS value of 1604.59 N (1.60 kN)**, and the **resulting dynamic gust torque (Tw,RMS) had an RMS value of 14441.35 N · m (14.44 kN·m)**. The simulation's internal diagnostics verified that the standard deviations of the generated force and torque time-series data matched these calculated RMS values, confirming the model's accuracy.

- **Visualizing Dynamic Effects**: To effectively communicate these dynamic effects, graphical representations are indispensable. The time-series plots generated by the simulation, showing the total force and torque over time, will visually demonstrate the random, rapid fluctuations around the mean values. Additionally, a histogram of the gust force component and a power spectral density (PSD) plot of the normalized gust signal are crucial diagnostic tools that should be included. They provide insight into the statistical distribution of the loads and the frequency content of the simulated turbulence.

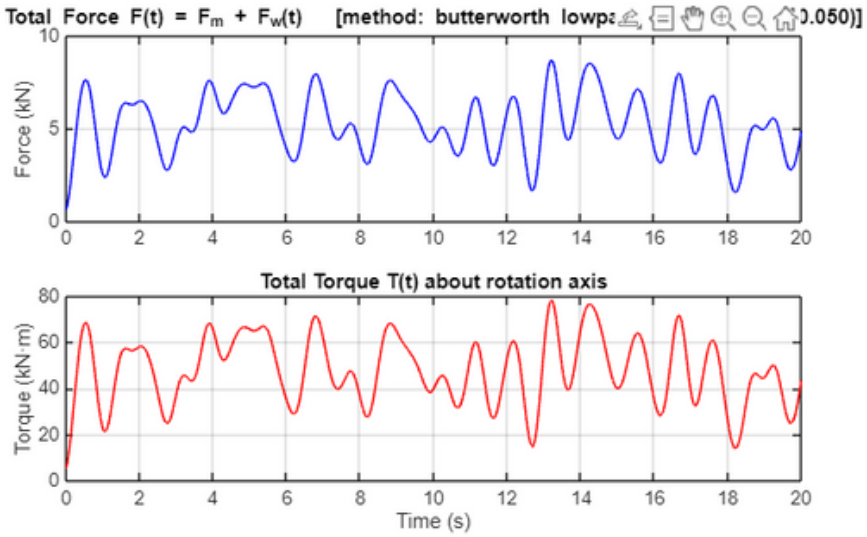

Figure 5.3.1 - Time-series of total force and total torque on the GMRT antenna during a simulated wind event. The fluctuations are driven by the Davenport-based wind gust model, is appropriate. [23]

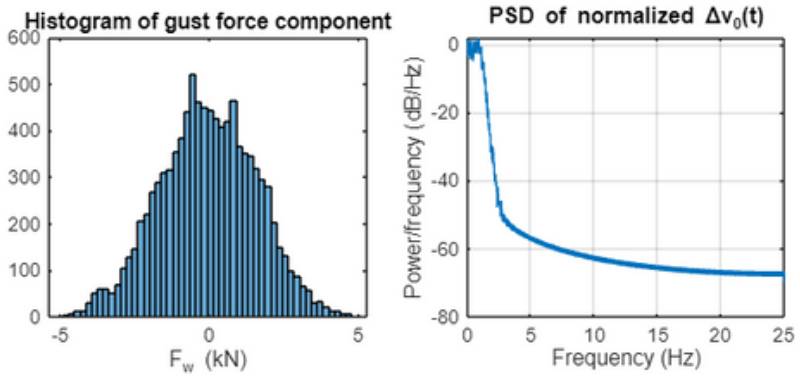

Figure 5.3.2 - Diagnostic plots showing the histogram of the simulated gust force and the power spectral density (PSD) of the normalized wind gust signal.[24]

NOTE : These all analysis and models have been simulated in MATLAB's Code, refer to the Source Codes in Appendix A, 9. GMRT wind force & torque simulation using Davenport, to obtain the force and torque outputs.

Chapter 6 : Conclusion and Future Scope

6.1 Conclusion

This project successfully addressed the critical challenge of modeling wind-induced disturbances for the **Giant Metrewave Radio Telescope (GMRT)**. Using site-specific wind data, **a mean wind speed of 3.15 m/s and a standard deviation of 1.09 m/s** were determined as representative parameters for the operating environment. A **4th-order transfer function was derived to replicate the Power Spectral Density (PSD)** characteristics of wind turbulence, and it was discretized using the Tustin method for digital simulation. The fitted model was validated through PSD and Bode plot comparisons, confirming strong agreement with the reference Davenport spectrum and demonstrating its suitability as a control-oriented, discrete-time representation of real wind gust behavior.

The validated model was then integrated into a comprehensive wind load simulation, which **quantified both steady-state and dynamic mechanical loads** on the GMRT antenna. The analysis revealed **a mean force of 5.44 kN, a steady torque of 49.00 kN·m, and a root-mean-square (RMS) gust torque of 14.44 kN·m**, highlighting the rapid oscillations that the antenna's control system must counteract. The close match between RMS values and the standard deviations of simulated signals confirms the model's high fidelity. Overall, this work provides a robust simulation framework and detailed understanding of wind-induced loads, forming a crucial foundation for designing advanced control strategies to maintain the antenna's sub-millimeter pointing precision.

6.2 Future Scope

The successful development and validation of the wind disturbance model establish a strong foundation for several avenues of future work aimed at enhancing the GMRT's performance and operational robustness.

