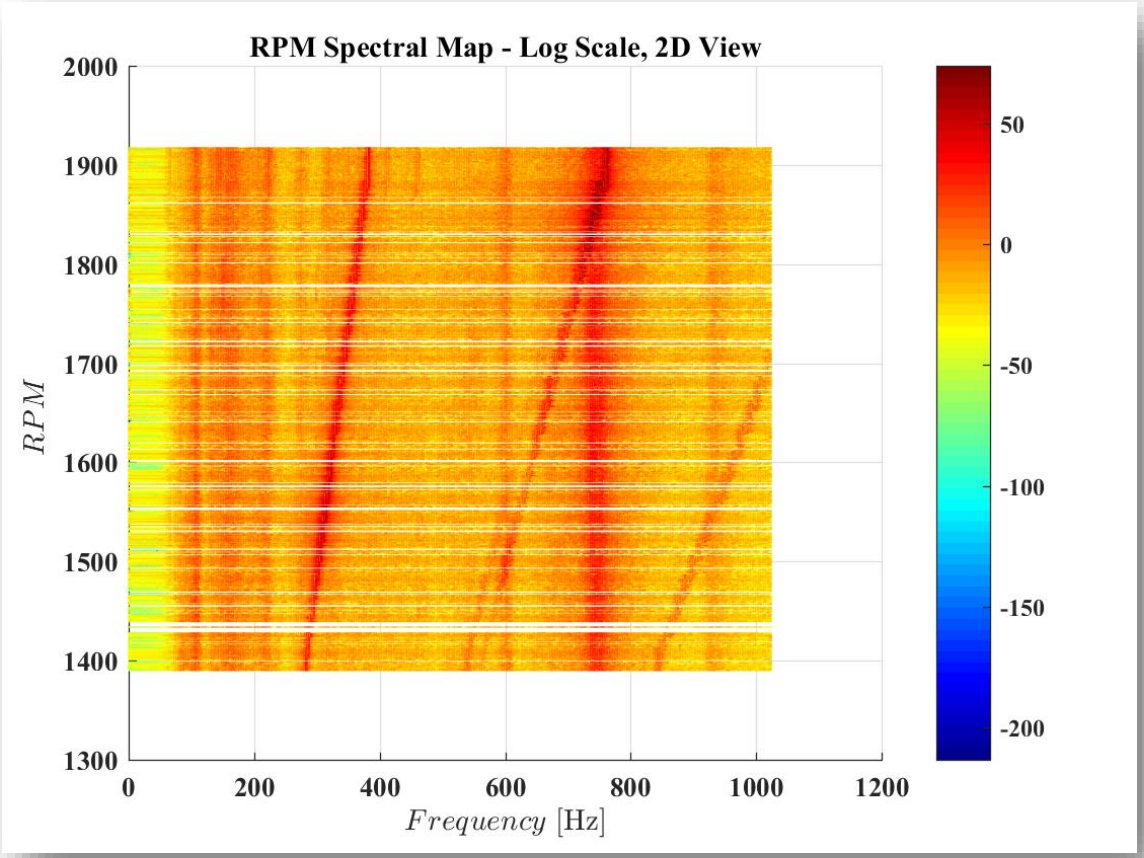


EXPERIMENTAL ANALYSIS OF ROTATING SYSTEMS – Course Report

Shashank Iyengar
M12934513

Abstract

Rotating systems are commonly found in present day machinery. Rotating shafts deliver power in an efficient manner with high reliability and far fewer complexities as compared to translational power delivery systems. Electric motors, jet engines, internal combustion engines, bearings, gear trains extensively use rotating components to deliver power.

These rotating systems have to be analyzed for vibration and noise responses in order to ensure reliable and durable operation along with comfort and convenience while preventing premature failure of components. Since rotating systems are usually transient (variable rotational speed with respect to time).

This course report is a bridged version of detailed concepts and theory behind the analysis of rotating systems. The flow of contents is presented in an equivalent chronological analysis procedure followed for rotating systems, i.e., tach signals, harmonics, spectral maps, Hilbert Transform, angle/order tracking, wavelets, computed order tracking (TVDFT, Kalman Filtering, Oversampling). Effectively, following this report as a problem-solving guide, the required harmonic analysis can be performed.

The homework assignment presented in the appendix are solved real world examples of signal processing for rotating systems. Homework 1 is based on observing time spectral maps and their characteristics. Homework 2 deals with reviewing response data for unbalanced and balanced systems. Homework 3 is computing instantaneous speed estimate for the given tach signal data. Homework 4 is about RPM spectral maps. Homework 4a is for RPM cepstral maps. Homework 5 is based on synthesized RPM maps. Homework 6 deals with Hilbert Transform. Homework 7 works with missing tooth tach data. Homework 8 is about order RPM maps and order splice. Homework 9 is TVDFT approach for HW8. Homework 10 is the Kalman Filter approach for HW8. Homework 11 is the oversampling/resampling approach for HW8.

Contents

Abstract 1

Tachometer Signal 4

Introduction 4

Types of Tachometer Sensors 4

Tachometer Sensor Issues 4

Tach Signal Processing 5

Harmonics (Order) 8

Introduction 8

Harmonics: Case Study 8

Balancing 11

Introduction 11

Balancing Post-Processing 11

Modulation 12

Introduction 12

Modulation Examples 12

Spectral Maps 15

Introduction 15

Time Maps 15

RPM Maps 16

Order Maps 16

Procedure 17

Cepstrum 19

Hilbert Transform 20

Introduction 20

Definition 20

Phasor Concept 20

Example 21

Insights 22

Angle/Order DSP 23

Introduction 23

Historical DSP Approach 24

Oversampling 24

Fixed Sample Rate, Variable Time Block Size 24

Variable Sample Rate, Fixed Block Size 25

Common Issues 25

Wavelets 26
Introduction 26
Definition 26
Insights 27
TVDFT 28
Kalman Filtering 29
Introduction 29
Vold-Kalman Filtering 30
References 31
Appendix 32
Homework 32

Tachometer Signal

Introduction

A tachometer is an instrument that is used in order to determine the rotational speed of a rotating system such as a shaft or gear in a motor. The data obtained from a tachometer is termed as tachometer signal. It is considered as the most important measurement that is made when working with a rotating system. Unlike other systems that undergo analysis for vibrations, rotating systems usually operate at a wide range of speeds. This would imply that transient rotational data needs to be recorded in order to accurately determine the changing system characteristics at different operating conditions.

The tachometer signal is used to estimate the instantaneous speed and further processed to obtain harmonic information. Since the dynamic responses are rapidly changing, the tachometer sensor should generate a signal that is not affected by transients, slew rate and should contain minimal noise possible.

Types of Tachometer Sensors

Commonly used tachometer sensors are: shaft encoders, fiber optic light sensors, infrared light sensors, laser sensors, variable reluctance tach sensor, conditioned variable reluctance tach sensor, magneto-resistive tach sensor, inductance/capacitance tach (Bentley Probe).

Tachometer Sensor Issues

Tach sensors can have signals wherein the amplitude of the output may vary with speed. Variable reluctance sensors give an output that is sensitive to speed, due to which they need a steel or magnetic tone wheel. Light based sensors have problems with ambient lighting or backlighting. Tachometer signals can have modulation depending on the off-center nature of the rotating system. A major drawback is that all tach sensors must be in close proximity to the system and this could be challenging depending on the design and accessibility of the system.

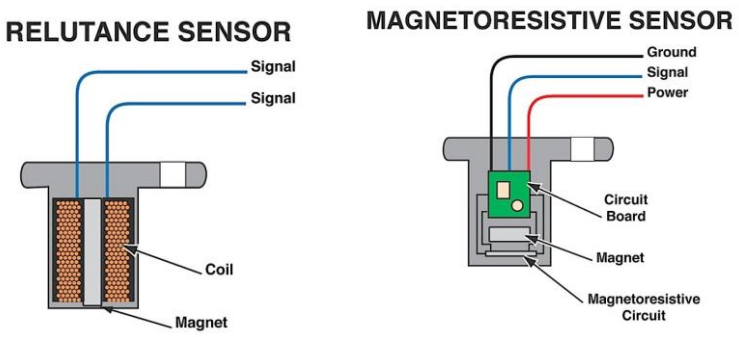
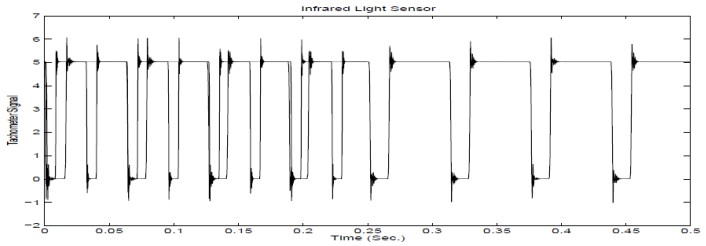


Fig. 1: Construction of Tach Sensors

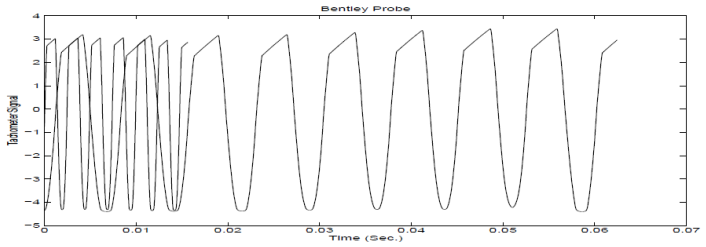
Tach Signal Processing

Examples of tachometer signals:

Light Tachometer Sensor



Bentley Probe Tachometer Sensor



Variable Reluctance Tachometer Sensor

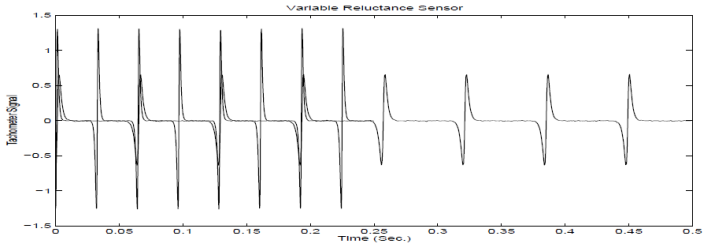


Fig. 2: Tachometer Signals

The above plots represent tachometer signals obtained from different types of sensors. A raw tach signal is a series of pulses generated and captured on the time axis. Since the signal is digitized, it is often not centered as the signal comes through a DC coupling.

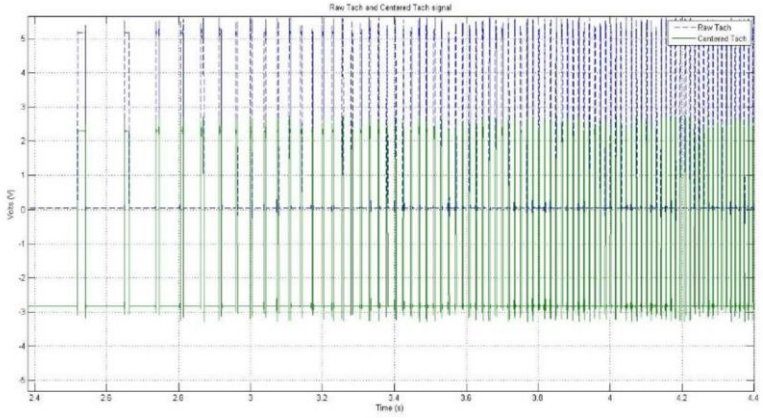


Fig. 3: Raw Tach vs. Centered Tach

- Centering the signal is important as it provides better accuracy in estimating the zero-crossings of the signal and more accurate RPM estimate.
- Centering is done by subtracting the data values from the mean of the entire data set.
- Zero-crossings are then determined by for positive slopes in the data set. This is done by looking for a change from negative to positive data values using the *sign* function in MATLAB. The points of positive zero-crossings are stored in a new data set.
- The time axis is then linearly interpolated for the zero-cross points.

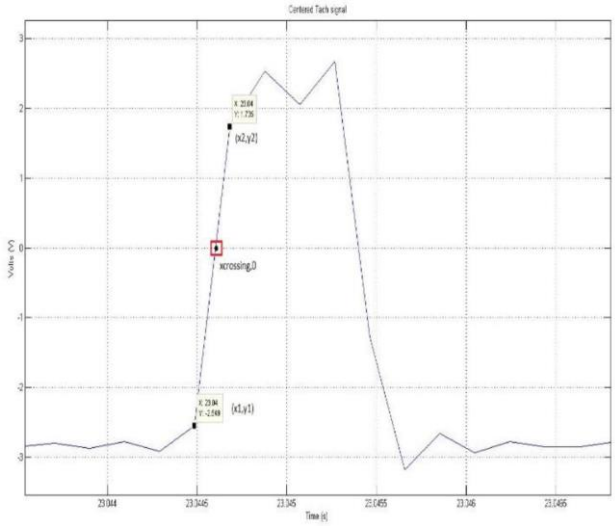


Fig. 4: Zero-crossing

$$xcrossing = \left((0 - y1) \times \frac{x2 - x1}{y2 - y1} \right) + x1$$

- Without interpolation for zero crossing, machines which rotate at high speeds cause considerable error in the tachometer signal.

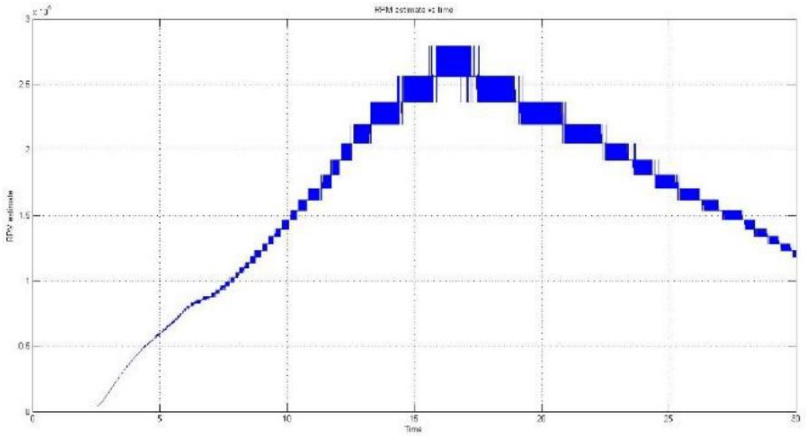


Fig. 5: RPM Estimate

- The sizeable variation in RPM estimate cannot exist physically due to the inertia of the system.
- The RPM between two consecutive pulses is calculated by:

$$RPM = \frac{60}{t2 - t1}$$

Where tn= n^{th} instant of pulse detection.

The time points corresponding to these RPM estimates are calculated as:

$$ti = \frac{t1 + t2}{2}$$

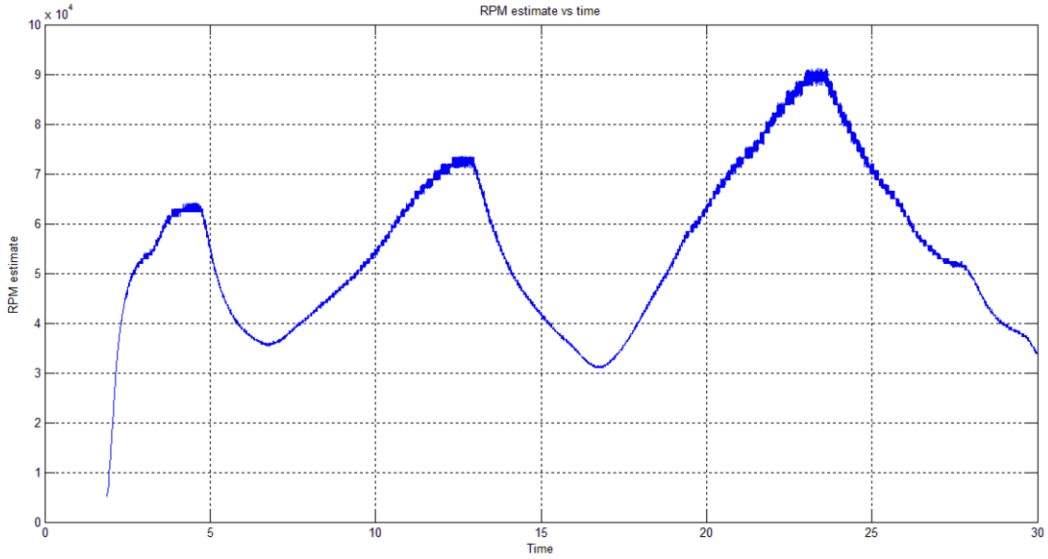


Fig. 6: First RPM Estimate

- Spline fit is used to obtain a smooth curve which better represents the actual operating condition of the system.
- This is done by segmenting the data into multiple blocks and applying a cubic spline fit between each segment. Continuity is enforced to obtain smooth fit between segments.

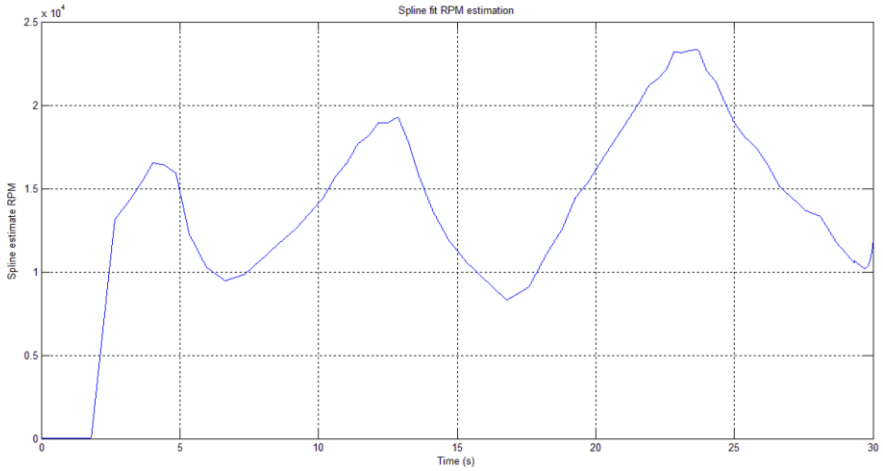


Fig. 7: Spline Fit RPM Estimate

Harmonics (Order)

Introduction

Harmonic or Order refers to vibration or noise at a frequency that is an integer multiple of the rotating axis speed. This speed is determined from the tachometer signal. The harmonic information is designated as 1X, 2X and so on implying that the frequency is an integer multiple of the rotating axis speed.

Harmonic relationship can also take any rational order such as 0.37X, 1.4X, 11.7X due to geometric conditions on a secondary shaft in the system which is not being monitored by the tachometer signal. At times, fractional orders are by design. By having at least one tachometer for each shaft, the appearance of fractional orders can be avoided.

Harmonics are important because they represent the forcing function being applied to the system. The forcing function will excite the system causing vibrations, noise and strain. When this forcing function operates near a natural frequency, it is known as resonance. In a rotating system, if the harmonic excites a natural frequency when operating at specific speeds, these speeds are called critical speeds.

Harmonics: Case Study

- Rotating unbalance is the largest cause of a first order harmonic (1X).

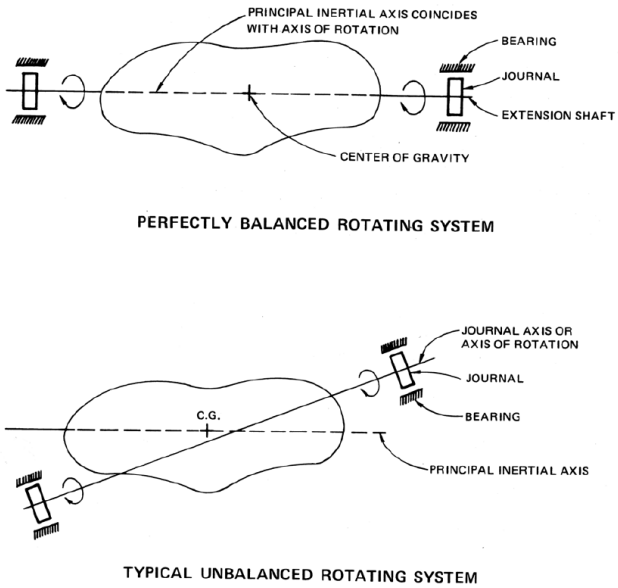
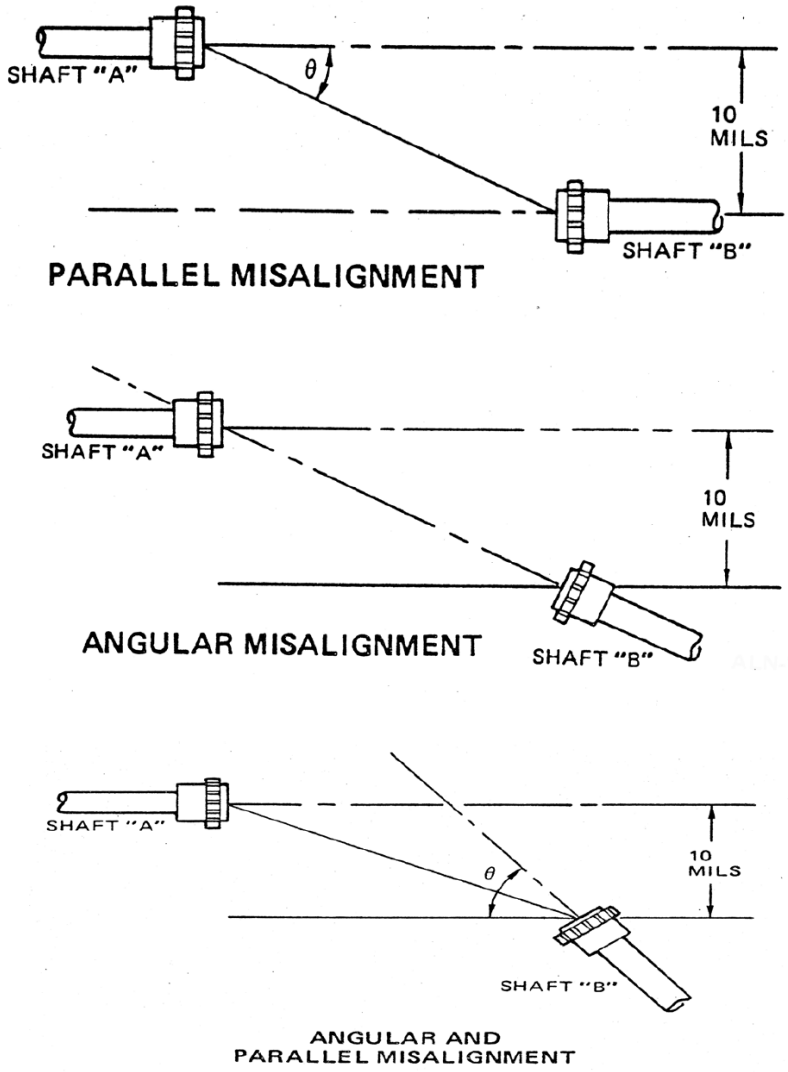
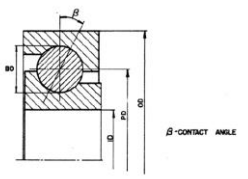


Fig. 8: Balanced v. Unbalanced Rotating System

- Misalignment: Shaft couplings are a major source of shaft misalignment. It usually results in a large first order harmonic (1X) with accompanying second order (2X) and third order (3X).



- Gears/Chains/Belts: A kinematic relationship that allows the rotation of one axis to generate rotation in another axis results in a harmonic or speed ratio change between the axes.
- Bearings:



Bearing Nomenclature:

- *N - Number of balls, or rollers, in bearing*
- *BD - Ball, or roller, Diameter*
- *OD - Outer Diameter*
- *ID - Inner Diameter*
- *PD - Pitch Diameter*
- *β - Contact angle*

Fig. 10: Construction of Bearing

- Engines: Engine will be balanced but shaking forces will still be present. These shaking forces will be at harmonics of the engine speed. Engine configuration, number and location of cylinders will be contributing parameters for vibration response.
- Motors: Harmonics of motors will be a function of the number of electro-magnetic poles in both the rotor and the stator.
- Non-contacting Bearings: Fluid bearings will exhibit responses that occur due to oil whirl and oil whip. Oil whirl is a characteristic response that follows the speed variation and begins at speed that is 0.35 to 0.47 times the natural frequency of the oil bearing. Oil whip is a self-excited vibration response.

Balancing

Introduction

Balancing a rotating system is an important aspect of the design and operation of a mechanical system that involves a rotating shaft. Rotating systems are rarely perfectly balanced. Unbalance is generally caused by an unbalanced mass, located at some eccentricity, spinning about the center of rotation at a constant frequency.

Static balancing refers to a procedure that adds or subtracts mass at some eccentricity to balance the vector forces. Dynamic balancing refers to a procedure that adds or subtracts mass at some eccentricity and location along the axis of rotation to balance the unbalance moments. Dynamic balancing provides a better balance whenever the rotating shaft is long, and the number of unbalances occur at many locations along the axis of rotation. If the shaft of the rotating system has the first natural frequency in bending above the operating speed (rigid), dynamic balancing can theoretically balance the system perfectly using only two arbitrary planes of balance. If the shaft is flexible, the only way to balance perfectly is to find every plane of unbalance and to balance each plane separately.

Balancing Post-Processing

The FRFs between the accelerometers and the tach signal can be used to identify that the magnitude of response in the system has decreased after balancing the system.

Modulation

Introduction

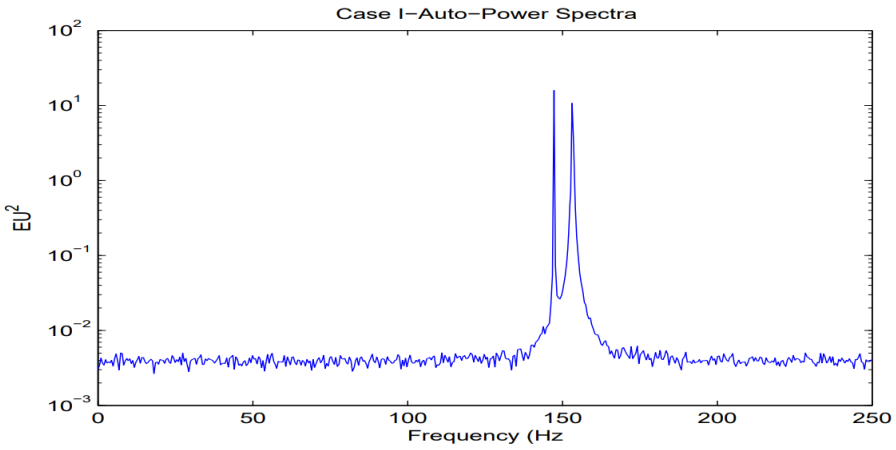
Modulation of frequencies occur whenever two frequencies with significant magnitude are close. It is clearly shown as angled lines of magnitude information that originates from a crossing of a harmonic forcing function and a system characteristic frequency. With respect to rotating systems, modulation usually refers to amplitude modulation. Other forms of modulation can include phase modulation and pulse amplitude modulation. The crossing of a harmonic forcing function and system characteristic frequency is referred to as beating.

Modulation Examples

Source of Sum and Difference Frequencies:

- Case 1:

$$A\ cos(\omega_1\ t)\ B\ cos(\omega_2\ t) = \frac{A\ B}{2}\left[\cos((\omega_1 - \omega_2)\ t) + \cos((\omega_1 + \omega_2)\ t)\right]$$

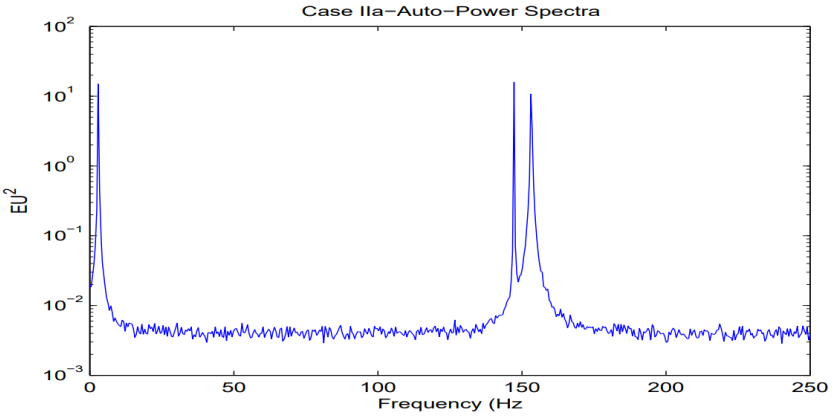


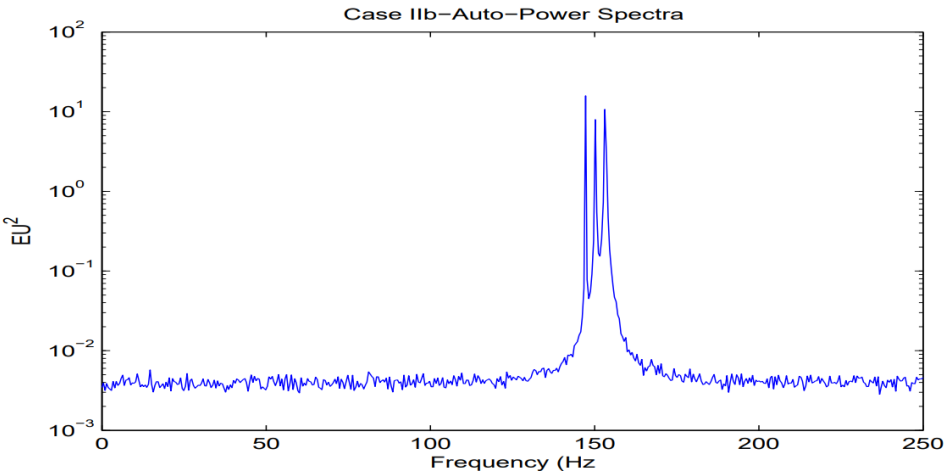
- Case 2:

$$\left[C + A\ cos(\omega_1\ t) \right] B\ cos(\omega_2\ t) =$$

$$\frac{A\ B}{2} \left[\cos((\omega_1 - \omega_2)\ t) + \cos((\omega_1 + \omega_2)\ t) \right] +$$

$$B\ C\ cos(\omega_2\ t)$$



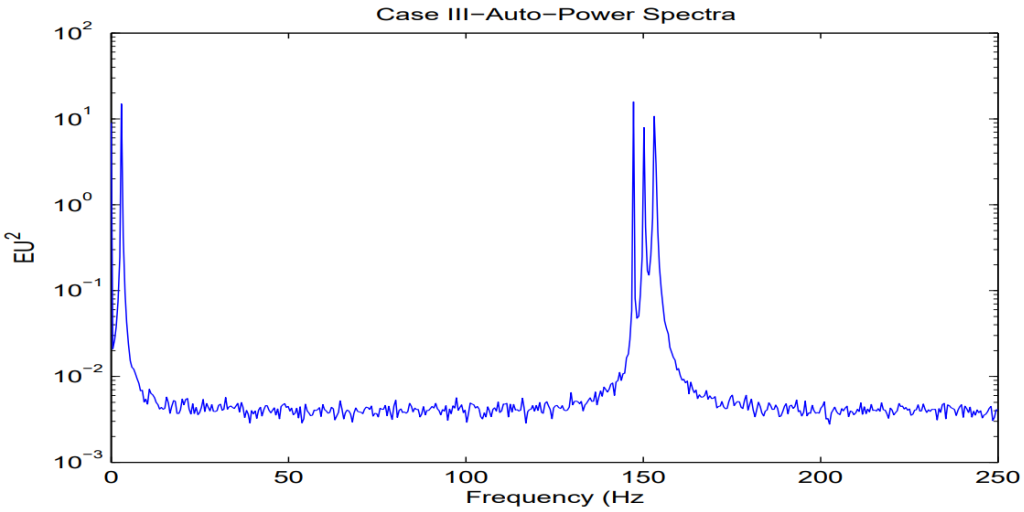


- Case 3:

$$\left[C + A\ cos(\omega_1\ t) \right]\left[D + B\ cos(\omega_2\ t) \right] =$$

$$\frac{A\ B}{2}\left[\cos((\omega_1 - \omega_2)\ t) + \cos((\omega_1 + \omega_2)\ t)\right] +$$

$$A\ D\ cos(\omega_1\ t) + B\ C\ cos(\omega_2\ t) + C\ D$$



Spectral Maps

Introduction

Response data obtained from a non-transient system is displayed as a function of frequency to understand the characteristics of that system. This is a typical FFT based 2D plot. A rotating system is transient; thereby, data is observed in three dimensions as a waterfall or spectrogram plot. Typically, amplitude information is displayed as a function of frequency as a third parameter is allowed to vary. Two-dimensional information can be extracted from this three-dimensional map for further analysis.

Time Maps

- A time spectral map is the auto power spectrum displayed as a function of time as the third parameter.
- A time cepstrum map is the auto power cepstrum displayed as a function of time as the third parameter.

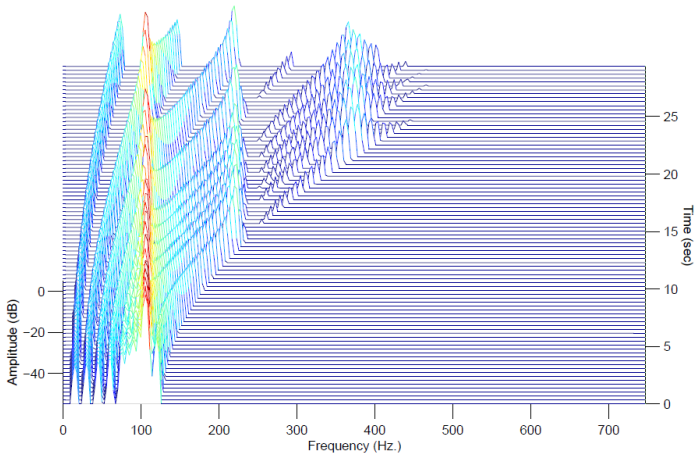


Fig. 15: Time Spectral Map

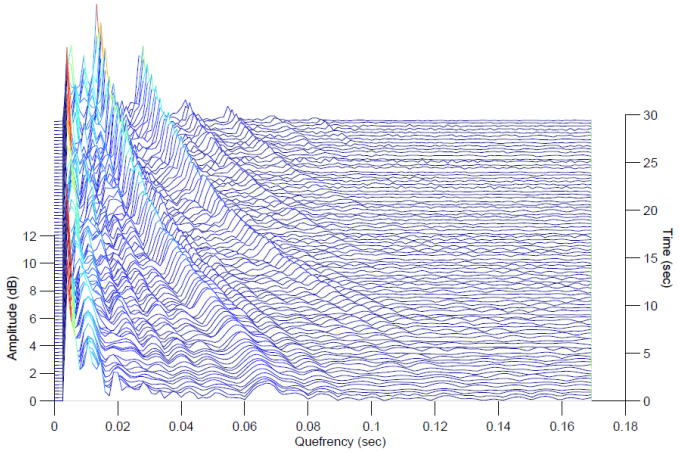


Fig. 16: Time Cepstral Map

RPM Maps

- RPM Spectral Map is the auto power spectrum displayed as a function of RPM as the third parameter.
- Harmonic information appears as angled lines in the magnitude information with slopes equal to the order (1X, 2X, 3X, 3.7X)
- Natural frequency information appears as vertical lines in the magnitude information indicating that the frequency is constant for all RPMs. Curving could occur if the rotational speed is very high to cause centrifugal weakening/stiffening.
- RPM Cepstral Map is the auto power cepstrum displayed as a function of RPM as the third parameter.

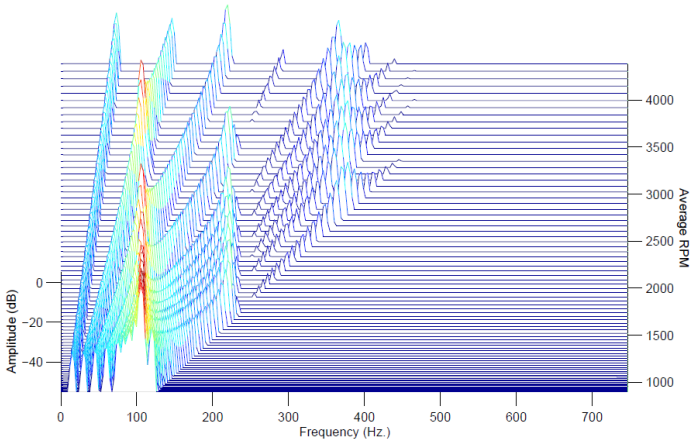


Fig. 17: RPM Spectral Map

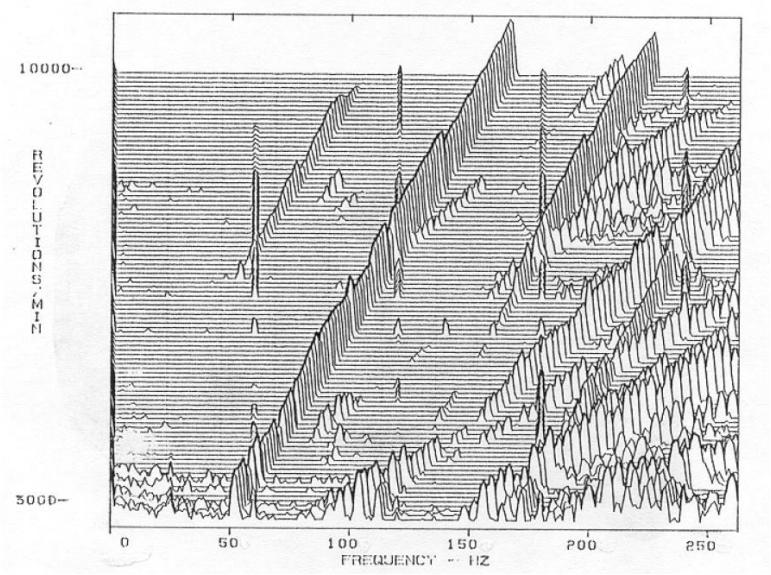


Fig. 18: RPM Spectral Map with Modulation

Order Maps

- Order maps are RPM spectral maps with the frequency axis replaced by the order axis of the system.

- Order Time Map is similar to a time spectral map, replacing the frequency axis with an order axis to visualize how the order changes as a function of time.
- Order RPM Map is similar to an RPM spectral map, replacing the frequency axis with an order axis to visualize how the order changes as a function of RPM.
- Harmonic information appears as vertical lines indicating that the order is constant for all RPMs.
- Natural frequency information appears as curved lines.

