# Power-Specification Frequency-Domain Test and Analysis Methodology for Large Dynamic Loads

Tyler Boehmer and Deanna Temkin

January 2, 2018, Rev. 0

## 1 Overview

A new frequency-domain requirement (see Figure 1) is being proposed for emerging Navy systems that exhibit large dynamic loads. For these requirements to be applied as intended, it is crucial that the test and analysis methodology be clearly defined. Changes in either the measuring equipment or the method of Discrete Fourier Transform (DFT) analysis can cause sizable changes in results (particularly the DFT analysis). As such, this document details a proposed test and analysis methodology as it relates to compliance with the new frequency-domain power requirement.

The motivation for this new requirement is to protect the generator and prime mover from potential stressful load profiles while also maintaining good power bus quality for the electric plant. Hence, this new requirement would be imposed at the system's interface with the ship's electric power bus. The proposed requirement is as follows: a new system shall limit its combined three-phase instantaneous power ripple, as seen by the shipboard generator(s), at any single frequency from 0.01 Hz to 2 kHz to be less than the limits shown in Figure 1. In Figure 1, the average power is the mean of the three-phase instantaneous power over a 100-second averaging window, and the alternating current (AC) power is defined as the amplitude of the three-phase instantaneous power (zero to peak) at each of the individual decomposed frequency components (e.g., at 1 Hz, the allowed amplitude of the ripple is 3 % of the average load, where the average load is determined over the 100-second window).

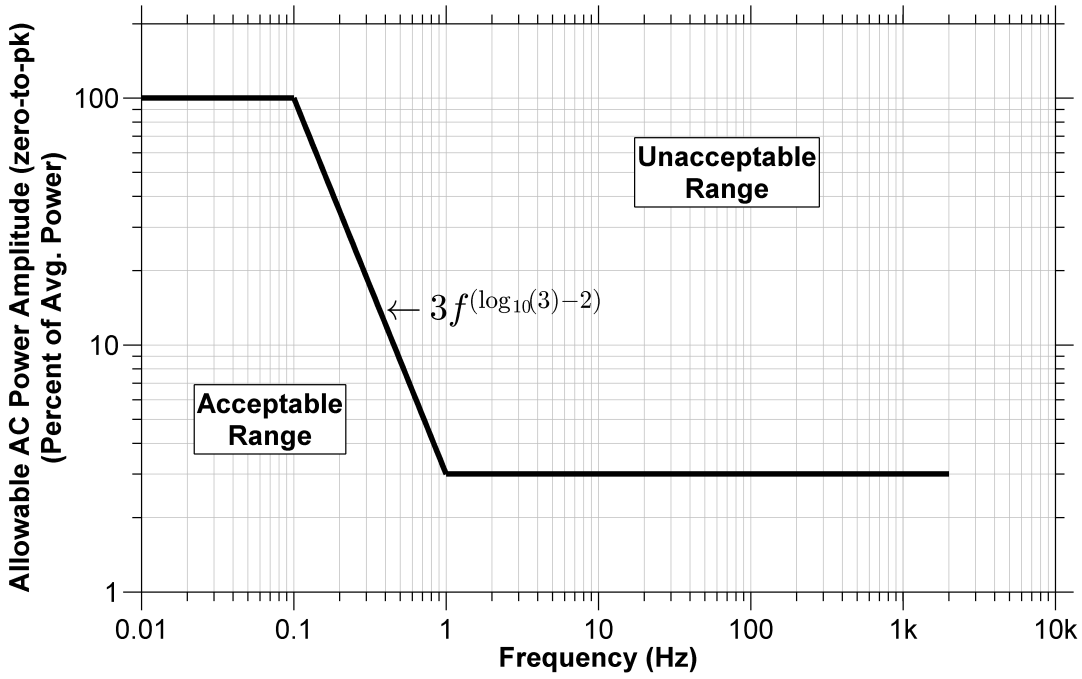


Figure 1: The recently proposed frequency-domain power specification for large dynamic loads.

# 2 Test Methodology

## 2.1 Hardware

The hardware used to measure the power is critical in achieving accurate results; however, that does not dictate a need for expensive equipment. Many workable solutions exist, from high-accuracy low-bandwidth oscilloscopes to dedicated power meters. As such, only very general specifications are provided here.

The measurement setup will consist of three line-to-neutral voltage measurements and three current measurements. Each instantaneous voltage–current measurement pair (one pair for each phase) will be multiplied, and the three powers summed together to find the total three-phase active power that the prime mover will see. There are two critical factors in this measurement: the accuracy of each individual measurement and the skew between the various measurements. The first is more readily identifiable by examining the full-scale accuracies of the probes and measuring instruments, as well as the number of effective bits of resolution. The latter, however, is trickier. The measurements need to be very close to simultaneous to produce an accurate measurement. Consequently, a truly in-phase voltage and current at the maximum measured frequency should result in a measurement of an in-phase (or very close to it) voltage and current. This is particularly important with current probes, which can exhibit more phase delays than voltage probes at these lower frequencies, as well as the multichannel data acquisition equipment (skew between channels).

The effective bits of resolution in the combined digitizer and probes is another important factor in the measuring instrumentation selection. Resolution affects the accuracy of the instrument and determines the dynamic range it can measure, which is important because the measurement must accurately distinguish AC signals smaller than 3 % of the average signal. An effective 6 bits of resolution, for example, will provide only 1.6 % resolution of the full-scale signal. The effective bit resolution also takes into account noise inherent to the measuring instrument and probe. For example, a 10-bit analog-to-digital converter may have only 9 effective bits when the noise in the measuring instrument and probe is considered because any signal smaller than the noise is indistinguishable. Sampling at a higher frequency followed by digital filtering may improve the effective number of bits if the probe and digitizer noise follows a normal distribution; however, the filter should not filter out frequencies of interest (frequencies at or below 2 kHz in this case).

With these factors in mind, the general specifications for the test equipment are as follows:

- The combined full-scale accuracy of the measuring instrument and full-scale accuracy of the measuring probes shall be better than 3 %.
  - Errors in the full-scale measurement will be primarily due to gain and offset errors in the measured amplitudes and subsequent total signal distortion (TSD) results. This is separate from resolution, which must have higher precision.
  - Assuming all error in the measurement is due to gain and offset errors (i.e., negligible nonlinearity of the measuring instrument and probe), this results in a reported DFT amplitude percentage of ±0.1 %. For example, if a power ripple frequency exists at 3 % of the average power, the reported amplitude will be between 2.9 and 3.1 %.
- The combined resolution of the measuring instrument digitizer and sensor probes shall have at least 10 effective bits of resolution.
  - The resolution is needed to accurately capture the power dynamics and not just noise.
  - 10 effective bits of resolution provides approximately 0.1 % full-scale resolution.
- The skew between all simultaneous measurements (three current and three voltage measurements), at all required frequencies, shall be better than 15 µs (∼11° at 2 kHz).
  - When two sine waves separated by 11° are multiplied (i.e., voltage and current), the resulting sine wave (i.e., power) has an amplitude within 0.2 % of the two original sine wave amplitudes being multiplied together (no phase shift).
- Proper anti-aliasing filters shall be used to prevent energy at frequencies higher than the Nyquist frequency from being folded into the frequency spectrum of interest.