- **Advanced Control System Design**: The most logical and impactful next step is to leverage this validated model to design and implement advanced control strategies. Traditional PID controllers often struggle with the complex, dynamic nature of wind loads. Future work should explore modern control techniques such as Linear-Quadratic Gaussian (LQG) and H∞ controllers. An LQG controller, for instance, could be designed to minimize the antenna's pointing error by optimally balancing control effort against performance criteria.

This would involve integrating the wind model as a process disturbance and designing a state estimator, like a Kalman filter, to accurately estimate the antenna's true position and velocity in real-time, even in the presence of sensor noise and wind-induced motion.

- **Refining the Wind Model**: While the current model is effective, it can be further refined. Future efforts could involve developing higher-order filters to more accurately capture the nuances of the wind PSD, particularly at higher frequencies where structural resonances may be excited. This could lead to a more precise and predictive simulation model.

- **Incorporating Non-Linear Dynamics**: The current simulation primarily focuses on linear dynamics. Real-world antenna systems, however, are affected by non-linearities such as drive backlash, stiction, and motor saturation. Future work should integrate these non-linear elements into the simulation environment to create a more realistic digital twin of the antenna system. The control strategies developed with these non-linearities in mind, such as anti-windup techniques and dither signals, are expected to yield more robust performance in practice.

- **Multi-Axis and Multi-Domain Modeling**: This project focused on single-axis disturbances. Expanding the model to a multi-axis system would allow for the investigation of cross-coupling effects between the azimuth and elevation drives.

Additionally, integrating thermal models that simulate temperature-induced structural deformations would provide a holistic view of all significant environmental disturbances, leading to a more comprehensive and resilient control system design. This expanded framework would enable the development of integrated control solutions that simultaneously address aerodynamic, mechanical, and thermal challenges.

NOTE : That under this project only Force and Torque with Davenport filter in it were simulated, and 4th Order Transfer Function was considered.

Appendix

A : Source Codes

1. **Mean_Standard deviation_PSD_calculation_through wind DATA**

```
clear; close all;

%% === Step 1: Load and Clean Wind Data ===
A = importdata('31Oct2024.dat');
wind_speed = A.data(:,1); % Assuming wind speed in column 1

% Remove NaNs and non-physical values
valid_idx = ~isnan(wind_speed) & wind_speed > 0;
wind_clean = wind_speed(valid_idx);

% Despike using median filter
wind_despiked = medfilt1(wind_clean, 5);

% Remove ±3σ outliers
mean_wind = mean(wind_despiked);
std_wind = std(wind_despiked);
upper_limit = mean_wind + 3 * std_wind;
lower_limit = mean_wind - 3 * std_wind;
wind_final = wind_despiked(wind_despiked >= lower_limit & wind_despiked <=
upper_limit);

%% === Step 2: Compute Statistics ===
mean_clean = mean(wind_final);
std_clean = std(wind_final);
cov_rw_cw = cov(wind_clean(1:length(wind_final)), wind_final);
cov_value = cov_rw_cw(1,2);

%% === Step 3: Plot Raw vs Cleaned and Statistics ===
figure;
subplot(2,1,1);
axis off;
text(0, 0.8, sprintf('\\bfMean of Cleaned Wind Data: %.4f m/s', mean_clean), 'FontSize', 12);
text(0, 0.5, sprintf('\\bfStandard Deviation: %.4f m/s', std_clean), 'FontSize', 12);
text(0, 0.2, sprintf('\\bfCovariance (Raw vs Cleaned): %.4f', cov_value), 'FontSize', 12);
title('Wind Data Statistics');
```

```
subplot(2,1,2);
plot(wind_clean, 'b'); hold on;
plot(find(wind_despiked >= lower_limit & wind_despiked <= upper_limit), wind_final, 'r');
grid on;
xlabel('Time Index');
ylabel('Wind Speed [m/s]');
legend('Raw Wind', 'Cleaned Wind');
title('Raw vs Cleaned Wind Speed');


%% === Step 4: PSD using FFT ===
Fs = 10; % Sampling frequency in Hz
N = length(wind_final);
Y = fft(wind_final);
P2 = abs(Y/N).^2;
P1 = P2(1:floor(N/2)+1);
P1(2:end-1) = 2*P1(2:end-1);
f_fft = Fs*(0:floor(N/2))/N; % Linear frequency in Hz
P1_dB = 10*log10(P1);


figure;
plot(f_fft, P1_dB, 'b', 'LineWidth', 1.5);
grid on;
xlabel('Frequency (Hz)');
ylabel('Power/Frequency (dB/Hz)');
title('One-Sided PSD (FFT Method)');
xlim([0 0.5]);
ylim([-160 20]);


%% === Step 5: Davenport Spectrum (Theoretical) ===
U = mean_clean; % Mean wind speed from cleaned data
z = 23.16; % Antenna height (m)
z0 = 0.03; % Roughness length (m)
b = 600 / (pi * U);
k = (1 / (2.5 * log(z/z0)))^2;
% Frequency range and Davenport formula
f_theory = linspace(0.001, 3, 1500);
w_theory = 2 * pi * f_theory;
Sv = (4800 * U * b * k) ./ ((1 + (b^2) * w_theory.^2).^(4/3));
Sv_dB = 10*log10(Sv);
```

```
figure;
loglog(f_theory, Sv, 'r', 'LineWidth', 2);
grid on;
xlabel('Frequency (Hz)');
ylabel('PSD (m^2/s)');
title('Davenport Wind Velocity Spectrum');

%% === Step 6: PSD using Welch ===
window = hamming(256);
noverlap = 128;
nfft = 512;
[pxx, f_welch] = pwelch(wind_final, window, noverlap, nfft, Fs);
pxx_dB = 10*log10(pxx);
figure;
plot(f_welch, pxx_dB, 'LineWidth', 2);
grid on;
xlabel('Frequency [Hz]');
ylabel('PSD [dB/Hz]');
title('Power Spectral Density using Welch Method');

%% === Step 7: Compare Welch PSD with Davenport ===
w_welch = 2 * pi * f_welch;
Sv_welch = (4800 * U * b * k) ./ ((1 + (b^2) * w_welch.^2).^(4/3));
Sv_welch_dB = 10*log10(Sv_welch);
figure;
plot(f_welch, pxx_dB, 'b', 'LineWidth', 2); hold on;
plot(f_welch, Sv_welch_dB, 'r--', 'LineWidth', 2);
xlabel('Frequency [Hz]');
ylabel('PSD [dB/Hz]');
title('Welch PSD vs Davenport Spectrum');
legend('Welch PSD','Davenport Spectrum');
grid on;

%% === Step 8: Compare FFT PSD with Davenport (Angular Frequency) ===
w_fft = 2 * pi * f_fft;
Sv_interp = interp1(w_theory, Sv_dB, w_fft, 'linear', 'extrap');
figure;
plot(w_fft, P1_dB, 'b', 'LineWidth', 2); hold on;
plot(w_fft, Sv_interp, 'r--', 'LineWidth', 2);
xlabel('Angular Frequency [rad/s]');
ylabel('PSD [dB/(rad/s)]');
title('FFT PSD vs Davenport Spectrum');
legend('FFT PSD','Davenport Spectrum');
grid on;
```