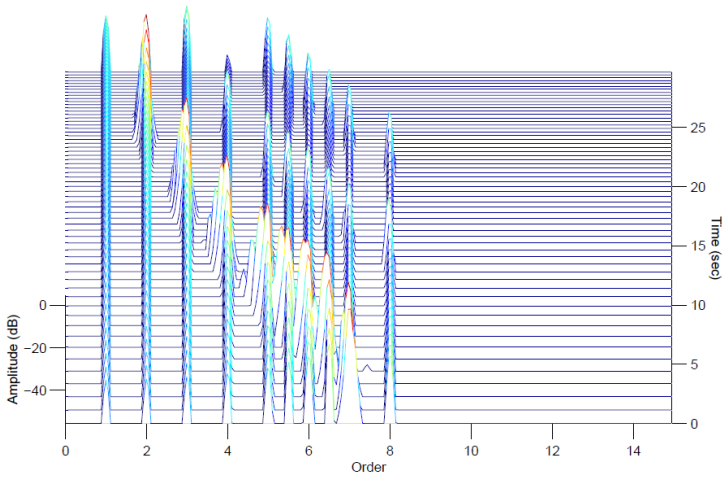


Fig. 19: Order Time Map

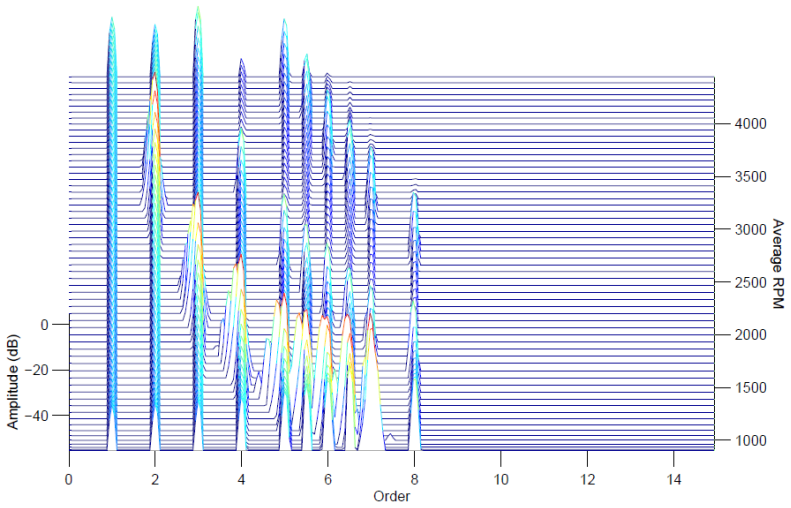


Fig. 20: Order RPM Map

Procedure

- The tach signal is processed to obtain the instantaneous speed curve for the given rotating system.
- The response data and time axis are split into blocks of size the power of 2s in order to have higher computational efficiency (example: 1024, 2048 block size).

- These blocks are overlapped by a certain user specified percentage (25, 50, 75, 87.5, 93.75%).
- FFT and auto power spectrum is performed on each block.
- To obtain a Time Spectral Map, the auto power spectrum(x-z axis) is plotted against time (y axis).
- To obtain an RPM Spectral Map, the auto power spectrum is plotted against instantaneous speed previously computed.
- To obtain an Order Time Map, the magnitude of auto power spectrum is plotted against the order axis (x axis) along time (y axis).
- To obtain an Order RPM Map, the magnitude of auto power spectrum is plotted against the order axis (x axis) along RPM data (y axis).

Cepstrum

Cepstrum is defined as the spectrum of the log of the spectrum of a time waveform. It was originally used to evaluate signals that contained echoes. Cepstrum is a deconvolution process that turns normal convolution into an addition process.

$$x(t) = s(t) + \alpha s(t - \tau)$$

$$|X(f)|^2 = X(f) * X(f)^* = |S(f)|^2 * [1 + \alpha^2 + 2\alpha\cos(2\pi f\tau)]$$

$$\log(|X(f)|^2) = \log(|S(f)|^2) * \log([1 + \alpha^2 + 2\alpha\cos(2\pi f\tau)])$$

$$C_x(\tau) = |FFT\{\log[F_{xx}(f)]\}^2| = FFT[\log(X(f) * X(f)^*)]$$

Where,

τ is Quefrency in units of time; $C_x(\tau)$ is dimensionless magnitude

Other terms defined with cepstrum are rahmonics and liftering. Rahmonics are the peaks in the cepstrum and liftering is the linear filtering of the log of the spectrum.

Power cepstrum is defined as the power spectrum of the logarithm of the power spectrum. It is advantageous over autocorrelation due to the fact that it is far less sensitive to frequency shape. Due to the lack of sensitivity to global spectrum shape, power cepstrum finds use in detection of echo in signals, speech analysis (measure of voice pitch), detecting harmonic patterns in machine vibration spectra and to detect & separate families of sidebands in a spectrum.

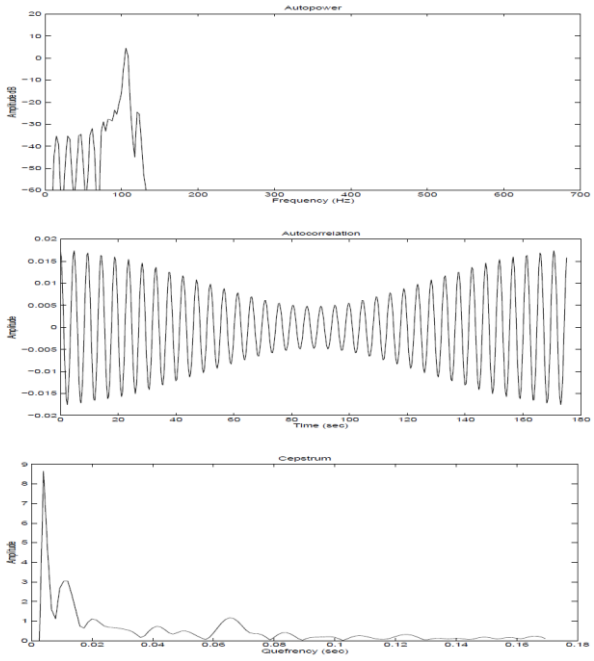


Fig. 21: Power Cepstrum

Complex cepstrum is defined as the inverse Fourier Transform of the complex logarithm of the complex spectrum (Fourier Transform of the time signal).

Hilbert Transform

Introduction

Hilbert Transform takes advantage of the causality to estimate the orthogonal aspects of a signal. In the time domain, it creates a complex-valued phasor that is useful in the demodulation of signals and the computation of instantaneous frequency. In the frequency domain, it creates a complex-valued function that can be used to detect nonlinearities.

Definition

$$H[x(t)] \equiv x'(t) \equiv \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{x(\tau)}{t-\tau}\, d\tau \equiv x(t) * \frac{1}{\pi t}$$

- Hilbert Transform of a time/frequency domain function is a time/frequency domain function.
- This transform is equivalent to a modified convolution.
- It gives a localized perspective of time-frequency behaviour.
- Integration is infinite. Hence, time/frequency truncation will introduce errors into the calculations.
- The convolution can be calculated by shifting, multiplying and integrating over the range of the function.
- Hilbert Transform shifts positive frequency spectral components by $-\frac{\pi}{2}$ and negative frequency spectral components by $+\frac{\pi}{2}$.
- FFT can be used to compute the discrete Fourier transform; then shift the phase of the spectrum and inverse FFT to get the Hilbert Transform.

Phasor Concept

- An analytic signal is complex and represents phasing of a signal.
- The real part of the analytic signal is the measured data and the imaginary part of the signal is the Hilbert Transform of the measured data.

$$z(t) \equiv x(t) + j\ x_h(t) \equiv A(t)\ e^{j\ \psi(t)}$$

where:

- $A(t) = \sqrt{x^2(t) + x^{2_h}(t)}$
- ***A(t) is the instantaneous magnitude or modulus.***
- ***A(t) is also referred to as the quadrature average.***
- $\psi(t) = \tan^{-1} \dfrac{x_h(t)}{x(t)}$
- $\psi(t)$ ***is the instantaneous phase.***
- $\Omega(t) = \dfrac{d(\psi(t))}{dt}$
- $\Omega(t)$ ***is the derivative of instantaneous phase which is the instantaneous frequency (proportional to RPM).***

Example

Following is an example of a chirp signal.

$$x(t) = \cos[2\pi(f_0 t)t]$$

$$z(t) \equiv x(t) + j\ x_h(t) = \cos[2\pi(f_0 t)t] + j \sin[2\pi(f_0 t)t]$$

$$\psi(t) = \tan^{-1} \frac{x_h(t)}{x(t)} = \tan^{-1} \left\{ \frac{\sin[2\pi(f_0 t)t]}{\cos[2\pi(f_0 t)t]} \right\} = [2\pi(f_0 t)t]$$

$$\Omega(t) = \frac{d(\psi(t))}{dt} = \frac{d(2\pi(f_0 t)t)}{dt}$$

$$\Omega(t) = 4\pi f_0 t$$

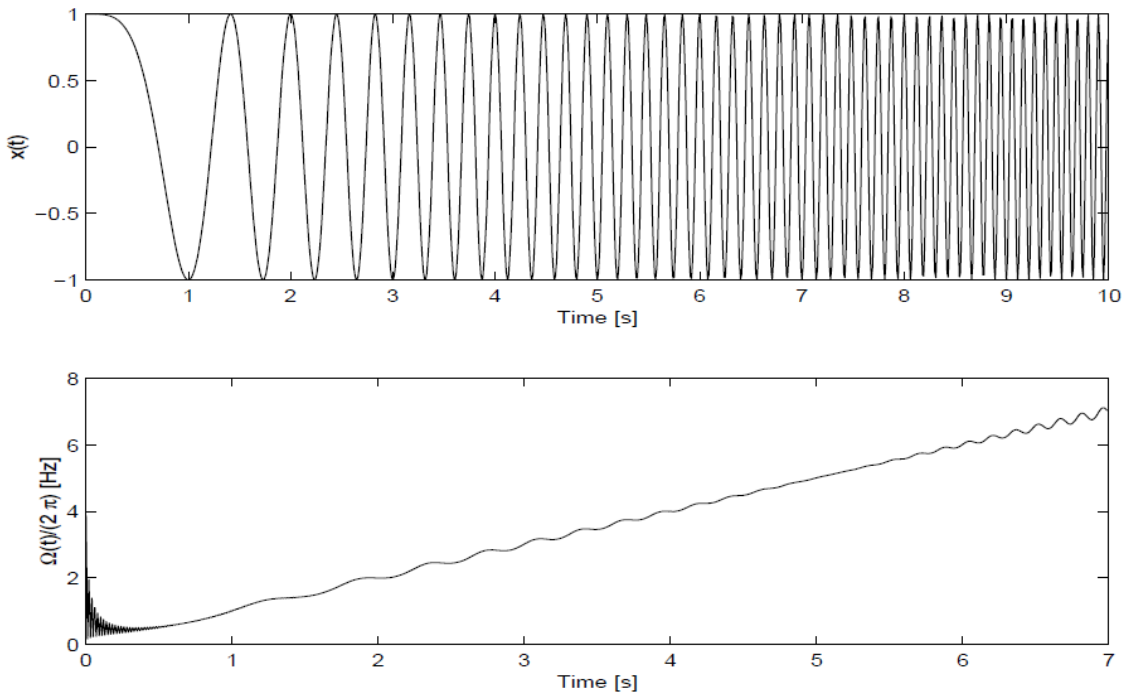


Fig. 22: Instantaneous Frequency obtained from Hilbert Transform

Insights

- Low and high frequency residuals must be compensated. This can be done by including the negative frequency information.
- The instantaneous phase must be unwrapped before computing the instantaneous frequency.
- The instantaneous frequency and backbone curve must be low pass filtered for smoothness.
- Multi-frequency responses must be filtered to obtain single frequency responses.
- Significant coupling in MDOF systems creates inaccuracies.

Angle/Order DSP

Introduction

Traditional FFT digital signal processing will work effectively for systems where the first order (fundamental harmonic) is not changing when measures are taken to minimize leakage. If the fundamental harmonic changes over time with varying slew rates, the data must be sampled in a different way in order to change the independent variable, time, to a companion independent variable, angle. Historically, since number of samples per revolution was fixed, the sample rate was slower at slower rotational speed and higher at higher rotational speeds.

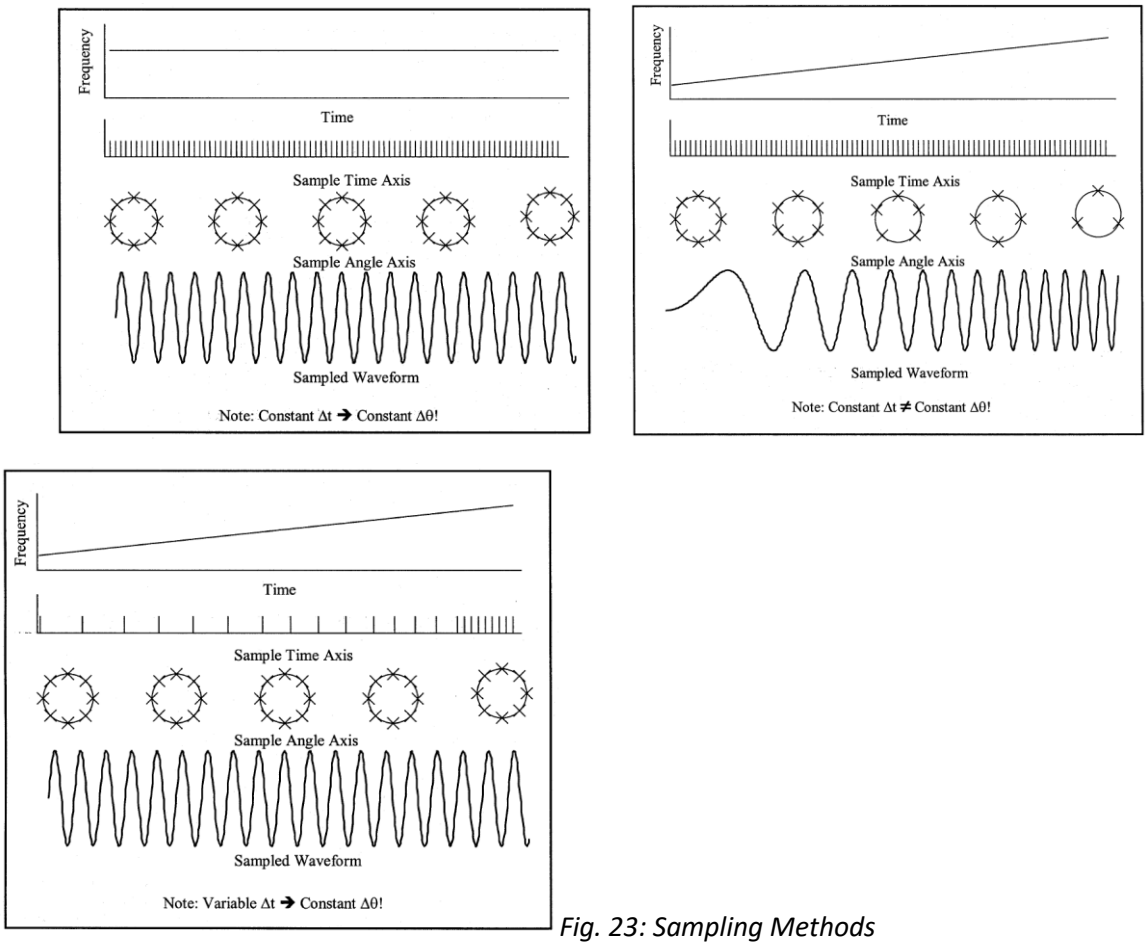


Fig. 23: Sampling Methods

Domain	
Time/Frequency	***Angle (Revolutions)/Order***
Samples/Second	***Samples/Revolution***
Total Time (T)	***Total Angle/Revolutions (R)***
Time Resolution (Δt)	***Angle Resolution*** ($\Delta\theta$)
Frequency Resolution (Δf)	***Order Resolution*** (ΔO)
Nyquist Frequency (F_{Nyq})	***Nyquist Order*** (O_{Nyq})
Maximum Frequency ($F_{\max}$)	***Maximum Order*** ($O_{\max}$)

Shannon's Sampling Theorem

$$O = \frac{1}{\Delta\theta} = O_{nyq} * 2$$

$$O_{nyq} \geq O_{max}$$

Rayleigh's Criteria

$$\Delta O = \frac{1}{R}$$

Historical DSP Approach

The Tracking Ratio Synthesizer converts the tachometer signal into a TTL level tachometer signal at a multiple of the incoming tachometer signal using a phase locked loop (PLL). The ratio is chosen from an integer numerator, integer denominator pair of number giving almost any output tachometer signal. The tracking anti-aliasing filter protects against aliasing. Due to the extensive use of these two instruments, the historical approach depends upon an external sampling capability in the ADC. O_{max} will be stable and synchronous with the FFT in the angle/order domain as long as the tracking ratio tuner can produce a stable frequency from the tach signal. Fast changes in speed cannot be processed using this approach.

With fixed sample rate and block size, O_{max} and ΔO will change with each block of data. For slow rotational speeds, O_{max} and ΔO will be high. This will result in more orders with less resolution. This is conversely true. Since consistency is required across all blocks of data, a fixed sample rate and block size approach cannot be used for systems with changing speed to prevent leakage.

Oversampling

As the name suggests, oversampling is sampling the data at extremely high sample rates than what is required (in the order of 10 to 1000 times the maximum frequency of the system). The data is then downsampled or resampled to obtain a particular O_{max} and ΔO.

Fixed Sample Rate, Variable Time Block Size

During sampling of the data, a fixed sample rate is chosen with a variable block size in order to keep ΔO constant among all blocks. By following this approach, the blocksize used by FFT is no longer multiples of the powers of 2. This is combated by computational capabilities of present age computers. Since the sampling rate is fixed, O_{max} will still vary between blocks with instantaneous speed. To obtain the O_{max} required, the data is digitally filtered and downsampled. The necessary criteria for this system to efficiently work is that the speed must remain constant within the block of observation.

Variable Sample Rate, Fixed Block Size

Here, the block size is fixed, but the sampling rate can vary between blocks. The sampling frequency is changed from one block to another to span an integer number of revolutions. This integer number of revolutions is required for ΔO and is then used in an arbitrary length FFT allowing for the number of samples per revolution to be consistent and also the number of revolutions per blocksize. This method yields leakage free measurement in the angle/order domain even at considerable changes in speed. Similar to the previous case, the speed must be constant within a block of observation.

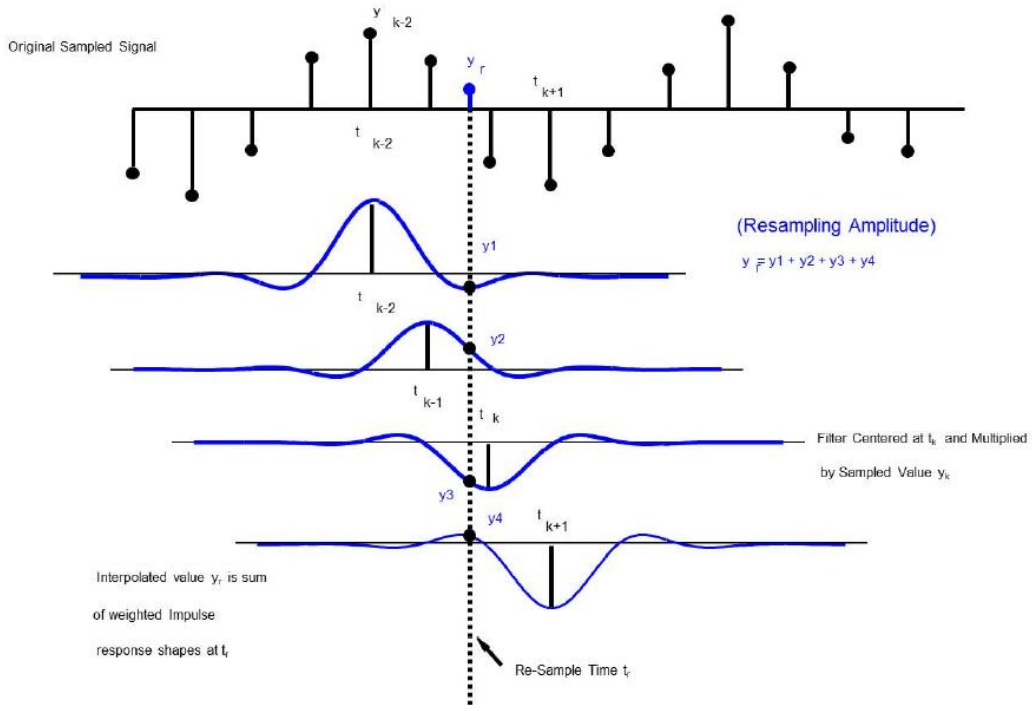


Fig. 24: Resampling using Digital Filter

Common Issues

- Frequencies that do not change with speed will give distorted amplitudes with leakage errors.
- Harmonics that cannot be described within O_{max} and ΔO relationships will still cause leakage errors.
- Despite having perfect periodicity with respect to rotating speed, there will be leakage if during the observation period associated with the FFT, the speed changes.
- As speed increases, even though ΔO remains constant, the total observation time decreases. This implies that frequency resolution increases and thereby the ability to separate frequency content that is not proportional to speed will decrease with increasing speed.

Wavelets

Introduction

Wavelets are used to localize response information in both time and frequency domains. This is especially important because there is a need to understand the transient responses that are mixed in with the stationary responses. Fourier transform approaches are not designed to process this sort of data since the kernel of the Fourier transform is stationary and the data being analysed is non-stationary. Data for these cases must be analysed both as a function of time and frequency using localization techniques in the time and frequency domains.

Wavelets permit long time windows when accurate low frequency information is needed and short time windows when accurate high frequency information is needed. Real wavelets detect sharp signal transitions, while analytic wavelets identify instantaneous frequency evolution. They can also be classified as continuous (integral) or discrete (summation). Wavelets normally restrict the basis function (mother wavelet) to be short and oscillatory to ensure that the integration of the transform is finite. The function must have an average of zero and decay quickly at both ends. These are known as admissibility conditions.

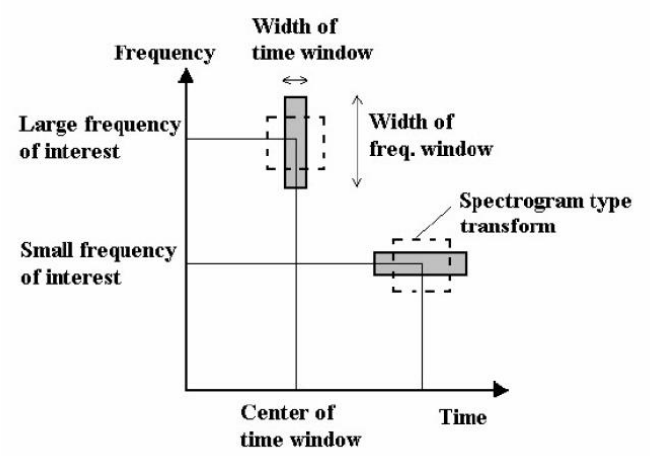


Fig. 25: Time-Frequency Windows for kernels (Wavelets)

Definition

$$W[x(t)] \equiv \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} x(t) \Psi^* \left(\frac{t-b}{a} \right) dt$$

- $\varphi(t)$ is the basic wavelet
- b is the time shift
- a is the time dilation (scaling) constant
- Wavelets are usually designed using octave frequency bands

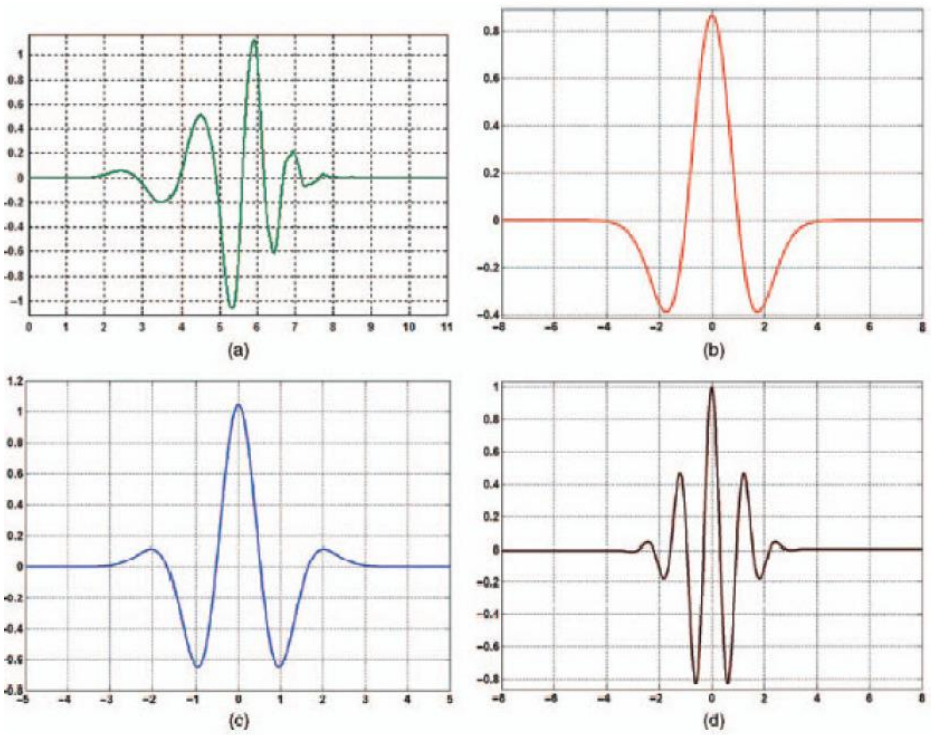


Fig. 26: Types of Wavelets: a) Daubechies wavelet, b) Mexican hat wavelet, c) Gaussian wavelet, d) Morlet wavelet

Insights

- Wavelets can be used to analyse signals with multiple frequency components.
- High frequency resolution in wavelet maps requires more time points. Computation time is significantly increased.
- Wavelet analysis capabilities are not found in most dynamic signal analyzers, implying that post processing is required.

TVDFT

Time Variant Discrete Fourier Transform (TVDFT) uses a modified kernel of the FFT that varies with the frequency of the order of interest. This is because the FFT is not particularly good for analyzing signals that are transient within a block of data. This limitation exists because the kernel of the FFT is stationary. It is given by:

$$a_m = \frac{1}{N}\sum_{n=1}^{N} x(n\Delta t)\cos(2\pi f_m n\Delta t)$$

$$b_m = \frac{1}{N}\sum_{n=1}^{N} x(n\Delta t)\sin(2\pi f_m n\Delta t)$$

In the angle/order domain:

$$a_m = \frac{1}{N}\sum_{n=1}^{N} x(n\Delta\theta)\cos(2\pi O_m n\Delta\theta)$$

$$b_m = \frac{1}{N}\sum_{n=1}^{N} x(n\Delta\theta)\sin(2\pi O_m n\Delta\theta)$$

TVDFT method reduces smearing and leakage issues that arise from changing speed within the observed data block. The TVDFT kernel is given by:

$$a_m = \frac{1}{N}\sum_{n=1}^{N} x(n\Delta\theta)\cos\left(\int_0^{n\Delta t}\left(2\pi O_m\left(\frac{RPM}{60}\right)\Delta t\right)dt\right)$$

$$b_m = \frac{1}{N}\sum_{n=1}^{N} x(n\Delta\theta)\sin\left(\int_0^{n\Delta t}\left(2\pi O_m\left(\frac{RPM}{60}\right)\Delta t\right)dt\right)$$

The TVDFT process does not guarantee that an integer number of revolutions is processed. Due to this, some leakage could occur. Leakage could also occur due to the other nearby orders that can contaminate the information associated with the order of interest. Unlike a fixed frequency or fixed order analysis, the kernels of the TVDFT may not be completely orthogonal. This is overcome through the use of an orthogonality compensation matrix.

Kalman Filtering

Introduction

Kalman Filtering is a popular method of Computed Order Tracking (COT). Computed order tracking is used to process data sampled at fixed sampling rates into order domain data. This method has multiple advantages over hardware order tracking. The primary advantage is that it does not require specialized hardware previously mentioned in the historic approach section. Secondly, computed order tracking does not have speed change limitations that hardware order tracking has.

Kalman filtering is a least squares method for processing nonstationary data. This method needs an accurate estimate of the instantaneous speed versus time in order to tune the filter to the proper harmonic. The width of the filter is chosen via a weighting parameter. Usually, the weighting parameter is the ratio of the standard deviation of the error in the structure equation to the standard deviation of the error in the data equation.

A narrow filter requires a longer period of time to stabilize the filter. A wider filter yields less accurate results. It is usually implemented as a second order filter. The two equations solved in the least squares method is the structure equation and the data equation. This involves three successive data points.

Data Equation (Measurement equation):

$$y(t_n) = x(t_n) + \eta(t_n)$$

Where,

$y(t_n)$ is the measured signal,

$x(t_n)$ is the desired harmonic signal to be extracted

$\eta(t_n)$ is the random noise plus all periodic components except the desired harmonic

Structure Equation (Process equation):

$$x(t_n) - 2\cos(\omega\Delta t)\, x(t_{n-1}) + x(t_{n-2}) = 0$$

Where, ω is the instantaneous frequency estimate

Usually, an error term is included in the structure equation,

$$x(t_n) - 2\cos(\omega\Delta t)\, x(t_{n-1}) + x(t_{n-2}) = \varepsilon(t_n)$$

These equations can be assembled in a matrix to yield:

$$\begin{bmatrix} 1 & -2\cos(\omega\,\Delta t) & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x(t_{n-2}) \\ x(t_{n-1}) \\ x(t_n) \end{Bmatrix} = \begin{Bmatrix} \varepsilon(t_n) \\ y(t_n) - \eta(t_n) \end{Bmatrix}$$

Applying a weighting function $W(t_n)$***:***

$$\begin{bmatrix} 1 & -2\cos(\omega\,\Delta t) & 1 \\ 0 & 0 & W(t_n) \end{bmatrix} \begin{Bmatrix} x(t_{n-2}) \\ x(t_{n-1}) \\ x(t_n) \end{Bmatrix} = \begin{Bmatrix} \varepsilon(t_n) \\ W(t_n)\,y(t_n) - \eta(t_n) \end{Bmatrix}$$

Assuming a minimum variance unbiased solution:

$$\begin{bmatrix} 1 & -2\cos(\omega\,\Delta t) & 1 \\ 0 & 0 & W(t_n) \end{bmatrix} \begin{Bmatrix} x(t_{n-2}) \\ x(t_{n-1}) \\ x(t_n) \end{Bmatrix} = \begin{Bmatrix} 0 \\ W(t_n)\,y(t_n) \end{Bmatrix}$$

The weighting function is defined as

$$W(t_n) = \frac{\varepsilon(t_n)}{\eta(t_n)}$$

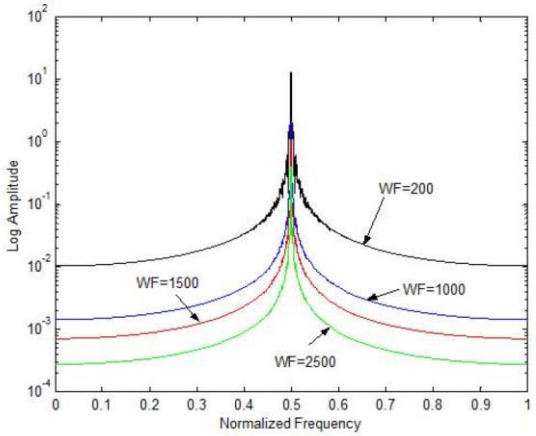


Fig. 27: Effects of different weighting functions

Vold-Kalman Filtering

The structure equation can be presented in an exponential form which would require one less data point for the solution. The data equation is then manipulated by multiplying the exponential kernel to reduce the complexity of the combined equations. This results in a simpler form of equations and allows for implementations of first, second and third order filters.

$$x(t_{n+1}) - e^{(j\omega\Delta t)}\,x(t_n) = 0 = \varepsilon(n)$$

References

1. Allemang, R., *"Rotating System Measurements, Background Concepts, Tachometer Signals"*, University of Cincinnati, Aug. 2013
2. Deshpande, S., *"Estimating Instantaneous Speed from Raw Tachometer Signal"*, University of Cincinnati, Jan. 2014
3. Allemang, R., *"Rotating System Measurements, Background Concepts, Harmonic/Orders"*, Apr. 2018
4. Enrich, F., *"Handbook of Rotordynamics",* Section 4.6, McGraw-Hill Inc.
5. SDRL*, "Balancing Test Lab"* 20-263-571 Lab Notes, University of Cincinnati, Feb. 2007
6. Allemang, R., *"Rotating System Measurements, Background Concepts, Modulation"*, University of Cincinnati, Sep. 2019
7. Allemang, R., *"Rotating System Measurements, Background Concepts, Data Maps"*, Aug. 2013
8. Allemang, R*., "Rotating System Measurements, Background Concepts, Cepstrum"*, Sep. 2019
9. Allemang, R., *"Rotating System Measurements, Background Concepts, Hilbert Transform"*, Oct. 2019
10. Allemang, R., *"Rotating System Measurements, Background Concepts, Angle/Order DSP"*, Apr. 2014
11. Allemang, R., *"Rotating System Measurements, Background Concepts, Wavelets",* Mar. 2014
12. Allemang, R*., "Rotating System Measurements, Background Concepts, TVDFT"*, Oct. 2019
13. Blough, J., Brown, D., *"The Time Variant Discrete Fourier Transform as an Order Tracking Method"*, SAE 972006
14. Allemang, R., *"Rotating System Measurements, Background Concepts, Kalman Filter COT"*, Oct. 2019
15. Tuma, J., *"Vold-Kalman Order Tracking Filtration"*
16. Eichenberger, N., *"Experimental Analysis of Rotating Systems"*

Appendix

Homework

The following pages contain the codes for Homework 1-11 in order along with plots and results.