– This also drives the minimum sampling frequency of the measurement equipment, such that there is sufficient room between the maximum measured frequency component (2 kHz) and the Nyquist frequency for the anti-aliasing filter to reach the desired attenuation level. A minimum sampling frequency of 16 kHz is recommended.

Please note that these general requirements are only intended for testing the new frequency-domain power requirement. They are not sufficient for measuring single-line current harmonics, which requires specialized hardware that samples at higher frequencies and synchronizes to the fundamental line frequency.

## 2.2 Post-processing

The method in which the DFT of the captured data is taken significantly affects the resulting output. An overview of the DFT analysis is given, followed by an explanation of the windowing-function selection, and finishing with the determination of the properly sized DFT windows.

### 2.2.1 DFT Analysis Overview

Due to the nature of the DFT, any frequency content that does not fall exactly within a DFT frequency bin will result in spectral smearing. Sidelobes with an initial peak at −13 dB will be generated with the use of a rectangular DFT window (the DFT output points all lie on the corresponding sinc function). Because the specification allows frequency components with amplitudes up to 100 % of the load (such as at 0.1 Hz) while still detecting frequencies with amplitudes below 3 % (−30.5 dB), it is imperative that a proper DFT window is used in order to mitigate sidelobes that could skew the reported measurements. Furthermore, the DFT window will also have a strong effect on the accuracy of the total signal distortion (TSD) calculations because the sidelobes of a single frequency will sum with other frequencies and will subsequently count toward the TSD.

One of the goals of the specification is to ensure the generator(s) maintain control authority over their voltage and speed loops, which have a typical bandwidth in the vicinity of 1 Hz for non-steam generators (i.e., gas turbine and diesel). As such, a long time window that can capture low frequencies is desired for the DFT.

The other goal of the specification is to minimize higher frequency disturbances that could cause stress to the generator (e.g., excite a mechanical or subsynchronous resonance) or degrade the power-bus quality that could affect other loads. This will consist of frequencies that are above the controllable bandwidth of the generators(s); therefore, (relatively) high frequency knowledge is desired. If a long window is used in the DFT, however, short transients that could have detrimental effects on the generator(s) or the power-bus quality could get "washed out" because the short transient does not contain a significant portion of the energy of the entire signal.

Because of the potential of attenuating high frequency events, a long time window for the DFT cannot solely be relied upon. An approach to solve this problem was developed to minimize the extra work of testing and implementation while maintaining enough accuracy to meet the intended requirements. First, a long time window is used for a DFT of the measured power. This window is 100-seconds long to ensure accurate low-frequency calculations and will be used for calculating the frequency content below 1 Hz, as well as determining the average power level that will be used for normalizing all the results. Second, a shorter rolling time window is used to calculate the frequency content from 1 Hz and up. This window is 5-seconds long,[1] and each consecutive window has 80 % overlap (shifting the window 1 second at a time), ensuring transient events are not missed. Third and last, a Kaiser-Bessel window (with $\beta = 7$), whose equation and MATLAB code are shown in Table 1, is applied to the time window before taking the long and short DFTs. This window is particularly well suited for this application – it includes a relatively narrow main lobe and sufficiently attenuated sidelobes, and it provides excellent TSD accuracy because of its quick initial decay in sidelobe response (one of the driving functions in the window selection).

When taking the DFT of a signal, it is important to note that the output scales with the number of samples, $N$, input into the DFT. When a rectangular window with an amplitude of 1 is used on a signal

[1] Although this paper was originally developed with a 5-s long window with 80 % overlap, MIL-STD-1399-300-1 uses a 4-s long window with 75 % overlap to better capture transient events. The difference in overlap percentages is to maintain a 1-s shift between windows for both 5-s and 4-s window lengths.

(i.e., there is no modification of the signal before performing the DFT), the output of the DFT must be divided by $N$ to obtain the correct amplitudes at each frequency.

Windowing functions become necessary when a sampled signal is not periodic within the length of the window (which is usually the case), and discontinuities between the beginning point and the end point of the signal result in spectral smearing, where a single frequency of a signal is spread across multiple DFT frequency bins. When no shaping of the signal is applied before performing the DFT (i.e., a rectangular window is used), this spectral smearing is spread over sidelobes that are of a significant amplitude – approximately $-13$ dB from the main lobe.

Windowing functions are applied to the signal before performing the DFT to taper the start and end of the signal towards zero, either greatly reducing or completely eliminating any discontinuities between the start and end of the signal. This reduces the magnitude of the sidelobes, but this tapering also reduces the total energy of the signal. As a result, different window functions will result in different amplitudes of the DFT output. This change in the DFT gain is a result of what is known as coherent gain (CG), or sometimes referred to as processing gain. The CG of a window is defined as

$$CG = \sum_{n=0}^{N-1} w(n), \tag{1}$$

where $w(n)$ is the window function with samples 0 through $N-1$. Zero padding is not included in the window length, $N$, for this calculation. Note that a rectangular window results in $CG = N$. To obtain the correct amplitudes of a signal's frequency content from the output of a DFT, one must simply divide either the window or the DFT output by the CG. The literature typically normalizes CG by dividing by the number of samples, $N$, facilitating CG comparisons between the various window functions. When this normalization is performed, CG will be equal to 1 for a rectangular window and less than 1 for all other window functions.

Lastly, zero padding equal to nine (9) times the length of the signal is used for the DFT calculation. Zero padding adds more samples to the DFT output, which effectively eliminates amplitude errors, known as scalloping losses, that occur when frequencies of the actual signal do not fall directly in a DFT frequency bin (the signal frequency is spread across two or more DFT frequency bins). Although zero padding has a minimal effect on the TSD calculation (no signal energy is added or removed), scalloping losses cause the amplitude of signal frequencies that fall between DFT frequency bins to appear lower than they actually are, with the maximum loss occurring when a signal frequency falls directly between two adjacent DFT frequency bins. By increasing the frequency resolution (reducing the frequency span between bins) with sufficient zero padding, scalloping losses can be effectively eliminated and accurate frequency amplitudes are reported. Zero padding of a factor of 9 is chosen to increase the frequency resolution by a factor of 10 (if the signal is length $N$, the signal with zero padding is $N + 9N = 10N$).

If computation speeds are a concern, further zero padding may be added to achieve the next power of two so that the Fast Fourier Transform (FFT) algorithm may be used. If computation time is still a concern, zero padding may be reduced or eliminated so long as there are no frequencies close to the specified limit (where reducing scalloping losses is most important).