2. PSD_Compare_on_different_scales (dB and log log)

```
clc; clear; close all;

%% ---------- Step 1: Load and Clean Wind Data ---------- %%
A = importdata('31Oct2024.dat');
wind_speed = A.data(:,1);

% Remove NaN and bad values
valid_idx = ~isnan(wind_speed) & wind_speed > 0;
wind_clean = wind_speed(valid_idx);

% Median filter (despiking)
wind_despiked = medfilt1(wind_clean, 5);

% Remove ±3σ outliers
mean_wind = mean(wind_despiked);
std_wind = std(wind_despiked);
upper_limit = mean_wind + 3 * std_wind;
lower_limit = mean_wind - 3 * std_wind;
wind_final = wind_despiked(wind_despiked >= lower_limit & wind_despiked <= upper_limit);

%% ---------- Step 2: FFT-based PSD ---------- %%
Fs = 10; % Sampling frequency (Hz)
N = length(wind_final);
Y = fft(wind_final);
P2 = abs(Y/N).^2; % Two-sided power
P1 = P2(1:floor(N/2)+1); % One-sided power
P1(2:end-1) = 2*P1(2:end-1); % Energy correction
f_fft = Fs*(0:floor(N/2))/N; % Frequency vector (Hz)

%% ---------- Step 3: Welch-based PSD ---------- %%
window = hamming(256);
noverlap = 128;
nfft = 512;
[pxx_welch, f_welch] = pwelch(wind_final, window, noverlap, nfft, Fs);

%% ---------- Step 4: Davenport Spectrum ---------- %%
U = mean_wind; % Mean wind speed from cleaned data
z = 23.16; % Height (m)
z0 = 0.03; % Roughness length (m)
b = 600 / (pi * U);
k = (1 / (2.5 * log(z / z0)))^2;
```

```
% Match Davenport freq vector to FFT/Welch resolution
f_dav = f_fft;
w_dav = 2*pi*f_dav;
Sv_dav = (4800 * U * b * k) ./ ((1 + (b^2) * w_dav.^2).^(4/3));

%% ---------- Step 5: Ensure Units Match ---------- %%
% FFT and Welch already give linear PSD in [m^2/s^2/Hz]
% Davenport is also in [m^2/s^2/Hz] now (linear scale)
% For dB plot: 10*log10(PSD)


%% ---------- Step 6: Plot Both Scales ---------- %%
figure('Units','normalized','Position',[0.1 0.1 0.8 0.4]);
% ---- Left: dB scale ----
subplot(1,2,1);
plot(f_fft, 10*log10(P1), 'g'); hold on;
plot(f_welch, 10*log10(pxx_welch), 'b', 'LineWidth', 1.5);
plot(f_dav, 10*log10(Sv_dav), 'r--', 'LineWidth', 1.5);
grid on;
xlabel('Frequency [Hz]');
ylabel('PSD [dB/Hz]');
title('PSD Comparison (dB scale)');
legend('FFT','Welch','Davenport');
xlim([min(f_fft) max(f_fft)]);

% ---- Right: log–log linear PSD ----%
subplot(1,2,2);
loglog(f_fft, P1, 'g'); hold on;
loglog(f_welch, pxx_welch, 'b', 'LineWidth', 1.5);
loglog(f_dav, Sv_dav, 'r--', 'LineWidth', 1.5);
grid on;
xlabel('Frequency [Hz]');
ylabel('PSD [m^2/s^2/Hz]');
title('PSD Comparison (Log–Log)');
legend('FFT','Welch','Davenport');
xlim([min(f_fft(f_fft>0)) max(f_fft)]);
```