- Homework 1: Time Spectral Maps
- Homework 2: Balancing Experiment
- Homework 3: Instantaneous speed curve determination
- Homework 4: RPM Spectral Maps
- Homework 4a: RPM Cepstral Maps
- Homework 5: Synthesized RPM Spectral Map
- Homework 6: Hilbert Transform
- Homework 7: Missing Tooth Tach Instantaneous Speed
- Homework 8: Order RPM Map, Order Splice
- Homework 9: TVDFT Method
- Homework 10: Kalman Filter Method
- Homework 11: Oversampling/Resampling (VOW) Method

Contents

- Experimental Vibrations of Rotating Systems - Homework 1 - Shashank Iyengar (M12934513)
- Pre-processing
- Time Block Size = 512
- Time Block Size = 2048
- Time Block Size = 1024
- Plots - Linear Scale
- Plots - Log Scale

Experimental Vibrations of Rotating Systems - Homework 1 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('hwk1_data.mat')
```

Pre-processing

```
Fs = 5120;  %Sampling Frequency
Nc = 0;     %No cyclic averaging
```

Time Block Size = 512

```
Ntime = 512;
Navg = length(accel_data1)/Ntime;

dt = 1/Fs;                      % Time Step
N = length(accel_data1);        % Total number of sample points
Ttime = Ntime/Fs;               % Time for one block
Ttotal = N/Fs;                  % Time for complete signal
df = 1/Ttime;                   % Frequency Resolution

f_xaxis = 0:df:1200;            % Frequency axis
t_yaxis = 0:Ttime:Ttotal-Ttime; % Time axis

% No Overlap
n=1;
for i=1:Navg
    if i==1
        accel_nolap(i,1:Ntime) = accel_data1(1,1:Ntime);                    % Data chopped an
d saved
    else
        accel_nolap(i,1:Ntime) = accel_data1(1,(n*Ntime)+1:(n+1)*Ntime);    % Data chopped an
```

```
d saved
        n=n+1;
    end
end

% 50% overlap
n=1;
for i=1:Navg*2-1
    if i==1
        accel(i,1:Ntime) = accel_data1(1,1:Ntime);                          % Data chopped an
d saved
    else
        accel(i,1:Ntime) = accel_data1(1,n*(Ntime/2)+1:511+(n*(Ntime/2)+1));% Data chopped an
d saved
        n=n+1;
    end
end

% FFT and Power Spectrum
for i=1:Navg
    AP_nolap_Total(i,1:Ntime) = conj(fft(accel_nolap(i,:))).*(fft(accel_nolap(i,:)));       %
 Auto Power Spectrum - No overlap
end
for i=1:Navg*2-1
    AP_Total(i,1:Ntime) = conj(fft(accel(i,:))).*(fft(accel(i,:)));                         %
 Auto Power Spectrum - 50% overlap
end

AP_nolap = AP_nolap_Total(:,1:121);     % Limiting to first 1200Hz magnitudes - No overlap
AP = AP_Total(:,1:121);                 % Limiting to first 1200Hz magnitudes - 50% overlap
```

Time Block Size = 2048

```
Ntime1 = 2048;
Navg1 = length(accel_data1)/Ntime1;

dt1 = 1/Fs;                         % Time Step
Ttime1 = Ntime1/Fs;                 % Time for one block
df1 = 1/Ttime1;                     % Frequency Resolution

f_xaxis1 = 0:df1:1200;              % Frequency axis
t_yaxis1 = 0:Ttime1:Ttotal-Ttime1; % Time axis

% No Overlap
n=1;
for i=1:Navg1
    if i==1
        accel_nolap1(i,1:Ntime1) = accel_data1(1,1:Ntime1);
    else
        accel_nolap1(i,1:Ntime1) = accel_data1(1,(n*Ntime1)+1:(n+1)*Ntime1);
        n=n+1;
    end
end

% 50% overlap
n=1;
```

```
for i=1:Navg1*2-1
    if i==1
        accel1(i,1:Ntime1) = accel_data1(1,1:Ntime1);
    else
        accel1(i,1:Ntime1) = accel_data1(1,n*(Ntime1/2)+1:2047+(n*(Ntime1/2)+1));
        n=n+1;
    end
end

% FFT and Power Spectrum
for i=1:Navg1
    AP_nolap_Total1(i,1:Ntime1) = conj(fft(accel_nolap1(i,:))).*(fft(accel_nolap1(i,:)));
   % Auto Power Spectrum - No overlap
end
for i=1:Navg1*2-1
    AP_Total1(i,1:Ntime1) = conj(fft(accel1(i,:))).*(fft(accel1(i,:)));
   % Auto Power Spectrum - 50% overlap
end

AP_nolap1 = AP_nolap_Total1(:,1:481);     % Limiting to first 1200Hz magnitudes - No overlap
AP1 = AP_Total1(:,1:481);                 % Limiting to first 1200Hz magnitudes - 50% overlap
```

Time Block Size = 1024

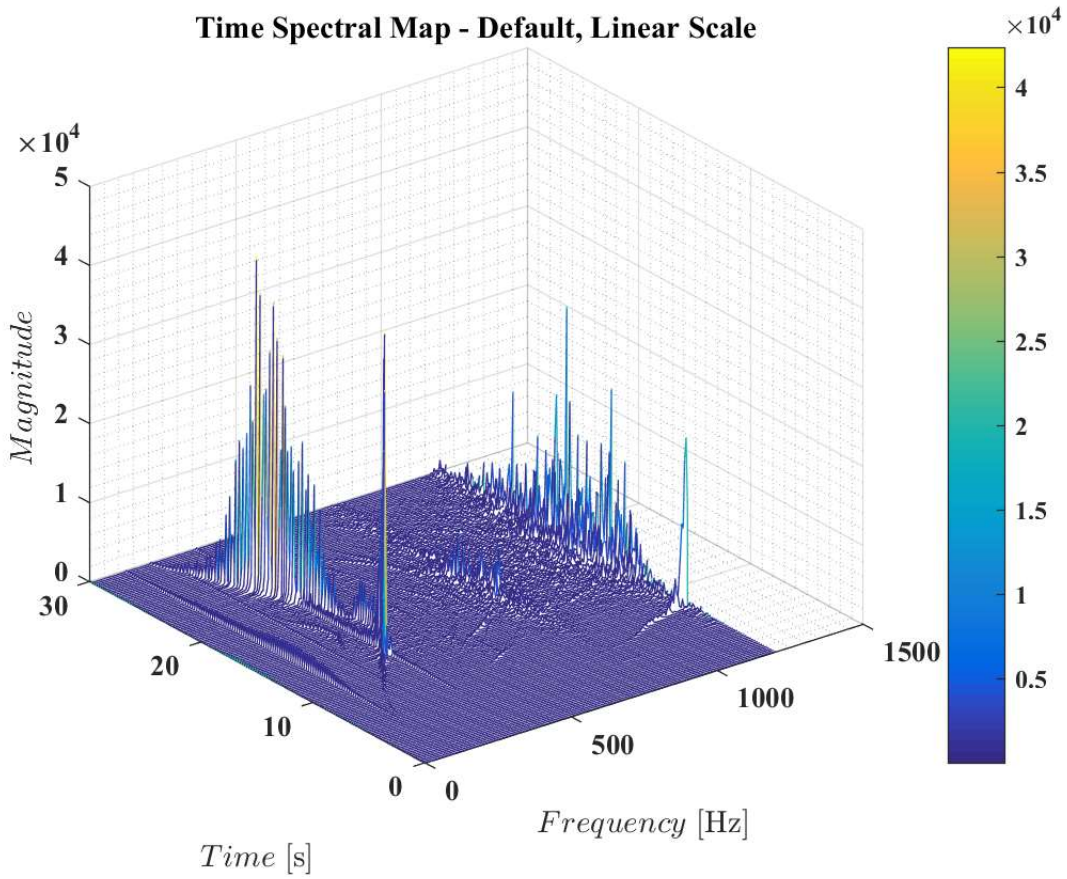
```
Ntime2 = 1024;
Navg2 = length(accel_data1)/Ntime2;

dt2 = 1/Fs;                         % Time Step
Ttime2 = Ntime2/Fs;                 % Time for one block
df2 = 1/Ttime2;                     % Frequency Resolution

f_xaxis2 = 0:df2:1200;               % Frequency axis
t_yaxis2 = 0:Ttime2:Ttotal-Ttime2; % Time axis

% No Overlap
n=1;
for i=1:Navg2
    if i==1
        accel_nolap2(i,1:Ntime2) = accel_data1(1,1:Ntime2);
    else
        accel_nolap2(i,1:Ntime2) = accel_data1(1,(n*Ntime2)+1:(n+1)*Ntime2);
        n=n+1;
    end
end

% 50% overlap
n=1;
for i=1:Navg2*2-1
    if i==1
        accel2(i,1:Ntime2) = accel_data1(1,1:Ntime2);
    else
        accel2(i,1:Ntime2) = accel_data1(1,n*(Ntime2/2)+1:1023+(n*(Ntime2/2)+1));
        n=n+1;
    end
end
```

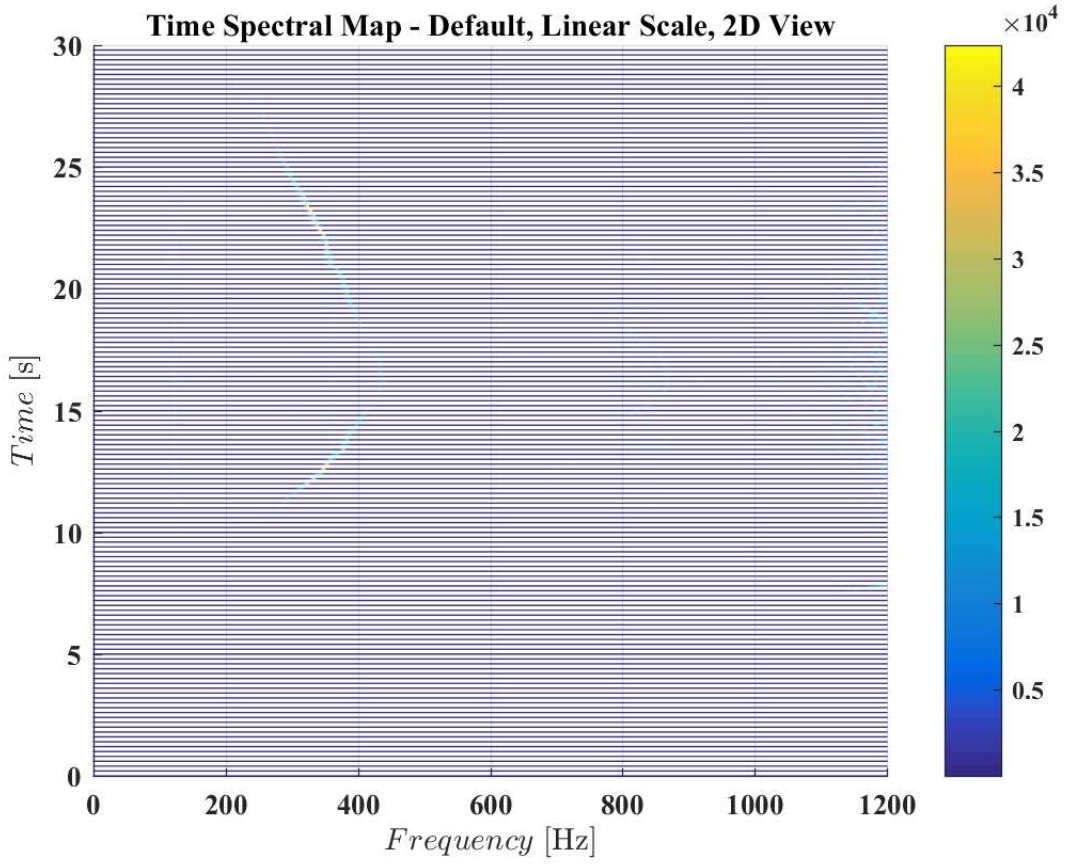
```
% FFT and Power Spectrum
for i=1:Navg2
    AP_nolap_Total2(i,1:Ntime2) = conj(fft(accel_nolap2(i,:))).*(fft(accel_nolap2(i,:)));
   % Auto Power Spectrum - No overlap
end
for i=1:Navg2*2-1
    AP_Total2(i,1:Ntime2) = conj(fft(accel2(i,:))).*(fft(accel2(i,:)));
   % Auto Power Spectrum - 50% overlap
end

AP_nolap2 = AP_nolap_Total2(:,1:241);     % Limiting to first 1200Hz magnitudes - No overlap
AP2 = AP_Total2(:,1:241);                 % Limiting to first 1200Hz magnitudes - 50% overlap
```

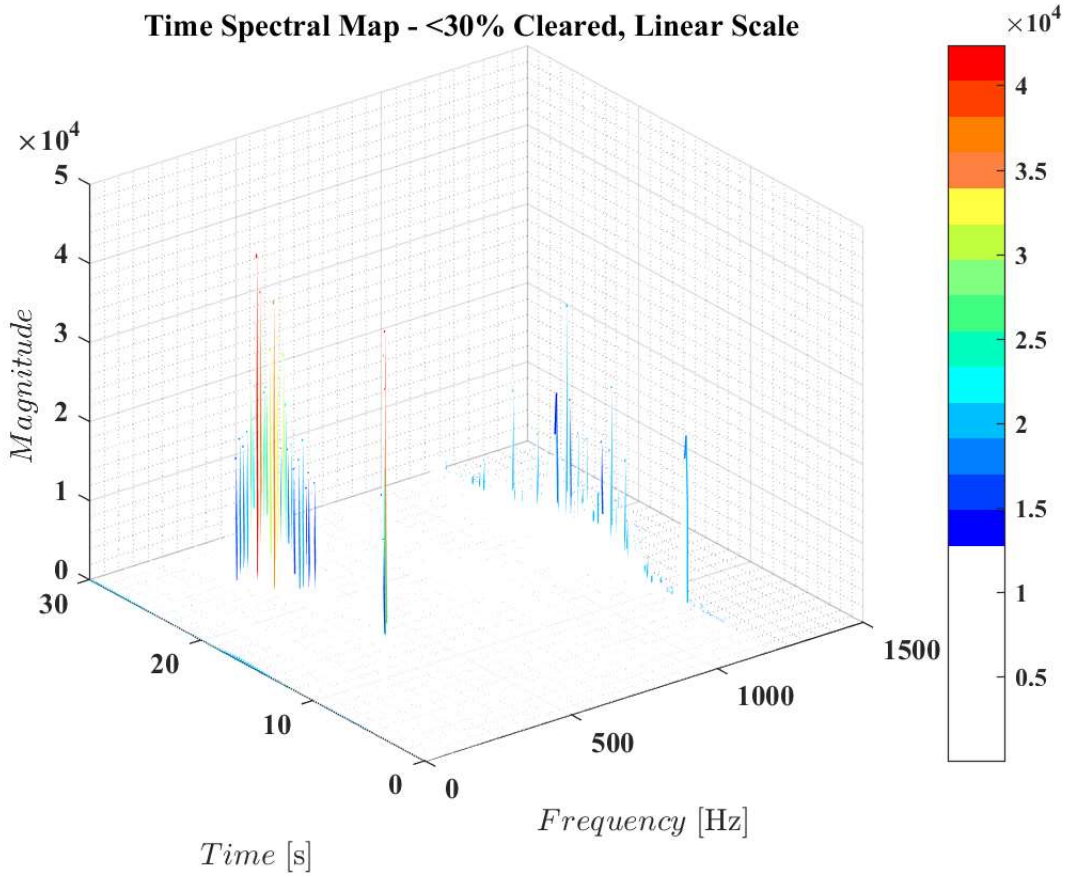
Plots - Linear Scale

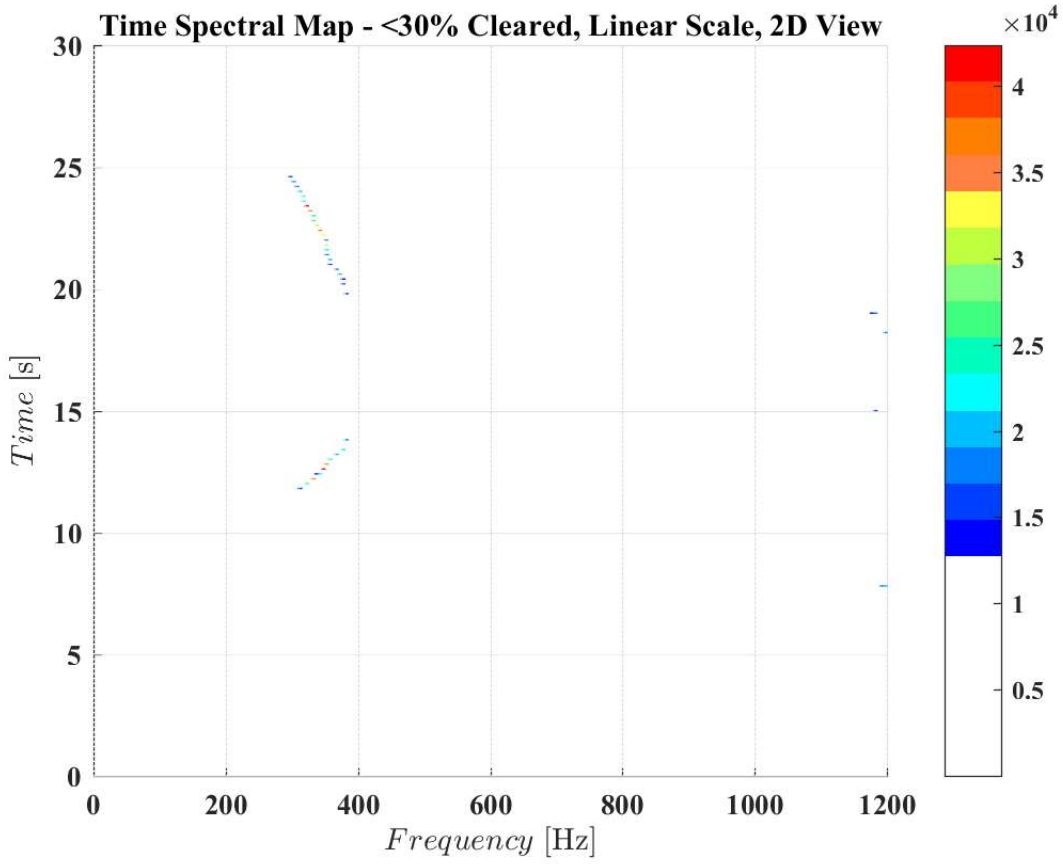
```
figure(1)
waterfall(f_xaxis2,t_yaxis2,AP_nolap2)      % 1024 time block; Default map; Linear scale
title('Time Spectral Map - Default, Linear Scale')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid minor
colorbar

figure(2)
waterfall(f_xaxis2,t_yaxis2,AP_nolap2)      % 1024 time block; Default map; Linear scale; 2D
View
title('Time Spectral Map - Default, Linear Scale, 2D View')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)

figure(3)
waterfall(f_xaxis2,t_yaxis2,AP_nolap2)      % 1024 time block; Linear scale; Below 30% cleare
d
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('Time Spectral Map - <30% Cleared, Linear Scale ')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid minor
colorbar

figure(4)
waterfall(f_xaxis2,t_yaxis2,AP_nolap2)      % 1024 time block; Default map; Linear scale; 2D
View; Below 30% cleared
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('Time Spectral Map - <30% Cleared, Linear Scale, 2D View')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
```

```
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)
grid on
```




Time Spectral Map - Default, Linear Scale, 2D View
×10^4
4
3.5
3
2.5
2
1.5
1
0.5
30
25
20
15
10
5
0
Time [s]
0
200
400
600
800
1000
1200
Frequency [Hz]

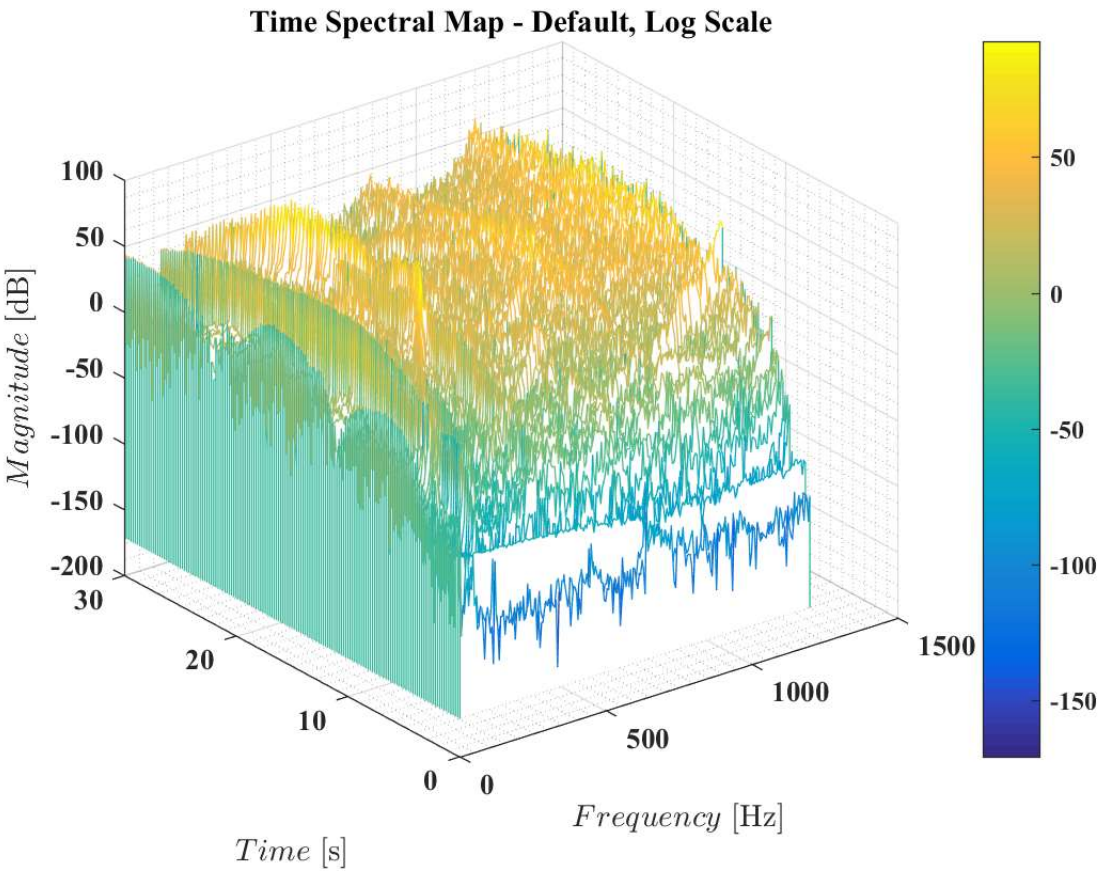

Time Spectral Map - <30% Cleared, Linear Scale
×10^4
4
3.5
3
2.5
2
1.5
1
0.5
×10^4
5
4
3
2
1
0
Magnitude
30
20
10
0
Time [s]
0
500
1000
1500
Frequency [Hz]

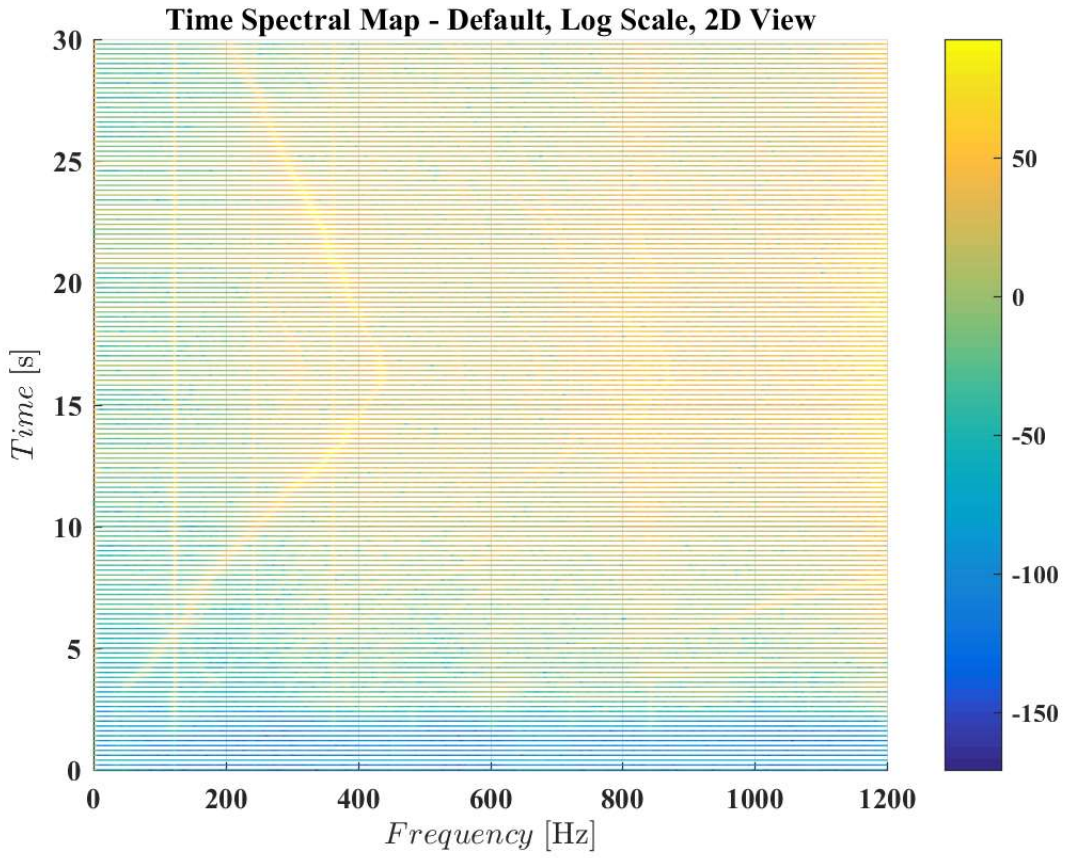


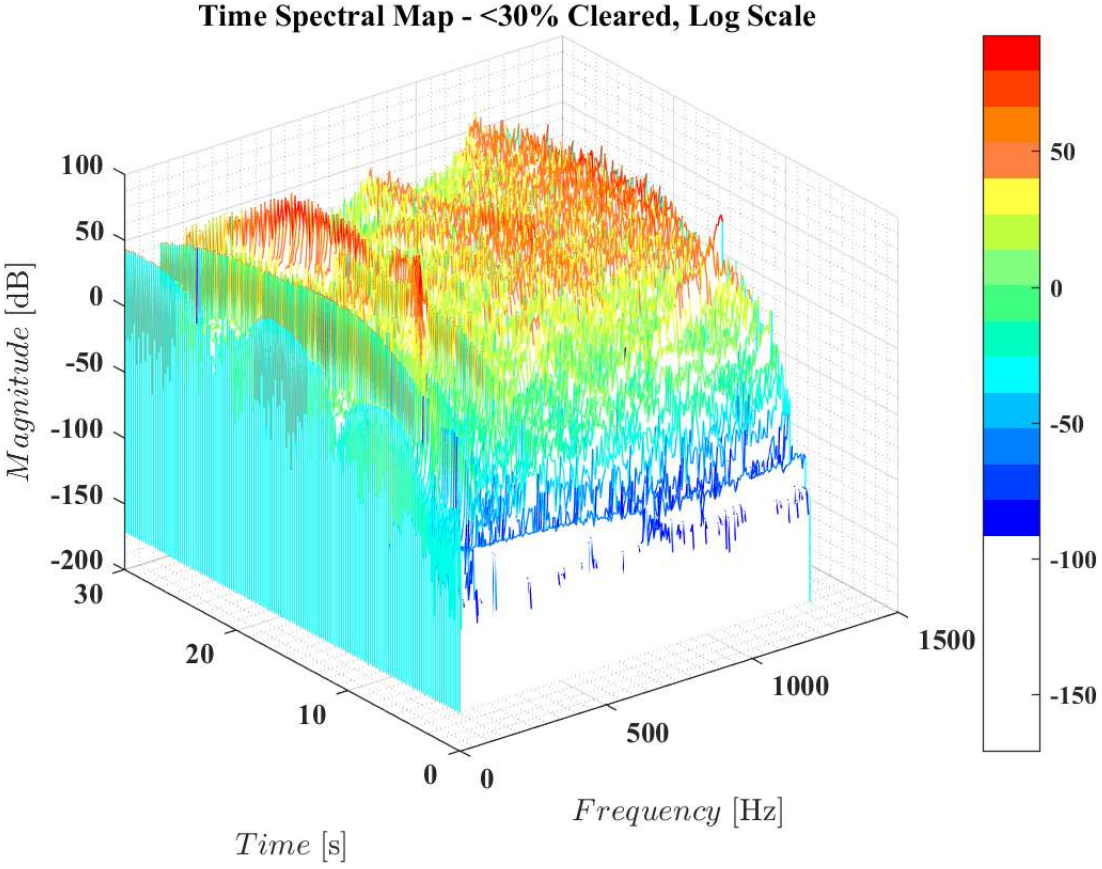
Plots - Log Scale

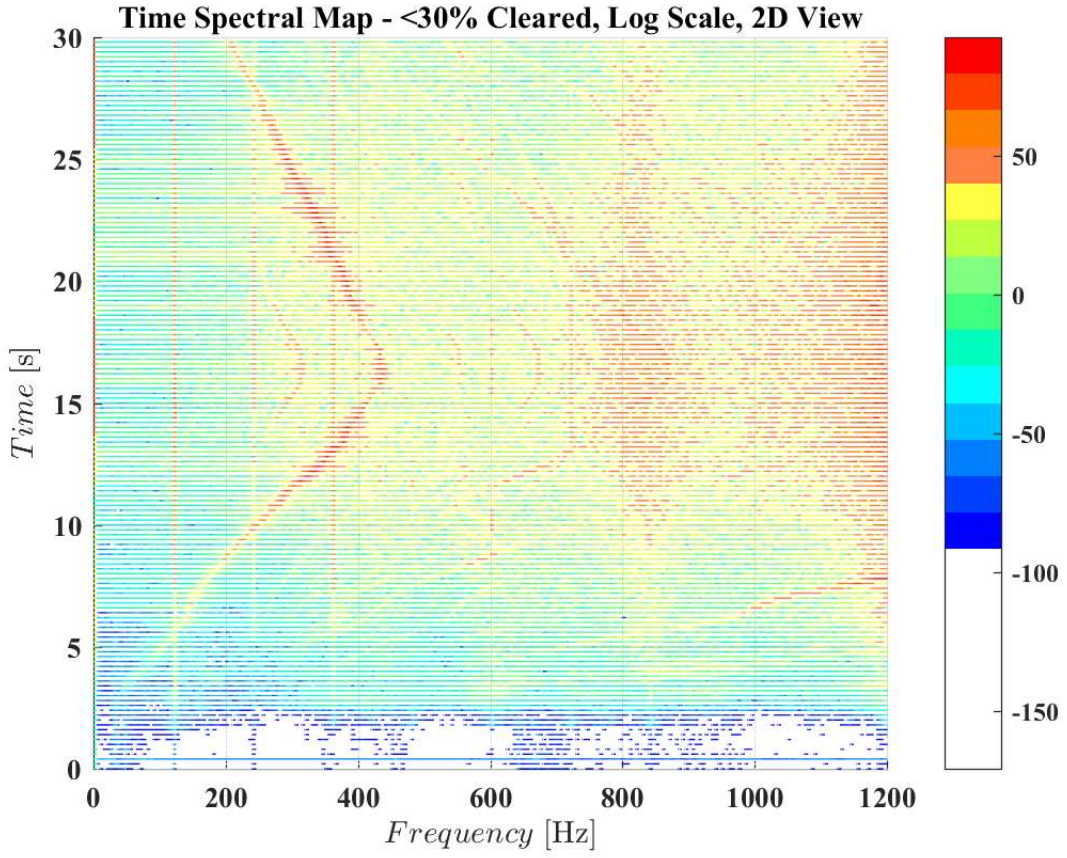
```
AP_nolap2_log = mag2db(AP_nolap2);
figure(5)
waterfall(f_xaxis2,t_yaxis2,AP_nolap2_log)      % 1024 time block; Default map; Linear scale
title('Time Spectral Map - Default, Log Scale')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{[dB]} $'],'interpreter','latex')
grid minor
colorbar

figure(6)
waterfall(f_xaxis2,t_yaxis2,AP_nolap2_log)      % 1024 time block; Default map; Linear scale;
 2D View
title('Time Spectral Map - Default, Log Scale, 2D View')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{[dB]} $'],'interpreter','latex')
colorbar
view(0,90)

figure(7)
waterfall(f_xaxis2,t_yaxis2,AP_nolap2_log)      % 1024 time block; Default map; Linear scale;
 Below 30% cleared
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('Time Spectral Map - <30% Cleared, Log Scale ')
```

```
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{[dB]} $'],'interpreter','latex')
grid minor
colorbar

figure(8)
waterfall(f_xaxis2,t_yaxis2,AP_nolap2_log)      % 1024 time block; Default map; Linear scale;
 2D View; Below 30% cleared
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('Time Spectral Map - <30% Cleared, Log Scale, 2D View')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{[dB]} $'],'interpreter','latex')
colorbar
view(0,90)
grid on
```




Time Spectral Map - Default, Log Scale, 2D View
Time [s]
Frequency [Hz]
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
50
0
-50
-100
-150


Time Spectral Map - <30% Cleared, Log Scale
Magnitude [dB]
Time [s]
Frequency [Hz]
100
50
0
-50
-100
-150
-200
30
20
10
0
0
500
1000
1500
50
0
-50
-100
-150



Published with MATLAB® R2015b

Contents

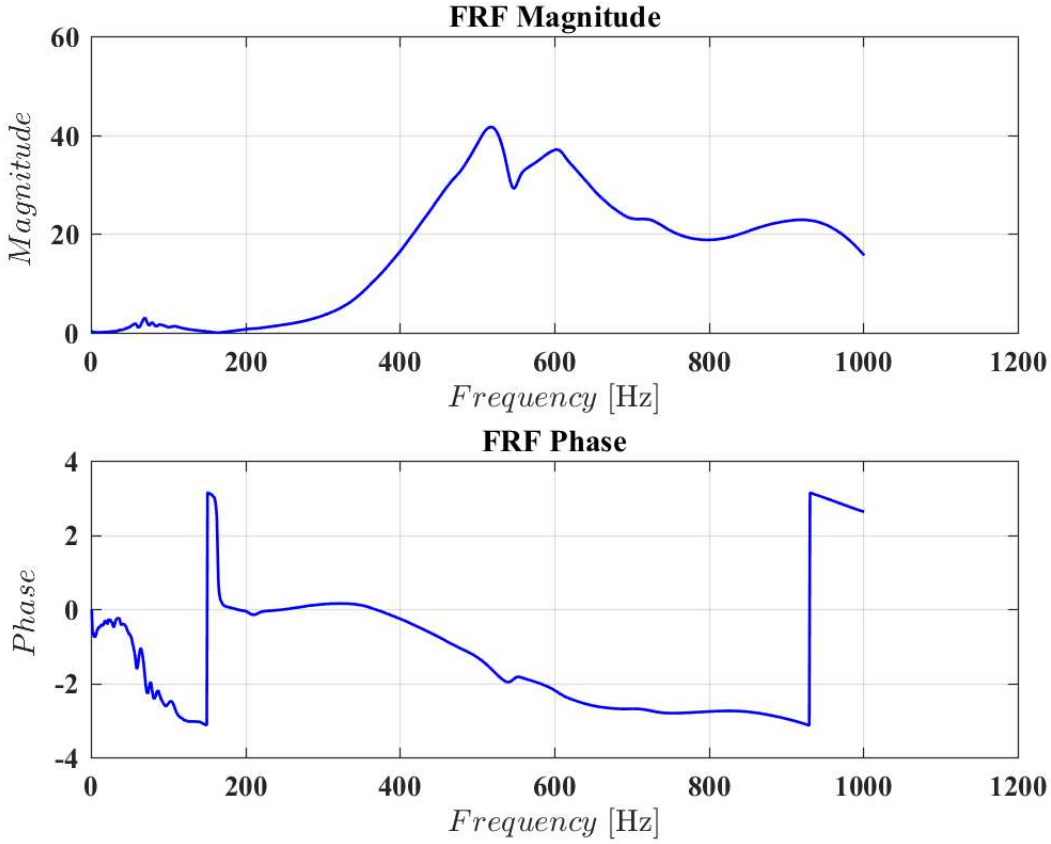
- Experimental Vibrations of Rotating Systems - Homework 2 - Shashank Iyengar (M12934513)
- Magnitude and Phase of FRF
- Ratio - This ratio represents Fa/del_Fa = m/del_m
- Final Response vs. Initial Response
- Single Plane Balancing - Magnitude and location of unbalance in plane A
- Calculation of theta

Experimental Vibrations of Rotating Systems - Homework 2 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('hwk2_data.mat')
```

Magnitude and Phase of FRF

```
figure(1)
subplot(2,1,1)
plot(1:FRF_freqlines,abs(FRF),'-b')
title('FRF Magnitude')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid on
subplot(2,1,2)
plot(1:FRF_freqlines,angle(FRF),'-b')
title('FRF Phase')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Phase\;\mathrm{} $'],'interpreter','latex')
grid on
```



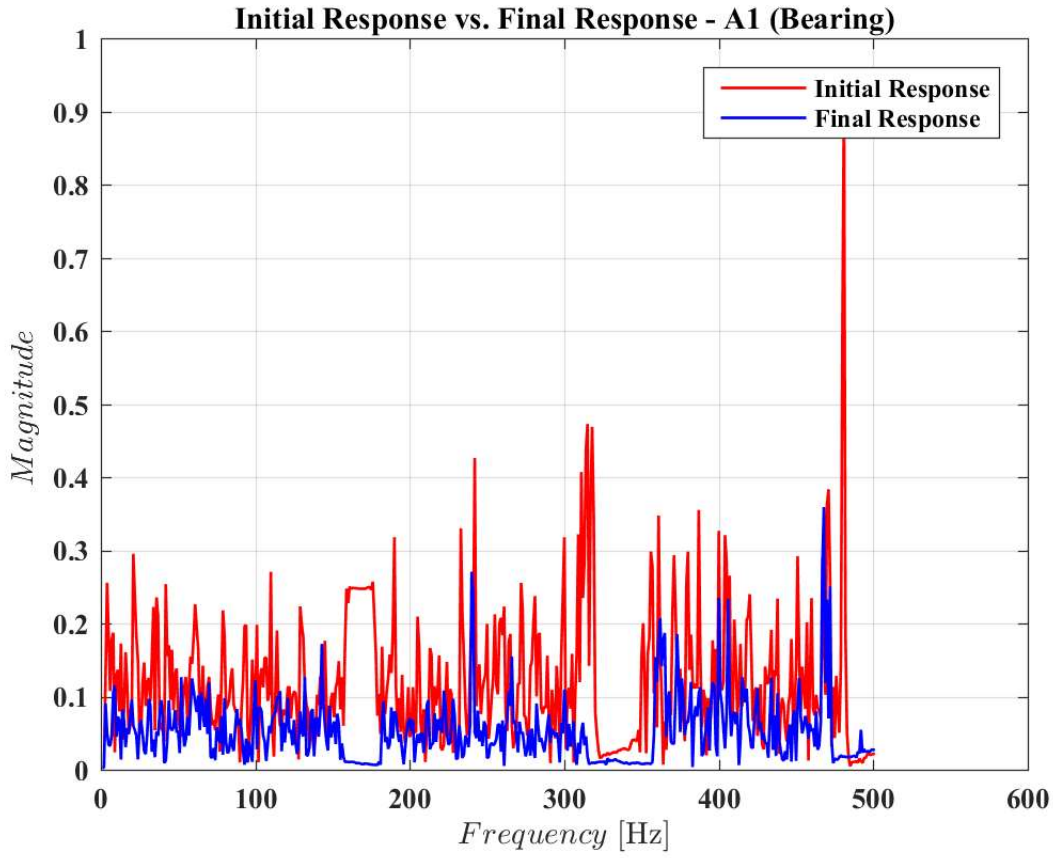
Ratio - This ratio represents Fa/del_Fa = m/del_m

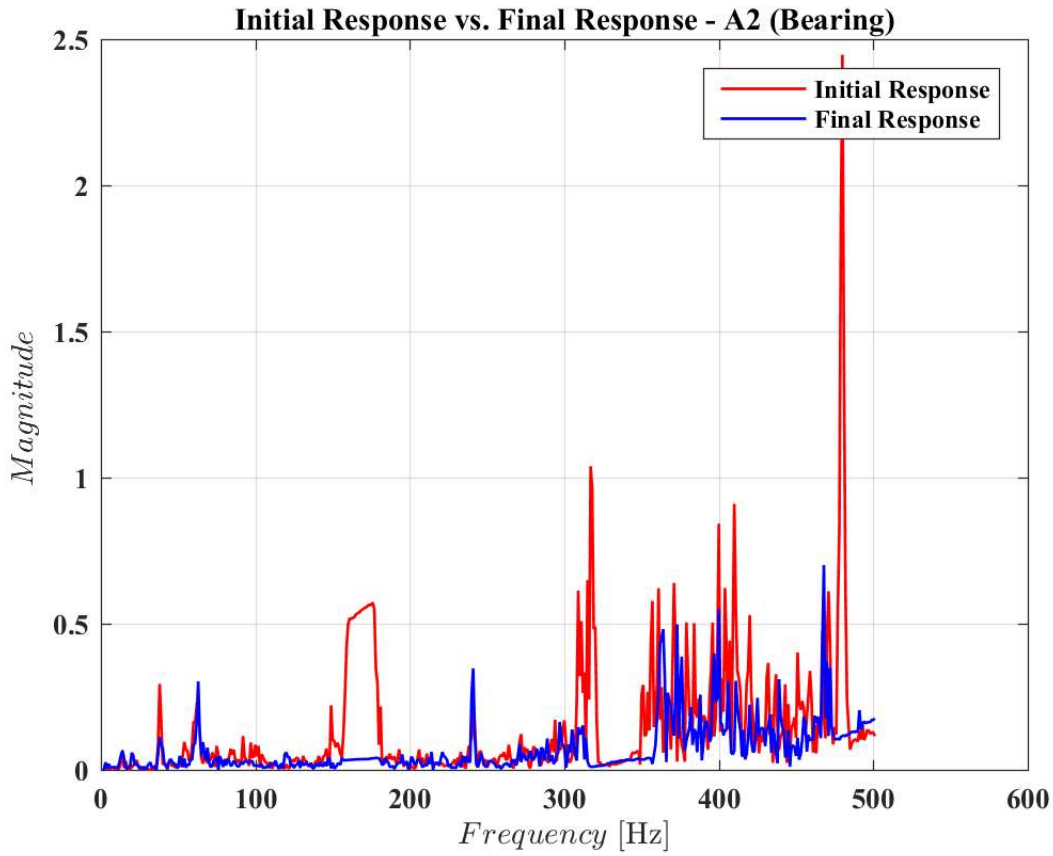
```
n=1;
for w=1:501
    Ratio_A(w) = (A1(w)*(A2b(w)-A2(w))-A2(w)*(A1b(w)-A1(w)))/((A2b(w)-A2(w))*(A1a(w)-A1(w))-(
A1b(w)-A1(w))*(A2a(w)-A2(w)));
    m_A(w) = Ratio_A(w)*(9/5);
    % m_A(w) is a complex number. The real part signifies the magnitude of
    % unbalance and the imaginary part signifies the location of the
    % unbalance with respect to the added mass

    % Similarly
    Ratio_B(w) = (A2(w)*(A1a(w)-A1(w))-A1(w)*(A2a(w)-A2(w)))/((A2b(w)-A2(w))*(A1a(w)-A1(w))-(
A1b(w)-A1(w))*(A2a(w)-A2(w)));
    m_B(w) = Ratio_B(w)*(5/9);
end
```

Final Response vs. Initial Response

```
figure(2)
plot(1:freqlines,abs(A1),'-r')
hold on
plot(1:freqlines,abs(A1f),'-b')
title('Initial Response vs. Final Response - A1 (Bearing)')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
legend('Initial Response','Final Response')
grid on
```

```
figure(3)
plot(1:freqlines,abs(A2),'-r')
hold on
plot(1:freqlines,abs(A2f),'-b')
title('Initial Response vs. Final Response - A2 (Bearing)')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
legend('Initial Response','Final Response')
grid on
```