### 2.2.2 Window Selection

Figure 2 shows a comparison of three different DFT-window responses, with the code to generate them in MATLAB included in Table 1. These are good windows for this application based on the following criteria:

- Minimal main-lobe width (better frequency resolution).
- Sidelobe response better than 0.3 % ($-50.46$ dB).
- TSD calculation error better than 0.2 % of a test signal calculated over 1 Hz <= $f$ <= 2 kHz.
  - Test signal: 0.2-Hz tone with a peak amplitude (0 to peak) of 34 % of average, summed with a 0.5-Hz square wave with a peak amplitude (0 to peak) of 6.75 % of average, which is shown in Figure 3. The signal is sampled at a rate of 8 kHz. No anti-aliasing filtering is applied because it is known that frequency harmonics above the 4-kHz Nyquist frequency have an amplitude

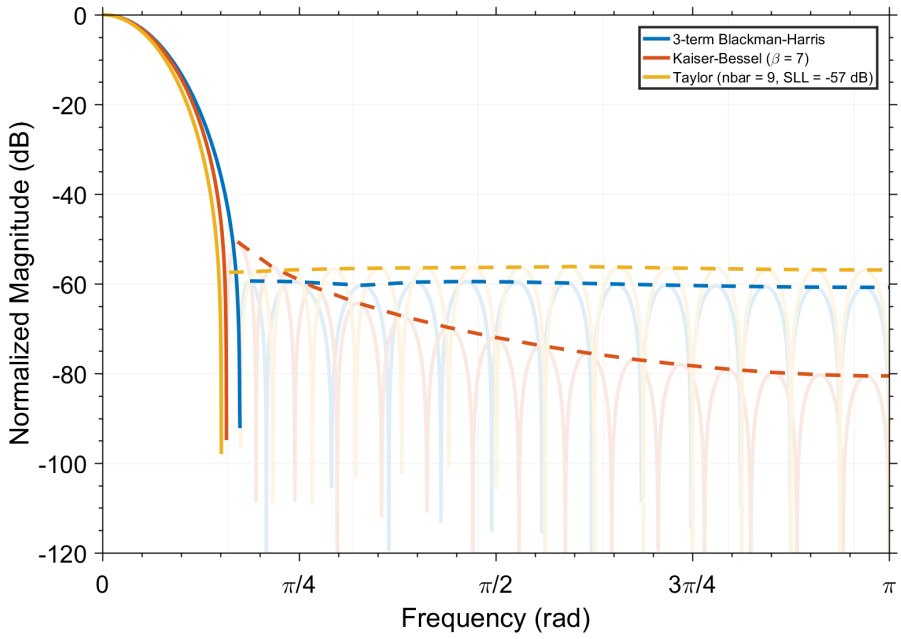


Figure 2: A comparison of different DFT windows. The dotted lines follow the peaks of the sidelobes.

less than −75 dB compared to the fundamental frequency. This is sufficiently low level as to cause minimal error.

– DFT Details: A 5-second window ($N = 40\,000$ when $T_s = 125\,\mu s$) is used for the short DFT with 80 % overlap. A 100-second window ($N = 800\,000$ when $T_s = 125\,\mu s$) is used for the long DFT. Zero padding equal to 9 times the length of the DFT window is used on both the long and short window DFTs. For instance, if the signal length is 5 seconds, an additional 45 seconds of zero padding is added. Using this zero padding length ensures that there are a sufficient number of frequency bins between the sidelobe nulls caused by the windowing. This guarantees the sidelobe energy is captured, preventing inadvertently sampling only at the sidelobe null points.

Although the test signal is not a practical real-life signal, it is designed as a stressing case for the DFT that is easily duplicated to verify the analysis is performed correctly. The exact TSD is due only to the harmonics of the square wave, which have a frequency and amplitude of

$$f_{harmonic} = (2k - 1) \cdot f_{fundamental}, \tag{2}$$

$$A_{harmonic} = \frac{4}{\pi} \cdot \frac{1}{2 \cdot k - 1}, \tag{3}$$

where $f_{harmonic}$ is the frequency of the harmonic, $f_{fundamental}$ is the fundamental frequency of the square wave, $k$ is an integer number from 1 to $\infty$, and $A_{harmonic}$ is the amplitude of each successive frequency component that makes up the square wave. As shown in the equation, a square wave generates only odd harmonics with decaying amplitudes as the harmonic number increases. Because the TSD is calculated from 1 Hz to 2 kHz, only harmonics $k = 2$ through 2000 are included for a 0.5-Hz square wave. Because the fundamental is below 1 Hz, it is excluded (hence $k$ starting at 2 instead of 1).

Table 2 shows the resulting TSD error of each window when compared to the known value using Equation (3) with $k = 2$ through 2000. Note that the driving design parameter of the Kaiser-Bessel window design is reducing the sidelobe response (can otherwise easily meet TSD error), whereas the driving design parameter of the Taylor window design is minimizing the TSD error (can otherwise easily meet minimum sidelobe attenuation). Although the Taylor window provides better frequency resolution as a result of its smaller main-lobe width, it is not a significant improvement over the Kaiser-Bessel window. Even though the initial sidelobes of the Kaiser-Bessel window are larger than those of the Taylor window, the Kaiser-Bessel sidelobes have a rapid initial decay, shown in Figure 2, providing

Table 1: Equations and MATLAB code of the examined DFT windows.

| **Window** | **Equation** | **MATLAB Code**<br>N = sample length of DFT window |
|---|---|---|
| 3-term Blackman-Harris | $w(n) = a_0 - a_1 \cdot cos\left(\frac{2\pi \cdot n}{N-1}\right) + a_2 \cdot cos\left(\frac{4\pi \cdot n}{N-1}\right)$ | `a = [0.44959,-0.49364,0.05677];`<br>`k = (0:2)';`<br>`n = 0:(N-1);`<br>`w = a*cos(2*pi*k*n/(N-1));` |
| Kaiser-Bessel | $w(n) = I_0\left(\beta \cdot \sqrt{1 - \left(\frac{2 \cdot n}{N-1} - 1\right)^2}\right)$<br>where $I_0(Z) = \sum_{k=0}^{\infty} \frac{\left(\frac{1}{4}Z^2\right)^k}{(k!)^2}$ | `beta = 7;`<br>`n = 0:(N-1);`<br>`Z = beta * ...`<br>`  sqrt(1-(2*n/(N-1)-1).^2);`<br>`kmax = 20;`<br>`w = zeros(1,N);`<br>`for i = 1:N`<br>` for k = 0:kmax`<br>`  w(i) = w(i)+(Z(i)^2/4)^k ...`<br>`  / factorial(k)^2;`<br>` end`<br>`end`<br><br>`OR`<br><br>`beta = 7;`<br>`n = 0:(N-1);`<br>`Z = beta * ...`<br>`  sqrt(1-(2*n/(N-1)-1).^2);`<br>`w = besseli(0, Z);`<br><br>`OR`<br><br>`beta = 7;`<br>`w = kaiser(N, beta);` |
| Taylor | | `nbar = 9;`<br>`SLL = -57;`<br>`w = taylorwin(N, nbar, SLL);` |

Table 2: DFT-Window Test Signal

| **Window** | **TSD Error (%)** |
|---|---|
| 3-term Blackman Harris | 0.082 |
| Kaiser-Bessel ($\beta = 7$) | 0.048 |
| Taylor ($\bar{n} = 9,\ \text{SLL} = -57\,\text{dB}$) | 0.186 |

better sidelobe performance further out from the main lobe. This is why the Kaiser-Bessel performs better than the Taylor window in TSD calculations. The output of the analysis using the Kaiser-Bessel window, along with the requirement limit, is shown in Figure 3.

