Damage Based Analysis (DBA): Theory, Derivation and Practical Application – using both an Acceleration and Pseudo-Velocity approach

Spacecraft – Launch Vehicle Workshop - 2016
Aerospace Corp, El Segundo, CA

Vince Grillo
Dynamic Environments, KSC LSP Structural Dynamics

Scope of the problem

- Launch and Space Vehicles are subjected to complex, vibroacoustic non-stationary flight environments which vary in time during certain events in a mission profile.
- Development, Qualification and Acceptance test specifications developed from historical data and analytical/empirical techniques don't always take into account the non-stationary aspect of some portion of the flight data.
- Flight data is assessed after each mission by comparing the flight results to prior qualification testing using a Maximax approach which can be potentially too conservative.
- The need exists for a consistent, quantitative and accurate way to characterize and compare non-stationary flight data with stationary test environments.

Historical Approach

- Maximax Approach – Vibroacoustic Data
  - Typically, vibroacoustic flight data processed from instruments such as accelerometers are treated as a series of overlapping stationary time segments.
  - Maximax is a technique used to capture the Power Spectral Density (PSD) during each time segment. The goal is to obtain the maximum value of the spectrum at each frequency independently and loop through the total number of time segments.
  - The Maximax results are then compared to the test specification.
- Limitations to Maximax Method
  - Peak responses in flight data can be considerably overestimated.
  - Maximax tends to result is excessive conservatism in the Qualification margin which may lead to more expensive designs and less reliability.
  - For a relatively immature design, component qualification may result in over-tests with excessive environments being applied which will impact cost and schedule.
  - Since Maximax is a purely mathematical transformation of the signal, results are dependent on the choice of processing parameters such as window size and overlap.
  - The physical properties of the system such as damping and fatigue from stress cycles are not considered.

Damage Potential

Absolute Acceleration and Pseudo-Velocity Approaches

Review of S.J.DiMaggio, B.H.Sako, S.Rubin , Absolute Acceleration and Scot I. McNeill Pseudo Velocity approaches.

Absolute Acceleration Method, Damage Indicators, G1 – G12

Using Equations (4 & 5) in Appendix A, we can solve for the power spectral density (PSD) by plugging in $\boldsymbol{\sigma^2}$ from eq(5) into eq(4) :

$$G = G_1 = \left(\frac{A_{max}^2}{Q\,\pi\, f_n \ln(f_n\, T_0)} \right) \qquad \text{eq(6)}$$

Eq(6) represents the power spectral density $\boldsymbol{G_1}$, which is a function of the max absolute acceleration response at each natural frequency represented by a SDOF system. eq(6) is solved by computing an SRS at each frequency for a series of SDOF systems, recovering the Accel-Time responses and finding the maximum amplitude at each frequency.

- $\boldsymbol{G_1}$ represents a $\boldsymbol{T_0}$=60sec equivalent PSD input that ensures the test response or "SDOF base-drive response" of a component, envelopes the maximum amplitude response of a component in flight.
- Note that G1 or amplitude damage indicator is related to maximum amplitude response of a component in flight and does not take into account fatigue.
- Fatigue will be represented by damage indicators such as G4, G8 & G12 which are functions of the slope b for stress cycles on the S-N curve and are derived in Appendix A. Each of these indicators have a fairly complex mathematical form and will need to be derived each time the slope b changes, see Appendix A.
- In order to illustrate the meaning of A_{max}^2/Q and G_1, figures (1 & 2) are shown in Appendix A.