3. TF(given)_vs_Davenport_PSD

```
clc; clear; close all;

%% --- Davenport Spectrum parameters ---
U = 3.15; % mean wind speed (m/s)
z = 23.16; % antenna height (m)
z0 = 0.03; % roughness length (m)

b = 600 / (pi * U);
k = (1 / (2.5 * log(z / z0)))^2;

f = linspace(0.001, 3, 2000); % frequency vector (Hz)
w = 2*pi*f; % angular frequency (rad/s)

Sv = (4800 * U * b * k) ./ ((1 + (b^2) * w.^2).^(4/3)); % Davenport PSD

%% --- Continuous-time TF H_fit(s) ---
num_c = [0.01197 0 -40.03 0 461.3]; % coefficients for s^4 ... s^0
den_c = [1 0 -482.7 0 65.13]; % coefficients for s^4 ... s^0

% Display transfer function in MATLAB-friendly format
H_fit = tf(num_c, den_c);
disp('Continuous-time transfer function H_fit(s):');
H_fit

% Compute frequency response of analog TF at frequencies w
Hjw = freqs(num_c, den_c, w); % Hjw is complex frequency response H(jw)

% Theoretical output PSD for unit-spectral-level white noise: S_out(w) = |H(jw)|^2
PSD_H = abs(Hjw).^2;

%% --- Option A: Raw comparison ---
figure('Name','Raw PSD comparison');
loglog(f, Sv, 'r-', 'LineWidth', 1.8); hold on;
loglog(f, PSD_H, 'b--', 'LineWidth', 1.4);
grid on;
xlabel('Frequency (Hz)');
ylabel('PSD');
title('Raw Comparison: Davenport (red) vs |H(j\omega)|^2 (blue)');
legend('Davenport Spectrum','|H(j\omega)|^2 (unit input)');
```

```
%% --- Option B: Normalized comparison ---
PSD_H_norm = PSD_H / max(PSD_H);
Sv_norm = Sv / max(Sv);
figure('Name','Normalized PSD comparison');
loglog(f, Sv_norm, 'r-', 'LineWidth', 1.8); hold on;
loglog(f, PSD_H_norm, 'b--', 'LineWidth', 1.4);
grid on;
xlabel('Frequency (Hz)');
ylabel('Normalized PSD (peak = 1)');
title('Normalized PSD Comparison (shape comparison)');
legend('Davenport (normalized)','|H(j\omega)|^2 (normalized)');


%% --- Option C: Scaled TF PSD to match Davenport peak ---
scale_factor = max(Sv) / max(PSD_H);
PSD_H_scaled = PSD_H * scale_factor;
figure('Name','Scaled PSD comparison');
loglog(f, Sv, 'r-', 'LineWidth', 1.8); hold on;
loglog(f, PSD_H_scaled, 'b--', 'LineWidth', 1.4);
grid on;
xlabel('Frequency (Hz)');
ylabel('PSD (same units)');
title('Davenport vs Scaled |H(j\omega)|^2 (peaks matched)');
legend('Davenport','Scaled |H(j\omega)|^2');


%% --- Display peak info ---
[~, idx_Sv_peak] = max(Sv);
f_Sv_peak = f(idx_Sv_peak);
Sv_peak = Sv(idx_Sv_peak);
[~, idx_H_peak] = max(PSD_H);
f_H_peak = f(idx_H_peak);
H_peak = PSD_H(idx_H_peak);
fprintf('Davenport peak: f = %.4f Hz, value = %.4e\n', f_Sv_peak, Sv_peak);
fprintf('TF peak: f = %.4f Hz, value = %.4e\n', f_H_peak, H_peak);
fprintf('Scale factor used (to match peaks): %.4e\n', scale_factor);
```

4. Comparision_DiscreteTFs_Bode_Plots

```
%% Compare Bode (freq response) of two discrete TFs
clc; clear; close all;

% ----------------- User settings -----------------
Fs = 10; % sampling frequency [Hz] (set to your data Fs)
Ts = 1/Fs;

nfft = 4096; % frequency samples for plotting

% ----------------- Original discrete TF (z^-1 form) -----------------
b_orig = [0.02679, -0.05951, 0.01895, 0.03348, -0.01971];
a_orig = [1, -3.952, 5.857, -3.857, 0.9523];

% Create discrete TF using z^-1 variable (sample time Ts)
H_orig = tf(b_orig, a_orig, Ts, 'variable','z^-1');

% ----------------- Fitted continuous TF -> discretize with Tustin -----------------
% (use your fitted continuous TF coefficients here)
% Continuous H_fit(s) = (0.01197 s^4 - 40.03 s^2 + 461.3) / (s^4 - 482.7 s^2 + 65.13)
s = tf('s');
Hc_fit = (0.01197*s^4 - 40.03*s^2 + 461.3) / (s^4 - 482.7*s^2 + 65.13);

% Discretize using bilinear (Tustin)
H_fit_d = c2d(Hc_fit, Ts, 'tustin');

% Get numerator/denominator as vectors (z^-1 form) for freqz if needed
[b_fit_d, a_fit_d] = tfdata(H_fit_d, 'v'); % returns in descending z powers but freqz accepts them

% ----------------- Frequency response (Hz axis) -----------------
% Use freqz on digital filter coefficients (freqz expects numerator/den denominator in z^-1 form)
% But tfdata with 'v' returns polynomials in z^-1 for discrete TFs created with 'z^-1' variable;
% if not, ensure you pass correct b,a to freqz.

% compute frequency vector (Hz)
[H1, w] = freqz(b_orig, a_orig, nfft, Fs); % returns frequency response at frequencies corresponding to Fs
[H2, ~] = freqz(b_fit_d, a_fit_d, nfft, Fs); % fitted discrete TF response

f = w; % w already in Hz because we used freqz(...,Fs)
```

```
%% --- Bode Plot (Magnitude and Phase) ---
figure('Name','Bode Plot of Continuous-Time Fitted TF','NumberTitle','off');
bode(Hc_fit, {1e-2, 1e2}); % frequency range from 0.01 to 100 rad/s
grid on;


% Improve formatting
title('Bode Magnitude and Phase Plots of H_{fit}(s)');
set(findall(gcf,'type','line'),'LineWidth',1.5); % thicker lines


% ----------------- Plot magnitude (dB) and phase (deg) -----------------
figure('Units','normalized','Position',[0.08 0.12 0.82 0.72]);
plot(f, 20*log10(abs(H1)+eps), 'b', 'LineWidth', 1.5); hold on;
plot(f, 20*log10(abs(H2)+eps), 'r--', 'LineWidth', 1.5);
xlim([0 Fs/2]);
ylabel('Magnitude (dB)');
title('Frequency Response Comparison (Discrete) — Magnitude');
legend('Original discrete TF','Fitted (Tustin) discrete TF','Location','best');
grid on;
```