Another nuance to the DFT stems from the fact that a purely real signal is being analyzed. In accordance with Euler's formula, a purely real cosine wave is represented as a summation of two complex waves: one rotating counterclockwise (due to the positive frequency), and one rotating clockwise (due to the negative frequency). When the DFT is performed on a single tone, two components appear in the spectrum, with one having a positive frequency, one having a negative frequency, and each component having a magnitude equal to half the amplitude of the original time domain signal. When the frequency of interest is close to the zeroth bin of the DFT, the sidelobes of the positive and negative frequency components are of a large enough magnitude that they will noticeably interact with one another and contribute some amount of error to the subsequent calculations. Because the Taylor window relies upon a number of equiripple sidelobes to achieve its small main-lobe width, the sidelobes of the positive and negative frequency components have a greater chance of interacting with one another and contributing more error.

Given that the Taylor window provides only a small increase in frequency resolution while having a

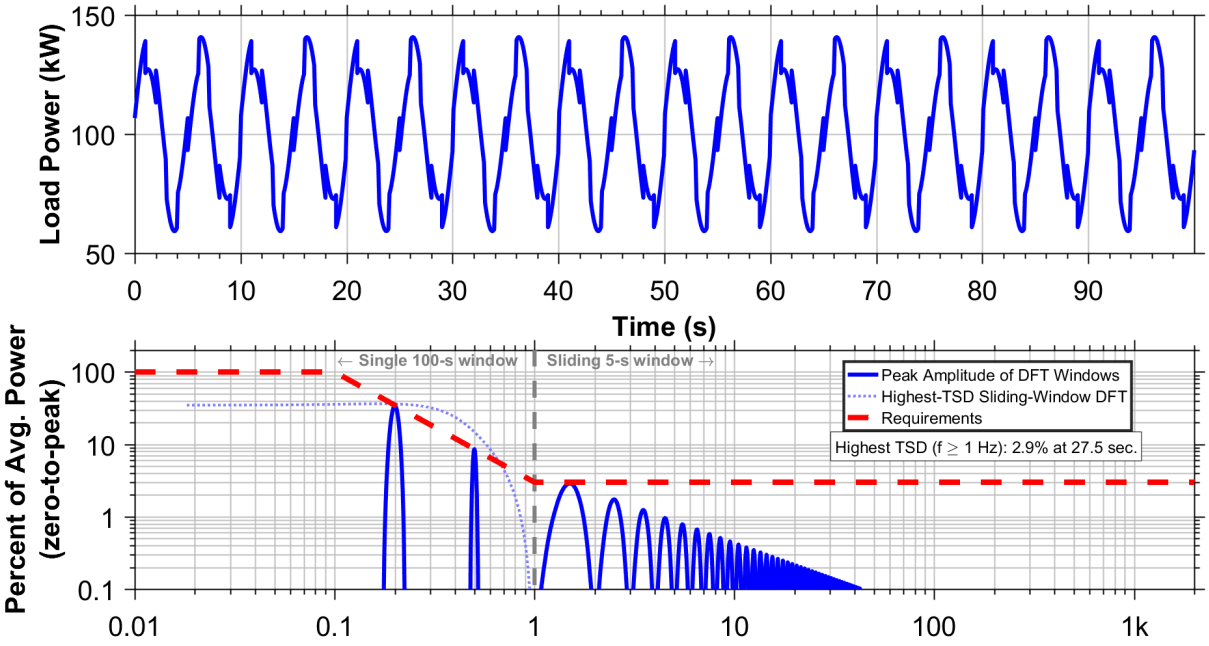


Figure 3: The DFT output of the test signal using the Kaiser-Bessel window.

less accurate TSD calculation, and the fact that a Kaiser-Bessel window is much simpler to generate, the Kaiser-Bessel window was selected as the DFT window of choice for this application.

### 2.2.3 Total Signal Distortion Calculation

The TSD calculation is performed using the results from the 5-second sliding DFT per the previously described method. The generic form of the TSD equation is

$$TSD\ (\text{percent}) = 100 \cdot \sqrt{\sum_{s} \left( \frac{P_s}{\sqrt{2}} \cdot \frac{1}{P_{avg}} \right)^2}, \tag{4}$$

where $P_s$ is the zero-to-peak power amplitude at individual signal frequency $s$, $P_{avg}$ is the average power, and $s$ ranges from 1 Hz to 2 kHz (inclusive). The $\sqrt{2}$ term is included to align the TSD requirement with the allowed total harmonic distortion of a single phase current. This equation also assumes perfect knowledge of the frequency content of the power signal. When performing a DFT on the power signal, which is an estimate of the frequency content, the noise characteristics of the DFT are affected by windowing and zero padding, which is primarily seen in the width of the main and side lobes of a particular signal frequency and the spacing between frequency bins. This effect is captured in a term called equivalent noise bandwidth (ENBW), which requires a modification to Equation (4).

The ENBW quantifies the effective spread of the main lobe of a given signal frequency due to the use of a DFT window. This widening of the main lobe spreads out the energy of an actual signal frequency across multiple DFT bins. As described earlier, the rectangular DFT window, being a finite length of time, results in discontinuities at the beginning and end of the sampled signal if the sampled signal length is not an exact integer multiple of the signal's period (note that most signals are not perfectly periodic, either). As a result, this discontinuity adds extra spectral content, sometimes referred to as spectral smearing, to the DFT output that shows up as a sinc function at each signal frequency, with each main lobe having the decaying sidelobes of the sinc function (all DFT bins for a single signal frequency fall on this function, though this may not be obvious if zero padding is not being used).

Non-rectangular windows taper the sampled signal toward zero at the ends to remove this discontinuity. The tapering, however, distorts the shape of the signal being analyzed. This distortion results in more spectral content within each main lobe, i.e., a wider main lobe. The benefit, however, is that the

sidelobes are decreased by increasing the width of the main lobe, with the exact amount specific to each windowing function.

When calculating the TSD, the ENBW is used to compensate for this increase in the main lobe bandwidth, and for any extra frequency bins sampled along the sinc-like function due to zero padding. If ENBW is not included, the resulting TSD value is dependent on which windowing technique is chosen, with the TSD value being inflated as the main lobe width or zero padding increases. For a rectangle window **with no zero padding**, the ENBW is equal to 1. For all other windows, the ENBW is greater than 1. From Harris 1978 [1], the ENBW is calculated by the sum of squares over the square of sums times the length of the window, written as

$$ENBW = N \cdot \frac{\sum_{n=0}^{N-1} \left[w(n)^2\right]}{\left[\sum_{n=0}^{N-1} w(n)\right]^2}, \tag{5}$$

where $w(n)$ is the window function with samples 0 through $N-1$. **It is important to note that any zero padding must be included in the window length, $N$**. For instance, a 64-sample rectangular window with no zero padding exhibits an ENBW of 1. If zero padding were added such that the resulting window length was 10 times longer, making it 640 samples, the ENBW would increase to 10 (in this example, the ratio of the power gains remained constant but $N$ increased by a factor of 10).