Pseudo-Velocity Method

- From Appendix B, It can be shown that the maximum stress σ_s is proportional to pseudo-velocity, σ_s = k σ_{pv} where σ_{pv} is the oscillator pseudo-velocity RMS.
- From eq(21) appendix A, we have:

$$G_1 = \frac{PV_{max}^2 4\pi f_n}{Q \ln[f_n T_0]} \qquad \text{eq(21)}$$

- The resulting eq(21) represents the power spectral density (PSD), which is a function of the max pseudo-velocity amplitude response, PV_{max} at each natural frequency represented by a SDOF system. This result is similar to eq(6).
- From Appendix B eqs(23-25), the fatigue damage indicators G4 – G12 are:

$$G_4 = \frac{4\pi f_n}{Q}\sqrt{\frac{D_F}{2 f_n T_0}} \qquad G_8 = \frac{4\pi f_n}{Q}\sqrt[\frac{1}{4}]{\frac{D_F}{24 f_n T_0}} \qquad G_{12} = \frac{4\pi f_n}{Q}\sqrt[\frac{1}{6}]{\frac{D_F}{720 f_n T_0}}$$

  - Where D_F is the number of fatigue damage cycles from flight data, counted using rainflow methods in a series of Amplitude Bins similar to the method used in the acceleration approach.

- Note that eqs(23-25) represent a much simpler form of the fatigue damage indicators compared to the acceleration method. Also, alternate fatigue damage indicators such as G6 or G10 could be easily derived for various values of slope b on the SN curve.

Practical Application - Python Implementation

Fatigue Damage Equivalent – fdepsd python code

pyyeti.fdepsd.**fdepsd**(*sig, sr, freq, Q, resp='absacce', hpfilter=5.0, nbins=300, T0=60.0, rolloff='lanczos', ppc=12, parallel='auto', maxcpu=14, verbose=False*)

Compute a fatigue damage equivalent PSD from a signal. See Appendix – C for more information

Parameters:

- **sig** (*1d array_like*) – Base acceleration signal.
- **sr** (*scalar*) – Sample rate.
- **freq** (*array_like*) – Frequency vector in Hz. This defines the single DOF (SDOF) systems to use.
- **Q** (*scalar > 0.5*) – Dynamic amplification factor $Q = 1/(2\zeta)$ where ζ is the fraction of critical damping.
- **resp** (*string; optional*) – The type of response to base the damage calculations on:

resp	Damage is based on
'absacce'	absolute acceleration [1]
'pvelo'	pseudo velocity [2]

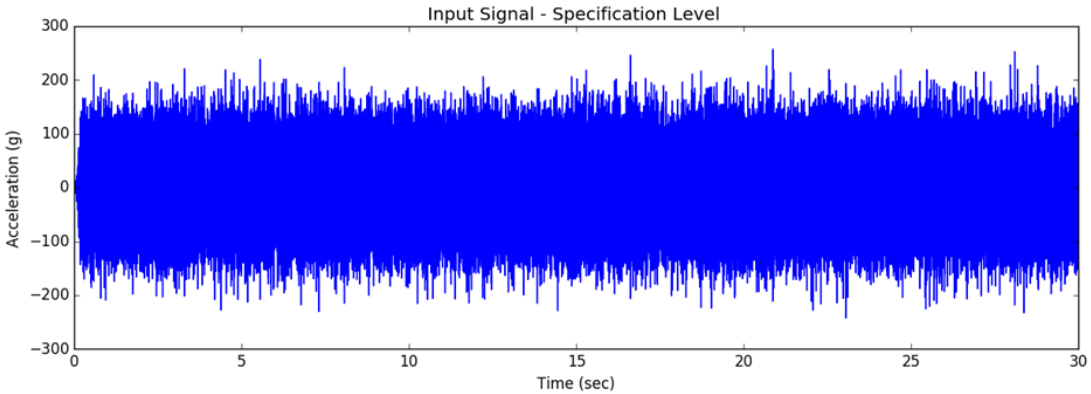
- **nbins** (*integer; optional*) – The number of amplitude levels at which to count cycles
- **T0** (*scalar; optional*) – Specifies test duration in seconds