5. DiscreteTF_Bode_PSD_Comparison

```
clc; clear; close all;

% Define continuous-time transfer function
s = tf('s');
H_fit = (0.01197*s^4 - 40.03*s^2 + 461.3) / ...
 (s^4 - 482.7*s^2 + 65.13);

% Sampling time (choose based on your data, e.g., 10 Hz → Ts = 0.1 s)
Ts = 0.1;

% Discretize using bilinear (Tustin) method
H_fit_d = c2d(H_fit, Ts, 'tustin');

% Display results
H_fit
H_fit_d

%% First discrete-time TF: H_fit_d
num1 = [0.413 0.1762 -1.402 0.1762 0.413];
den1 = [1 19.38 -40.79 19.38 1];
Ts = 0.1; % sample time in seconds
H1 = tf(num1, den1, Ts);

%% Second discrete-time TF: given one
num2 = [0.02679 -0.05951 0.01895 0.03348 -0.01971];
den2 = [1 -3.952 5.857 -3.857 0.9523];
H2 = tf(num2, den2, Ts);

%% Bode plot comparison
figure;
bode(H1, 'b', H2, 'r--');
grid on;
legend('H\_fit\_d', 'Given TF');
title('Bode Plot Comparison of Two Discrete-Time Systems');


%% Define transfer functions
num1 = [0.413 0.1762 -1.402 0.1762 0.413];
den1 = [1 19.38 -40.79 19.38 1];
```

```
num2 = [0.02679 -0.05951 0.01895 0.03348 -0.01971];
den2 = [1 -3.952 5.857 -3.857 0.9523];

Ts = 0.1; % sample time
%% Frequency vector for plotting
nfft = 1024;
[H1_w, w] = freqz(num1, den1, nfft, 1/Ts);
[H2_w, ~] = freqz(num2, den2, nfft, 1/Ts);


%% For white noise input with unit variance:
PSD1 = abs(H1_w).^2; % output PSD
PSD2 = abs(H2_w).^2;


%% Plot in dB/Hz
figure;
plot(w, 10*log10(PSD1), 'b', 'LineWidth', 1.5); hold on;
plot(w, 10*log10(PSD2), 'r--', 'LineWidth', 1.5);
grid on;
xlabel('Frequency (Hz)');
ylabel('PSD (dB/Hz)');
legend('H1', 'H2');
title('Theoretical Output PSD for Unit Variance White Noise Input');
```

6. PSD_Comparison_of_FFT_and_Welch

```
clc; clear; close all;

%% Transfer function coefficients
b = [0.02679, -0.05951, 0.01895, 0.03348, -0.01971];
a = [1, -3.952, 5.857, -3.857, 0.9523];

%% Sampling frequency
Fs = 10; % Hz (adjust as per your system)

%% Generate impulse response (long enough for PSD)
N = 2048; % Number of samples
imp = [1; zeros(N-1,1)]; % Unit impulse
h = filter(b, a, imp); % Impulse response

%% Remove any NaN or Inf (safety step)
h(~isfinite(h)) = 0;

%% ---- PSD via FFT ----
H_fft = fft(h, N);
PSD_fft = (abs(H_fft).^2) / Fs; % Power spectrum
f_fft = (0:N-1)*(Fs/N);

%% ---- PSD via Welch ----
[PSD_welch, f_welch] = pwelch(h, hamming(256), 128, 1024, Fs);

%% ---- Plot Results ----
figure;

subplot(2,1,1);
plot(f_fft(1:N/2), PSD_fft(1:N/2), 'b', 'LineWidth', 1.5);
xlabel('Frequency (Hz)'); ylabel('PSD (FFT)');
title('PSD using FFT');
grid on;

subplot(2,1,2);
plot(f_welch, PSD_welch, 'r', 'LineWidth', 1.5);
xlabel('Frequency (Hz)'); ylabel('PSD (Welch)');
title('PSD using Welch Method');
grid on;
```