Single Plane Balancing - Magnitude and location of unbalance in plane A

```
for w=1:501
    a1(w) = real(A1(w,1));
    b1(w) = imag(A1(w,1));
    a2(w) = real(A1a(w,1));
    b2(w) = imag(A1b(w,1));
end

% Calculation of B and C
for w=1:501
    B(w) = sqrt(a1(w)^2 + b1(w)^2);
    C(w) = sqrt((a2(w)-a1(w))^2 + (b2(w)-b1(w))^2);
end

R = B./C;
```

Calculation of theta

```
for w=1:501
    theta1 = atan2(b1(w),a1(w));
    theta2 = atan2((b2(w)-b1(w)),(a2(w)-a1(w)));
    theta(w) = theta1-theta2;
end
```

Published with MATLAB® R2015b

Contents

- Experimental Vibrations of Rotating Systems - Homework 3 - Shashank Iyengar (M12934513)
- Pre-processing
- Centering of tach signal
- Zero-level crossing
- RPM Estimate

Experimental Vibrations of Rotating Systems - Homework 3 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('hwk3_data.mat')       % Loaded variables: Tach1 and Tach2
```

Pre-processing

```
Fs = 5120;                  % Sampling Frequency
T = 30;                     % Total signal time
dt = 1/Fs;                  % Time instant
tt = 0:dt:T-dt;             % Time axis
```

Centering of tach signal

```
Tach1_mean = mean(Tach1);   % Mean of Tach1 data
Tach2_mean = mean(Tach2);   % Mean of Tach2 data

Tach1_centered = Tach1 - Tach1_mean;    % Centered tach signal
Tach2_centered = Tach2 - Tach2_mean;    % Centered tach signal
```

Zero-level crossing

Tach Signal 1

```
n=1;
for i=1:length(Tach1_centered)-1
    if sign(Tach1_centered(1,i+1)) > sign(Tach1_centered(1,i))
        z_cross1(n) = ((0 - Tach1_centered(1,i))*((tt(1,i+1)-tt(1,i))/(Tach1_centered(1,i+1)-
Tach1_centered(1,i)))) + tt(1,i);
        n=n+1;
    end
end
```

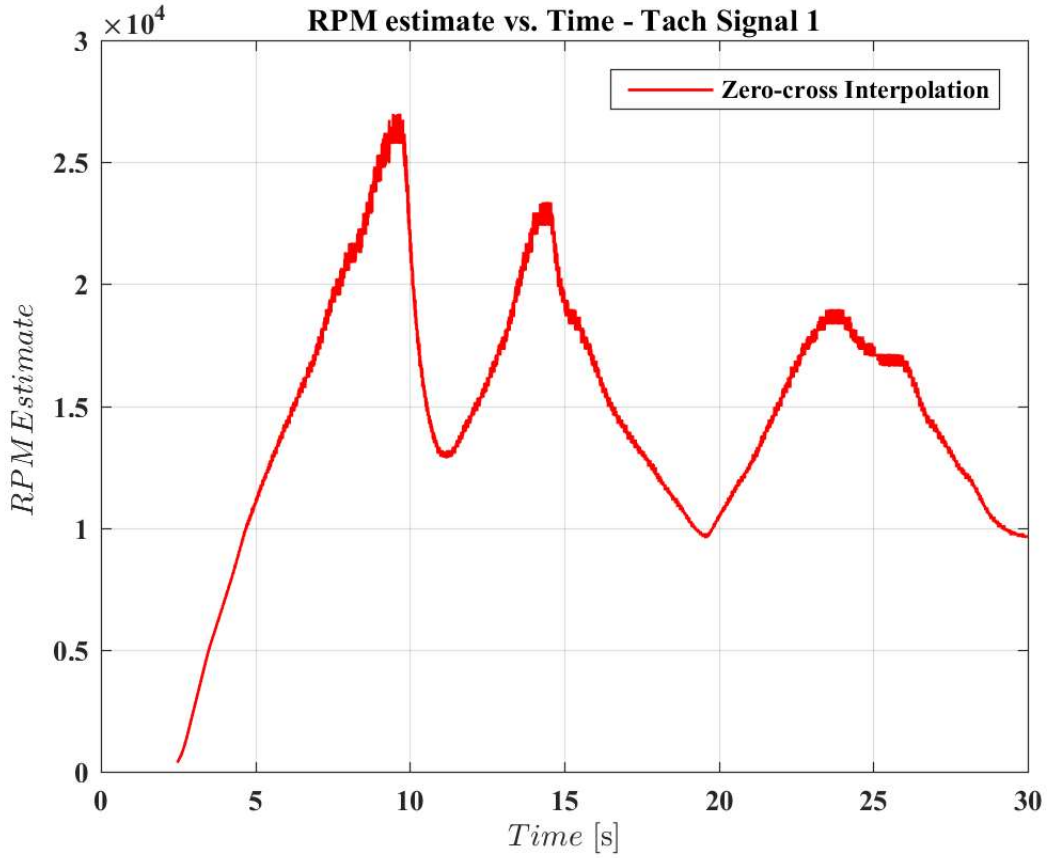
```
% Tach Signal 2
n=1;
for i=1:length(Tach2_centered)-1
    if sign(Tach2_centered(1,i+1)) > sign(Tach2_centered(1,i))
        z_cross2(n) = ((0 - Tach2_centered(1,i))*((tt(1,i+1)-tt(1,i))/(Tach2_centered(1,i+1)-
Tach2_centered(1,i)))) + tt(1,i);
        n=n+1;
    end
end
```

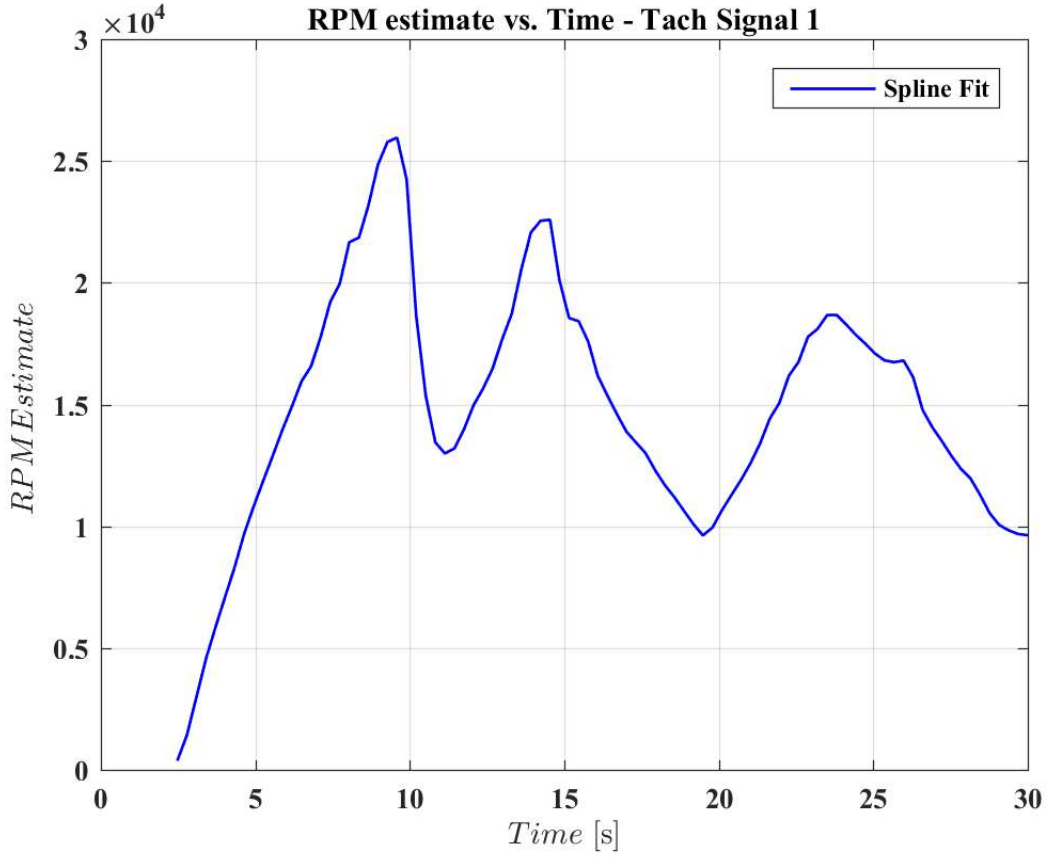
RPM Estimate

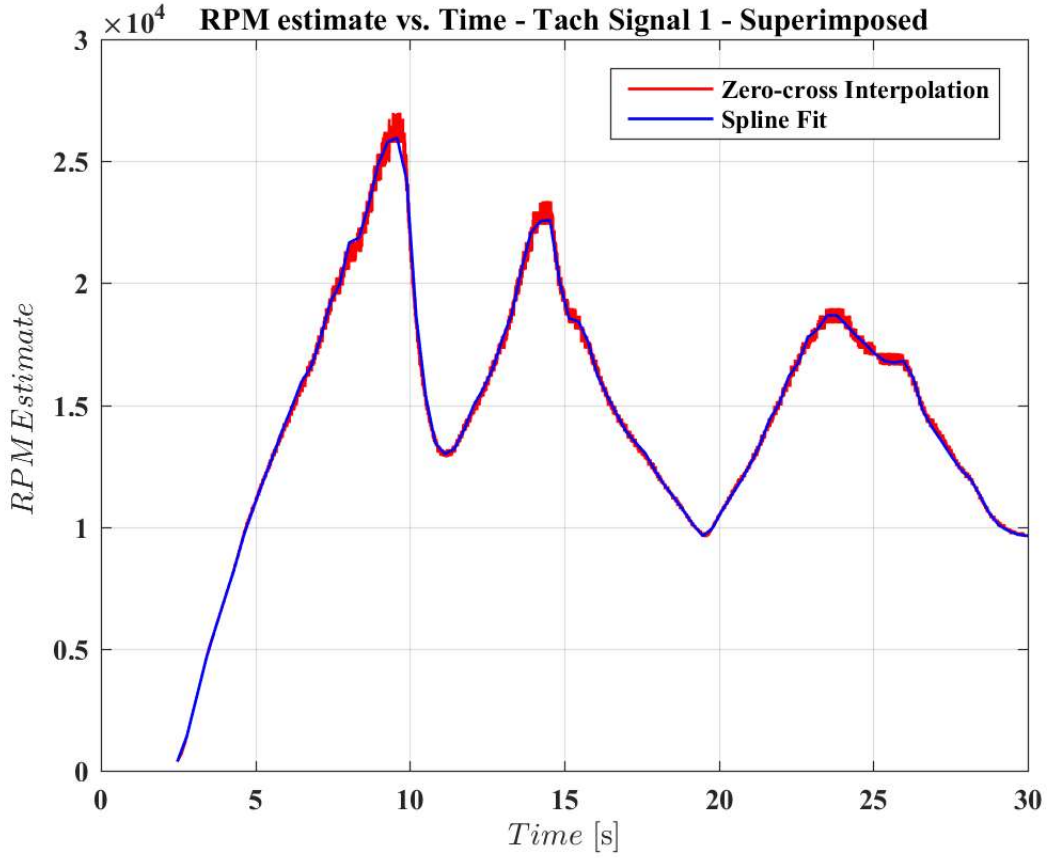
Tach Signal 1

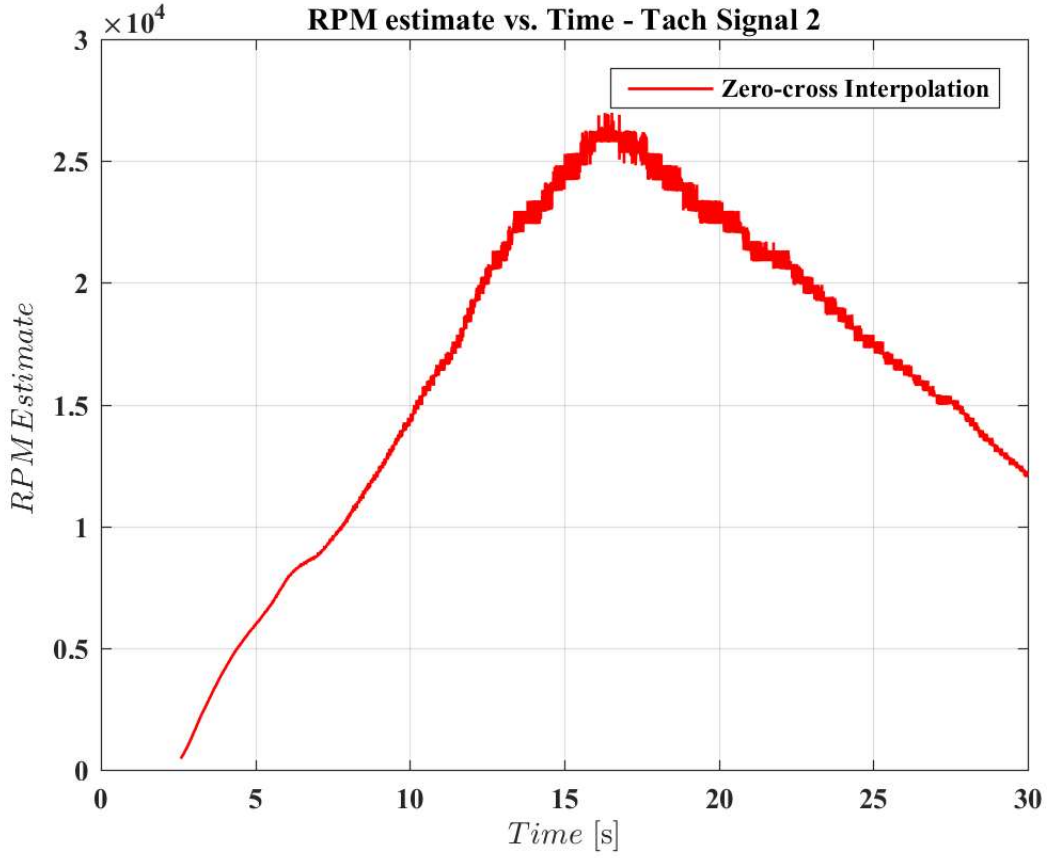
```
for i=1:length(z_cross1)-1
    RPM1(i) = 60/(z_cross1(1,i+1)-z_cross1(1,i));       % RPM estimate
    t_RPM1(i) = (z_cross1(1,i)+z_cross1(1,i+1))/2;      % Time
end
t_spline1 = linspace(t_RPM1(1),30,90);
RPM_spline1 = spline(t_RPM1,RPM1,t_spline1);            % RPM spline curve

figure(1)
plot(t_RPM1,RPM1,'-r')
title('RPM estimate vs. Time - Tach Signal 1')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Zero-cross Interpolation')
grid on

figure(2)
plot(t_spline1,RPM_spline1,'-b')
title('RPM estimate vs. Time - Tach Signal 1')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Spline Fit')
grid on

figure(3)
plot(t_RPM1,RPM1,'-r')
hold on
plot(t_spline1,RPM_spline1,'-b')
title('RPM estimate vs. Time - Tach Signal 1 - Superimposed')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Zero-cross Interpolation','Spline Fit')
grid on

% Tach Signal 2
for i=1:length(z_cross2)-1
    RPM2(i) = 60/(z_cross2(1,i+1)-z_cross2(1,i));       % RPM Estimate
    t_RPM2(i) = (z_cross2(1,i)+z_cross2(1,i+1))/2;      % Time
end
t_spline2 = linspace(t_RPM2(1),30,50);
RPM_spline2 = spline(t_RPM2,RPM2,t_spline2);            % RPM spline curve
```

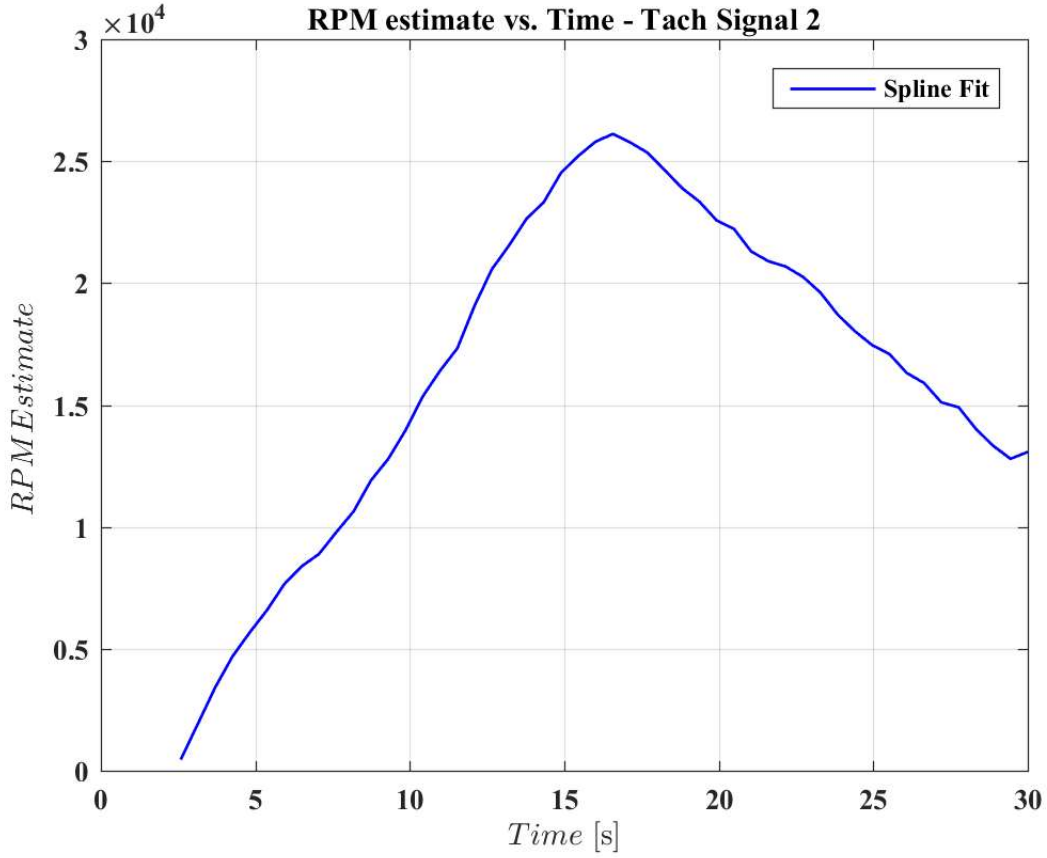
```
figure(4)
plot(t_RPM2,RPM2,'-r')
title('RPM estimate vs. Time - Tach Signal 2')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Zero-cross Interpolation')
grid on

figure(5)
plot(t_spline2,RPM_spline2,'-b')
title('RPM estimate vs. Time - Tach Signal 2')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Spline Fit')
grid on

figure(6)
plot(t_RPM2,RPM2,'-r')
hold on
plot(t_spline2,RPM_spline2,'-b')
title('RPM estimate vs. Time - Tach Signal 2 - Superimposed')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Zero-cross Interpolation','Spline Fit')
grid on
```

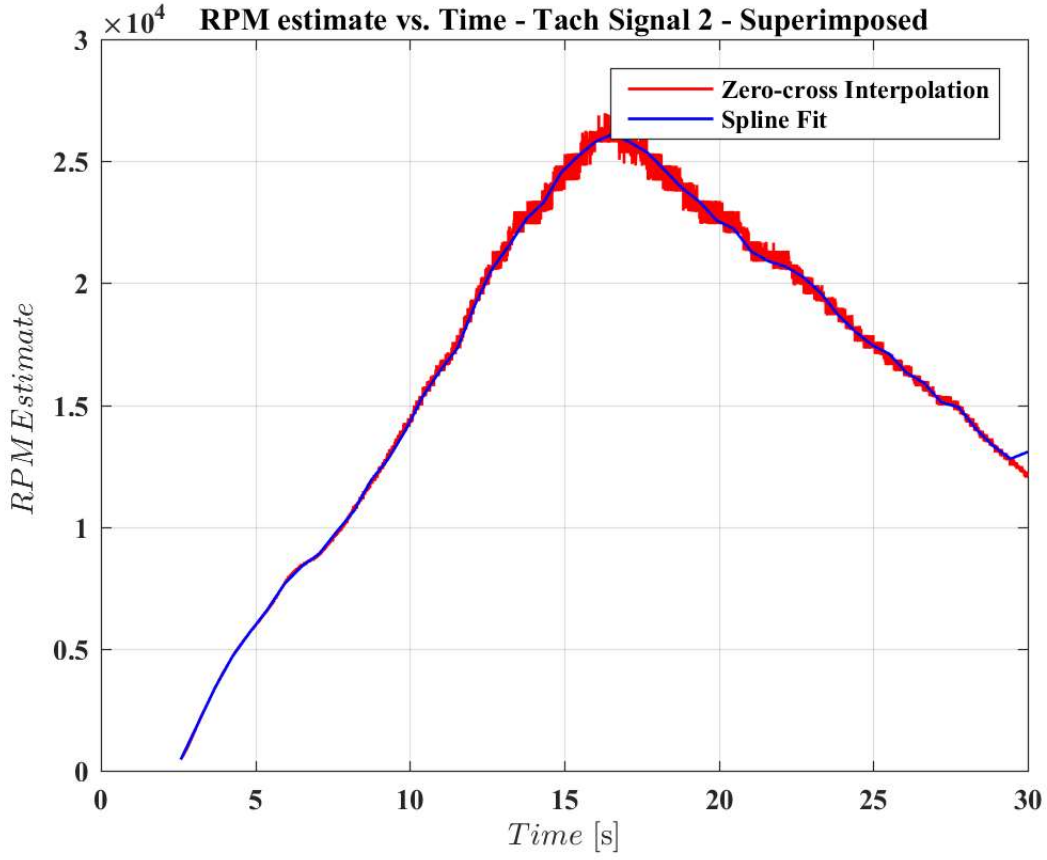



RPM estimate vs. Time - Tach Signal 1
×10^4
Spline Fit
RPMEstimate
Time [s]
0
0.5
1
1.5
2
2.5
3
0
5
10
15
20
25
30


RPM estimate vs. Time - Tach Signal 1 - Superimposed
×10^4
Zero-cross Interpolation
Spline Fit
RPMEstimate
Time [s]
0
0.5
1
1.5
2
2.5
3
0
5
10
15
20
25
30


RPM estimate vs. Time - Tach Signal 2
×10^4
Zero-cross Interpolation
RPM Estimate
Time [s]
0
0.5
1
1.5
2
2.5
3
0
5
10
15
20
25
30


RPM estimate vs. Time - Tach Signal 2
×10^4
Spline Fit
RPM Estimate
Time [s]
0
0.5
1
1.5
2
2.5
3
0
5
10
15
20
25
30



Published with MATLAB® R2015b

Contents

- Experimental Vibrations of Rotating Systems - Homework 4 - Shashank Iyengar (M12934513)
- Time Block Size = 512
- Time Block Size = 2048
- Time Block Size = 1024
- Plots - Linear Scale
- Plots - Log Scale

Experimental Vibrations of Rotating Systems - Homework 4 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 15)
set(0,'defaultlinelinewidth',0.1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('Hwk4_data.mat')       % Loaded variables: Tach1, Fsamp, data
```

Time Block Size = 512

```
Ntime = 512;
Fs = Fsamp;
Navg = length(data(1,:))/Ntime;

dt = 1/Fs;                          % Time Step
N = length(data(1,:));              % Total number of sample points
Ttime = Ntime/Fs;                   % Time for one block
Ttotal = N/Fs;                      % Time for complete signal
df = 1/Ttime;                       % Frequency Resolution

f_xaxis = 0:df:1200;                % Frequency axis
tt = 0:dt:Ttotal-dt; % Time axis

% No Overlap
n=1;
for i=1:Navg
    if i==1
        accel_nolap(i,1:Ntime) = data(1,1:Ntime);
    else
        accel_nolap(i,1:Ntime) = data(1,(n*Ntime)+1:(n+1)*Ntime);
        n=n+1;
    end
end

% FFT and Power Spectrum
for i=1:Navg
    AP_nolap_Total(i,1:Ntime) = conj(fft(accel_nolap(i,:))).*(fft(accel_nolap(i,:)));       %
 Auto Power Spectrum - No overlap
```

```
end

AP_nolap = AP_nolap_Total(:,1:121);         % Limiting to first 1200Hz magnitudes - No overla
p

% Centering of tach signal
Tach1_mean = mean(Tach1);
Tach1_centered = Tach1 - Tach1_mean;

% Zero-level crossing
n=1;
for i=1:length(Tach1_centered)-1
    if sign(Tach1_centered(1,i+1)) > sign(Tach1_centered(1,i))
        z_cross1(n) = ((0 - Tach1_centered(1,i))*((tt(1,i+1)-tt(1,i))/(Tach1_centered(1,i+1)-
Tach1_centered(1,i)))) + tt(1,i);
        n=n+1;
    end
end

% RPM Estimate
for i=1:length(z_cross1)-1
    RPM1(i) = 60/(z_cross1(1,i+1)-z_cross1(1,i));
    t_RPM1(i) = (z_cross1(1,i)+z_cross1(1,i+1))/2;
end
t_spline1 = linspace(0,30,300);
RPM_spline1 = spline(t_RPM1,RPM1,t_spline1);
```

Time Block Size = 2048

```
Ntime1 = 2048;
Fs = Fsamp;
Navg1 = length(data(1,:))/Ntime1;

dt = 1/Fs;                      % Time Step
N = length(data(1,:));          % Total number of sample points
Ttime1 = Ntime/Fs;               % Time for one block
Ttotal1 = N/Fs;                  % Time for complete signal
df = 1/Ttime1;                   % Frequency Resolution

f_xaxis = 0:df:1200;            % Frequency axis
tt = 0:dt:Ttotal-dt; % Time axis

% No Overlap
n=1;
for i=1:Navg1
    if i==1
        accel_nolap1(i,1:Ntime1) = data(1,1:Ntime1);
    else
        accel_nolap1(i,1:Ntime1) = data(1,(n*Ntime1)+1:(n+1)*Ntime1);
        n=n+1;
    end
end

% FFT and Power Spectrum
for i=1:Navg1
    AP_nolap_Total1(i,1:Ntime1) = conj(fft(accel_nolap1(i,:))).*(fft(accel_nolap1(i,:)));
```

```
    % Auto Power Spectrum - No overlap
end

AP_nolap1 = AP_nolap_Total1(:,1:481);          % Limiting to first 1200Hz magnitudes - No over
lap

% Centering of tach signal
Tach1_mean = mean(Tach1);
Tach1_centered = Tach1 - Tach1_mean;

% Zero-level crossing
n=1;
for i=1:length(Tach1_centered)-1
    if sign(Tach1_centered(1,i+1)) > sign(Tach1_centered(1,i))
        z_cross1(n) = ((0 - Tach1_centered(1,i))*((tt(1,i+1)-tt(1,i))/(Tach1_centered(1,i+1)-
Tach1_centered(1,i)))) + tt(1,i);
        n=n+1;
    end
end

% RPM Estimate
for i=1:length(z_cross1)-1
    RPM1(i) = 60/(z_cross1(1,i+1)-z_cross1(1,i));
    t_RPM1(i) = (z_cross1(1,i)+z_cross1(1,i+1))/2;
end
t_spline1 = linspace(t_RPM1(1),30,300);
RPM_spline1 = spline(t_RPM1,RPM1,t_spline1);
```

Time Block Size = 1024

```
Ntime2 = 1024;
Fs = Fsamp;
Navg2 = length(data(1,:))/Ntime2;

dt = 1/Fs;                          % Time Step
N = length(data(1,:));              % Total number of sample points
Ttime2 = Ntime2/Fs;                   % Time for one block
Ttotal2 = N/Fs;                     % Time for complete signal
df = 1/Ttime2;                      % Frequency Resolution

f_xaxis = 0:df:1200;                % Frequency axis
tt = 0:dt:Ttotal-dt;                % Time axis

% No Overlap
n=1;
for i=1:Navg2
    if i==1
        accel_nolap2(i,1:Ntime2) = data(1,1:Ntime2);
    else
        accel_nolap2(i,1:Ntime2) = data(1,(n*Ntime2)+1:(n+1)*Ntime2);
        n=n+1;
    end
end
% 87.5% overlap
n=1;
for i=1:Navg2*8-7
```

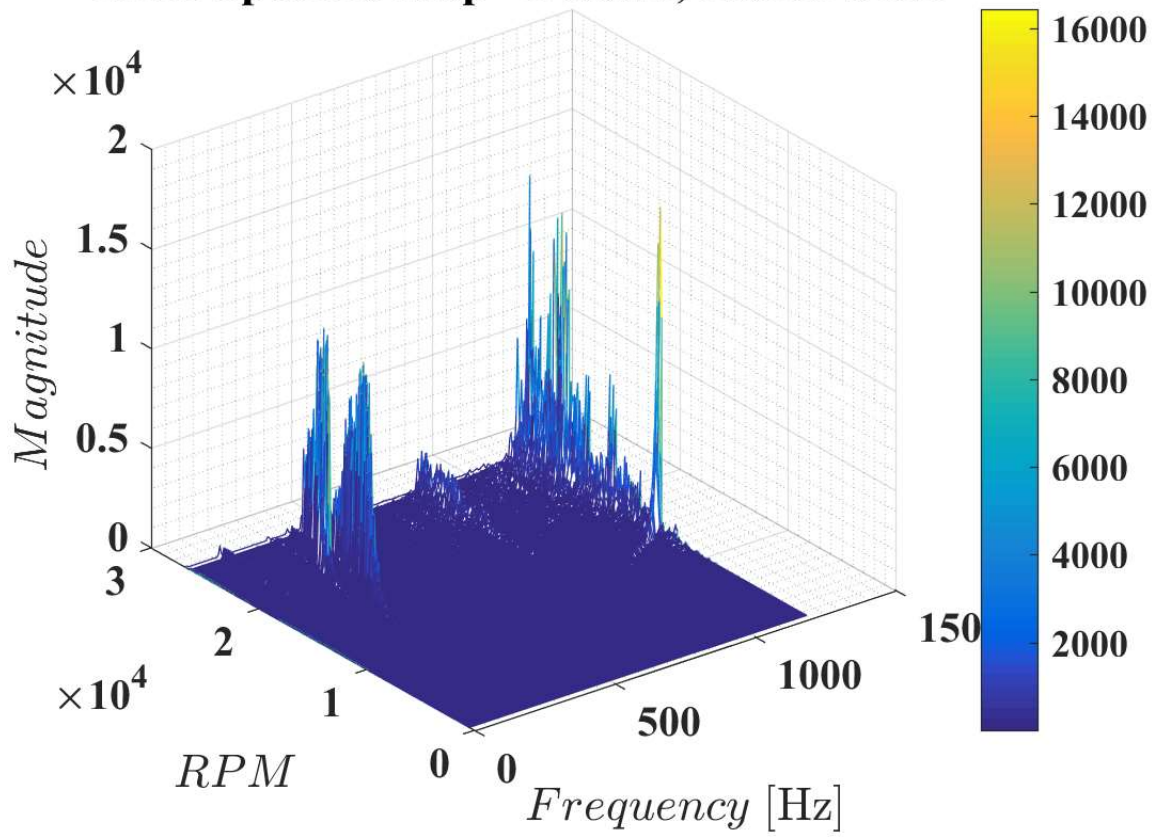
```
    if i==1
        accel2(i,1:Ntime2) = data(1,1:Ntime2);
        accel2_tr = accel2(i,1:Ntime2)';
        accel2_hann(i,1:Ntime2) = hann(Ntime2).*accel2_tr;
    else
        accel2(i,1:Ntime2) = data(1,n*(Ntime2/8)+1:1023+(n*(Ntime2/8)+1));
        accel2_tr = accel2(i,1:Ntime2)';
        accel2_hann(i,1:Ntime2) = hann(Ntime2).*accel2_tr;
        n=n+1;
    end
end

% FFT and Power Spectrum
for i=1:Navg2
    AP_nolap_Total2(i,1:Ntime2) = conj(fft(accel_nolap2(i,:))).*(fft(accel_nolap2(i,:)));
   % Auto Power Spectrum - No overlap
end
for i=1:Navg2*8-7
    AP_Total2(i,1:Ntime2) = conj(fft(accel2_hann(i,:))).*(fft(accel2_hann(i,:)));
             % Auto Power Spectrum - 50% overlap
end

AP_nolap2 = AP_nolap_Total2(:,1:241);         % Limiting to first 1200Hz magnitudes - No over
lap
AP2 = AP_Total2(:,1:241);                  % Limiting to first 1200Hz magnitudes - 50% overlap

% Centering of tach signal
Tach1_mean = mean(Tach1);
Tach1_centered = Tach1 - Tach1_mean;

% Zero-level crossing
n=1;
for i=1:length(Tach1_centered)-1
    if sign(Tach1_centered(1,i+1)) > sign(Tach1_centered(1,i))
        z_cross1(n) = ((0 - Tach1_centered(1,i))*((tt(1,i+1)-tt(1,i))/(Tach1_centered(1,i+1)-
Tach1_centered(1,i)))) + tt(1,i);
        n=n+1;
    end
end

% RPM Estimate
for i=1:length(z_cross1)-1
    RPM1(i) = 60/(z_cross1(1,i+1)-z_cross1(1,i));
    t_RPM1(i) = (z_cross1(1,i)+z_cross1(1,i+1))/2;
end
t_spline1 = linspace(t_RPM1(1),30,1193);
RPM_spline1 = spline(t_RPM1,RPM1,t_spline1);
```

Plots - Linear Scale

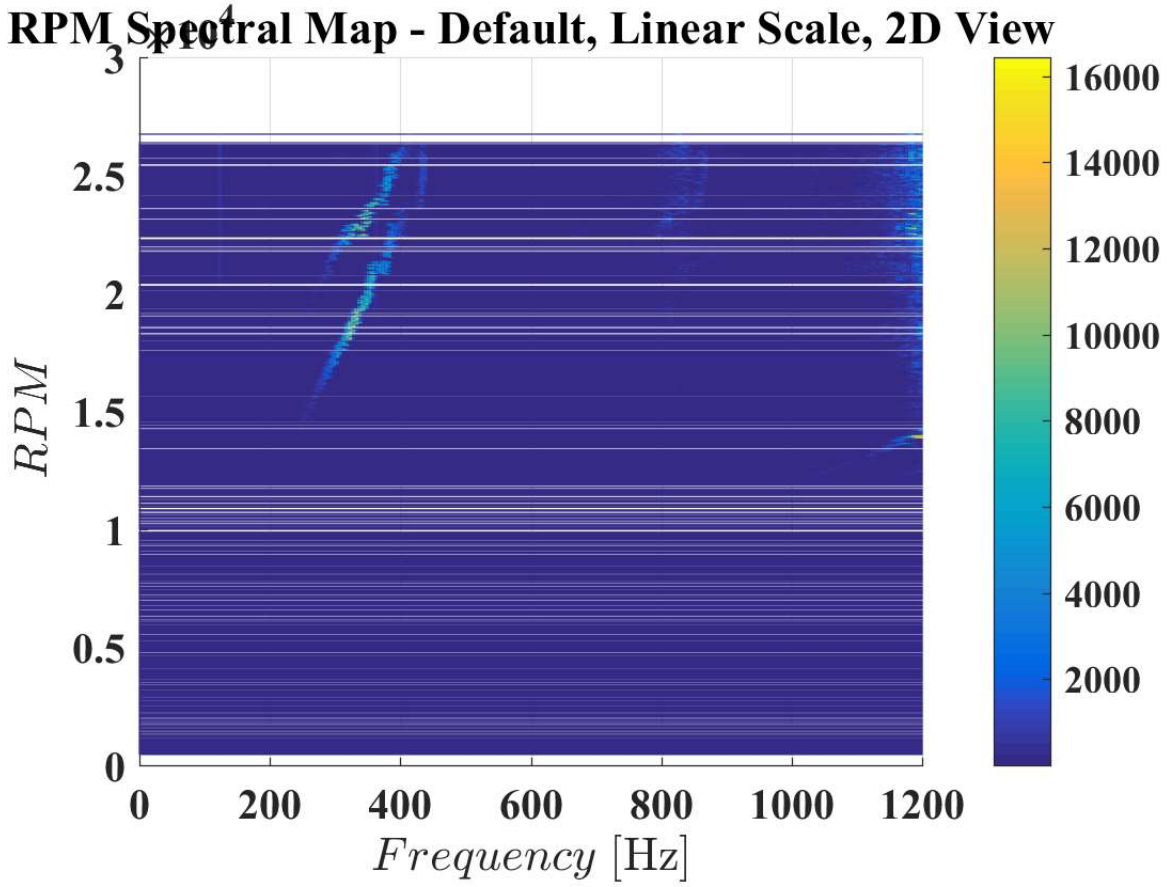
```
figure(1)
waterfall(f_xaxis,RPM_spline1,AP2)      % 1024 time block; Default map; Linear scale
title('RPM Spectral Map - Default, Linear Scale')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
```

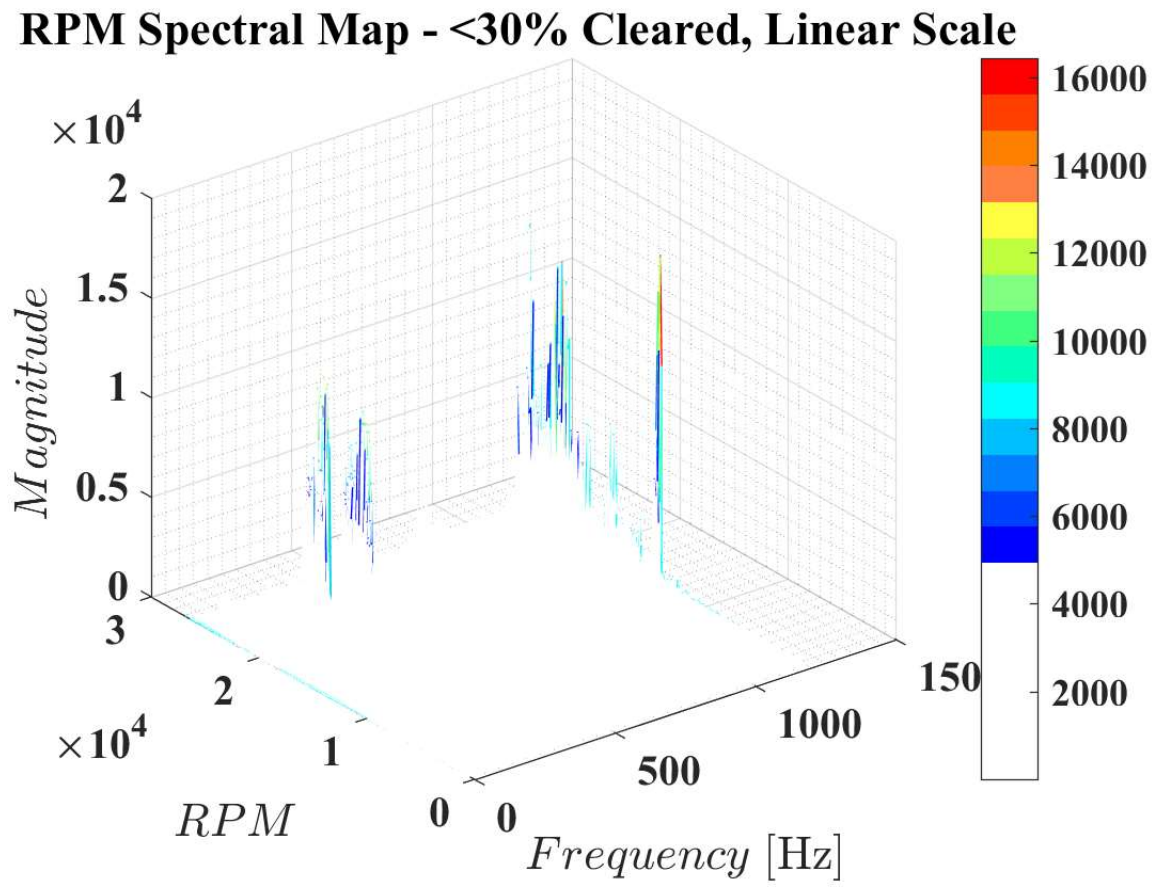
```
grid minor
colorbar

figure(2)
waterfall(f_xaxis,RPM_spline1,AP2)      % 1024 time block; Default map; Linear scale; 2D View
title('RPM Spectral Map - Default, Linear Scale, 2D View')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)

figure(3)
waterfall(f_xaxis,RPM_spline1,AP2)      % 1024 time block; Linear scale; Below 30% cleared
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('RPM Spectral Map - <30% Cleared, Linear Scale ')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid minor
colorbar

figure(4)
waterfall(f_xaxis,RPM_spline1,AP2)      % 1024 time block; Default map; Linear scale; 2D View
; Below 30% cleared
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('RPM Spectral Map - <30% Cleared, Linear Scale, 2D View')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)
grid on
```