The denominator in Equation (5) is proportional to the maximum power gain of the desired signal (the actual signal that has a tone matching the center frequency of the main lobe, or the coherent signal), and the numerator is proportional to the combined maximum power gain for all the frequency components in the DFT filter (the frequency components from adjacent bins due to the main lobe spreading and the corresponding energy in the sidelobes combined with the signal at the main lobe frequency, or the incoherent and coherent signals).

To make the TSD calculation, every single frequency bin from 1 Hz to 2 kHz is included. If only the peak values of each lobe from the DFT were considered (local maxima), information could be lost where two frequencies are very close to one another and form a single wider lobe. Using every single frequency bin, such that no signal power is lost in the frequency spectrum of interest, is the correct method and will result in high-accuracy calculations that are dependent on ENBW, which can be easily demonstrated with any known signal for comparison. The equation for TSD is

$$TSD = \frac{\sqrt{P_k^2 + P_{k+1}^2 + P_{k+2}^2 + \cdots + P_n^2}}{\sqrt{2 \cdot ENBW} \cdot P_{avg}}, \tag{6}$$

where $P_k$ is the 1-Hz DFT bin, $P_n$ is the 2-kHz bin, $ENBW$ is the equivalent noise bandwidth of the DFT window [see Equation (5)], and $P_{avg}$ is the mean three-phase instantaneous power calculated over the larger 100-second window. The TSD calculation assumes that the coherent gain (CG) of the DFT window was already used to obtain the correct amplitudes from the DFT calculation, such that the peak of a tone's lobe in the DFT results is equal to the tone's actual amplitude (zero to peak).

To summarize, first the amplitude of the DFT output must be corrected by the CG term, which is specific to the selected window. Second, when calculating the TSD, which uses every single bin from the DFT output within some frequency range, the ENBW must be included in the calculation to negate the accumulated noise-capturing effect of the selected window (because of the main-lobe and sidelobe spreading).

# 3 Additional DFT Notes

1. MATLAB's DFT function decomposes a time domain signal into a set of cosine and sine waveforms. The DFT function returns an array of real and imaginary numbers at positive and negative frequencies. When the components of both frequencies are appropriately combined, the real numbers represent the amplitude of the cosine waveforms and the imaginary numbers represent the amplitudes of the sine waveforms (i.e., Euler's formula). The location of the real and imaginary numbers in the array gives the bin number, or frequency value, for each cosine and sine wave, and the location in the array will indicate if the frequency is negative or positive for each real and imaginary number.

    - A single cosine real tone is represented as two real numbers in the frequency domain: one with a positive frequency and one with a negative frequency. The amplitude of each of the frequencies is half the value of the original single-tone amplitude. This result can be shown via Euler's formula. For purely real signals, the time domain amplitude can be computed by multiplying the DFT amplitude of the positive frequency components by 2.
    - For actual measured signals where the decomposed signal consists of sine or cosine terms with various phase shifts, imaginary numbers will also exist in the amplitudes of the associated frequency bin. By taking the absolute value of the DFT output at each positive frequency and multiplying by 2, the amplitude of each respective frequency of the original waveform can be determined. Recall that the summation of a cosine wave with a sine wave for a given frequency can be represented as a cosine wave at the same frequency with the added phase shift, $\theta$; $A\cos(x)+B\sin(x) = M\cos(x+\theta)$, where $M = \sqrt{(A^2+B^2)}$ and $\theta = \arctan(B/A)$ [2]. As such, because Euler's formula represents cosine terms as real and sine terms as imaginary, taking the absolute value will provide the amplitude of the equivalent cosine wave.

2. If the sampling frequency is $f_s$, then the highest frequency component that can be determined is $f_s/2$ (the Nyquist frequency).

3. The width (in time) of the signal window determines the frequency resolution: $df = 1/T_{\text{window}}$. For instance, to be able to resolve 61-Hz and 60-Hz components from one another, then one needs at least 1 second's worth of samples, not including any zero padding samples. As such, the length (in time) of the sampled signal must be long enough to capture at least one full cycle of the frequency of interest.

4. Scalloping loss is a term used to describe the reduction in amplitude of the DFT output when the frequency of the time-domain signal does not match a specific DFT frequency bin. The maximum loss in amplitude occurs when the frequency of the signal lies exactly between two bins. Because the DFT is a sampled version of the continuous sinc-like function one would see for a single tone, zero padding can be used to add more samples along the sinc-like function. Zero padding does not add information (or power) to the DFT output, but can provided greater insight into the resulting out.

5. Zero padding can be used to effectively eliminate scalloping losses, where the peak value of a signal frequency that falls between two DFT frequency bins would not otherwise be captured. Zero padding increases the number of frequency bins, subsequently decreasing the space between frequency bins. Zero padding means you are doing the DFT for more frequencies, achieving more resolution, such that $df = f_s/N_{dft}$, where $N_{dft}$ is the total number of samples, including the zero padding length. Zero padding is helpful when there is more than a full cycle of data but the window is not an integer number of cycles, resulting in the signal frequency components falling between the frequency bins if zero padding is not used.

    - Note that zero padding does not add any information to the signal; it only helps in determining the exact frequency of a signal by increasing the frequency resolution of the DFT (decreasing the span between adjacent frequency bins), which consequently reduces, or nearly eliminates, scalloping loss and allows a local maxima to be determined. Zero padding simply adds more sampled data points to the DFT, or more data points along the sinc-like shape of the DFT output, and will not shrink the width of the main lobe.

6. The frequency of a bin is found by: (Bin Number) $\cdot f_s$/(total # of samples including zero padding).

7. If using a rectangle window and zero padding the time domain signal prior to taking the DFT (by simply adding zeros to the time-domain signal), $1/T_{\text{window}}$ is the space between nulls of the sidelobes of the resulting frequency domain sinc function, whereby $T_{\text{window}}$ does not include the zero padding samples, and $2/T_{\text{window}}$ is the width of the main lobe between nulls of the sinc function.

8. Windowing (such as using the Taylor or Kaiser windows) allows the resolution of low-level signals when strong signals are in the near vicinity because of the decreased sidelobes.

9. Windowing functions increase the main lobe width but decrease the sidelobe amplitude when compared with a rectangular window. By increasing the signal window length, the main lobe width can be decreased, hence increasing the frequency resolution.

10. The amplitudes of the DFT must be divided by a correction factor, the Coherent Gain (CG), specific to the window. The CG compensates for the change in amplitude of the DFT output due to applying the window function as well as the window length, $N$. For the simplest case of using a rectangle window, CG equals $N$, where $N$ does not include zero-padding.