7. Advanced_TF_Analysis_and_Comparison

```
clc; clear; close all;

%% --- Step 1: Load Wind Data ---
A = importdata('31Oct2024.dat');
wind_speed = A.data(:,1); % assuming wind speed in column 1

% Clean data: remove NaNs and negatives
valid_idx = ~isnan(wind_speed) & wind_speed > 0;
wind_clean = wind_speed(valid_idx);

fs = 10; % Sampling frequency in Hz

%% --- Step 2: Compute Realistic PSD from Data (Welch method) ---
[PSD_real, f_real] = pwelch(wind_clean - mean(wind_clean), ...
 hamming(1024), 512, 1024, fs, 'onesided');

%% --- Step 3: Davenport PSD Parameters ---
U_mean = 3.15; % Mean wind speed (m/s)
sigma_u = std(wind_clean); % Std deviation from real data
L = 120; % Turbulence length scale (m)
f = logspace(-3, 1, 200); % Frequency range (Hz)

% Davenport formula
Sv = (4 * sigma_u^2 * L / U_mean) ./ ...
 (1 + 6*f*L/U_mean).^(5/3);

%% --- Step 4: Transfer Function H(s) ---
s = tf('s');
H = (0.72*s^4 - 862.5*s^2 + 1.402e4) / ...
 (s^4 - 221*s^2 + 312.2);

% Compute |H(jw)|^2
w = 2*pi*f;
[mag, ~] = bode(H, w);
mag = squeeze(mag);
PSD_from_H = mag.^2; % For unit variance white noise input

%% --- Step 5: Find Best α to Fit PSD_from_H to Davenport over all f ---
alpha = (Sv(:)' * PSD_from_H(:)) / (PSD_from_H(:)' * PSD_from_H(:));
PSD_from_H_scaled = alpha * PSD_from_H;
```

```
%% --- Step 6: Show Transfer Functions ---
disp('Transfer Function for unit variance white noise input:')
H

% Scaling TF output by sqrt(alpha) so that its PSD matches best-fit scaling
H_scaled = sqrt(alpha) * H;
disp('Best-fit scaled Transfer Function:')
H_scaled


%% --- Step 7: Plot Comparison ---
figure;
loglog(f, Sv, 'b', 'LineWidth', 2); hold on;
loglog(f, PSD_from_H, 'r--', 'LineWidth', 1.5);
loglog(f, PSD_from_H_scaled, 'g', 'LineWidth', 2);
loglog(f_real, PSD_real, 'm', 'LineWidth', 1.5);
grid on;
xlabel('Frequency (Hz)');
ylabel('PSD (m^2/s^3)');
legend('Davenport PSD', ...
 'H(s) PSD (unit variance input)', ...
 'H(s) PSD (best fit scaled)', ...
 'Realistic PSD from data');
title('PSD Comparison: Davenport vs Transfer Function vs Real Data');


%% Continuous-time transfer function
num_ct = [0.1083 0 -129.7 0 2108]; % s^4 ... s^0
den_ct = [1 0 -221 0 312.2];
Hc = tf(num_ct, den_ct);
%% Choose sampling time
Ts = 0.01; % seconds (adjust as needed)
%% Discretize using bilinear (Tustin) method
Hd = c2d(Hc, Ts, 'tustin');
%% Display discrete transfer function
disp('Discrete-time transfer function (bilinear method):');
Hd

%% Extract numerator, denominator
[num_d, den_d] = tfdata(Hd, 'v');
%% Poles and zeros
```

```
[z_d, p_d, k_d] = zpkdata(Hd, 'v');
disp('Discrete-time poles:');
disp(p_d);
disp('Discrete-time zeros:');
disp(z_d);

%% Convert to state-space
sysd = ss(Hd);
A = sysd.A;
B = sysd.B;
C = sysd.C;
D = sysd.D;


%% Stability check (all poles inside unit circle)
isStable = all(abs(p_d) < 1);
disp(['Is system stable? ', string(isStable)]);

%% Output variance for unit-variance white noise input
% Only valid if system is stable
if isStable

 % Solve discrete Lyapunov equation: A P A' - P + B B' = 0
 P = dlyap(A, B*B');
 sigma_y2 = C * P * C' + D*D';
 disp('Output variance for unit-variance white noise input:');
 disp(sigma_y2);
else
 warning('System is unstable, output variance not computed.');
end
%% First TF
num1 = [0.02679 -0.05951 0.01895 0.03348 -0.01971];
den1 = [1 -3.952 5.857 -3.857 0.9523];
Ts1 = 0.01;
Hd1 = tf(num1, den1, Ts1);

%% Second TF
num2 = [0.1056 -0.4356 0.6599 -0.4356 0.1056];
den2 = [1 -4.022 6.044 -4.022 1];
Ts2 = 0.01;
Hd2 = tf(num2, den2, Ts2);
```

```
%% Frequency response (magnitude squared = PSD for white noise input)
nfft = 2048; % number of frequency points
[H1, w] = freqz(num1, den1, nfft, 1/Ts1); % w in Hz
H2 = freqz(num2, den2, nfft, 1/Ts2);
PSD1 = abs(H1).^2; % output PSD
PSD2 = abs(H2).^2;


%% Plot
figure;
plot(w, 10*log10(PSD1), 'LineWidth', 1.5); hold on;
plot(w, 10*log10(PSD2), 'LineWidth', 1.5);
grid on;
xlabel('Frequency (Hz)');
ylabel('PSD magnitude (dB)');
title('Output PSD for Unit-Variance White Noise Input');
legend('TF1', 'TF2');
```