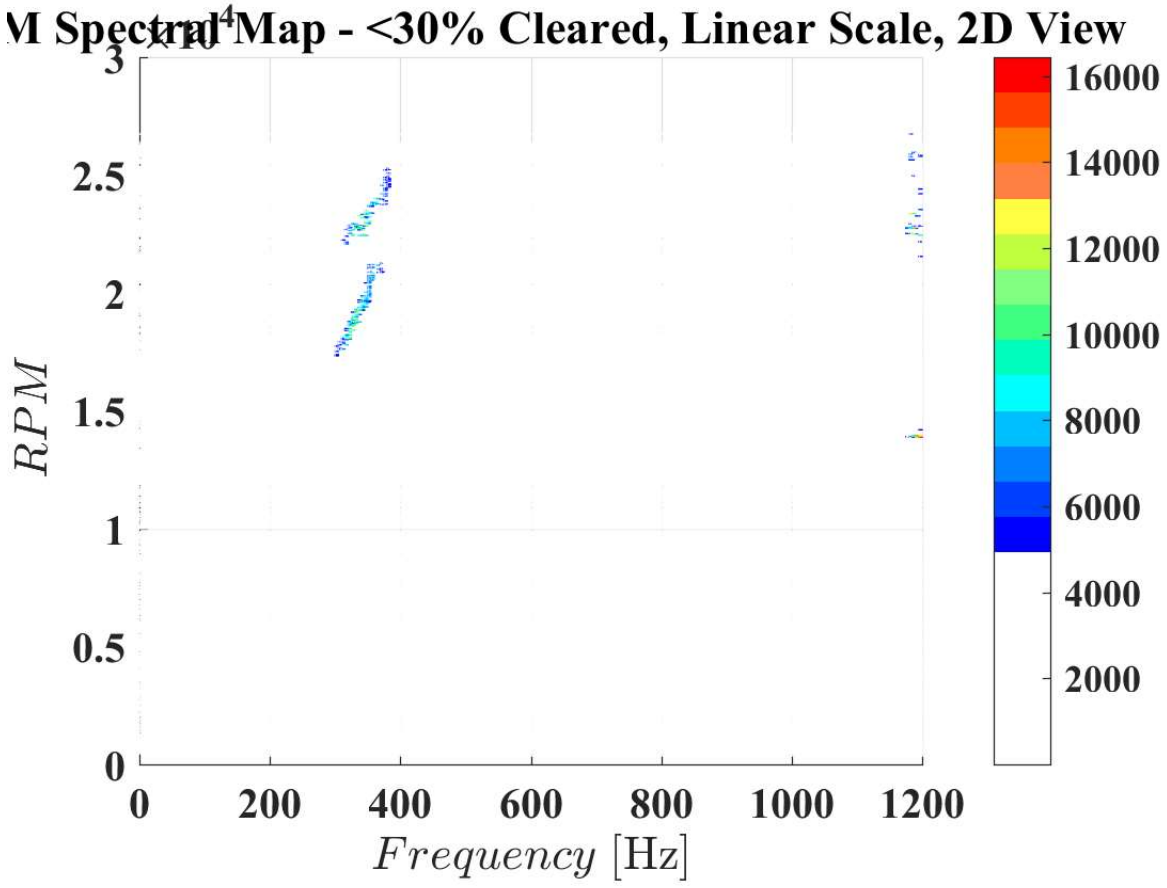
RPM Spectral Map - Default, Linear Scale



RPM Spectral Map - Default, Linear Scale, 2D View



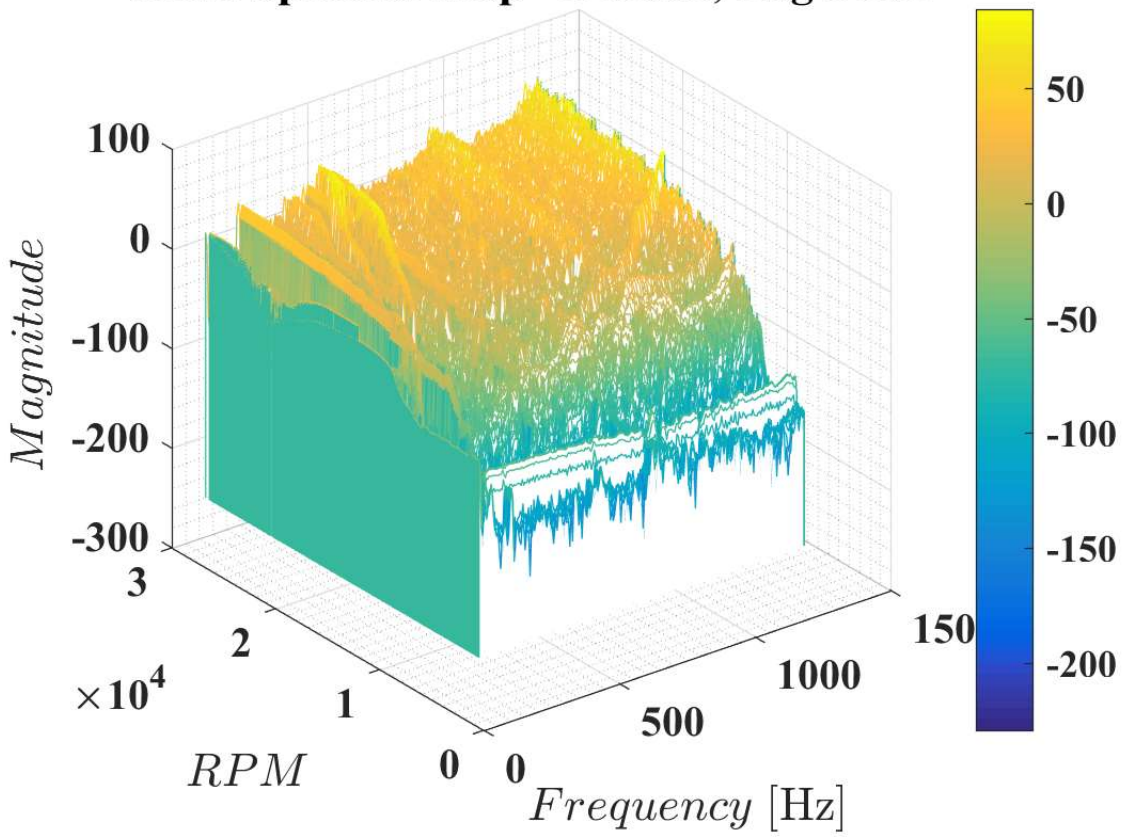




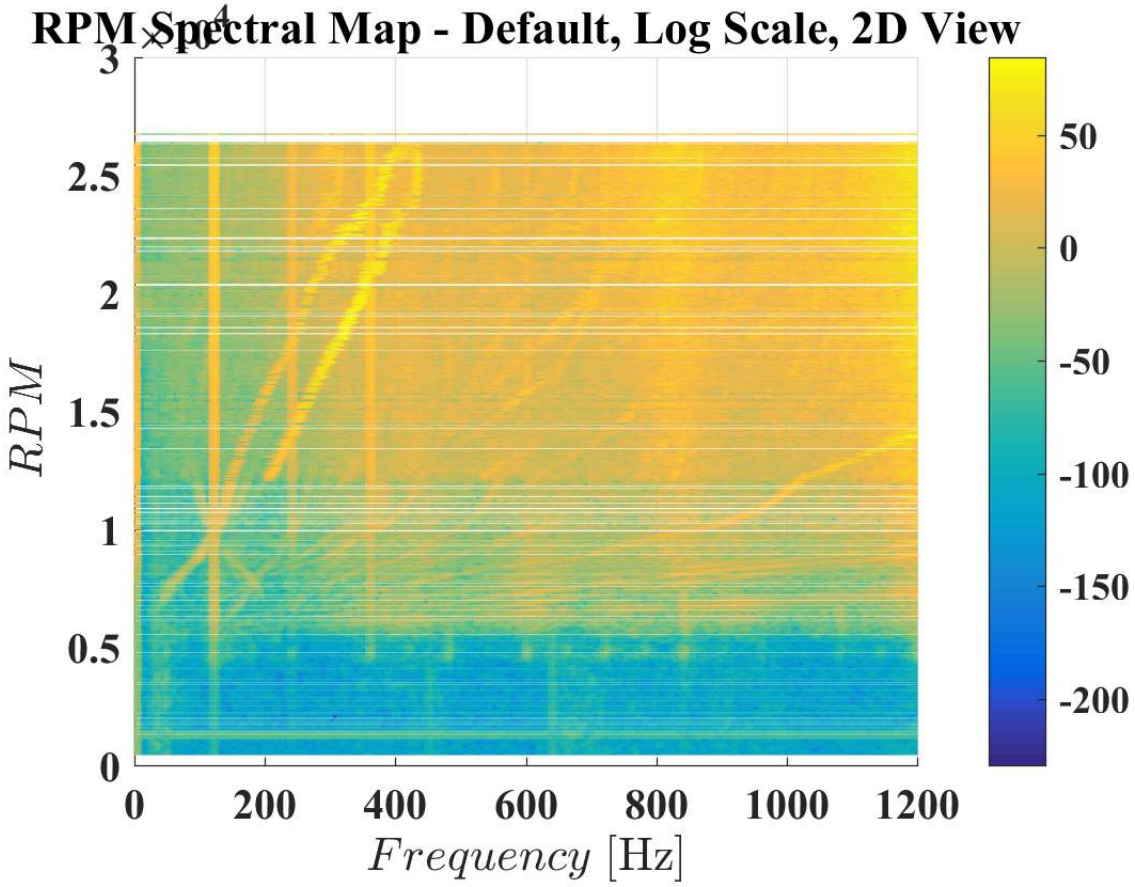
Plots - Log Scale

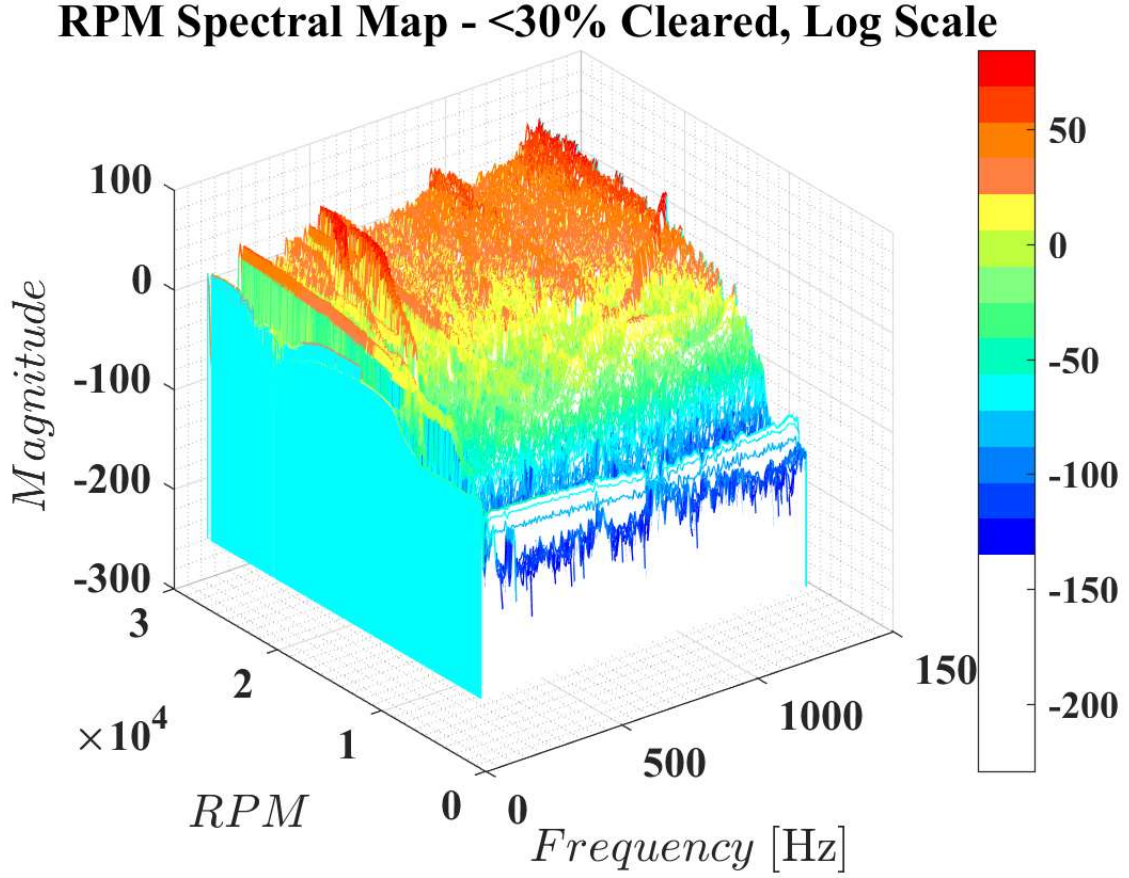
```
AP2_log = mag2db(AP2);
figure(5)
waterfall(f_xaxis,RPM_spline1,AP2_log)      % 1024 time block; Default map; Log scale
title('RPM Spectral Map - Default, Log Scale')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid minor
colorbar

figure(6)
waterfall(f_xaxis,RPM_spline1,AP2_log)      % 1024 time block; Default map; Log scale; 2D Vie
w
title('RPM Spectral Map - Default, Log Scale, 2D View')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)

figure(7)
waterfall(f_xaxis,RPM_spline1,AP2_log)      % 1024 time block; Log scale; Below 30% cleared
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('RPM Spectral Map - <30% Cleared, Log Scale ')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid minor
colorbar

figure(8)
waterfall(f_xaxis,RPM_spline1,AP2_log)      % 1024 time block; Default map; Log scale; 2D Vie
w;
map = [0 0.625 1;0 0.625 1;0 0.625 1;0 0.75 1;0 0.75 1;0 0.75 1;0 0.75 1;0 0.875 1;0 0.875 1;
0 0.875 1;0 1 1;0 1 1;0 1 0.875;0 1 0.75;0 1 0.5;0.25 1 0.5;0.5 1 0.5;0.75 1 0.5;0.75 1 0.25;
0.875 1 0.25;1 1 0.25;1 1 0.25;1 0.875 0.25;1 0.875 0.25;1 0.75 0.25;1 0.75 0.25;1 0.5 0.25;1
 0.5 0.25;1 0.5 0;1 0.5 0;1 0.5 0;1 0.25 0;1 0 0];
colormap(jet)
title('RPM Spectral Map - Log Scale, 2D View')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)
grid on
```

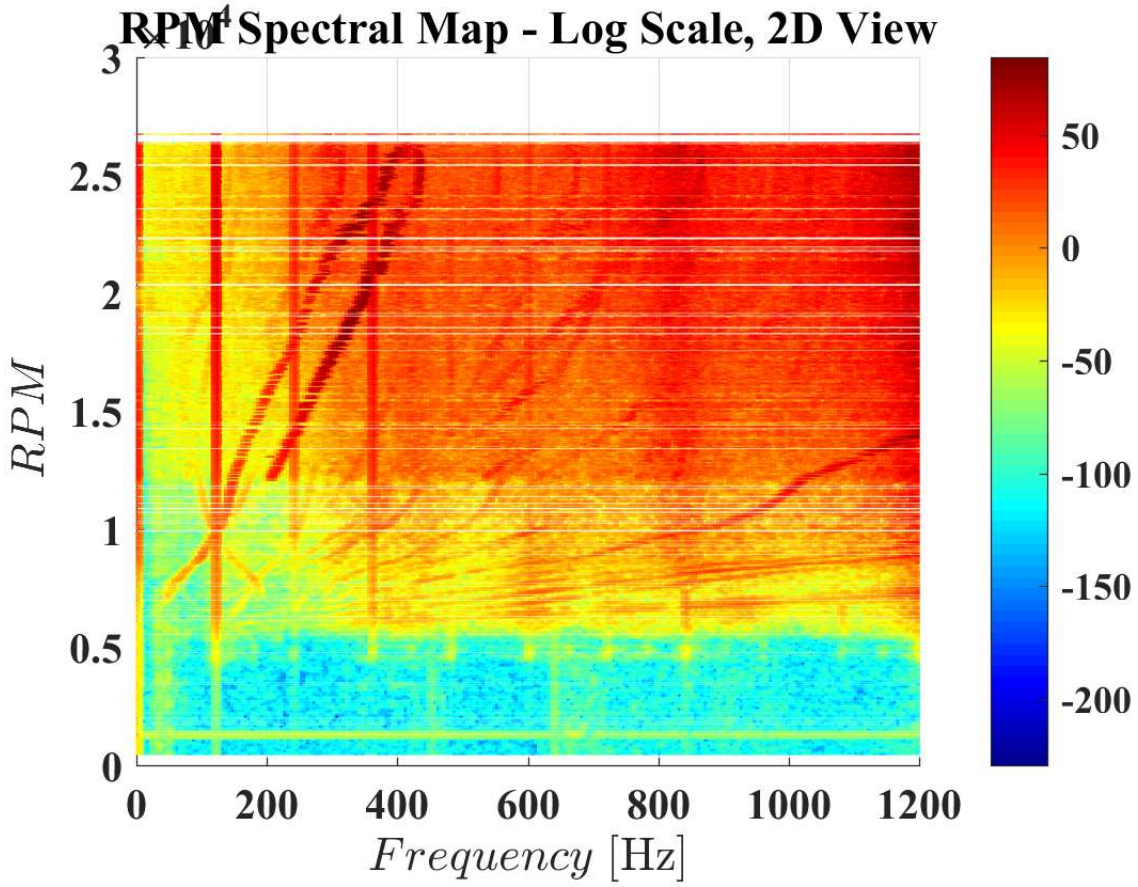
RPM Spectral Map - Default, Log Scale



RPM Spectral Map - Default, Log Scale, 2D View




RPM Spectral Map - <30% Cleared, Log Scale
Magnitude
100
0
-100
-200
-300
3
2
×10^4
1
0
RPM
0
500
1000
150
Frequency [Hz]
50
0
-50
-100
-150
-200


RPM Spectral Map - Log Scale, 2D View
×10^4
3
2.5
2
1.5
1
0.5
0
RPM
0
200
400
600
800
1000
1200
Frequency [Hz]
50
0
-50
-100
-150
-200

Published with MATLAB® R2015b

Contents

- Experimental Vibrations of Rotating Systems - Homework 4a - Shashank Iyengar (M12934513)
- Time Block Size = 512
- Time Block Size = 2048
- Time Block Size = 1024
- Plots - Linear Scale
- Plots - Log Scale

Experimental Vibrations of Rotating Systems - Homework 4a - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',0.1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('Hwk4_data.mat')       % Loaded variables: Tach1, Fsamp, data
```

Time Block Size = 512

```
Ntime = 512;
Fs = Fsamp;
Navg = length(data(1,:))/Ntime;

dt = 1/Fs;                          % Time Step
N = length(data(1,:));              % Total number of sample points
Ttime = Ntime/Fs;                   % Time for one block
Ttotal = N/Fs;                      % Time for complete signal
df = 1/Ttime;                       % Frequency Resolution

f_xaxis = 0:df:1200;                % Frequency axis
tt = 0:dt:Ttotal-dt; % Time axis

% No Overlap
n=1;
for i=1:Navg
    if i==1
        accel_nolap(i,1:Ntime) = data(1,1:Ntime);
    else
        accel_nolap(i,1:Ntime) = data(1,(n*Ntime)+1:(n+1)*Ntime);
        n=n+1;
    end
end

% FFT and Power Spectrum
for i=1:Navg
    AP_nolap_Total(i,1:Ntime) = conj(fft(accel_nolap(i,:))).*(fft(accel_nolap(i,:)));       %
 Auto Power Spectrum - No overlap
```

```
end

AP_nolap = AP_nolap_Total(:,1:121);         % Limiting to first 1200Hz magnitudes - No overla
p

% Centering of tach signal
Tach1_mean = mean(Tach1);
Tach1_centered = Tach1 - Tach1_mean;

% Zero-level crossing
n=1;
for i=1:length(Tach1_centered)-1
    if sign(Tach1_centered(1,i+1)) > sign(Tach1_centered(1,i))
        z_cross1(n) = ((0 - Tach1_centered(1,i))*((tt(1,i+1)-tt(1,i))/(Tach1_centered(1,i+1)-
Tach1_centered(1,i)))) + tt(1,i);
        n=n+1;
    end
end

% RPM Estimate
for i=1:length(z_cross1)-1
    RPM1(i) = 60/(z_cross1(1,i+1)-z_cross1(1,i));
    t_RPM1(i) = (z_cross1(1,i)+z_cross1(1,i+1))/2;
end
t_spline1 = linspace(0,30,300);
RPM_spline1 = spline(t_RPM1,RPM1,t_spline1);
```

Time Block Size = 2048

```
Ntime1 = 2048;
Fs = Fsamp;
Navg1 = length(data(1,:))/Ntime1;

dt = 1/Fs;                      % Time Step
N = length(data(1,:));          % Total number of sample points
Ttime1 = Ntime/Fs;               % Time for one block
Ttotal1 = N/Fs;                  % Time for complete signal
df = 1/Ttime1;                   % Frequency Resolution

f_xaxis = 0:df:1200;            % Frequency axis
tt = 0:dt:Ttotal-dt; % Time axis

% No Overlap
n=1;
for i=1:Navg1
    if i==1
        accel_nolap1(i,1:Ntime1) = data(1,1:Ntime1);
    else
        accel_nolap1(i,1:Ntime1) = data(1,(n*Ntime1)+1:(n+1)*Ntime1);
        n=n+1;
    end
end

% FFT and Power Spectrum
for i=1:Navg1
    AP_nolap_Total1(i,1:Ntime1) = conj(fft(accel_nolap1(i,:))).*(fft(accel_nolap1(i,:)));
```

```
    % Auto Power Spectrum - No overlap
end

AP_nolap1 = AP_nolap_Total1(:,1:481);          % Limiting to first 1200Hz magnitudes - No over
lap

% Centering of tach signal
Tach1_mean = mean(Tach1);
Tach1_centered = Tach1 - Tach1_mean;

% Zero-level crossing
n=1;
for i=1:length(Tach1_centered)-1
    if sign(Tach1_centered(1,i+1)) > sign(Tach1_centered(1,i))
        z_cross1(n) = ((0 - Tach1_centered(1,i))*((tt(1,i+1)-tt(1,i))/(Tach1_centered(1,i+1)-
Tach1_centered(1,i)))) + tt(1,i);
        n=n+1;
    end
end

% RPM Estimate
for i=1:length(z_cross1)-1
    RPM1(i) = 60/(z_cross1(1,i+1)-z_cross1(1,i));
    t_RPM1(i) = (z_cross1(1,i)+z_cross1(1,i+1))/2;
end
t_spline1 = linspace(t_RPM1(1),30,300);
RPM_spline1 = spline(t_RPM1,RPM1,t_spline1);
```

Time Block Size = 1024

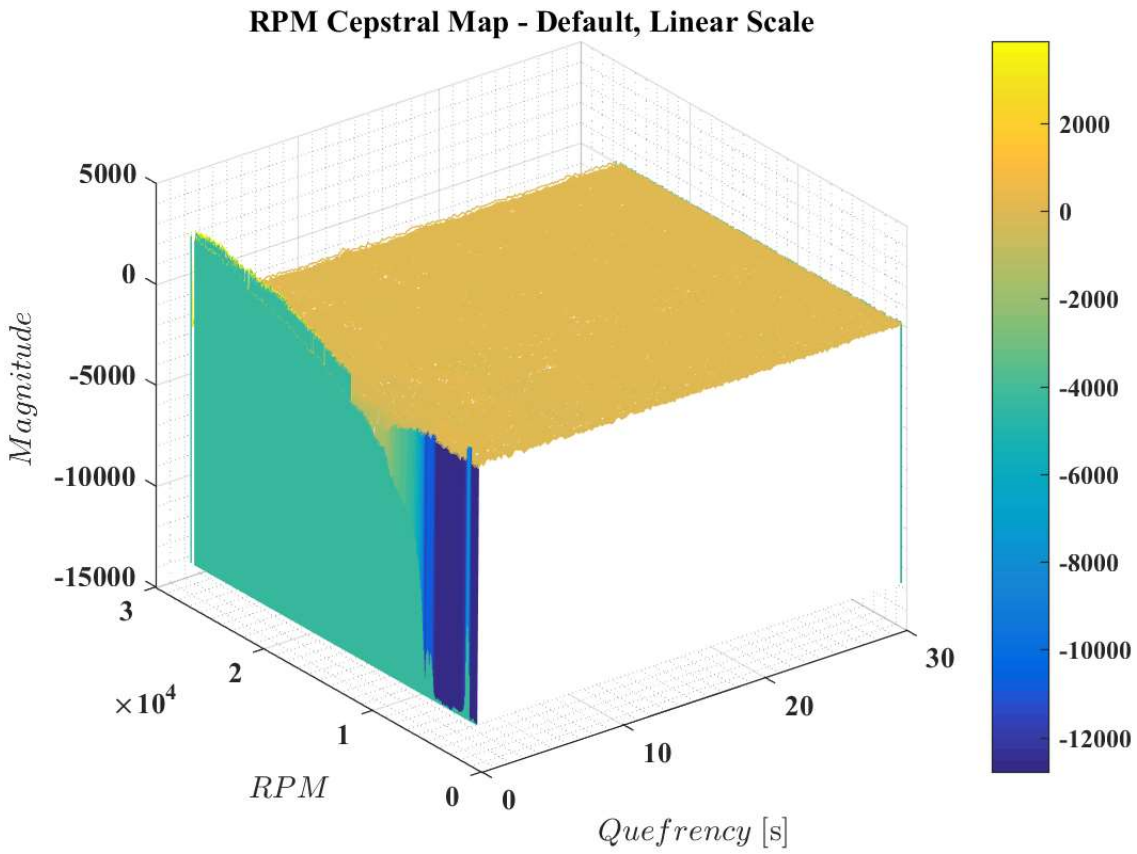
```
Ntime2 = 1024;
Fs = Fsamp;
Navg2 = length(data(1,:))/Ntime2;

dt = 1/Fs;                          % Time Step
N = length(data(1,:));              % Total number of sample points
Ttime2 = Ntime2/Fs;                   % Time for one block
Ttotal2 = N/Fs;                     % Time for complete signal
df = 1/Ttime2;                      % Frequency Resolution

f_xaxis = 0:df:1200;                % Frequency axis
tt = 0:dt:Ttotal-dt;                % Time axis

% No Overlap
n=1;
for i=1:Navg2
    if i==1
        accel_nolap2(i,1:Ntime2) = data(1,1:Ntime2);
    else
        accel_nolap2(i,1:Ntime2) = data(1,(n*Ntime2)+1:(n+1)*Ntime2);
        n=n+1;
    end
end
% 87.5% overlap
n=1;
for i=1:Navg2*8-7
```

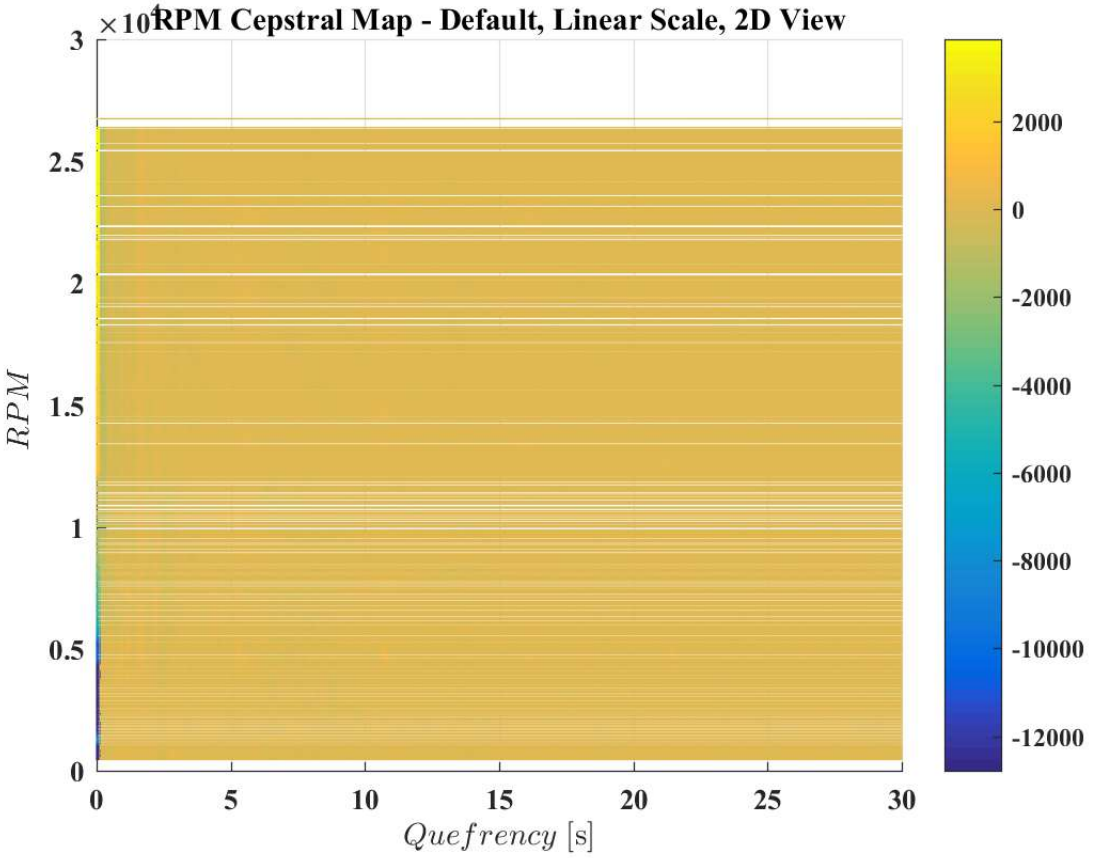
```
    if i==1
        accel2(i,1:Ntime2) = data(1,1:Ntime2);
        accel2_tr = accel2(i,1:Ntime2)';
        accel2_hann(i,1:Ntime2) = hann(Ntime2).*accel2_tr;
    else
        accel2(i,1:Ntime2) = data(1,n*(Ntime2/8)+1:1023+(n*(Ntime2/8)+1));
        accel2_tr = accel2(i,1:Ntime2)';
        accel2_hann(i,1:Ntime2) = hann(Ntime2).*accel2_tr;
        n=n+1;
    end
end

% FFT and Power Cepstrum
for i=1:Navg2
    AP_nolap_Total2(i,1:Ntime2) = fft(log(conj(fft(accel_nolap2(i,:))).*(fft(accel_nolap2(i,:
))));       % Auto Power Spectrum - No overlap
end
for i=1:Navg2*8-7
    AP_Total2(i,1:Ntime2) = fft(log(conj(fft(accel2_hann(i,:))).*(fft(accel2_hann(i,:)))));
                        % Auto Power Spectrum - 50% overlap
end

AP_nolap2 = AP_nolap_Total2(:,1:241);         % Limiting to first 1200Hz magnitudes - No over
lap
AP2 = AP_Total2(:,1:241);                 % Limiting to first 1200Hz magnitudes - 50% overlap

% Centering of tach signal
Tach1_mean = mean(Tach1);
Tach1_centered = Tach1 - Tach1_mean;

% Zero-level crossing
n=1;
for i=1:length(Tach1_centered)-1
    if sign(Tach1_centered(1,i+1)) > sign(Tach1_centered(1,i))
        z_cross1(n) = ((0 - Tach1_centered(1,i))*((tt(1,i+1)-tt(1,i))/(Tach1_centered(1,i+1)-
Tach1_centered(1,i)))) + tt(1,i);
        n=n+1;
    end
end

% RPM Estimate
for i=1:length(z_cross1)-1
    RPM1(i) = 60/(z_cross1(1,i+1)-z_cross1(1,i));
    t_RPM1(i) = (z_cross1(1,i)+z_cross1(1,i+1))/2;
end
t_spline1 = linspace(t_RPM1(1),30,1193);
RPM_spline1 = spline(t_RPM1,RPM1,t_spline1);
t = linspace(0,30,241);
```

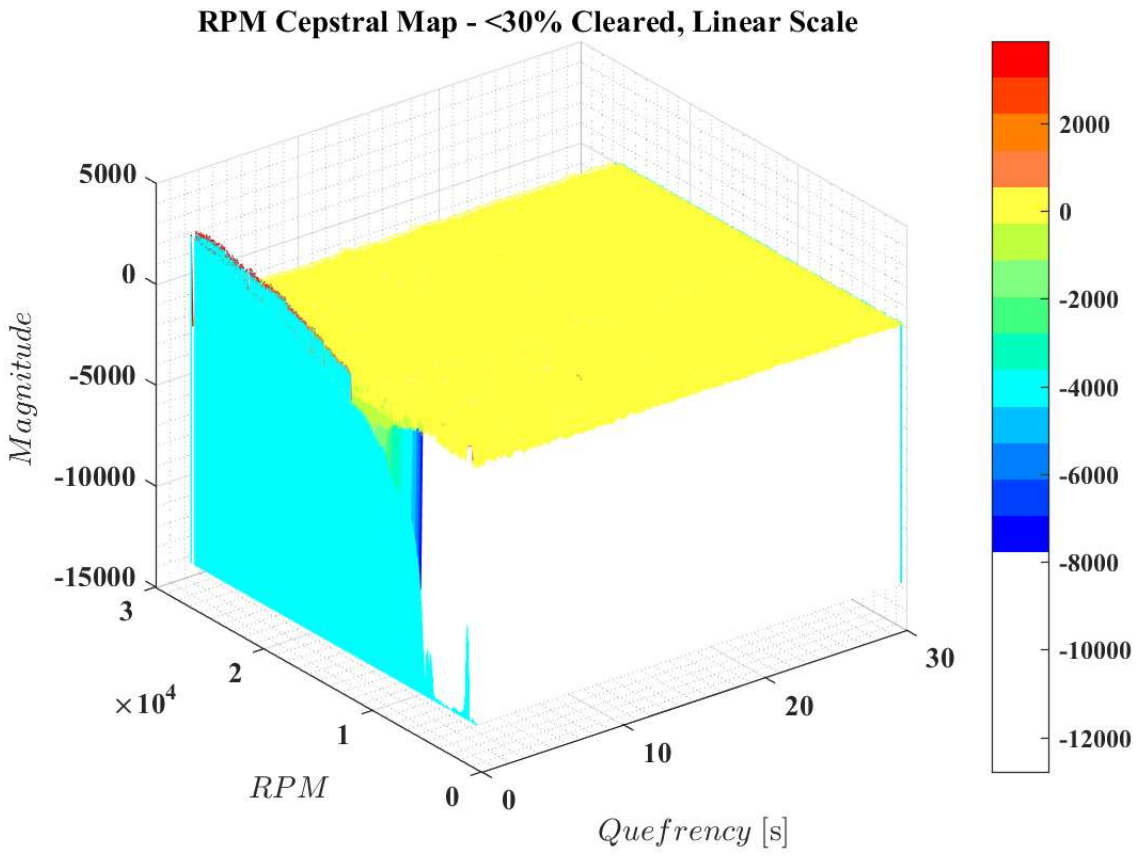
Plots - Linear Scale

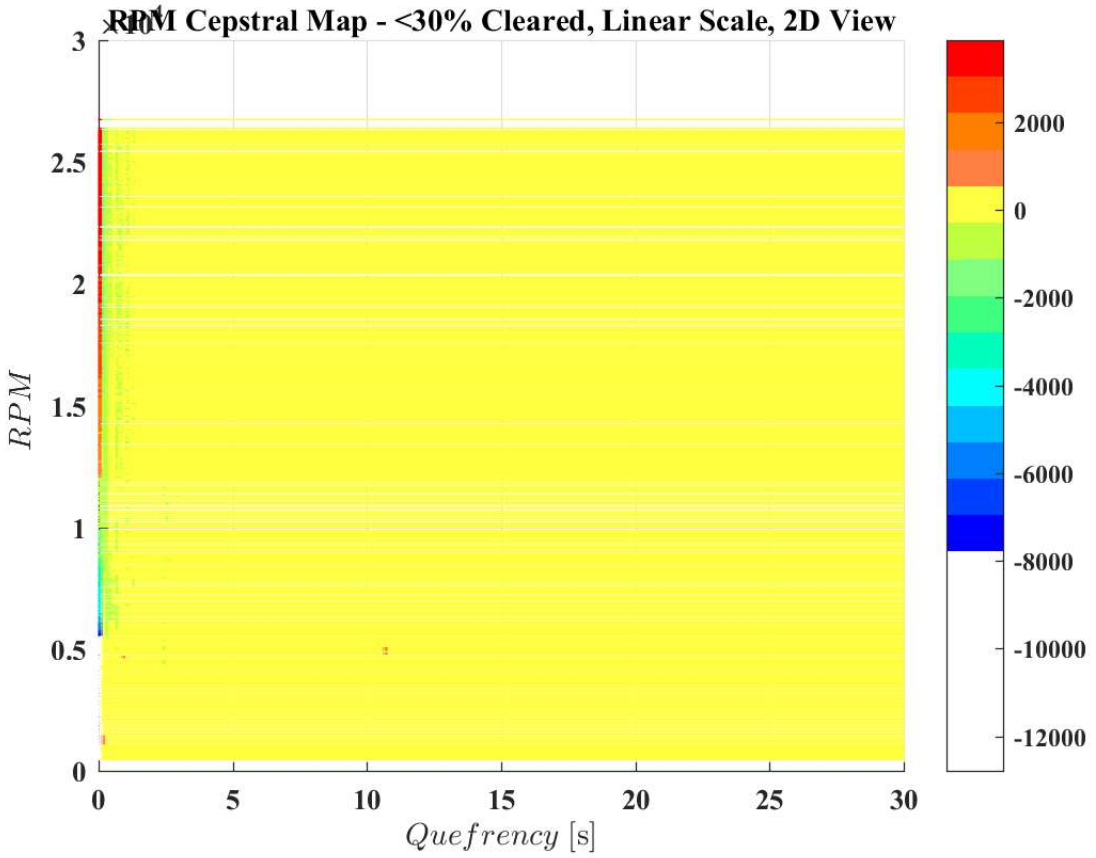
```
figure(1)
waterfall(t,RPM_spline1,real(AP2))      % 1024 time block; Default map; Linear scale
title('RPM Cepstral Map - Default, Linear Scale')
xlabel(['$ Quefrency\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
```