11. When the TSD is calculated, a correction term, called Equivalent Noise Bandwidth (ENBW), is used to negate the inflation of the TSD value due to the energy that appears in the sidelobes as well as the additional energy due to the main lobe bandwidth increase.

12. The MATLAB functions dft(...) and fft(...) output frequency bins 0 to $N-1$. The bins between 0 and $N/2$ are positive frequencies, and the bins from $N/2$ to $N-1$ are negative frequencies. Sample N/2 corresponds to frequency $f_s/2$, the Nyquist frequency. This assumes an even number of samples were fed into the dft or fft function.

    - In MATLAB, the order of dft and fft frequency bins is defined as $[0, 1, 2, \cdots, (N/2)-1, -(N/2), (-N/2)+1, \cdots, -2, -1]$. For example, performing a DFT on a time-domain signal with eight samples, or $N = 8$, will produce the frequency bin order $[0, 1, 2, 3, -4, -3, -2, -1]$.

Recommended steps to performing the DFT and computing the TSD:

Step 1 Compute the average value of the 100-second sampled signal and subtract this average value from the signal prior to performing the DFT.

Step 2 Sample multiply the 100-second signal by the windowing function and divide by the CG of the window, which corrects for the combined gain of the windowing function and DFT process.

- The windowing function should be a Kaiser-Bessel with $\beta = 7$.

Step 3 Add zero padding, equal to at least 9 times the length of the signal.

Step 4 Perform the long DFT on the windowed and zero-padded signal, and calculate the absolute value of each frequency component.

Step 5 For real signals, because the negative frequencies have the same information as the positive frequencies, multiply all the amplitudes (the absolute values of the complex numbers stored in each bin) of the positive frequency components (except 0 Hz and the Nyquist frequency, bins 0 and $N/2$) by 2 to achieve the proper amplitude.

Step 6 All negative frequency components may be discarded. The bin frequency spacing is $f_s/N$. The bin numbers are now 0 to $N/2$. With only the positive frequencies present (plus 0 Hz, bin 0), convert the bin numbers to frequencies by multiplying the bin numbers by the sampling frequency, $f_s$, and dividing by the total number of samples, $N$.

Step 7 Save the spectrum from 0.01 Hz to 1 Hz. The higher frequencies will be determined from the 5-second rolling window.

Step 8 Perform the multiple (rolling) short DFTs on the 100-second time-domain signal by using a rolling 5-second window, whereby each subsequent 5-second window will have an 80 % overlap with the preceding window (shift by 1 second). Perform the same DFT process as done for the 100-second DFT in Step 1 to Step 6 for each short DFT window. Record the results of each 5-second DFT calculation.

Step 9 Record the maximum amplitude at each frequency bin across all 5-second DFTs from the preceding step. These maximums will represent the DFT results for frequencies from 1 Hz to 2 kHz. This is similar to the "max hold" function on a spectrum analyzer.

Step 10 Perform a TSD calculation from 1 Hz to 2 kHz on each separate 5-second rolling window DFT from Step 8. The maximum TSD of all 5-second rolling-window DFTs is to be the reported TSD value to be compared against the 5 % requirement.

# 4 Algorithm

Algorithms 1 and 2 are example pseudo-code algorithms to assist in the understanding and development of the power-ripple analysis.

**Algorithm 1** Power-ripple analysis

**Precondition:** $P$ is the measured and summed instantaneous three-phase power of the load with constant sample rate recorded for 100 seconds. $T_s$ is the sample rate.

**function** POWERANALYSIS($P,\ T_s$)
  $windowLength \leftarrow 5$
  $overlapPercent \leftarrow 80$
  $time \leftarrow [0,\ T_s,\ 2 \cdot T_s,\ \ldots\ ,\ (\text{length}(P) - 1) \cdot T_s]$
  $[fLong,\ DFTLong,\ AvgLong,\ ENBWLong] \leftarrow \text{powerDFT}(P,\ T_s,\ \text{length}(P) \cdot T_s,\ 0)$
    ▷*Run the DFT analysis for the single long window.*
  $[f,\ DFT,\ Avg,\ ENBW] \leftarrow \text{powerDFT}(P,\ T_s,\ windowLength,\ overlapPercent)$
    ▷*Run the DFT analysis with the sliding window.*

  $index1HzLong \leftarrow \text{round}(\ 1.0 \cdot T_s \cdot (\text{length}(fLong) - 1) \cdot 2 + 1)$ ▷*The index of the long-DFT frequency array that is closest to 1 Hz.*
  $index1Hz \leftarrow \text{round}(\ 1.0 \cdot T_s \cdot (\text{length}(f) - 1) \cdot 2 + 1)$ ▷*The index of the sliding-DFT frequency array that is closest to 1 Hz.*
  $index2kHz \leftarrow \text{round}(\ 2000 \cdot T_s \cdot (\text{length}(f) - 1) \cdot 2 + 1)$ ▷*The index of the sliding-DFT frequency array that is closest to 2 kHz.*

  **for** $i \leftarrow 1$ to {Number of sliding DFTs} **do**
    ▷*Calculate the Total Signal Distortion as defined earlier*
    $TSD_i \leftarrow \sqrt{\sum DFT_i(index1Hz \text{ to } index2kHz)^2}/(\sqrt{2 \cdot ENBW} \cdot AvgLong) \cdot 100$
  **end for**

  **for** $i \leftarrow 1$ to length($f$) **do**
    $maxDFT \leftarrow \max(DFT_{1 \text{ to } end}(i))$ ▷*Find the maximum value of each frequency across all sliding DFT results.*
  **end for**

  $highestTSD \leftarrow \max(TSD)$ ▷*Find the highest TSD.*
  $finalDFT \leftarrow [\ DFTLong(1 \text{ to } index1HzLong - 1), maxDFT(index1Hz \text{ to } end)\ ]/AvgLong \cdot 100$
    ▷*This outputs the final DFT result, using a 100-s window below 1 Hz, and a 5-s sliding window 1 Hz and above.*
**end function**

**Algorithm 2** Power-ripple analysis DFT function

**function** POWERDFT($P$, $T_s$, $windowLength$, $overlapPercent$)
  $NDFT \leftarrow round(windowLength/T_s)$ ▷*Converter time to number of data points*
  $zeroPadMultiplier \leftarrow 9 + 1$ ▷*How much zero padding to add, multiples of the length of data*
  $FFTslide \leftarrow$ round( $(1 - overlapPercent/100) \cdot NDFT$ ) ▷*# of samples the DFTs overlap*
  $N \leftarrow$ floor( $(\text{length}(P) - NDFT)/FFTslide$ ) $+ 1$ ▷*Number of DFTs to run*

  $win \leftarrow \text{kaiser}(NDFT, 7)$ ▷*Create a Kaiser window of length $NDFT$ with $\beta = 7$*
  $CG \leftarrow \text{sum}(win)$ ▷*Coherent Gain*
  $ENBW \leftarrow \text{sum}(win^2)/\text{sum}(w)^2 \cdot NDFT \cdot zeroPadMultiplier$ ▷*Equivalent Noise Bandwidth*