8. White_Noise_TF_PSD_Analysis

```
clc; clear; close all;

%% First TF
num1 = [0.02679 -0.05951 0.01895 0.03348 -0.01971];
den1 = [1 -3.952 5.857 -3.857 0.9523];
Ts1 = 0.01;
Hd1 = tf(num1, den1, Ts1);

%% Second TF
num2 = [0.1056 -0.4356 0.6599 -0.4356 0.1056];
den2 = [1 -4.022 6.044 -4.022 1];
Ts2 = 0.01;
Hd2 = tf(num2, den2, Ts2);

%% Frequency response (magnitude squared = PSD for white noise input)
nfft = 2048;
[H1, w] = freqz(num1, den1, nfft, 1/Ts1); % w in Hz
H2 = freqz(num2, den2, nfft, 1/Ts2);

PSD1 = abs(H1).^2; % output PSD
PSD2 = abs(H2).^2;

%% Generate white noise and compute its PSD
N = 100000; % number of samples
white_noise = randn(N,1); % unit variance white noise

[PSD_white, f_white] = pwelch(white_noise, hamming(1024), 512, nfft, 1/Ts1);

%% Plot all three
figure;
plot(w, 10*log10(PSD1), 'LineWidth', 1.5); hold on;
plot(w, 10*log10(PSD2), 'LineWidth', 1.5);
plot(f_white, 10*log10(PSD_white), 'k--', 'LineWidth', 1);
grid on;
xlabel('Frequency (Hz)');
ylabel('PSD (dB)');
title('Output PSD for Unit-Variance White Noise Input');
legend('TF1 Output PSD', 'TF2 Output PSD', 'White Noise Input PSD');

%% Define transfer functions
num1 = [0.413 0.1762 -1.402 0.1762 0.413];
den1 = [1 19.38 -40.79 19.38 1];
```

```
num2 = [0.02679 -0.05951 0.01895 0.03348 -0.01971];
den2 = [1 -3.952 5.857 -3.857 0.9523];
Ts = 0.1; % sample time

%% Frequency vector for plotting
nfft = 1024;
[H1_w, w] = freqz(num1, den1, nfft, 1/Ts);
[H2_w, ~] = freqz(num2, den2, nfft, 1/Ts);
%% For white noise input with unit variance:
PSD1 = abs(H1_w).^2; % output PSD
PSD2 = abs(H2_w).^2;

%% Plot theoretical PSDs for H1 and H2
figure;
plot(w, 10*log10(PSD1), 'b', 'LineWidth', 1.5); hold on;
plot(w, 10*log10(PSD2), 'r--', 'LineWidth', 1.5);
grid on;
xlabel('Frequency (Hz)');
ylabel('PSD (dB/Hz)');
legend('H1', 'H2');
title('Theoretical Output PSD for Unit Variance White Noise Input');

%% Generate and display white noise
N = 2000; % number of samples
rng(1); % reproducible random numbers
white_noise = randn(N,1); % mean 0, variance 1

% Check variance
fprintf('White noise variance: %.4f\n', var(white_noise));

%% Plot white noise in time domain
figure;
plot((0:N-1)*Ts, white_noise, 'k');
xlabel('Time (s)');
ylabel('Amplitude');
title('Generated Unit Variance White Noise');
grid on;
%% Plot PSD of the generated white noise
[PSD_in, f_in] = pwelch(white_noise, hamming(256), 128, 1024, 1/Ts);
figure;
plot(f_in, 10*log10(PSD_in), 'm', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('PSD (dB/Hz)');
title('Measured PSD of Input White Noise');
grid on;
```