```
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid minor
colorbar

figure(2)
waterfall(t,RPM_spline1,real(AP2))      % 1024 time block; Default map; Linear scale; 2D View
title('RPM Cepstral Map - Default, Linear Scale, 2D View')
xlabel(['$ Quefrency\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)

figure(3)
waterfall(t,RPM_spline1,real(AP2))      % 1024 time block; Linear scale; Below 30% cleared
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('RPM Cepstral Map - <30% Cleared, Linear Scale ')
xlabel(['$ Quefrency\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid minor
colorbar

figure(4)
waterfall(t,RPM_spline1,real(AP2))      % 1024 time block; Default map; Linear scale; 2D View
; Below 30% cleared
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('RPM Cepstral Map - <30% Cleared, Linear Scale, 2D View')
xlabel(['$ Quefrency\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)
grid on
```


RPM Cepstral Map - Default, Linear Scale
Magnitude
5000
0
-5000
-10000
-15000
3
2
1
0
×10^4
RPM
0
10
20
30
Quefrency [s]
2000
0
-2000
-4000
-6000
-8000
-10000
-12000

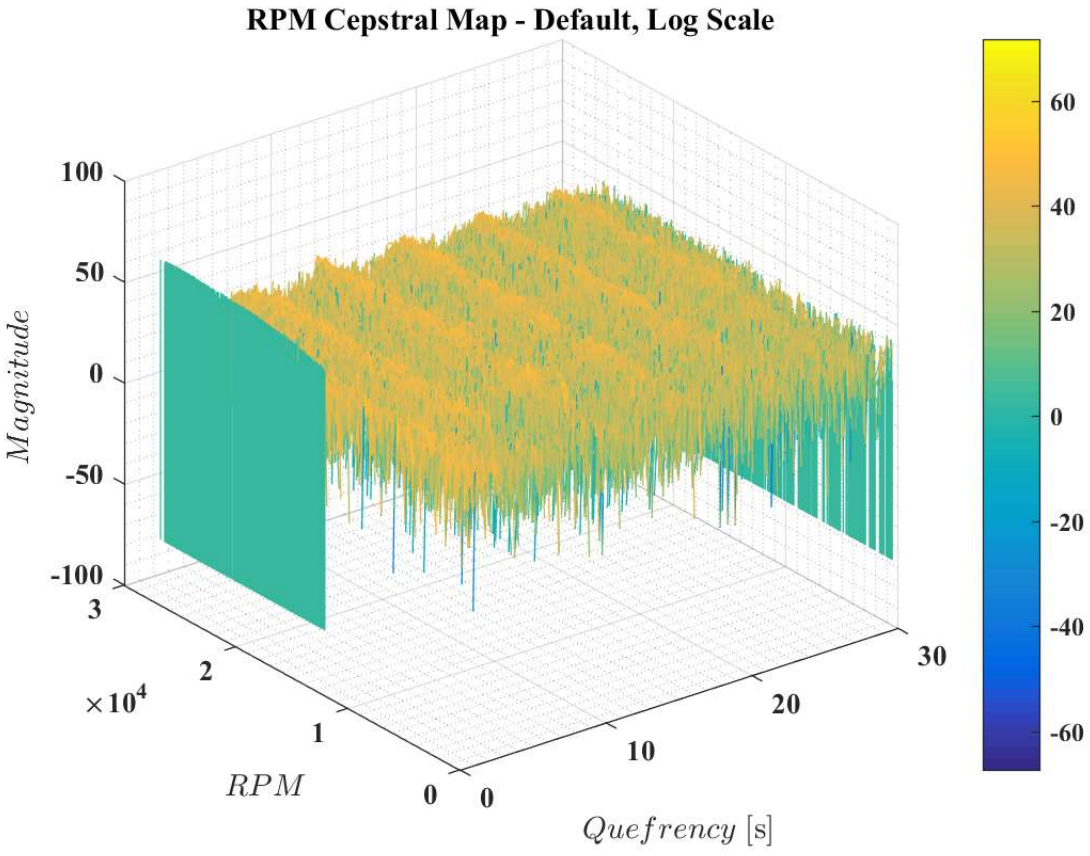

RPM Cepstral Map - Default, Linear Scale, 2D View
×10^4
3
2.5
2
1.5
1
0.5
0
RPM
0
5
10
15
20
25
30
Quefrency [s]
2000
0
-2000
-4000
-6000
-8000
-10000
-12000

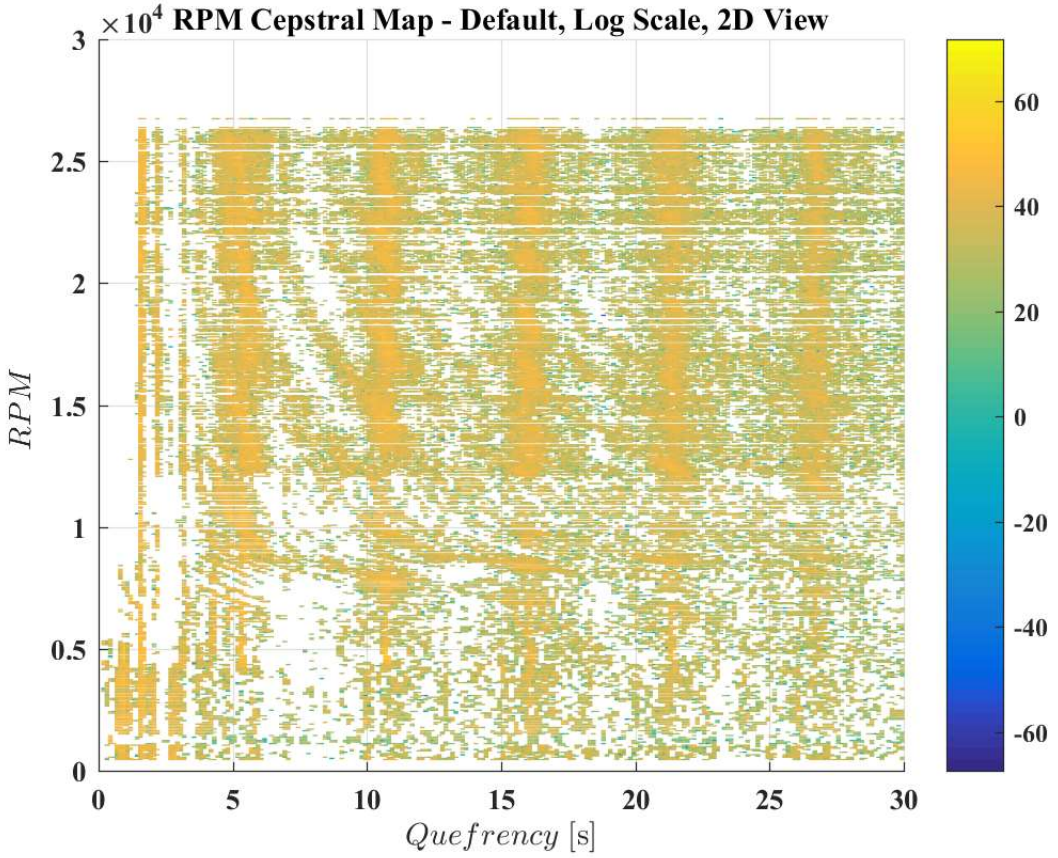


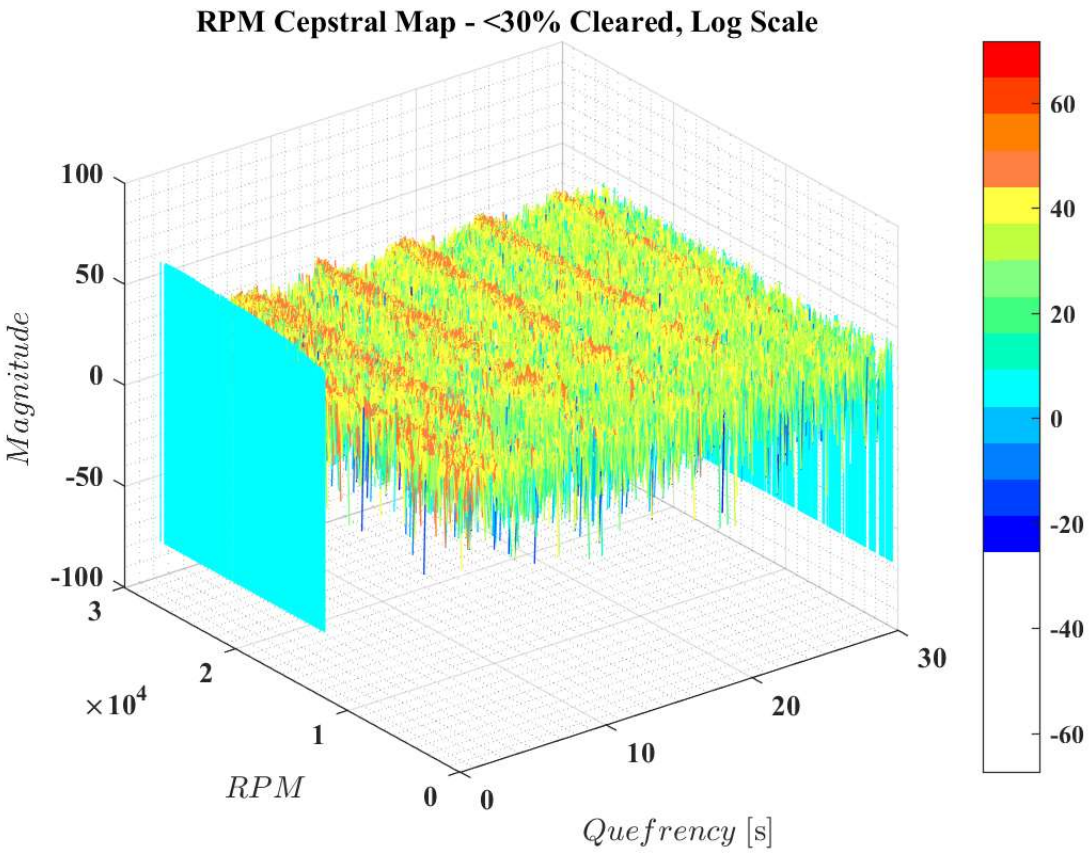


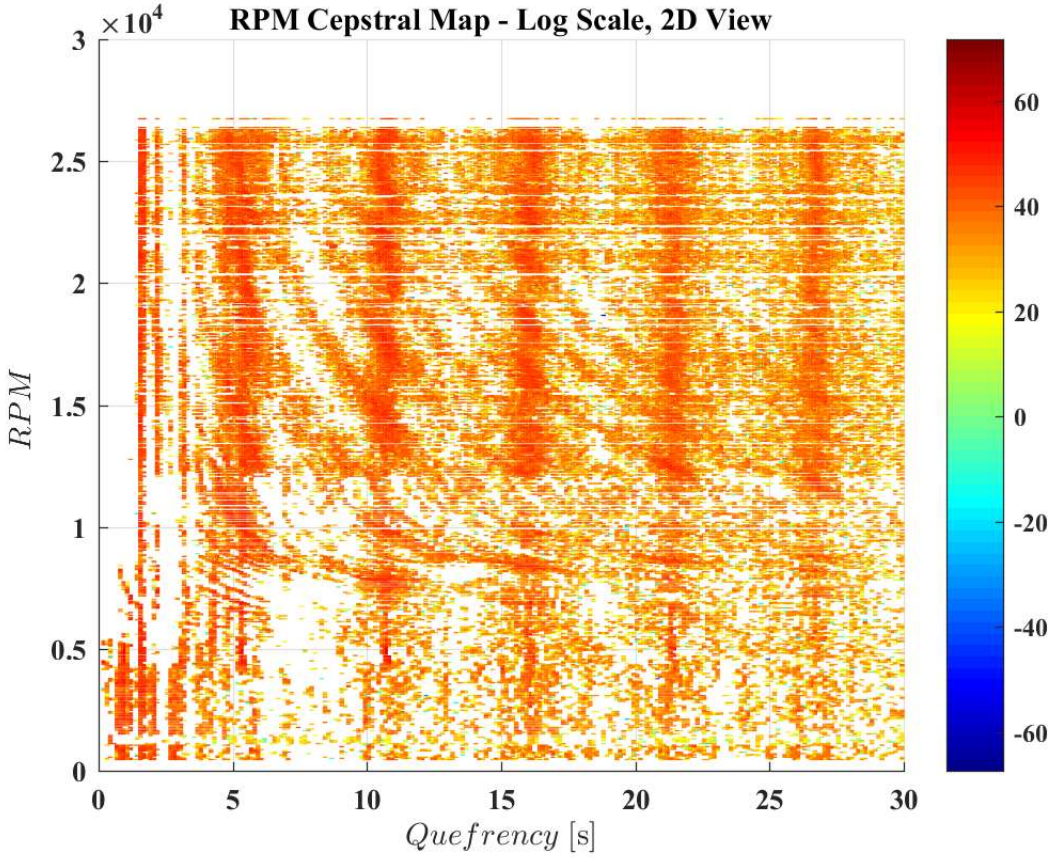
Plots - Log Scale

```
AP2_log = mag2db(AP2);
figure(5)
waterfall(t,RPM_spline1,real(AP2_log))      % 1024 time block; Default map; Log scale
title('RPM Cepstral Map - Default, Log Scale')
xlabel(['$ Quefrency\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid minor
colorbar

figure(6)
waterfall(t,RPM_spline1,real(AP2_log))      % 1024 time block; Default map; Log scale; 2D Vie
w
title('RPM Cepstral Map - Default, Log Scale, 2D View')
xlabel(['$ Quefrency\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)

figure(7)
waterfall(t,RPM_spline1,real(AP2_log))      % 1024 time block; Log scale; Below 30% cleared
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('RPM Cepstral Map - <30% Cleared, Log Scale ')
xlabel(['$ Quefrency\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
grid minor
colorbar


figure(8)
waterfall(t,RPM_spline1,real(AP2_log))      % 1024 time block; Default map; Log scale; 2D Vie
w;
map = [0 0.625 1;0 0.625 1;0 0.625 1;0 0.75 1;0 0.75 1;0 0.75 1;0 0.75 1;0 0.875 1;0 0.875 1;
0 0.875 1;0 1 1;0 1 1;0 1 0.875;0 1 0.75;0 1 0.5;0.25 1 0.5;0.5 1 0.5;0.75 1 0.5;0.75 1 0.25;
0.875 1 0.25;1 1 0.25;1 1 0.25;1 0.875 0.25;1 0.875 0.25;1 0.75 0.25;1 0.75 0.25;1 0.5 0.25;1
 0.5 0.25;1 0.5 0;1 0.5 0;1 0.5 0;1 0.25 0;1 0 0];
colormap(jet)
title('RPM Cepstral Map - Log Scale, 2D View')
xlabel(['$ Quefrency\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)
grid on
```


RPM Cepstral Map - Default, Log Scale
Magnitude
100
50
0
-50
-100
3
2
×10^4
1
0
RPM
0
10
20
30
Quefrency [s]
60
40
20
0
-20
-40
-60


×10^4 RPM Cepstral Map - Default, Log Scale, 2D View
RPM
3
2.5
2
1.5
1
0.5
0
0
5
10
15
20
25
30
Quefrency [s]
60
40
20
0
-20
-40
-60


RPM Cepstral Map - <30% Cleared, Log Scale
Magnitude
100
50
0
-50
-100
3
2
×10^4
1
0
RPM
0
10
20
30
Quefrency [s]
60
40
20
0
-20
-40
-60


RPM Cepstral Map - Log Scale, 2D View
×10^4
3
2.5
2
1.5
1
0.5
0
RPM
0
5
10
15
20
25
30
Quefrency [s]
60
40
20
0
-20
-40
-60

Published with MATLAB® R2015b

Contents

- Experimental Vibrations of Rotating Systems - Homework 5 - Shashank Iyengar (M12934513)
- Signal Synthesis
- FFT and Power Spectrum
- Plots

Experimental Vibrations of Rotating Systems - Homework 5 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',0.1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')
```

Signal Synthesis

```
Fs = 5120;                                                                  %Sampling Fre
quency
RPMs = linspace(1000,9000,40);                                              %40 equally s
paced RPMs
dt = 1/Fs;                                                                  %Time resolut
ion
% Ntime = input('Enter the blocksize(512, 1024, 2048, 4096 or 5120): ');          %Block Size
Ntime = 5120;
Ttime = dt*Ntime;                                                           %Time for one
 block
Ttotal = Ttime*length(RPMs);                                                %Time for tot
al sample space
tt = 0:1/Fs:Ttime-1/Fs;                                                     %Time axis
df = 1/Ttime;                                                               %Frequency Re
solution
Fnyq = Fs/2;                                                                %Nyquist Freq
uency
f_xaxis = 0:df:1500;                                                        %Frequency ax
is

for i=1:length(RPMs)
    S1(i,:) = 10*sin(2*pi*440*tt);                                          %Natural Freq
uency = 440Hz
    S2(i,:) = 14*sin(2*pi*765*tt);                                          %Natural Freq
uency = 765Hz
    S2(i,:) = 15*sin(2*pi*(RPMs(i)/60)*tt);                                 %First Order
    S3(i,:) = 15*sin(2*pi*(RPMs(i)/60)*2*tt);                               %Second Order
    S4(i,:) = 15*sin(2*pi*(RPMs(i)/60)*3*tt);                               %Third Order
    S5(i,:) = 15*sin(2*pi*(RPMs(i)/60)*7*tt);                               %Seventh Orde
r
    S6(i,:) = 15*sin(2*pi*(RPMs(i)/60)*12*tt);                              %Twelfth Orde
r
```

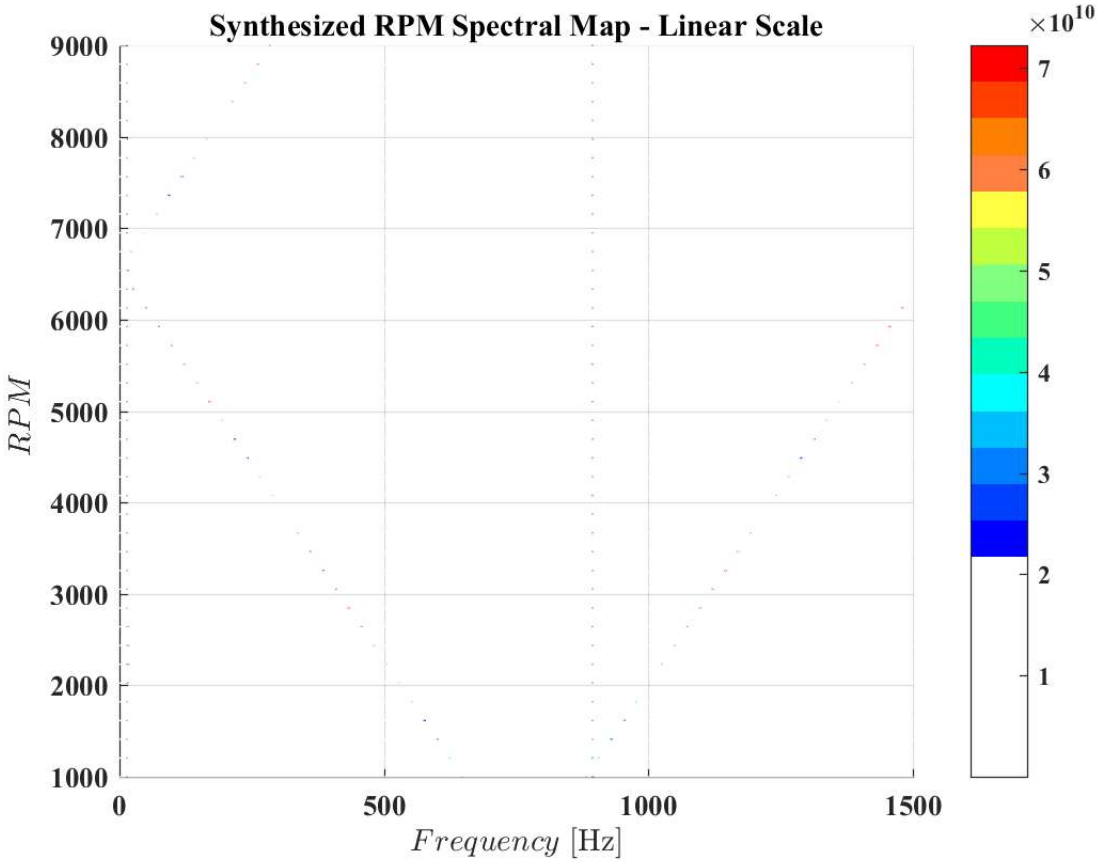
```
    S7(i,:) = 15*sin(2*pi*(RPMs(i)/60)*11.3*tt);                                %Fractional O
rder
    S8(i,:) = 15*sin(2*pi*(RPMs(i)/60)*0.37*tt);                                %Fractional O
rder
    S9(i,:) = (10*cos(2*pi*440*tt)).*(12*cos(2*pi*455*tt));                     %Amplitude Mo
dulation with Natural Frequency
    S10(i,:) = (14*cos(2*pi*765*tt)).*(15*cos(2*pi*(RPMs(i)/60)*7*tt));         %Amplitude Mo
dulation with Orders
    sum(i,:) = S1(i,:)+S2(i,:)+S3(i,:)+S4(i,:)+S5(i,:)+S6(i,:)+S7(i,:)+S8(i,:)+S9(i,:)+S10(i,
:);
end
```

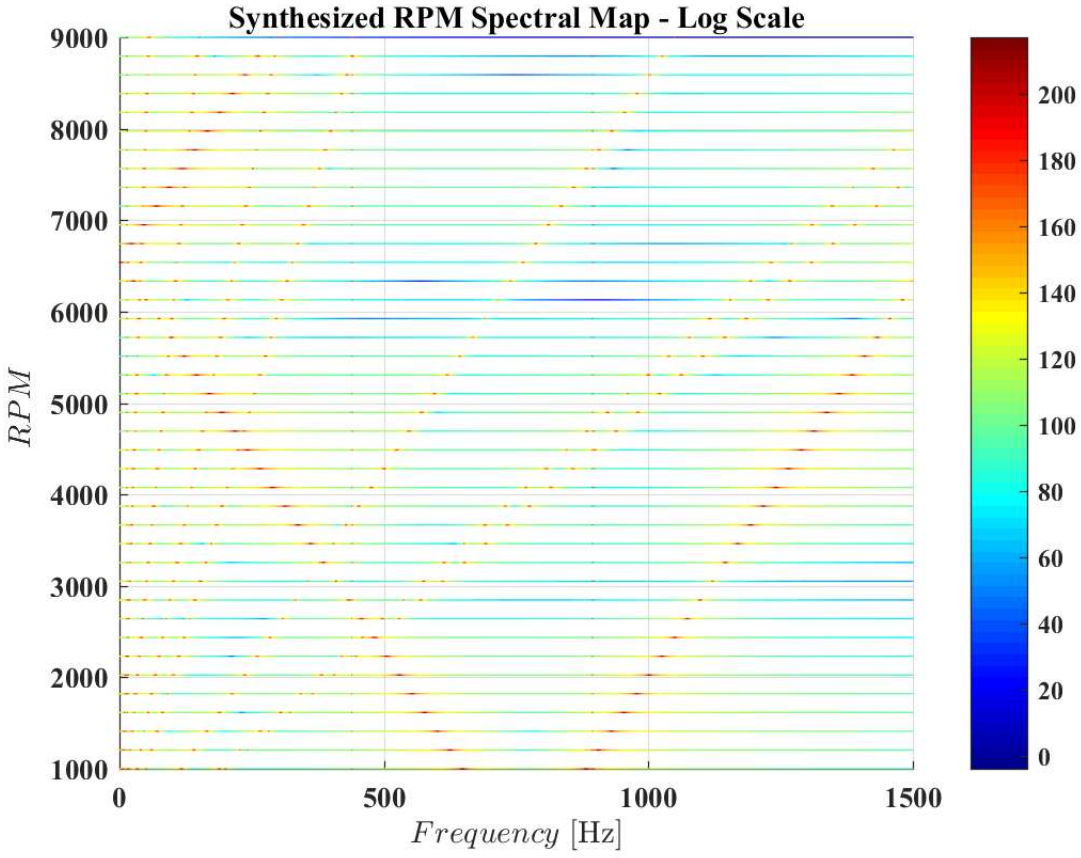
FFT and Power Spectrum

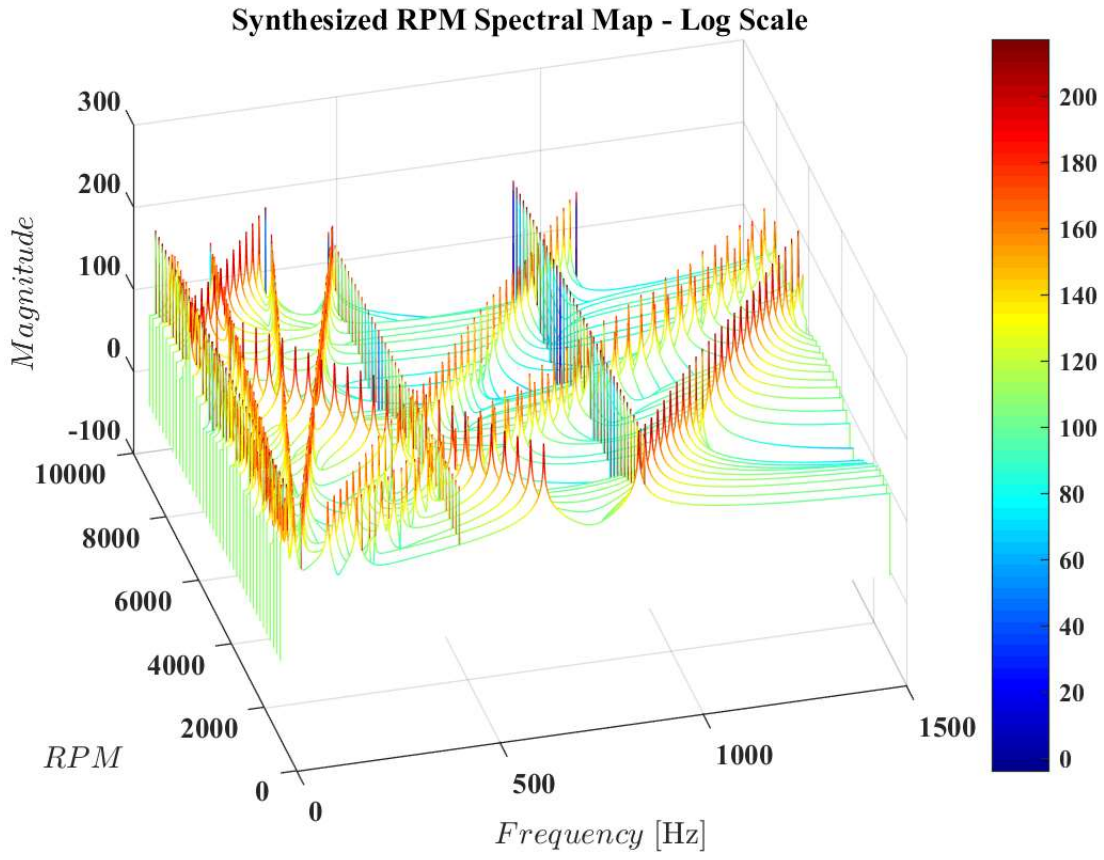
```
for i=1:length(RPMs)
    APS(i,:) = conj(fft(sum(i,:))).*(fft(sum(i,:)));
end
```

Plots

```
%Linear Scale
figure(1)
waterfall(f_xaxis,RPMs,APS(:,1:length(f_xaxis)))
map = [1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;1 1 1;0 0 1;0 0.25 1;0 0.5 1;0 0.75 1;0 1 1;0 1 0.75;0.2
5 1 0.5;0.5 1 0.5;0.75 1 0.25;1 1 0.25;1 0.5 0.25;1 0.5 0;1 0.25 0;1 0 0];
colormap(map)
title('Synthesized RPM Spectral Map - Linear Scale')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)

%Log Scale
APS_log = mag2db(APS);
figure(2)
waterfall(f_xaxis,RPMs,APS_log(:,1:length(f_xaxis)))
colormap(jet)
title('Synthesized RPM Spectral Map - Log Scale')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)

figure(3)
waterfall(f_xaxis,RPMs,APS_log(:,1:length(f_xaxis)))
colormap(jet)
title('Synthesized RPM Spectral Map - Log Scale')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(-15,45)
```


Synthesized RPM Spectral Map - Linear Scale
×10^10
7
6
5
4
3
2
1
9000
8000
7000
6000
5000
4000
3000
2000
1000
RPM
0
500
1000
1500
Frequency [Hz]


Synthesized RPM Spectral Map - Log Scale
200
180
160
140
120
100
80
60
40
20
0
9000
8000
7000
6000
5000
4000
3000
2000
1000
RPM
0
500
1000
1500
Frequency [Hz]



Published with MATLAB® R2015b

Contents

- Experimental Vibrations of Rotating Systems - Homework 6 - Shashank Iyengar (M12934513)
- Pre-processing
- Centering of tach signal
- Hilbert Transform
- Zero-level crossing
- RPM Estimate

Experimental Vibrations of Rotating Systems - Homework 6 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('hwk3_data.mat')       % Loaded variables: Tach1 and Tach2
```

Pre-processing

```
Fs = 5120;                  % Sampling Frequency
N = length(Tach1);          % Total Sample Points
Ttotal = N/Fs;              % Total signal time
dt = 1/Fs;                  % Time instant
tt = 0:dt:Ttotal-dt;        % Time axis
ff = 0:1/Ttotal:Fs-1/Ttotal;
```

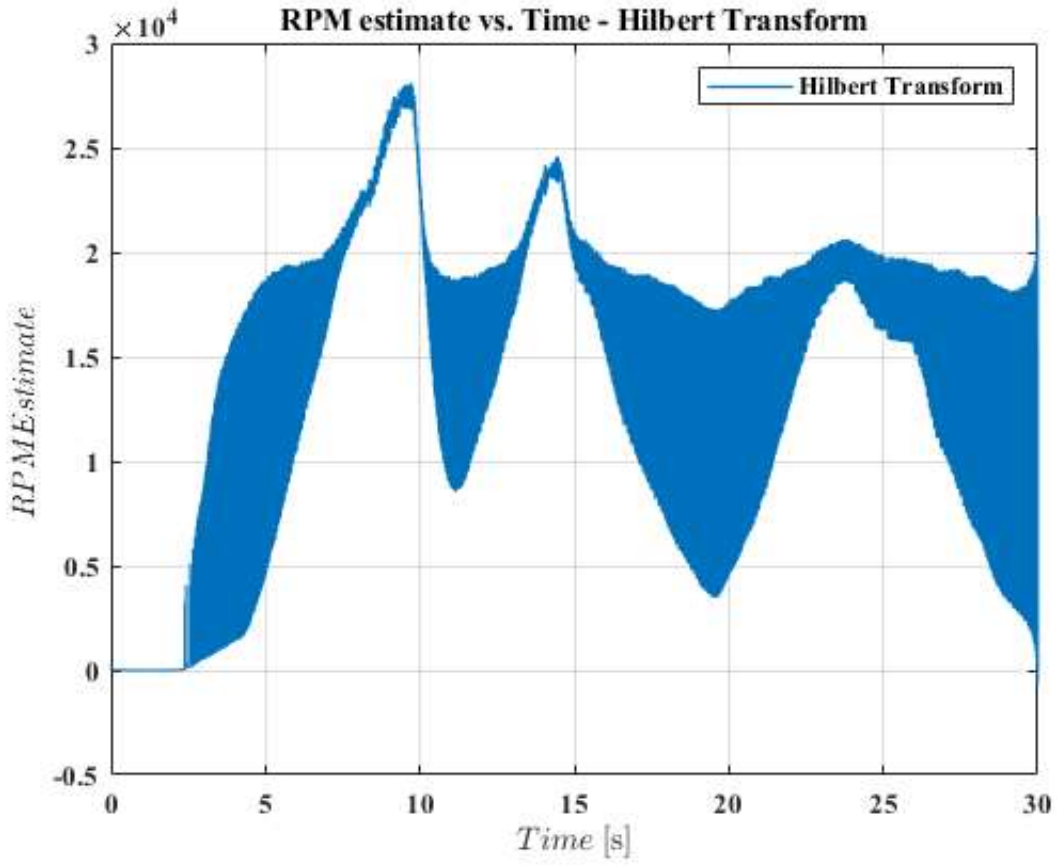
Centering of tach signal

```
Tach1_mean = mean(Tach1);   % Mean of Tach1 data
Tach1_centered = Tach1 - Tach1_mean;    % Centered tach signal
```

Hilbert Transform

```
h_Tach = hilbert(Tach1_centered);
psi = unwrap(angle(h_Tach));
inst_freq = diff(psi)*Fs;
inst_freq_filtered = lowpass(inst_freq(1,:),10,Fs);
inst_RPM_Hilbert = inst_freq_filtered*10;

figure(1)
plot(tt(1:length(inst_freq_filtered)),inst_RPM_Hilbert)
title('RPM estimate vs. Time - Hilbert Transform')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
```

```
legend('Hilbert Transform')
grid on
```



Zero-level crossing

Tach Signal 1

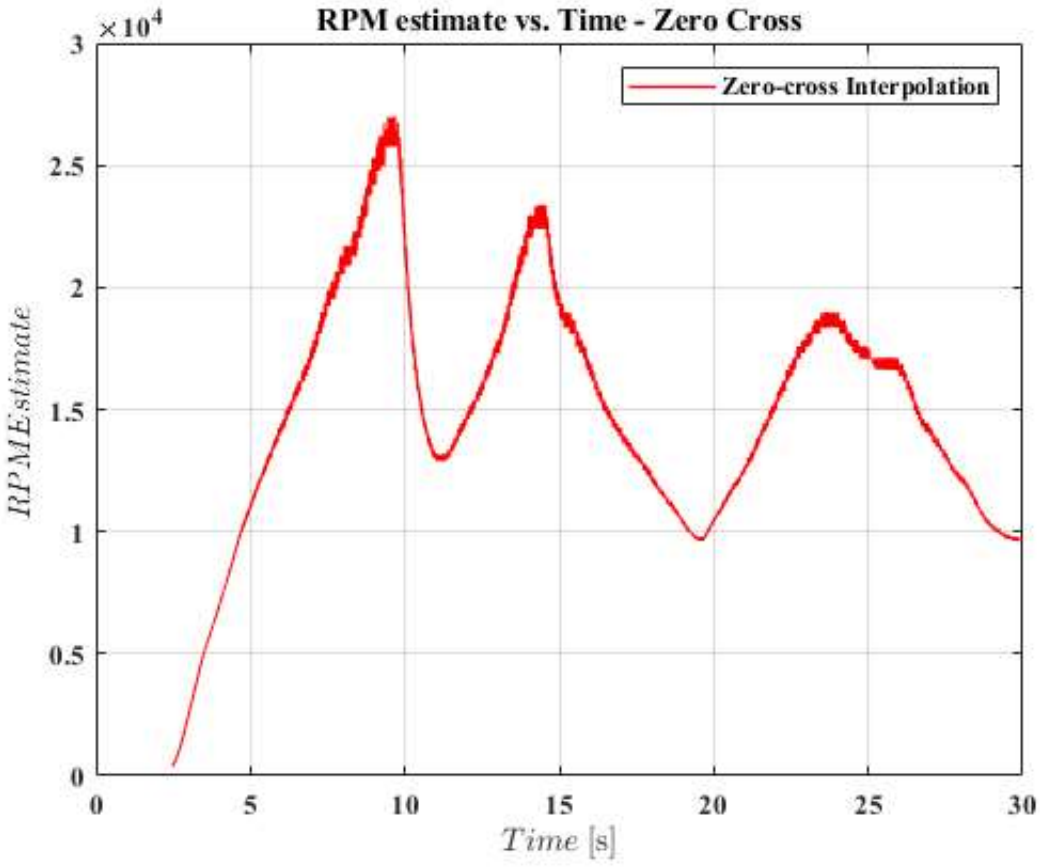
```
n=1;
for i=1:length(Tach1_centered)-1
    if sign(Tach1_centered(1,i+1)) > sign(Tach1_centered(1,i))
        z_cross1(n) = ((0 - Tach1_centered(1,i))*((tt(1,i+1)-tt(1,i))/(Tach1_centered(1,i+1)-
Tach1_centered(1,i)))) + tt(1,i);
        n=n+1;
    end
end
```

RPM Estimate

Tach Signal 1

```
for i=1:length(z_cross1)-1
    RPM1(i) = 60/(z_cross1(1,i+1)-z_cross1(1,i));        % RPM estimate
    t_RPM1(i) = (z_cross1(1,i)+z_cross1(1,i+1))/2;       % Time
end

figure(2)
```

```
plot(t_RPM1,RPM1,'-r')
title('RPM estimate vs. Time - Zero Cross')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Zero-cross Interpolation')
grid on

% figure(3)
% plot(t_RPM1,RPM1,'-r')
% hold on
% plot(tt(1:length(inst_freq_filtered)),inst_RPM_Hilbert,'-y')
% title('RPM estimate vs. Time')
% xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
% ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
% legend('Zero-cross Interpolation','Hilbert Transform')
% grid on
```



Published with MATLAB® R2018a

Contents

- Experimental Vibrations of Rotating Systems - Homework 7 - Shashank Iyengar (M12934513)
- Pre-processing
- Zero-level crossing
- FFT Method

Experimental Vibrations of Rotating Systems - Homework 7 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('HW7.mat')       % Loaded variables: Tach1 and Tach2
```