  **for** $i \leftarrow 1$ to $N$ **do**
    $istart \leftarrow 1 + (i - 1) \cdot FFTslide$ ▷*Where to start the DFT within P*
    $iend \leftarrow start + NDFT - 1$ ▷*Where to end the DFT within P*

    $X \leftarrow P(istart \text{ to } iend)$
    $DFTavg_i \leftarrow \text{mean}(X)$
    $X \leftarrow X - DFT_{avg}$ ▷*Subtract the average value in order to obtain better DFT results.*

    $Y \leftarrow \text{dft}(X \cdot win/CG,\ NDFT \cdot zeroPadMultiplier)$ ▷*Take the DFT with a window length of $NDFT \cdot zeroPadMultiplier$*
    $Y \leftarrow Y \cdot 2$ ▷*We only want positive frequencies, but the DFT creates both positive and negative frequencies with half amplitude of a purely real input. This corrects for the half amplitude.*
    $Y(1) \leftarrow Y(1)/2$ ▷*The zeroith bin of the DFT does not need to be doubled, so this recorrects it.*

    $DFT_i \leftarrow Y(1 \text{ to } NDFT \cdot zeropadMultiplier/2)$ ▷*Take only the zeroith bin and positive. frequencies of the fft output. This assumes the first half of the array is positive frequencies*
  **end for**

  ▷*Generate the frequency vector for the DFTs.*
  $frequency \leftarrow [0,\ 1,\ \ldots,\ NDFT \cdot zeroPadMultiplier/2]/(T_s \cdot NDFT \cdot zeroPadMultiplier)$

  **return** $frequency,\ DFT,\ DFTavg,\ ENBW$
**end function**

# A MATLAB Example Code

This section includes MATLAB code that can be used to run the power-specification frequency-domain analysis. To use the code, simply copy and paste it into a new MATLAB script and save the script with the appropriate file name listed in the subsection header (i.e., powerRippleAnalysis.m).

The algorithm function, `powerRippleAnalysis`, contains the overall structure of the analysis, while the `powerDFT` function performs the actual DFT analysis (including the sliding of the small DFT window). The code in `example.m` generates the test signal used in this paper and demonstrates how to use the included code. An example output is shown below in Figure 4.

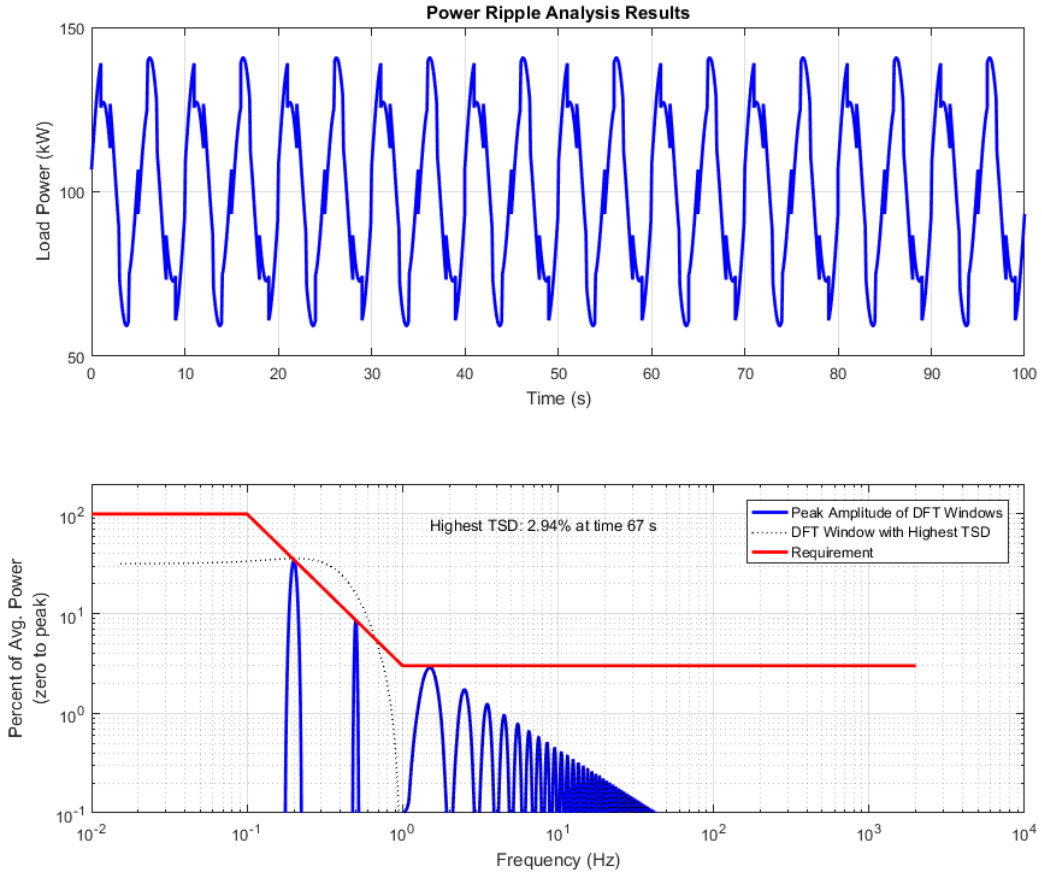


Figure 4: The DFT output of the test signal using the provided code.