9. GMRT wind force & torque simulation using Davenport

```
%% gmrt_wind_full.m
% Full GMRT wind force & torque simulation using Davenport (or fallback) filter

clear; close all; clc

% ----------------------
% USER INPUTS
% ----------------------
D = 45; % dish diameter (m)
R = D/2; % radius (m)
rho = 1.15; % air density (kg/m^3) - Khodad approx
Cd = 0.60; % mesh drag coefficient (dimensionless)
vn = 3.15; % mean wind speed (m/s)
z = 23.16; % antenna height (m)
z0 = 0.03; % terrain roughness (m)
rcp_fac = 0.4; % r_cp factor (fraction of radius)
rcp = rcp_fac * R; % effective moment arm (m)

use_user_filter = true; % true: try to use provided Davenport filter coefficients
rng_seed = 1; % random seed for reproducibility (set [] to disable)

% Provided discrete "Davenport" coefficients (user-supplied)
b_user = [0.413 0.1762 -1.402 0.1762 0.413]; % numerator (b)
a_user = [1 19.38 -40.79 19.38 1]; % denominator (a)

% ----------------------
% DERIVED / PHYSICAL QUANTITIES
% ----------------------
A = pi * D^2 / 4; % projected area (m^2)
kF = 0.5 * rho * Cd * A; % steady force coefficient (N/(m/s)^2)

kappa = 1 / ( (2.5 * log(z / z0))^2 );
alpha = sqrt(6 * kappa);

Fm = kF * vn^2; % steady (mean) force (N)
kf = 2 * kF * alpha * vn^2; % gust-force gain (N) (RMS of gust force when delta_v0 is unit
std)

Tm = Fm * rcp; % steady torque (N*m)
% Tw_RMS = kf * rcp; % gust torque RMS (N*m) % printed later
```

```
% Print summary
fprintf('\n--- GMRT Mesh Dish Wind Load ---\n');
fprintf('Diameter D = %.2f m, Radius R = %.2f m\n', D, R);
fprintf('Projected area A = %.2f m^2\n', A);
fprintf('k_F (steady coeff) = %.2f N/(m/s)^2\n', kF);
fprintf('F_m (steady force) = %.2f N (%.2f kN)\n', Fm, Fm/1e3);
fprintf('k_f (gust gain RMS) = %.2f N (%.2f kN)\n', kf, kf/1e3);
fprintf('T_m (steady torque) = %.2f N*m (%.2f kN*m)\n\n', Tm, Tm/1e3);

% ----------------------
% SIMULATION / TIME SETTINGS
% ----------------------
Ts = 0.02; % sampling time (s)
Tdur = 200; % duration (s)
N = round(Tdur / Ts);
t = (0:N-1)' * Ts;
% seed RNG if requested
if ~isempty(rng_seed)
 rng(rng_seed);
end

% ----------------------
% CREATE SHAPED UNIT-STANDARD GUST SERIES delta_v0(t)
% Try user filter if requested & stable, else fallback to Butterworth shaping
% ----------------------
wn = randn(N,1); % white noise, unit variance
use_fallback = false;
if use_user_filter
 % Check stability of user filter
 poles = roots(a_user);
 if all(abs(poles) < 1)
 % stable: use zero-phase filtering to avoid phase distortion
 try
 delta_v0 = filtfilt(b_user, a_user, wn);
 method_used = 'user discrete filter (filtfilt)';
 % sanity check on output
 if ~isfinite(std(delta_v0)) || std(delta_v0) == 0
 warning('User filter produced invalid output; switching to fallback shaping.');
 use_fallback = true;
 end
 catch ME
 warning('Error applying user filter: %s\nSwitching to fallback shaping.');
 use_fallback = true;
 end
```

```
else
 warning('User filter is unstable (poles outside unit circle). Using fallback shaping.');
 use_fallback = true;
 end
else
 use_fallback = true;
end
if use_fallback
  % Fallback: 4th-order lowpass Butterworth (normalized cutoff chosen to give low-freq
energy)
 ord = 4;
 Wn = 0.05; % normalized cutoff (0-1, 1 = Nyquist). adjust if needed.
 [b_but,a_but] = butter(ord, Wn);
 delta_v0 = filtfilt(b_but, a_but, wn);
 method_used = sprintf('butterworth lowpass (order %d, Wn=%.3f)', ord, Wn);
end
% Normalize to unit standard deviation safely
sigma = std(delta_v0);
if sigma > 0 && isfinite(sigma)
 delta_v0 = delta_v0 / sigma;
else
 error('Filtered gust signal has zero or invalid standard deviation. Check filter/coefs.');
end
% ----------------------
% COMPUTE FORCE & TORQUE TIME SERIES
% ----------------------
F_w_t = kf * delta_v0; % gust force component (N)
F_t = Fm + F_w_t; % total force (N)
T_t = F_t * rcp; % total torque about rotation axis (N*m)
% ----------------------
% PLOTTING
% ----------------------
figure('Name','Gust Force & Torque Time Series','NumberTitle','off','Position',[200 200 900
520]);
subplot(2,1,1);
plot(t, F_t/1e3, 'b'); grid on;
ylabel('Force (kN)');
title(sprintf('Total Force F(t) = F_m + F_w(t) [method: %s]', method_used));
xlim([0 min(Tdur,20)]); % show first 20 seconds for clarity; change as desired
subplot(2,1,2);
plot(t, T_t/1e3, 'r'); grid on;
ylabel('Torque (kN·m)');
xlabel('Time (s)');
title('Total Torque T(t) about rotation axis');
```

```
xlim([0 min(Tdur,20)]);
% ----------------------
% SUMMARY CHECKS / STATS
% ----------------------
fprintf('Filter method used: %s\n', method_used);
fprintf('std(F_w) = %.2f N (should be ~= k_f = %.2f N)\n', std(F_w_t), kf);
fprintf('std(T_w) = %.2f N*m (should be ~= k_f * rcp = %.2f N*m)\n\n', std(F_w_t)*rcp,
kf*rcp);
% Optionally show histograms or PSDs
figure('Name','Diagnostics','NumberTitle','off','Position',[300 300 900 350]);
subplot(1,2,1);
histogram(F_w_t/1e3,50); xlabel('F_w (kN)'); title('Histogram of gust force component');
subplot(1,2,2);
pwelch(delta_v0,[],[],[],1/Ts); title('PSD of normalized Δv_0(t)');
% Save data if desired
% save('gmrt_wind_timehistory.mat','t','delta_v0','F_t','F_w_t','T_t','Fm','kf','kF','rcp');
```

References

[1] Wodek Gawronski. "Modeling and Control of Antennas and Telescopes", chapter 5, pg. no. 51- 72, 2008.

[2] Giant Metrewave Radio Telescope (GMRT). Web. : http://gmrt.ncra.tifr.res.in/

[3] A. M. El-Saady, B. N. H. Hassan, and M. I. Aboud, "Wind Data Analysis and Turbulence Modeling for Antenna Control," International Journal of Modern Engineering Research, vol. 3, no. 5, pp. 3192-3197, 2013.

[4] D. Sun and Y. Liu, "Trajectory control of cable-suspended FAST telescope," in Proceedings of the 35th Chinese Control Conference, pp. 4934-4938, 2016.