Pre-processing

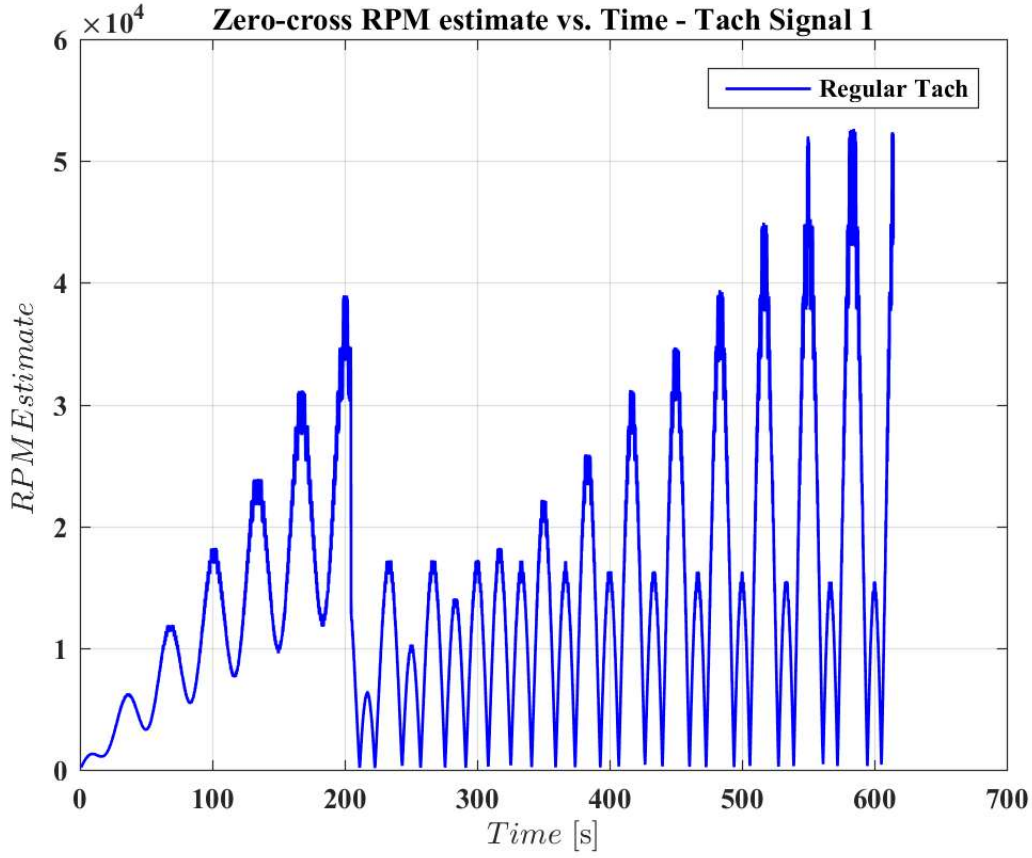
```
Fs = 5120;                  % Sampling Frequency
N = length(RegularTach);    % Total Sample Points
Ttotal = N/Fs;              % Total signal time
dt = 1/Fs;                  % Time instant
tt = 0:dt:Ttotal-dt;        % Time axis
```

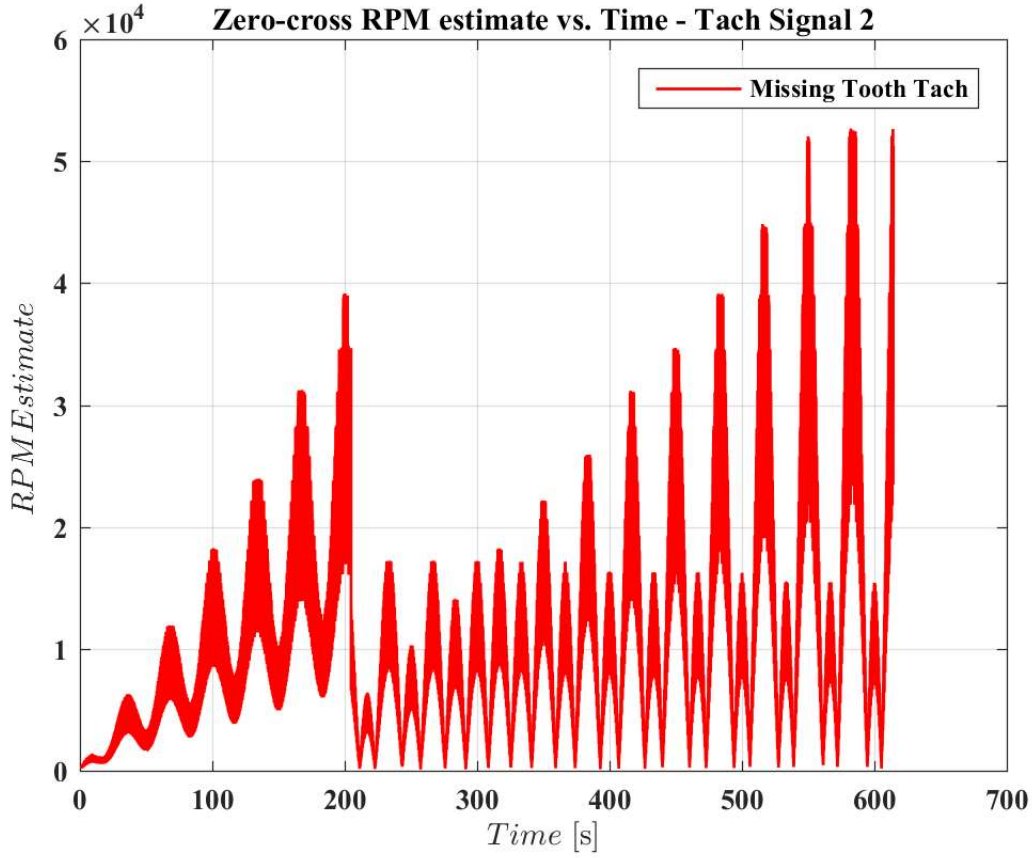
Zero-level crossing

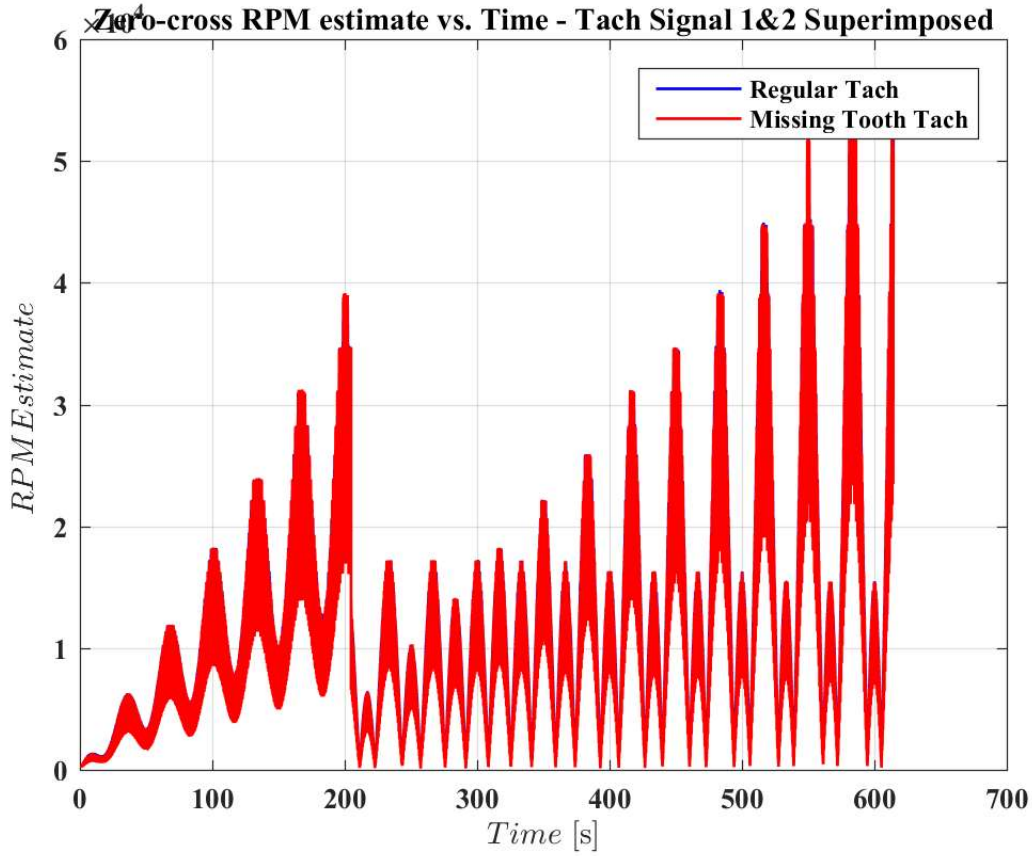
Tach Signal 1

```
n=1;
for i=1:length(RegularTach)-1
    if sign(RegularTach(1,i+1)) > sign(RegularTach(1,i))
        z_cross1(n) = ((0 - RegularTach(1,i))*((tt(1,i+1)-tt(1,i))/(RegularTach(1,i+1)-Regula
rTach(1,i)))) + tt(1,i);
        n=n+1;
    end
end
% Tach Signal 2
n=1;
for i=1:length(MissToothTach)-1
    if sign(MissToothTach(1,i+1)) > sign(MissToothTach(1,i))
        z_cross2(n) = ((0 - MissToothTach(1,i))*((tt(1,i+1)-tt(1,i))/(MissToothTach(1,i+1)-Mi
ssToothTach(1,i)))) + tt(1,i);
        n=n+1;
    end
end

%%RPM Estimate
```

```
% Tach Signal 1
for i=1:length(z_cross1)-1
    RPM1(i) = 60/(z_cross1(1,i+1)-z_cross1(1,i));       % RPM estimate
    t_RPM1(i) = (z_cross1(1,i)+z_cross1(1,i+1))/2;      % Time
end
figure(1)
plot(t_RPM1,RPM1,'-b')
title('Zero-cross RPM estimate vs. Time - Tach Signal 1')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Regular Tach')
grid on

% Tach Signal 2
for i=1:length(z_cross2)-1
    RPM2(i) = 60/(z_cross2(1,i+1)-z_cross2(1,i));       % RPM Estimate
    t_RPM2(i) = (z_cross2(1,i)+z_cross2(1,i+1))/2;      % Time
end
figure(2)
plot(t_RPM2,RPM2,'-r')
title('Zero-cross RPM estimate vs. Time - Tach Signal 2')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Missing Tooth Tach')
grid on

figure(3)
plot(t_RPM1,RPM1,'-b')
hold on
plot(t_RPM2,RPM2,'-r')
title('Zero-cross RPM estimate vs. Time - Tach Signal 1&2 Superimposed')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Regular Tach','Missing Tooth Tach')
grid on
```


Zero-cross RPM estimate vs. Time - Tach Signal 1
×10^4
Regular Tach
RPM Estimate
Time [s]
0
1
2
3
4
5
6
0
100
200
300
400
500
600
700

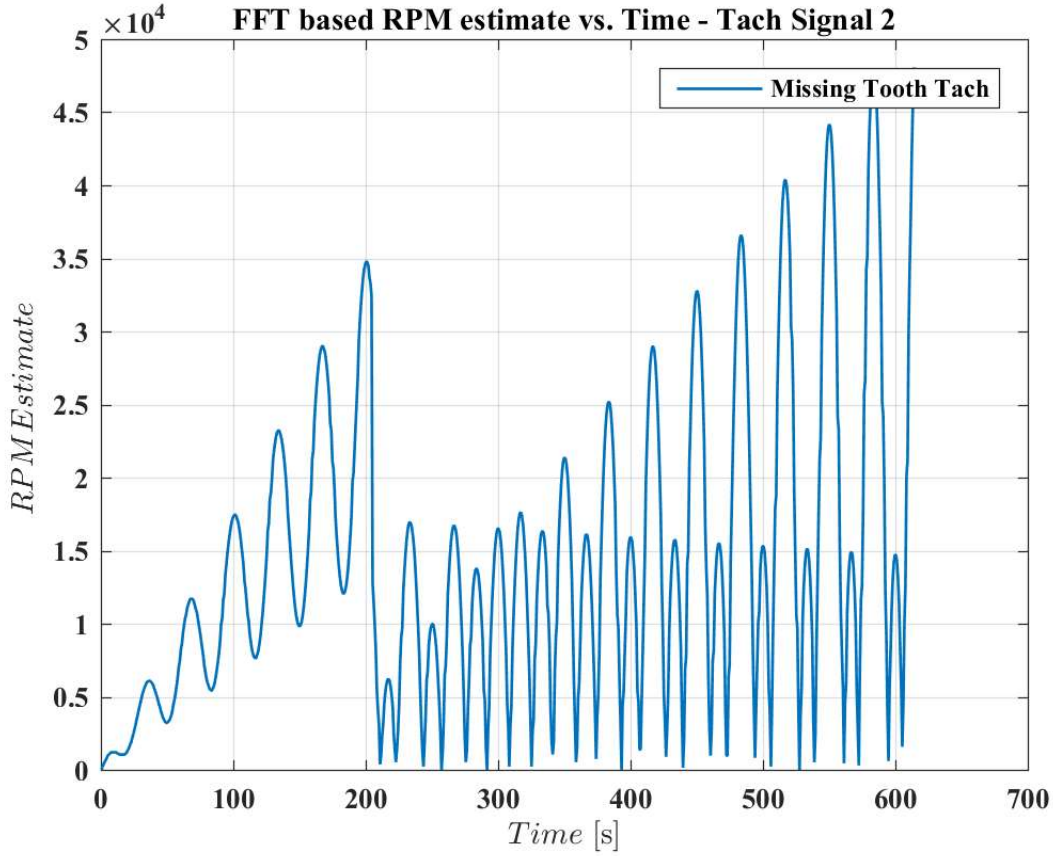

Zero-cross RPM estimate vs. Time - Tach Signal 2
×10^4
Missing Tooth Tach
RPM Estimate
Time [s]
0
1
2
3
4
5
6
0
100
200
300
400
500
600
700

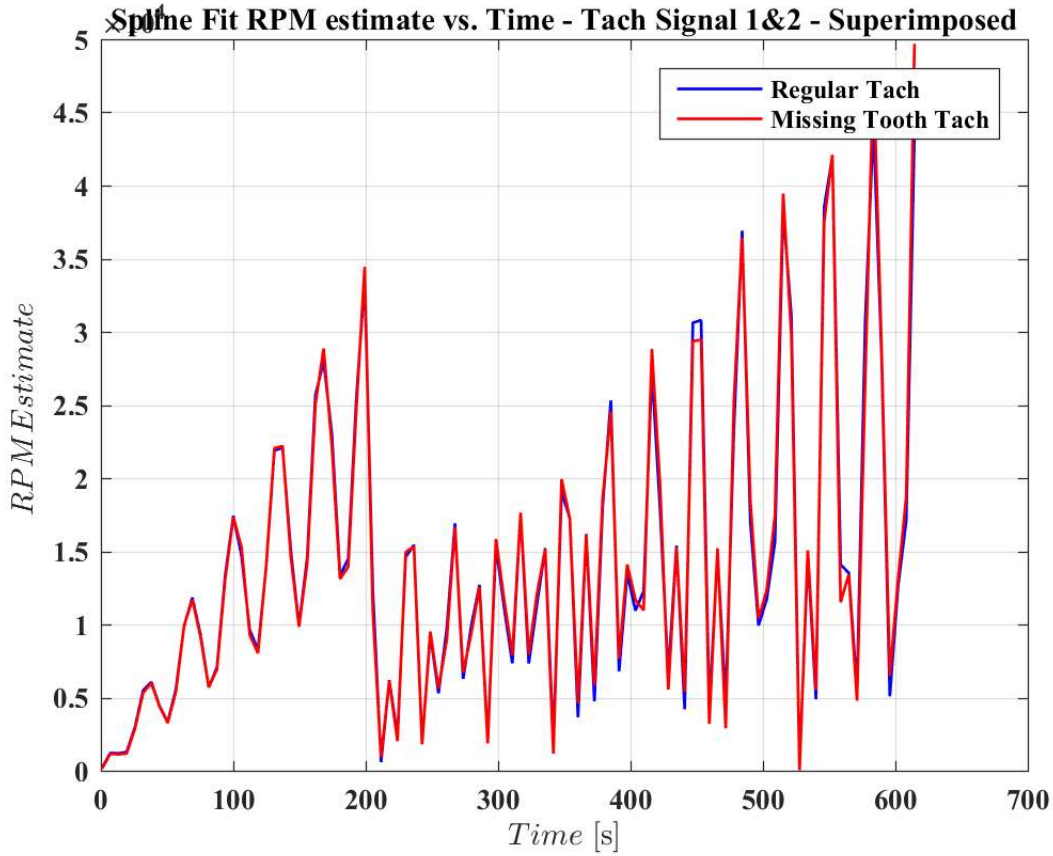


FFT Method

```
Ntime = 4096;                       %Block Size
olap_points = 0;                    %Overlapped Sample Points

% Overlapping
l=Ntime;
for i=1:length(MissToothTach(1,:))
    if l>=length(MissToothTach)
        break;
    else if i==1
        block_response(i,1:Ntime) = MissToothTach(1,1:Ntime);
        block_tt(i,1:Ntime) = tt(1,1:Ntime);
        k = Ntime-olap_points;
    else if k+Ntime<=length(MissToothTach)
        l = k+Ntime;
        block_response(i,1:Ntime) = MissToothTach(1,k+1:l);
        block_tt(i,1:Ntime) = tt(1,k+1:l);
        k = l-olap_points;
        end
        end
    end
end

Navg = size(block_response,1);      %Number of Blocks
Ttime = Ntime/Fs;                   %Time for one block
df = 1/Ttime;                       %Frequency Resolution
Fnyq = Fs/2;                        %Nyquist Frequency
f_xaxis = 0:df:Fs-df;               %Frequency axis (Upto 1/2 of Nyquist Frequency)
```

```
% FFT and Power Spectrum
for i=1:Navg
    APS(i,1:Ntime) = fft(block_response(i,:));       %Auto Power Spectrum
    APS_ign(i,:) = APS(i,2:end);                     %Ignoring the first peak at f=0Hz
end

% Calculating peak amplitude frequency block-by-block
for i=1:Navg
    [M I] = max(abs(APS_ign(i,:)));
    peak(i) = M;
    freq_index_peak(i) = I;
    freq_peak(i) = f_xaxis(I);
    block_tt_mean(i) = mean(block_tt(i,:));
end
RPMs = freq_peak.*60;

figure(4)
plot(block_tt_mean,RPMs)
title('FFT based RPM estimate vs. Time - Tach Signal 2')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Missing Tooth Tach')
grid on

t_spline1 = linspace(t_RPM1(1),t_RPM1(end),100);
RPM_spline1 = spline(t_RPM1,RPM1,t_spline1);                    % Regular Tach RPM spline curv
e
t_spline2 = linspace(t_RPM2(1),t_RPM2(end),100);
RPM_spline2 = spline(block_tt_mean,RPMs,t_spline2);             % Missing Tooth Tach RPM splin
e curve

figure(5)
plot(t_spline1,RPM_spline1,'-b')
hold on
plot(t_spline2,RPM_spline2,'-r')
title('Spline Fit RPM estimate vs. Time - Tach Signal 1&2 - Superimposed')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Regular Tach','Missing Tooth Tach')
grid on
```


FFT based RPM estimate vs. Time - Tach Signal 2
×10^4
Missing Tooth Tach
RPM Estimate
Time [s]
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
100
200
300
400
500
600
700


Spline Fit RPM estimate vs. Time - Tach Signal 1&2 - Superimposed
Regular Tach
Missing Tooth Tach
RPM Estimate
Time [s]
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
100
200
300
400
500
600
700

Published with MATLAB® R2015b

Contents

- Experimental Vibrations of Rotating Systems - Homework 8 - Shashank Iyengar (M12934513)
- Pre-processing
- 1. RPM Estimate
- 2. Speed Curve Variation
- 3. RPM Spectral Map
- 4. Order Spectral Map
- 5. 12th Order Extraction
- 12th Order Splice from Interactive Order Map Data

Experimental Vibrations of Rotating Systems - Homework 8 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('hwk8_data.mat')
```

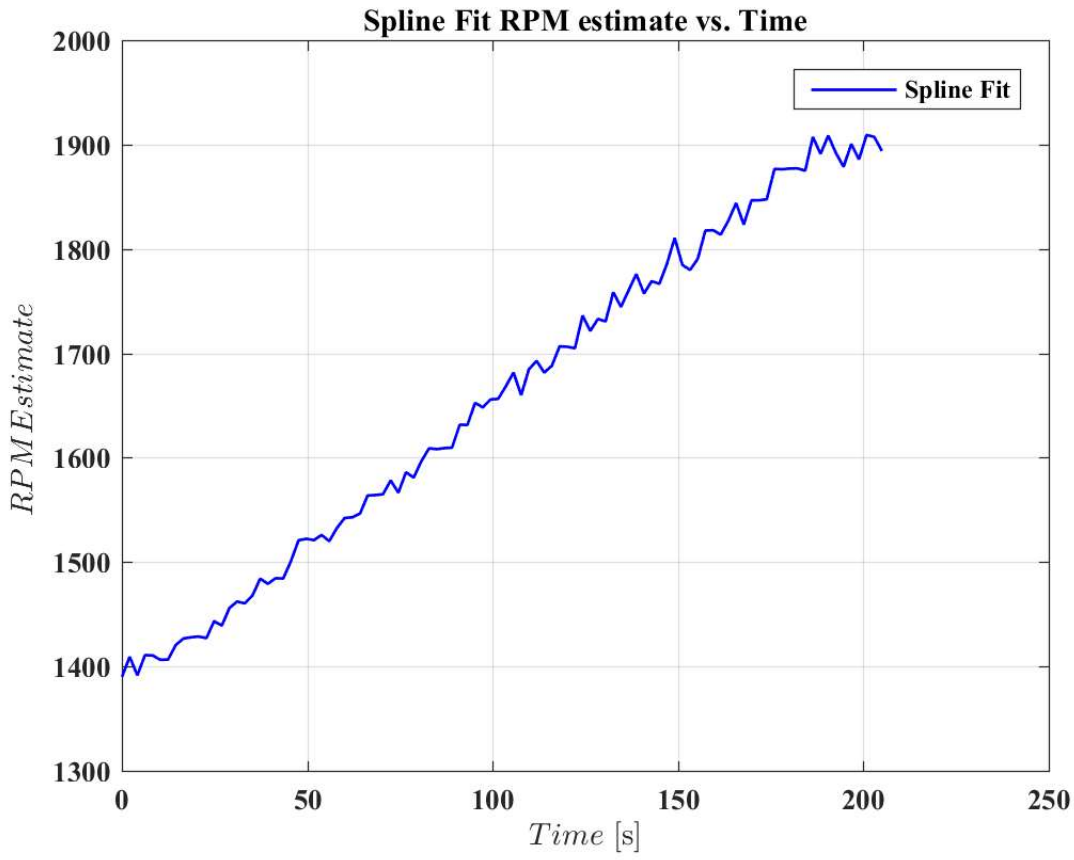
Pre-processing

```
Fs = 1/deltaT;                                  % Sampling Frequency
N = length(Response);                           % Number of sample points
Ttotal = N/Fs;                                  % Total signal time
tt = 0:deltaT:Ttotal-deltaT;                    % Time axis
```

1. RPM Estimate

```
%Zero-level crossing
n=1;
for i=1:length(tachsig)-1
    if sign(tachsig(1,i+1)) > sign(tachsig(1,i))
        z_cross(n) = ((0 - tachsig(1,i))*((tt(1,i+1)-tt(1,i))/(tachsig(1,i+1)-tachsig(1,i))))
 + tt(1,i);
        n=n+1;
    end
end
for i=1:length(z_cross)-1
    RPM(i) = 60/(z_cross(1,i+1)-z_cross(1,i));        % RPM estimate
    t_RPM(i) = (z_cross(1,i)+z_cross(1,i+1))/2;       % Time
end
t_spline = linspace(t_RPM(1),round(Ttotal),100);      % Time axis blocks
RPM_spline = spline(t_RPM,RPM,t_spline);              % RPM spline curve
```

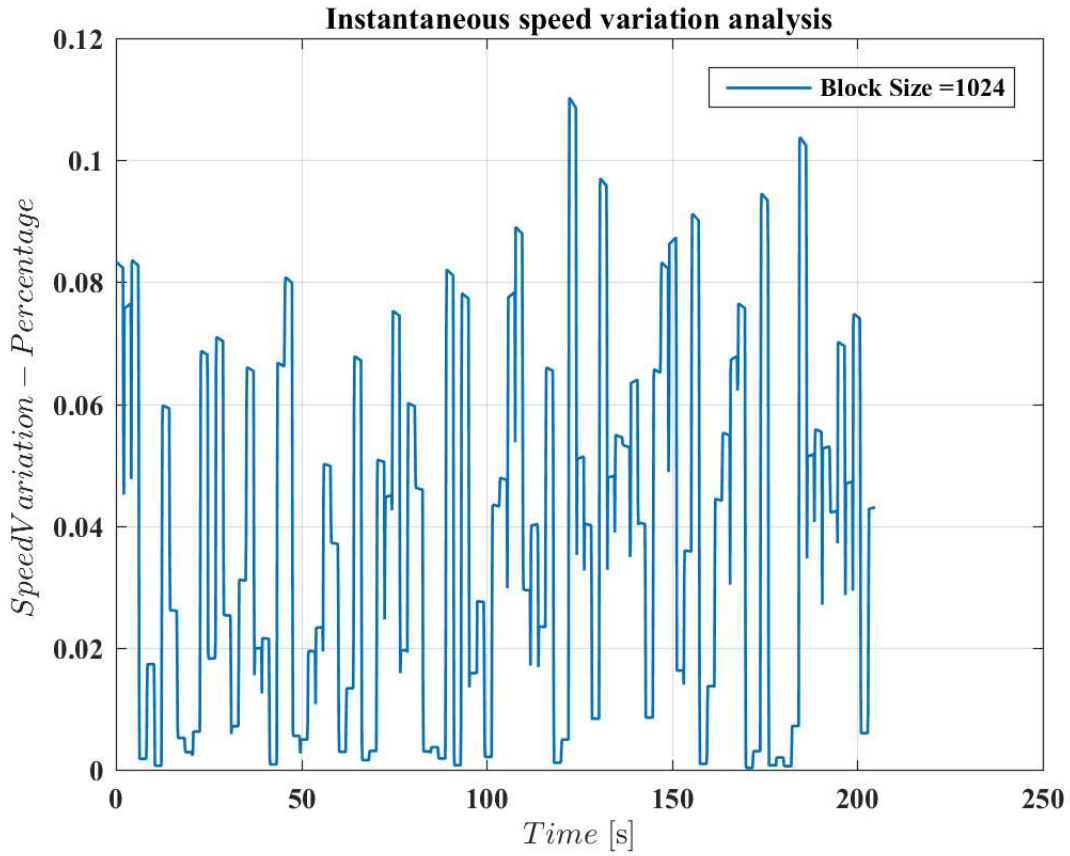
```
figure(1)
plot(t_spline,RPM_spline,'-b')
title('Spline Fit RPM estimate vs. Time')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ RPM Estimate\;\mathrm{} $'],'interpreter','latex')
legend('Spline Fit')
grid on
```



2. Speed Curve Variation

```
t_spline2 = linspace(t_spline(1),t_spline(end),length(Response));
RPM_spline2 = interp1(t_spline,RPM_spline,t_spline2);
             %RPM interpolation curve
% Ntime = input('Enter the blocksize for speed curve variance plot(512, 1024, 2048 or 4096):
');             %Block Size
% olap = input('Enter overlap percentage for speed curve variance plot(0-99): ');
                %Overlap Percentage
Ntime = 1024;
olap = 50;
olap_points = floor((Ntime*olap)/100);
             %Overlap Points
l=Ntime;
for i=1:length(RPM_spline2(1,:))
    if l>=length(RPM_spline2)
        break;
    else if i==1
        block_RPM(i,1:Ntime) = RPM_spline2(1,1:Ntime);
        block_tt(i,1:Ntime) = t_spline2(1,1:Ntime);
```

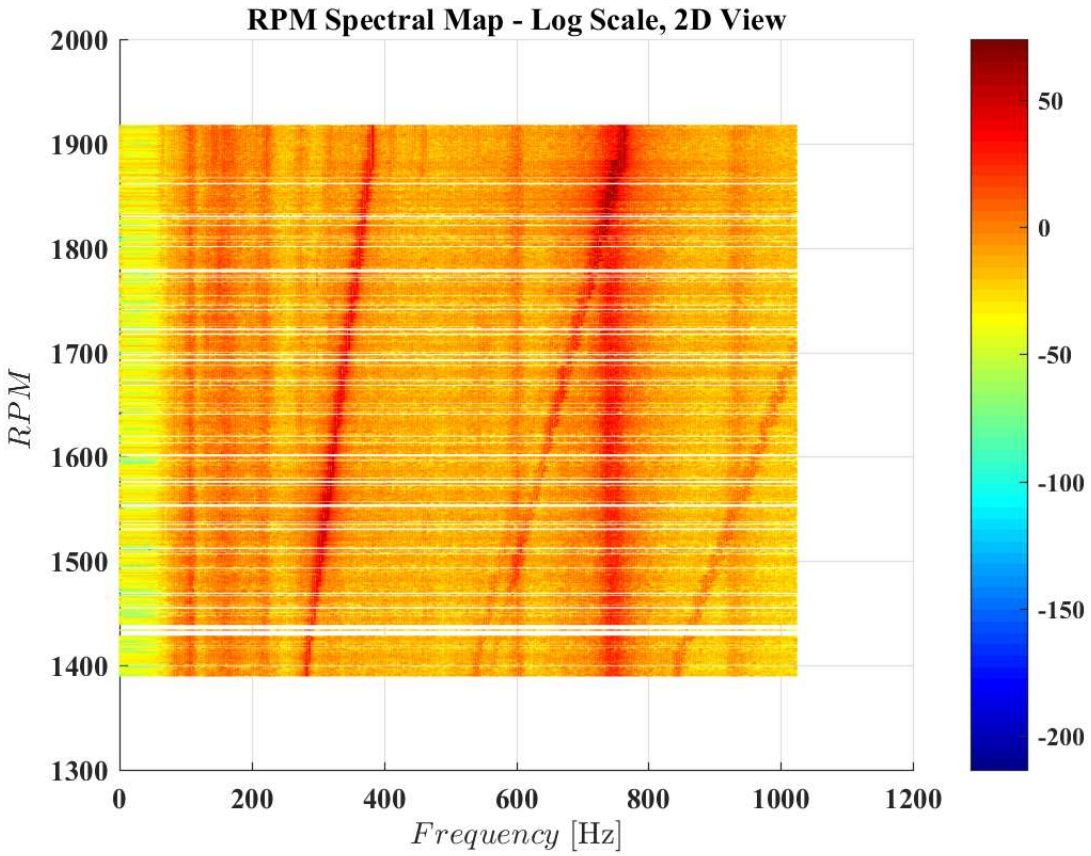
```
        k = Ntime-olap_points;
    else if k+Ntime<=length(RPM_spline2)
        l = k+Ntime;
        block_RPM(i,1:Ntime) = RPM_spline2(1,k+1:l);
        block_tt(i,1:Ntime) = t_spline2(1,k+1:l);
        k = l-olap_points;
        end
        end
    end
end
for i=1:size(block_RPM,1)
    mean_RPM(i) = mean(block_RPM(i,:));
    mean_tt(i) = mean(block_tt(i,:));
    max_RPM(i) = max(block_RPM(i,:));
    spdvar(i) = ((max_RPM(i)-mean_RPM(i))/mean_RPM(i))*100;
end
figure(2)
plot(mean_tt,spdvar)
title('Instantaneous speed variation analysis')
xlabel(['$ Time\;\mathrm{[s]} $'],'interpreter','latex')
ylabel(['$ Speed Variation - Percentage\;\mathrm{} $'],'interpreter','latex')
legend(strcat('Block Size = ', num2str(Ntime)))
grid on
```



3. RPM Spectral Map

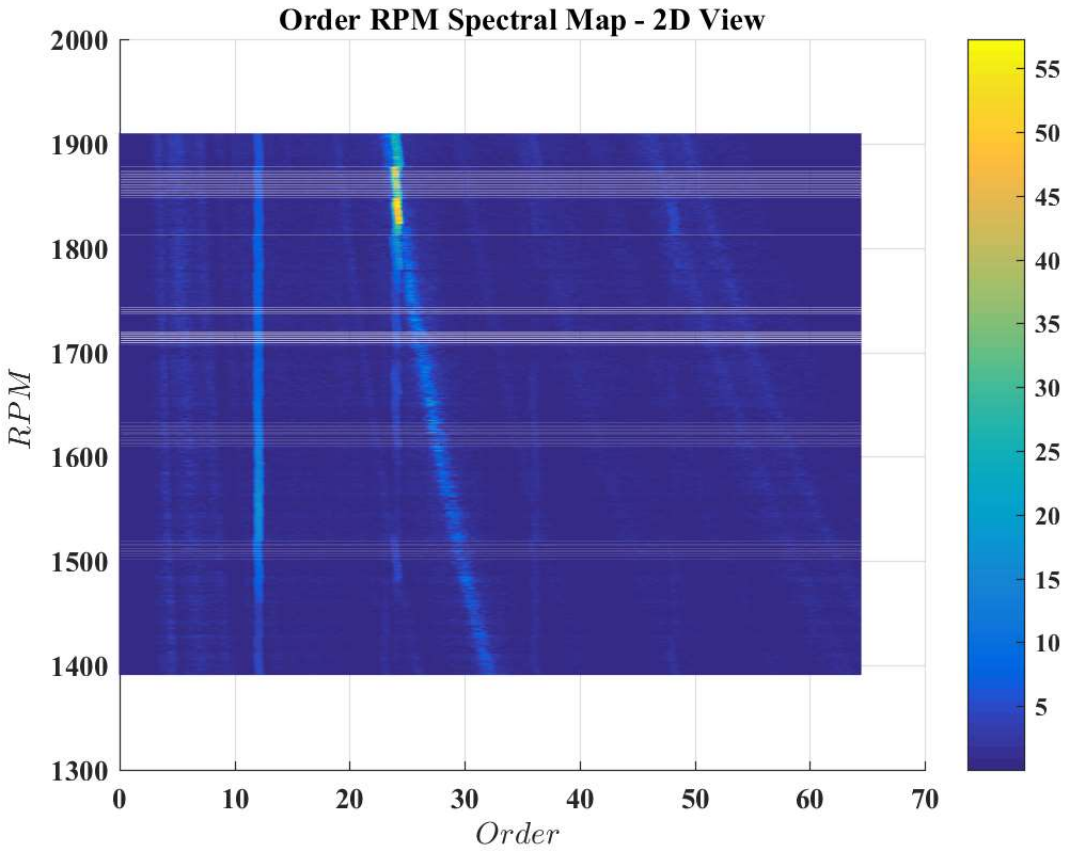
Ntime1 = input('Enter the blocksize for RPM Spectral Map(512, 1024, 2048 or 4096): '); %Block Size olap1 = input('Enter overlap percentage for RPM Spectral Map(0-99): '); %Overlap Percentage

```
Ntime1 = 1024;
olap1 = 50;
olap_points1 = floor((Ntime1*olap1)/100);
     %Overlapped Sample Points

% Overlapping
l=Ntime1;
for i=1:length(Response(1,:))
    if l>=length(Response)
        break;
    else if i==1
        block_response(i,1:Ntime1) = Response(1,1:Ntime1);
        k = Ntime1-olap_points1;
    else if k+Ntime1<=length(Response)
        l = k+Ntime1;
        block_response(i,1:Ntime1) = Response(1,k+1:l);
        k = l-olap_points1;
        end
        end
    end
end
Navg1 = size(block_response,1);                                    %Number of Blocks
Ttime1 = Ntime1/Fs;                                                %Time for one block
df1 = 1/Ttime1;                                                    %Frequency Resolution
Fnyq = Fs/2;                                                       %Nyquist Frequency
f_xaxis = 0:df1:Fnyq/2;                                            %Frequency axis (Upto 1/2 of
Nyquist Frequency)
t_spline = linspace(t_RPM(1),round(Ttotal),Navg1);                 %Time axis blocks
RPM_spline = spline(t_RPM,RPM,t_spline);                           %RPM spline curve

% FFT and Auto Power Spectrum
for i=1:Navg1
    fft_scaled(i,:) = (2/Ntime1)*fft(block_response(i,:));
    APS(i,:) = (fft_scaled(i,:)).*conj(fft_scaled(i,:));           %Auto Power Spectrum
end
APS_trunc = APS(1:length(RPM_spline),1:length(f_xaxis));
APS_scaled = APS_trunc;

% Plots - Log Scale
APS_log = mag2db(APS_scaled);
figure(3)
waterfall(f_xaxis,RPM_spline,APS_log)
colormap(jet)
title('RPM Spectral Map - Log Scale, 2D View')
xlabel(['$ Frequency\;\mathrm{[Hz]} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)
```



4. Order Spectral Map

```
[map,mapOrder,mapRPM,mapTime] = rpmordermap(Response,Fs,RPM_spline2,0.5);
freq_ordermap = linspace(0,Fnyq/2,length(mapTime));
[OR RPM_order] = meshgrid(mapOrder,mapRPM);