## A.1 Algorithm: powerRippleAnalysis.m

```
function [ f_final, DFT_final, TSD_max ] = powerRippleAnalysis( P, Ts )
%powerRippleAnalysis - Calculates and plots the DFT analysis for the
%                      MIL-STD-1399-300-1 power ripple requirements.
%-
%   Usage:
%   [ f_final, DFT_final, TSD_max ] = powerRippleAnalysis( P, Ts )
%-
%   Input:
%   P is the input power waveform and Ts is the sampling rate. It is
%   recommended to downsample the power waveform to under 20 kHz in order
%   to keep the memory usage to a reasonable amount (~1 GB at 20 kHz for
%   100 seconds).
%-
%   Output:
%   f_final is the pieced-together frequency bins from the 100-s DFT and
%   the 5-s DFT windows. DFT_final is the pieced-together DFT outputs from
%   the 100-s DFT (<1 Hz) and the maximum for each frequency across all 5-s
%   DFTs (>=1Hz). TSD_max is the maximum TSD of any single 5-s DFT.

    % Define the window length in seconds
    windowLength = 5;
    % Define the overlap percentage
    overlapPercent = 80;
    % Create a time vector
    time = ( 0:(length(P)-1) ) * Ts;

    % Run the DFT analysis with the long window
    [f_long,DFT_long,Pavg_long,ENBW_long] = powerDFT(P,Ts,time(end)+Ts,0);
    % Run the DFT analysis with the rolling short window
    [f, DFT, Pavg, ENBW] = powerDFT(P, Ts, windowLength, overlapPercent);

    % Calculate the 1-Hz bin indexes
    index1Hz_long =  ceil( 1.0 * Ts * (length(f_long)-1)*2 + 1 );
    index1Hz      = round( 1.0 * Ts * (length(f)-1)*2 + 1 );
    % Find the bin number for 2 Hz in the rolling DFT outputs
    index2kHz     = round( 2e3 * Ts * (length(f)-1)*2 + 1 );
    % If 4-kHz or lower sampling rate is used, use the last non-Nyquist bin
    if (index2kHz > size(DFT,2))
        index2kHz = size(DFT,2);
    end

    % Initialize array
    TSD = zeros(1,size(DFT,1));
    for i = 1:size(DFT,1)
        % Calculate the total power distortion (TSD) using the short window
        % for 1Hz <= f <= 2kHz
        TSD(i) = sqrt( sum( DFT(i,index1Hz:index2kHz).^2 ) ) ...
                / ( sqrt(ENBW) * sqrt(2) * Pavg_long ) * 100;
    end

    % Find the maximum amplitude of each frequency across all DFT windows
    DFT_peaks = max(DFT,[],1);

    % Find the DFT window with the highest TSD
    [TSD_max, TSD_max_index] = max(TSD);
    % Find what time (in seconds) the highest TSD occured.
    TSD_max_time = TSD_max_index * windowLength * ...
                   (1 - overlapPercent/100) + windowLength/2;

    % Generate the final results by concatenating the long window DFT with
    % frequencies below 1 Hz and the rolling window peaks (maximum of each
    % frequency over all windows) with frequencies 1 Hz and above.
    DFT_final = [DFT_long(1:index1Hz_long-1), DFT_peaks(index1Hz:end)] ...
                / Pavg_long * 100;
    f_final = [f_long(1:index1Hz_long-1), f(index1Hz:end)];
```

```
    % Generate the requirement to compare against
    req_f   = [0.01, 0.1,                logspace(-1,0,100), 1, 2e3];
    req_amp = [ 100, 100, 3*logspace(-1,0,100).^(log10(3)-2), 3,   3];

    %% Plot the results
    figure(1)
    set(gcf,'Position',[100 100 1000 800])

    subplot(211)
    plot((0:length(P)-1)*Ts, P/1e3, '-b', 'LineWidth',2)
    grid on
    xlabel('Time (s)')
    ylabel('Load Power (kW)')
    title('Power Ripple Analysis Results')

    subplot(212)
    loglog(f_final, DFT_final, '-b','LineWidth',2); hold on
    loglog(f, DFT(TSD_max_index,:)/Pavg_long*100, ':k','LineWidth',1)
    loglog(req_f, req_amp, '-r','LineWidth',2); hold off
    axis([0.01, 10e3, 0.1, 200])
    grid on
    xlabel('Frequency (Hz)')
    ylabel({'Percent of Avg. Power','(zero to peak)'})
    legend('Peak Amplitude of DFT Windows','DFT Window with Highest TSD','Requirement')
    text(1.5,80,['Highest TSD: ' num2str(TSD_max,'%5.2f') '% at time ' ...
         num2str(TSD_max_time,'%3.0f') ' s'])
end
```

## A.2 DFT Function: powerDFT.m

```
function [ frequency, DFT, Pavg, ENBW ] = powerDFT( P, Ts, windowLength, overlapPercent )
%powerDFT - outputs the results of the DFT process used for the
%MIL-STD-1399-300-1 power ripple requirement. This is a support function
%that is called from powerRippleAnalysis(). Please use powerRippleAnalysis
%for evaluating against the power ripple requirement.

    % Convert window length to the number of samples
    N_win = round( windowLength / Ts );
    % How much zero padding to add in multiples of the length of data, plus
    % one for the original data length
    zeroPadMultiplier = 9 + 1;
    % Define the length of the DFT that will be run. Make sure it's even.
    NDFT = round(N_win*zeroPadMultiplier);
    if ( mod(NDFT,2) )
        NDFT = NDFT + 1;
    end
    % If faster speed is desired, increase NDFT to the next power of 2 so
    % that the Fast Fourier Transform algorithm can be used
    NDFT = 2^nextpow2(NDFT);
    % Define the number of samples the DFTs slide between windows
    FFTslide = round( (1 - overlapPercent/100) * N_win );
    % Define the number of DFTs to run
    N = floor( (length(P)-N_win)/FFTslide ) + 1;

    % Create the Kaiser window of length N_win with Beta = 7
    beta = 7;
    % same as kaiser(NDFT,beta)
    n = 0:(N_win-1);
    win = besseli( 0, beta*sqrt(1-(2*n/(N_win-1)-1).^2) )';
    % Calculate the Coherent Gain of the window
    CG = sum(win);
    % Calculate the Equivalent Noise Bandwidth of the window
    ENBW = sum(win.^2)/sum(win)^2 * NDFT;

    % Initialize arrays
    DFT  = zeros(N,NDFT/2);
    Pavg = zeros(1,N);

    % Calculate the DFTs
    for i = 1:N
        % Where to start and end the DFT within P
        iStart = 1 + (i-1)*FFTslide;
        iEnd = iStart + N_win - 1;

        % Grab the part of the waveform we want
        X = P(iStart:iEnd);
        % Find the average value
        Pavg(i) = mean(X);
        % Subtract the average before running the DFT
        X = X - Pavg(i);
        % Perform the DFT with a window length of NDFT
        Y = fft( X(:) .* win/CG, NDFT);

        % We only want positive frequencies, but the DFT creates both
        % positive and negative frequencies with half amplitude of a purely
        % real input. This corrects for the half amplitude. The zeroth bin
        % for the DFT does not need to be doubled. We also do not want the
        % complex values, but the absolute values. The Nyquist bin is
        % thrown out.
        DFT(i,:) = [abs(Y(1)) 2*abs(Y(2:(NDFT/2))')];
    end

    % Generate the frequency vector for the DFTs.
    % f = (bin number) * (sampling frequency) / (DFT length with zero padding)
    frequency = ( 0 : NDFT/2-1 ) * (1/Ts) / NDFT;
end
```

## A.3 Test Code: example.m

```
%% Example Power Ripple
% Use the following as a test signal to ensure the functions have been
% copied or implemented correctly. The main lobes at 0.2 Hz, 0.5 Hz, and
% 1.5 Hz should all be very close to the requirement. The remaining lobes
% above 1.5 Hz should be placed in 1 Hz increments above 1.5 Hz (e.g. 2.5
% Hz, 3.5 Hz, etc.) and are gradually decreasing. The highest TSD should be
% 2.938%.

Ts = 1/8e3;
t  = 0:Ts:(100-Ts);
P  = 100e3 + 34e3*sin(0.2*2*pi*t) + 6.75e3*square(0.5*2*pi*t);

[ f_final, DFT_final, TSD_max ] = powerRippleAnalysis(P, Ts);
```

# References

[1] F. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, Jan 1978.

[2] S. W. Smith, *The Scientist & Engineer's Guide to Digital Signal Processing.* California Technical Pub, 1997. [Online]. Available: http://www.dspguide.com/pdfbook.htm

# Revision History

**Revision 0**, 2 January 2018:

- Initial release