% rpmordermap(Response,Fs,RPM_spline2,0.5)            %Interactive Order Time Map - Order Res
olution = 0.5
figure(4)                                             %Order RPM Map
waterfall(OR,RPM_order,map')
title('Order RPM Spectral Map - 2D View')
xlabel(['$ Order\;\mathrm{} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(0,90)
```

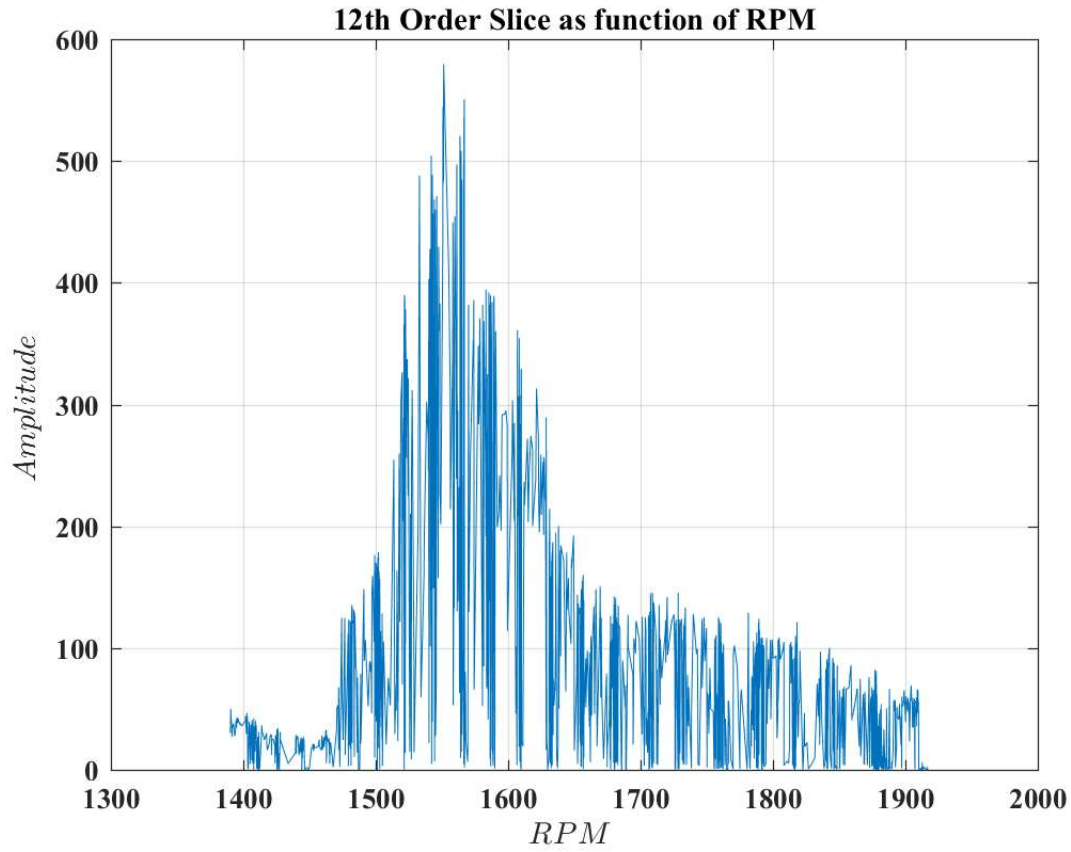


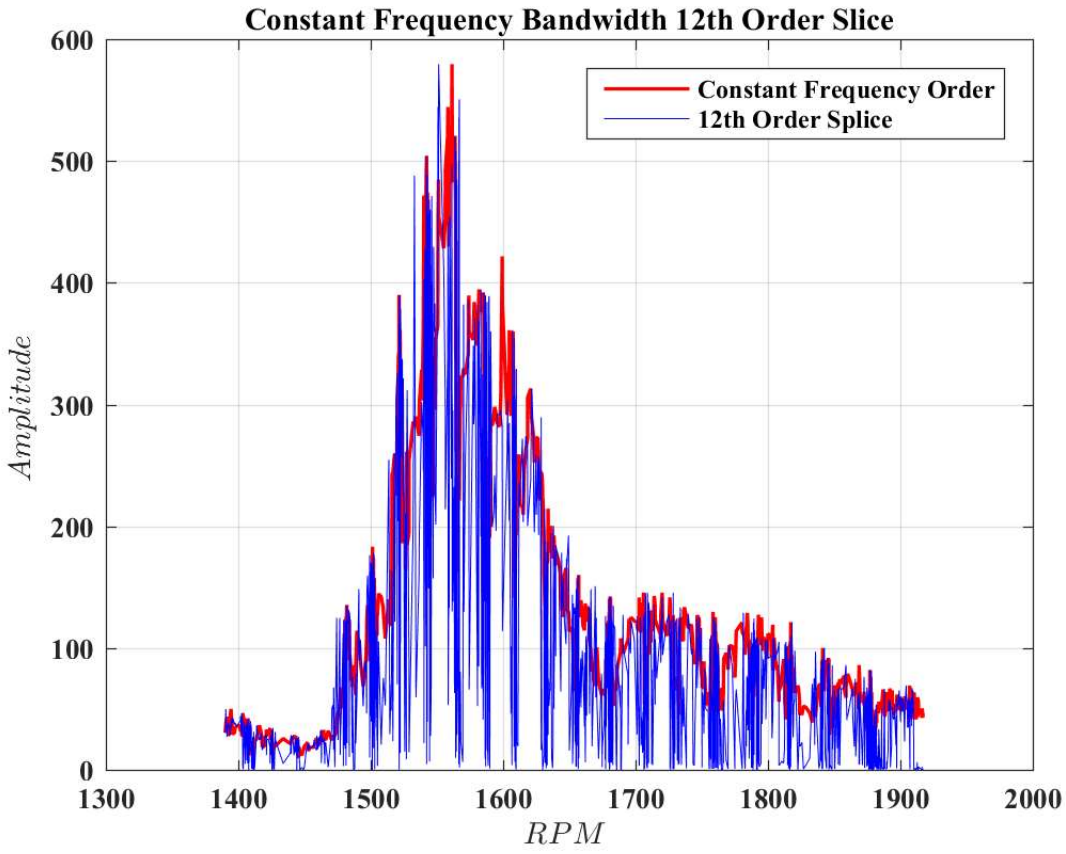
5. 12th Order Extraction

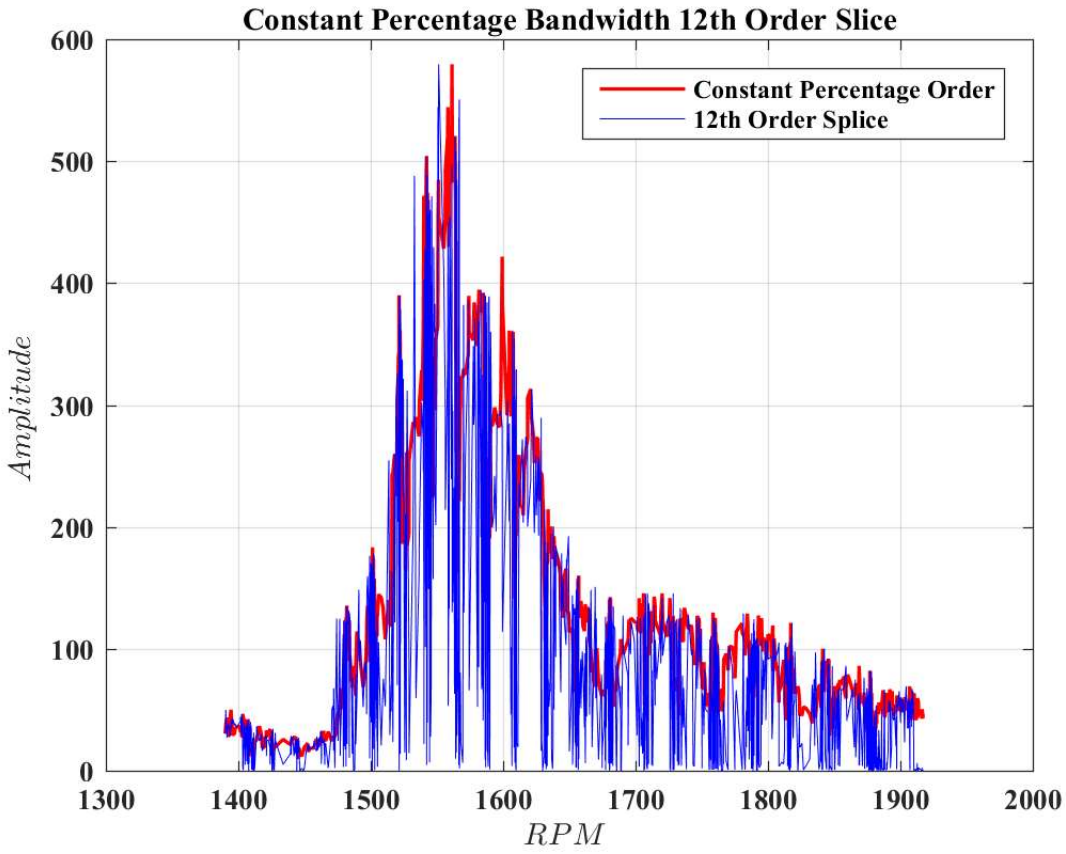
```
RPM_freq = RPM_spline./60;      %Frequency Equivalent RPM
RPM_freq_12th = RPM_freq.*12;   %12th Order Frequency Equivalent RPM
for i=1:length(RPM_freq_12th)
    for j=1:length(f_xaxis)
        temp(j) = abs(RPM_freq_12th(i)-f_xaxis(j));
    end
    [fr(i) id(i)] = min(temp);
    temp1=APS_scaled(i,:);
    mag_12th(i) = temp1(id(i));
end
[RPM_sorted sort_id] = sort(RPM_spline);
for i=1:length(mag_12th)
    ord(i) = mag_12th(1,sort_id(i));
end
figure(5)
plot(RPM_sorted,ord,'LineWidth',0.000001)
title('12th Order Slice as function of RPM')
xlabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
ylabel(['$ Amplitude\;\mathrm{} $'],'interpreter','latex')
grid on

%Constant Frequency Banwidth
for i=1:length(id)
    lb(i) = id(i)-10;
    ub(i) = id(i)+10;
end
for i=1:length(RPM_freq_12th)
```

```
    temp1 = APS_scaled(i,:);
    temp2=lb(i);
    for x=1:21
        cbmagtemp(i,x) = temp1(1,temp2);
        temp2=temp2+1;
    end
    cbmag(i) = max(cbmagtemp(i,:));
end
figure(6)
plot(RPM_sorted,cbmag,'-r','LineWidth',1.25)
hold on
plot(RPM_sorted,ord,'-b','LineWidth',0.000001)
title('Constant Frequency Bandwidth 12th Order Slice')
xlabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
ylabel(['$ Amplitude\;\mathrm{} $'],'interpreter','latex')
grid on
legend('Constant Frequency Order','12th Order Splice')

%Constant Percentage Bandwidth
for i=1:length(id)
    if i==1 && i<=100
        lb(i) = (id(i)-10)-floor(id(i)*12/100);
        ub(i) = (id(i)+10)+floor(id(i)*12/100);

    else if i==101 && i<=200
        lb(i) = (id(i)-20)-floor(id(i)*12/100);
        ub(i) = (id(i)+20)+floor(id(i)*12/100);
    else if i==201 && i<=300
        lb(i) = (id(i)-30)-floor(id(i)*12/100);
        ub(i) = (id(i)+30)+floor(id(i)*12/100);
    else
        lb(i) = (id(i)-40)-floor(id(i)*12/100);
        ub(i) = (id(i)+40)+floor(id(i)*12/100);
        end
        end
    end
end
for i=1:length(RPM_freq_12th)
    temp1 = APS_scaled(i,:);
    temp2=lb(i);
    if i==1 && i<=100
        for x=1:(ub(i)-lb(i))
            cbmagtemp(i,x) = temp1(1,temp2);
            temp2=temp2+1;
        end
    else if i==101 && i<=200
        for x=1:(ub(i)-lb(i))
            cbmagtemp(i,x) = temp1(1,temp2);
            temp2=temp2+1;
        end
    else if i==201 && i<=300
        for x=1:(ub(i)-lb(i))
            cbmagtemp(i,x) = temp1(1,temp2);
            temp2=temp2+1;
        end
    else
        for x=1:(ub(i)-lb(i))
```

```
            cbmagtemp(i,x) = temp1(1,temp2);
            temp2=temp2+1;
        end
        end
        end
    end
    cbmag1(i) = max(cbmagtemp(i,:));
end
figure(7)
plot(RPM_sorted,cbmag1,'-r','LineWidth',1.25)
hold on
plot(RPM_sorted,ord,'-b','LineWidth',0.000001)
title('Constant Percentage Bandwidth 12th Order Slice')
xlabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
ylabel(['$ Amplitude\;\mathrm{} $'],'interpreter','latex')
grid on
legend('Constant Percentage Order','12th Order Splice')
```







12th Order Splice from Interactive Order Map Data

```
for j=1:length(mapOrder) if floor(mapOrder(j))==12 ind = j; break; end end figure(5) plot(mapRPM,map(ind,:)) title('12th Order Slice') xlabel(['$ RPM\;\mathrm{}
```
$$'],'interpreter','latex')ylabel(['$ Amplitude\;\mathrm{} $'],'interpreter','latex')$$
```
grid on
```

Published with MATLAB® R2015b

Contents

- Experimental Vibrations of Rotating Systems - Homework 9 - Shashank Iyengar (M12934513)
- Pre-processing
- RPM Estimate
- RPM Spectral Map
- 12th Order Extraction
- TVDFT Method

Experimental Vibrations of Rotating Systems - Homework 9 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('hwk8_data.mat')
load('BlockData.mat')
```

Pre-processing

```
Fs = 1/deltaT;                                  % Sampling Frequency
N = length(Response);                           % Number of sample points
Ttotal = N/Fs;                                  % Total signal time
tt = 0:deltaT:Ttotal-deltaT;                    % Time axis
```

RPM Estimate

```
%Zero-level crossing
n=1;
for i=1:length(tachsig)-1
    if sign(tachsig(1,i+1)) > sign(tachsig(1,i))
        z_cross(n) = ((0 - tachsig(1,i))*((tt(1,i+1)-tt(1,i))/(tachsig(1,i+1)-tachsig(1,i))))
 + tt(1,i);
        n=n+1;
    end
end
for i=1:length(z_cross)-1
    RPM(i) = 60/(z_cross(1,i+1)-z_cross(1,i));       % RPM estimate
    t_RPM(i) = (z_cross(1,i)+z_cross(1,i+1))/2;      % Time
end
t_spline = linspace(t_RPM(1),round(Ttotal),100);     % Time axis blocks
RPM_spline = spline(t_RPM,RPM,t_spline);             % RPM spline curve
```

RPM Spectral Map

```
display('Block Size = 1024');
display('Overlap = 50%');
display('Above parameters selected from Homework 8');
Ntime1 = 1024;            %Block Size
olap1 = 50;                          %Overlap Percentage
olap_points1 = floor((Ntime1*olap1)/100);                                    %Overlapped Sa
mple Points

% Overlapping
l=Ntime1;
for i=1:length(Response(1,:))
    if l>=length(Response)
        break;
    else if i==1
        block_response(i,1:Ntime1) = Response(1,1:Ntime1);
        k = Ntime1-olap_points1;
    else if k+Ntime1<=length(Response)
        l = k+Ntime1;
        block_response(i,1:Ntime1) = Response(1,k+1:l);
        k = l-olap_points1;
        end
        end
    end
end
Navg1 = size(block_response,1);                              %Number of Blocks
Ttime1 = Ntime1/Fs;                                          %Time for one block
df1 = 1/Ttime1;                                              %Frequency Resolution
Fnyq = Fs/2;                                                 %Nyquist Frequency
f_xaxis = 0:df1:Fnyq/2;                                      %Frequency axis (Upto 1/2 of Nyqu
ist Frequency)
t_spline = linspace(t_RPM(1),round(Ttotal),Navg1);           %Time axis blocks
RPM_spline = spline(t_RPM,RPM,t_spline);                     %RPM spline curve

% FFT and Power Spectrum
for i=1:Navg1
    fft_scaled(i,:) = (2/Ntime1)*fft(block_response(i,:));
    APS(i,:) = (fft_scaled(i,:)).*conj(fft_scaled(i,:));       %Auto Power Spectrum
end
APS_trunc = APS(1:length(RPM_spline),1:length(f_xaxis));
APS_scaled = APS_trunc;
```

```
Block Size = 1024
Overlap = 50%
Above parameters selected from Homework 8
```

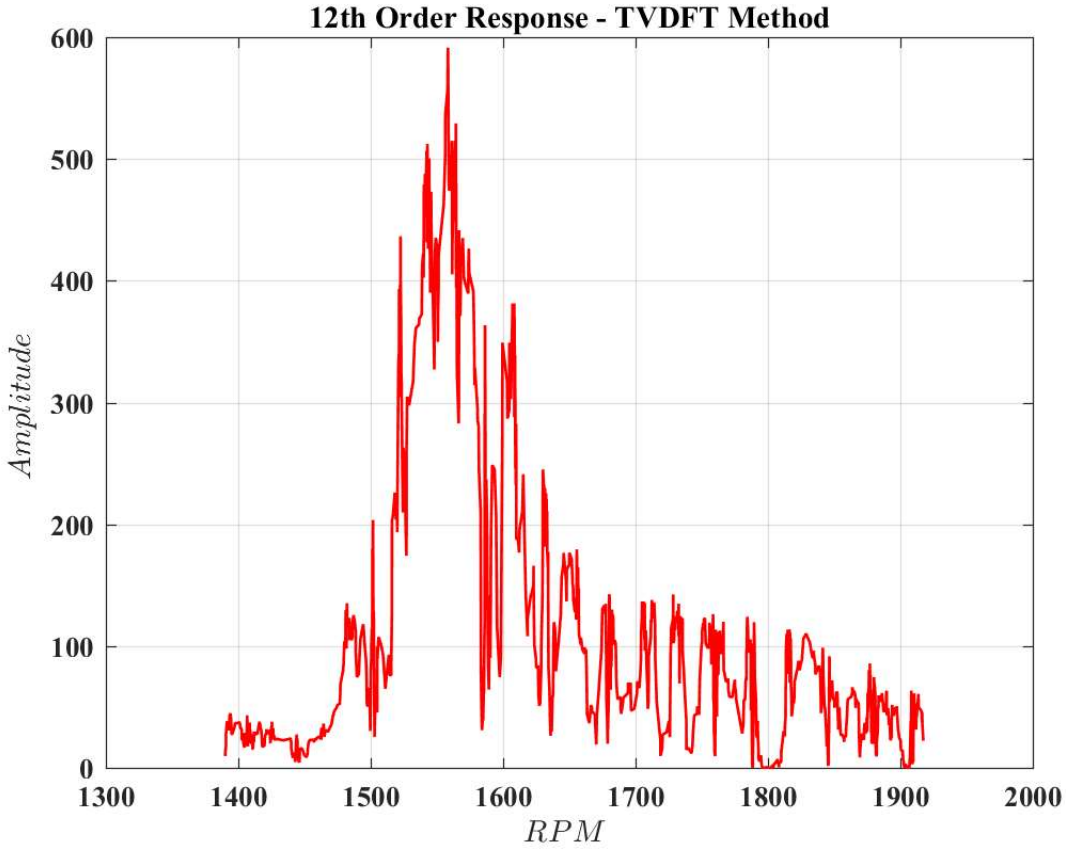
12th Order Extraction

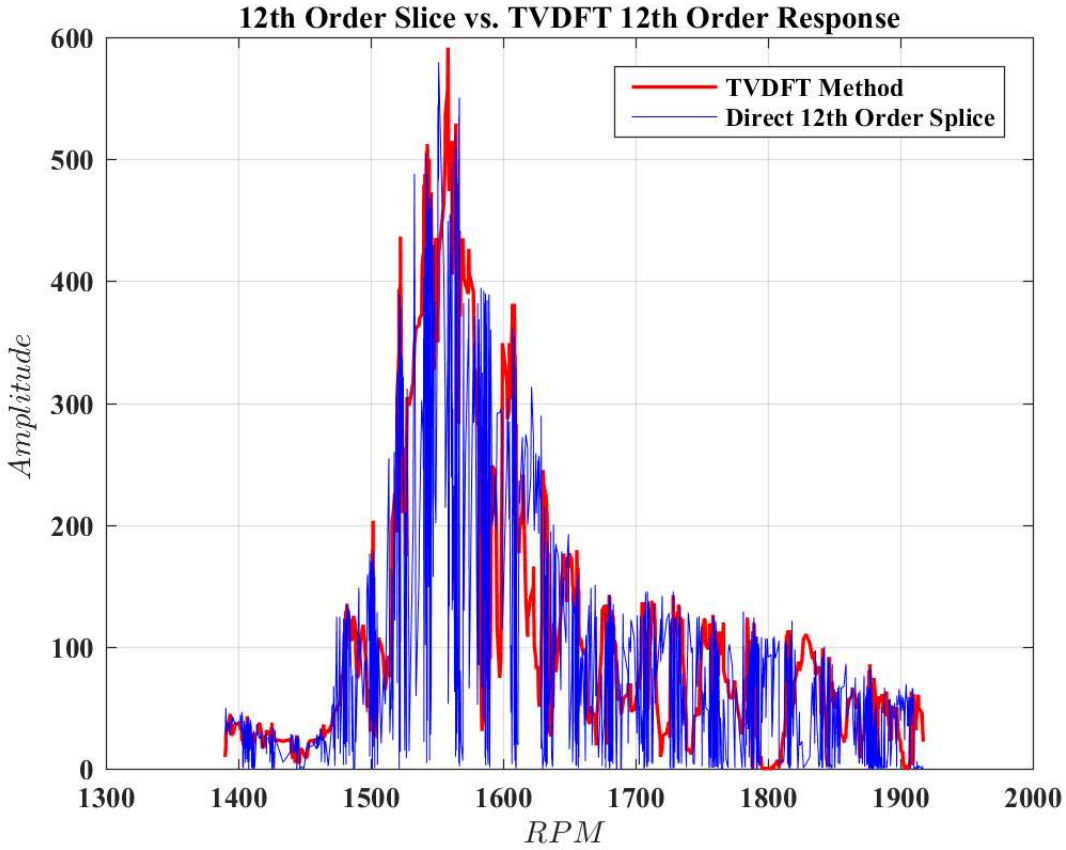
```
RPM_freq = RPM_spline./60;       %Frequency Equivalent RPM
RPM_freq_12th = RPM_freq.*12;    %12th Order Frequency Equivalent RPM
for i=1:length(RPM_freq_12th)
    for j=1:length(f_xaxis)
        temp(j) = abs(RPM_freq_12th(i)-f_xaxis(j));
    end
```

```
    [fr(i) id(i)] = min(temp);
    temp1=APS_scaled(i,:);
    mag_12th(i) = temp1(id(i));
end
[RPM_sorted sort_id] = sort(RPM_spline);
for i=1:length(mag_12th)
    ord(i) = mag_12th(1,sort_id(i));
end
```

TVDFT Method

```
t_spline2 = linspace(t_spline(1),t_spline(end),length(Response));
RPM_spline2 = interp1(t_spline,RPM_spline,t_spline2);

block_response_t=block_response';
block_RPM_t=block_RPM';
size_matrix=size(block_response_t);
a=zeros(1,size_matrix(2));
b=zeros(1,size_matrix(2));
for i=1:length(block_response(:,1))
    a = cos(2*pi*12*deltaT*cumsum(block_RPM_t(:,i)')/60)*block_response_t(:,i);     %Coeffici
ent a_m
    b = sin(2*pi*12*deltaT*cumsum(block_RPM_t(:,i)')/60)*block_response_t(:,i);     %Coeffici
ent b_m
    a = 2*a/Ntime1;                                                                    %Scal
ing
    b = 2*b/Ntime1;                                                                    %Scal
ing
    tvdft_final(i)=sqrt(a^2+b^2);                                                  %TVDFT
    tvdft_ap(i) = a^2+b^2;                                                         %TVDFT Au
to Power
end
figure(1)
plot(RPM_sorted,tvdft_ap,'-r')
title('12th Order Response - TVDFT Method')
xlabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
ylabel(['$ Amplitude\;\mathrm{} $'],'interpreter','latex')
grid on

figure(2)
plot(RPM_sorted,tvdft_ap,'-r','LineWidth',1.25)
hold on
plot(RPM_sorted,ord,'-b','LineWidth',0.000001)
title('12th Order Slice vs. TVDFT 12th Order Response')
xlabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
ylabel(['$ Amplitude\;\mathrm{} $'],'interpreter','latex')
grid on
legend('TVDFT Method','Direct 12th Order Splice')
```


12th Order Response - TVDFT Method
Amplitude
RPM
0
100
200
300
400
500
600
1300
1400
1500
1600
1700
1800
1900
2000


12th Order Slice vs. TVDFT 12th Order Response
TVDFT Method
Direct 12th Order Splice
Amplitude
RPM
0
100
200
300
400
500
600
1300
1400
1500
1600
1700
1800
1900
2000

Published with MATLAB® R2015b

Contents

- Experimental Vibrations of Rotating Systems - Homework 10 - Shashank Iyengar (M12934513)
- Pre-processing
- RPM Estimate
- Kalman Filter Code Outline obtained from: Tuma, J., "Vold-Kalman Order Trackiong Filtration"
- First Generation Vold-Kalman Filter
- Second Generation Vold-Kalman Filter

Experimental Vibrations of Rotating Systems - Homework 10 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('hwk8_data.mat')
load('BlockData.mat')
load('TVDFTdata.mat')
```

Pre-processing

```
Fs = 1/deltaT;                                  % Sampling Frequency
N = length(Response);                           % Number of sample points
Ttotal = N/Fs;                                  % Total signal time
tt = 0:deltaT:Ttotal-deltaT;                    % Time axis
```

RPM Estimate

```
%Zero-level crossing
n=1;
for i=1:length(tachsig)-1
    if sign(tachsig(1,i+1)) > sign(tachsig(1,i))
        z_cross(n) = ((0 - tachsig(1,i))*((tt(1,i+1)-tt(1,i))/(tachsig(1,i+1)-tachsig(1,i))))
 + tt(1,i);
        n=n+1;
    end
end
for i=1:length(z_cross)-1
    RPM(i) = 60/(z_cross(1,i+1)-z_cross(1,i));        % RPM estimate
    t_RPM(i) = (z_cross(1,i)+z_cross(1,i+1))/2;       % Time
end
t_spline = linspace(t_RPM(1),round(Ttotal),100);      % Time axis blocks
RPM_spline = spline(t_RPM,RPM,t_spline);              % RPM spline curve

t_spline2 = linspace(t_spline(1),t_spline(end),length(Response));
```

```
RPM_2 = interp1(t_spline,RPM_spline,t_spline2);
RPM_freq = RPM_2./60;
```

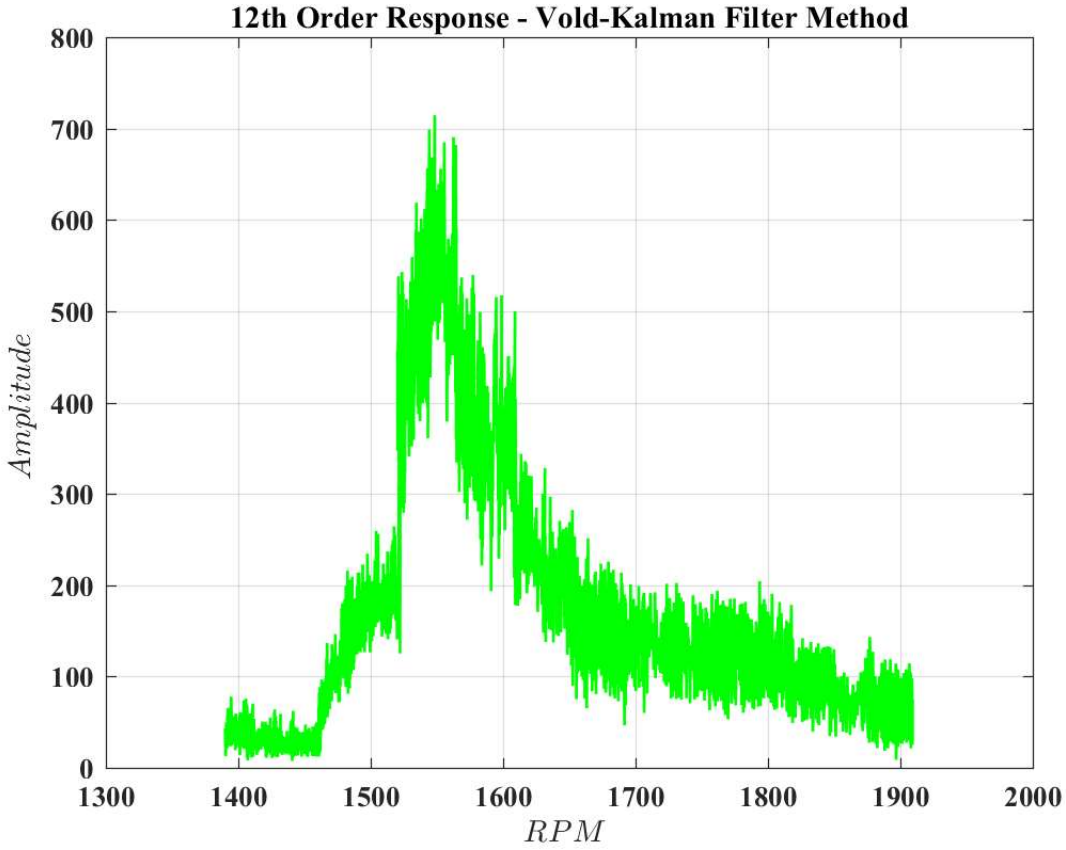
Kalman Filter Code Outline obtained from: Tuma, J., "Vold-Kalman Order Trackiong Filtration"

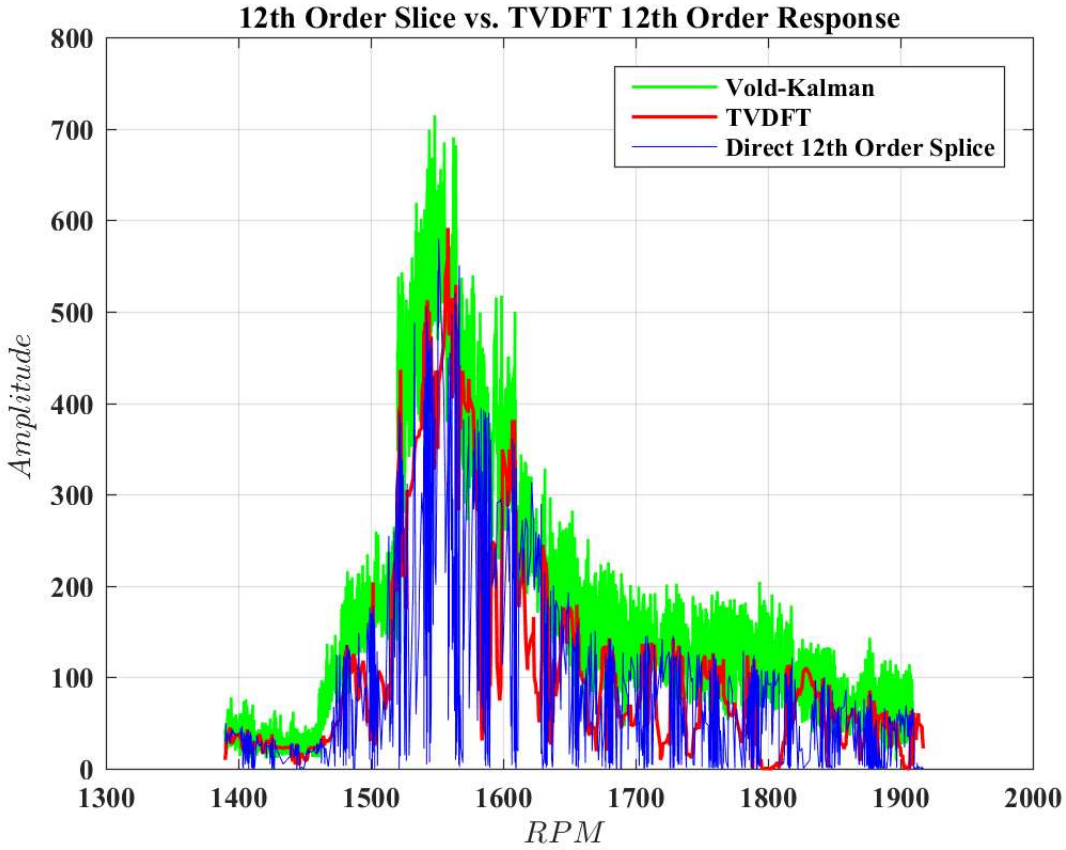
First Generation Vold-Kalman Filter

```
r = 1;
c = 2*cos(2*pi*RPM_freq'*deltaT*12);
N2 = N-2;
e = ones(N2,1);
A = spdiags([e -2*c(1:N2) e],0:2,N2,N);
AA = r*r*A'*A + speye(N,N);
x = AA\Response';
AP = x.*conj(x);
```

Second Generation Vold-Kalman Filter

```
NR = N-2;
r1 = 1000;
e1 = ones(NR,1);
A1 = spdiags([e1 -2*e1 e1],0:2,NR,N);
AA1 = r1*r1*A1'*A1 + speye(N);
yy1 = exp(-j*2*pi*cumsum(RPM_freq)*deltaT*12).*Response;
x1 = 2*(AA1\yy1');
AP1 = x1.*conj(x1);

figure(1)
plot(RPM_2,AP1,'-g')
title('12th Order Response - Vold-Kalman Filter Method')
xlabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
ylabel(['$ Amplitude\;\mathrm{} $'],'interpreter','latex')
grid on

figure(2)
plot(RPM_2,AP1,'-g')
hold on
plot(RPM_sorted,tvdft_ap,'-r','LineWidth',1.25)
hold on
plot(RPM_sorted,ord,'-b','LineWidth',0.000001)
title('12th Order Slice vs. TVDFT 12th Order Response')
xlabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
ylabel(['$ Amplitude\;\mathrm{} $'],'interpreter','latex')
grid on
legend('Vold-Kalman','TVDFT','Direct 12th Order Splice')
```


12th Order Response - Vold-Kalman Filter Method
Amplitude
RPM
0
100
200
300
400
500
600
700
800
1300
1400
1500
1600
1700
1800
1900
2000


12th Order Slice vs. TVDFT 12th Order Response
Vold-Kalman
TVDFT
Direct 12th Order Splice
Amplitude
RPM
0
100
200
300
400
500
600
700
800
1300
1400
1500
1600
1700
1800
1900
2000

Published with MATLAB® R2015b

Contents

- Experimental Vibrations of Rotating Systems - Homework 11 - Shashank Iyengar (M12934513)
- Pre-processing
- RPM Estimate
- Order Processing

Experimental Vibrations of Rotating Systems - Homework 11 - Shashank Iyengar (M12934513)

```
close all
clear all
clc
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 10)
set(0,'defaultlinelinewidth',1)
set(0,'DefaultLineMarkerSize', 6)
set(0,'defaultAxesFontWeight','bold')

load('hwk8_data.mat')
load('BlockData.mat')
load('TVDFTdata.mat')
```

Pre-processing

```
Fs = 1/deltaT;                                  % Sampling Frequency
N = length(Response);                           % Number of sample points
Ttotal = N/Fs;                                  % Total signal time
tt = 0:deltaT:Ttotal-deltaT;                    % Time axis
pulses = 4;
```

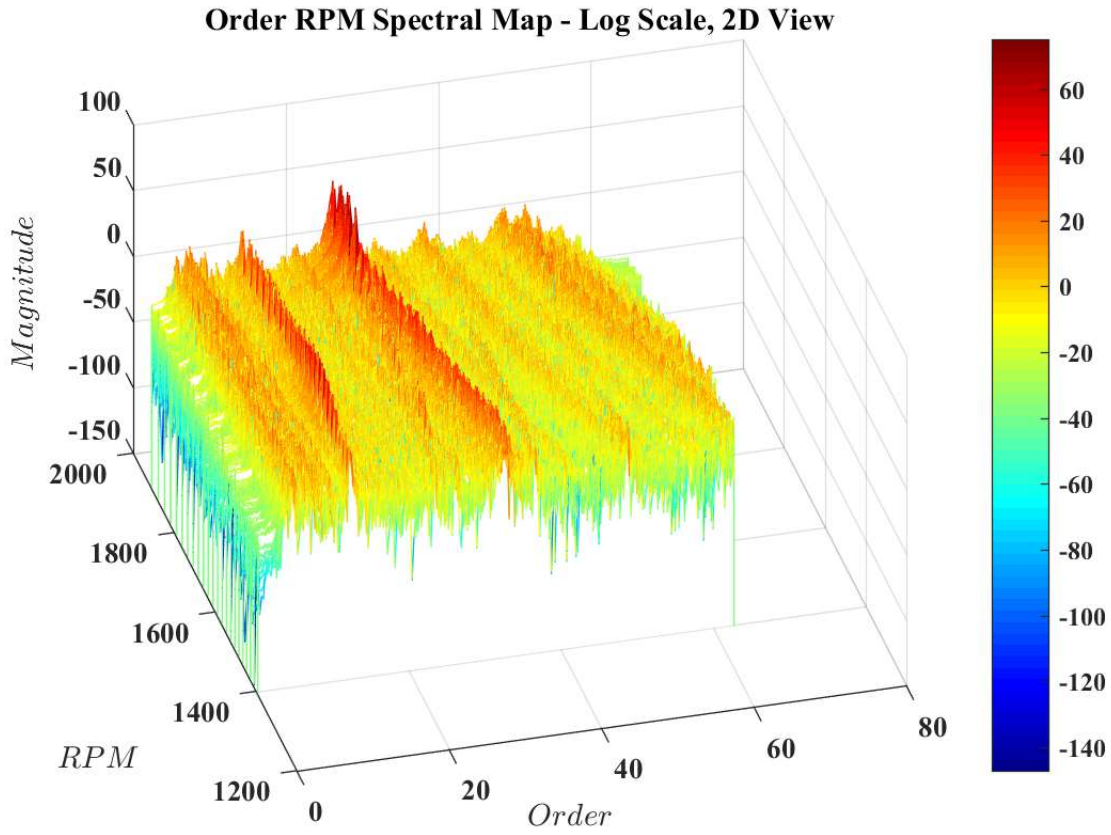
RPM Estimate

```
%Zero-level crossing
n=1;
for i=1:length(tachsig)-1
    if sign(tachsig(1,i+1)) > sign(tachsig(1,i))
        z_cross(n) = ((0 - tachsig(1,i))*((tt(1,i+1)-tt(1,i))/(tachsig(1,i+1)-tachsig(1,i))))
 + tt(1,i);
        d(n) = i;
        n=n+1;
    end
end
for i=1:length(z_cross)-1
    RPM(i) = 60/(z_cross(1,i+1)-z_cross(1,i));       % RPM estimate
    t_RPM(i) = (z_cross(1,i)+z_cross(1,i+1))/2;      % Time
end
t_spline = linspace(t_RPM(1),round(Ttotal),100);     % Time axis blocks
RPM_spline = spline(t_RPM,RPM,t_spline);             % RPM spline curve

t_spline2 = linspace(t_spline(1),t_spline(end),length(Response));
RPM_2 = interp1(t_spline,RPM_spline,t_spline2);
```

```
RPM_freq = RPM_2./60;
```

Order Processing

```
o_max = (2000/max(RPM_spline))*60;
% o_max_u = input('Enter maximum order to be tracked: ');
% delta_o = input('Enter order resolution: ');
o_max_u = 12;
delta_o = 0.25;
R = 1/delta_o;

RPM_pulse = RPM(1:R:floor(length(RPM/R))-R);
N = length(RPM_pulse);

for i=1:N
    inst_RPM(i) = RPM_pulse(i);         %Instantaneous RPM
    inst_T = R*60/inst_RPM(i);            %Instantaneous time for one block
    inst_block = round(inst_T/deltaT); %Speed dependant block size
    id1 = d(4*(i-1)+1);
    id2 = inst_block+id1-1;
    inst_Response = Response(id1:id2);
    block_fft = (2/inst_block)*fft(inst_Response);
    inst_fft(i,:) = block_fft(1:floor(o_max/delta_o));
end

o_axis = 0:delta_o:(round(o_max/delta_o)-1)*delta_o;
AP = inst_fft.*conj(inst_fft);

% Plots - Log Scale
APS_log = mag2db(AP);
figure(1)
waterfall(o_axis,RPM_pulse,APS_log)
colormap(jet)
title('Order RPM Spectral Map - Log Scale, 2D View')
xlabel(['$ Order\;\mathrm{} $'],'interpreter','latex')
ylabel(['$ RPM\;\mathrm{} $'],'interpreter','latex')
zlabel(['$ Magnitude\;\mathrm{} $'],'interpreter','latex')
colorbar
view(-15,45)
```



Published with MATLAB® R2015b