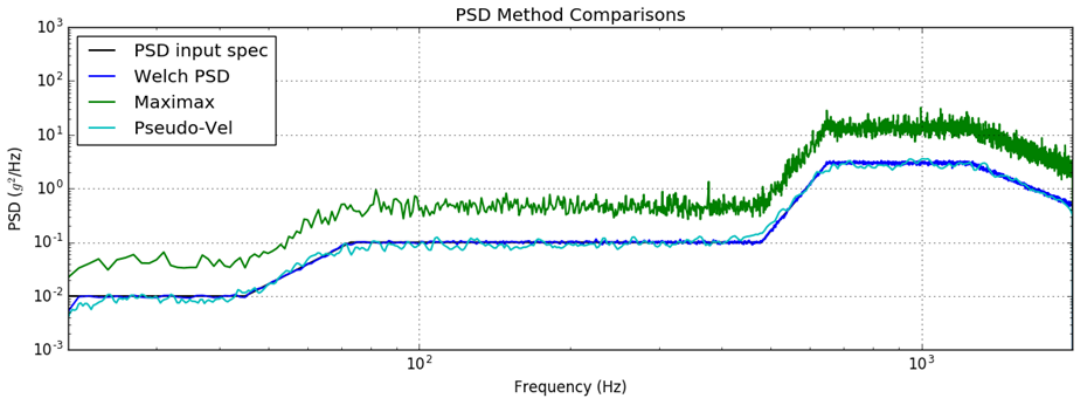
rolloff (*string or function or None; optional*) – Indicate which method to use to account for the SRS roll off when the minimum *ppc* value is not met 'linear'

- **parallel** (*string; optional*) – Controls the parallelization of the calculations:
- **maxcpu** (*integer or None; optional*) – Specifies maximum number of CPUs to use. If None, it is internally set to 4/5 of available CPUs (as determined from `multiprocessing.cpu_count()`.
- **psd** (2d ndarray; `len(freq) x 5`) – The five columns are: [G1, G2, G4, G8, G12]:

PSD Method Comparisons

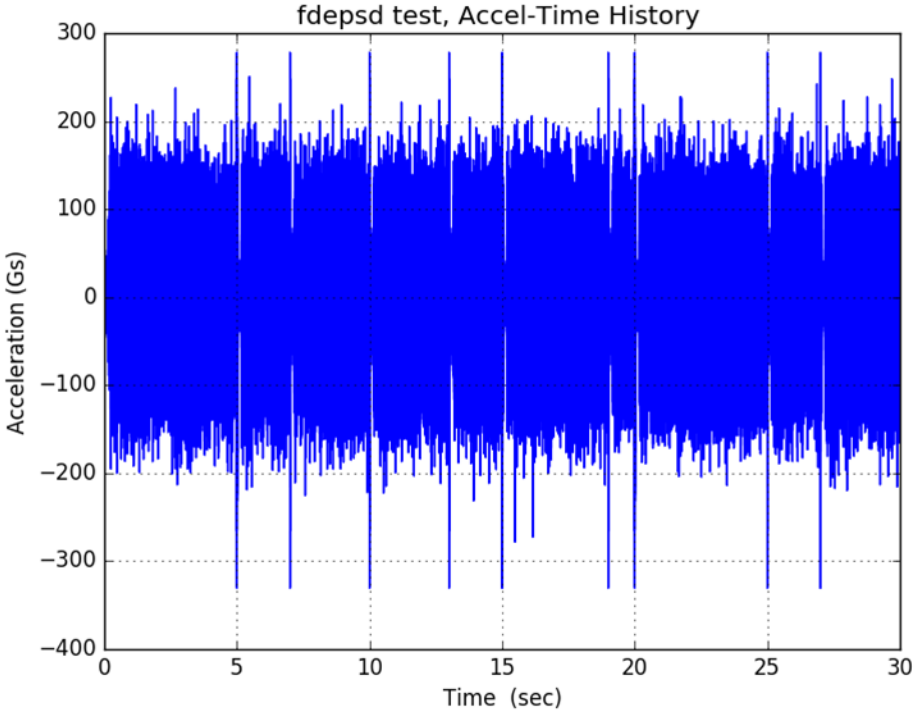
- Input Signal – based on arbitrary PSD input spec, converted to Acceleration-Time history.
- 30-sec signal, sample rate=40kHz, 1.2M samples

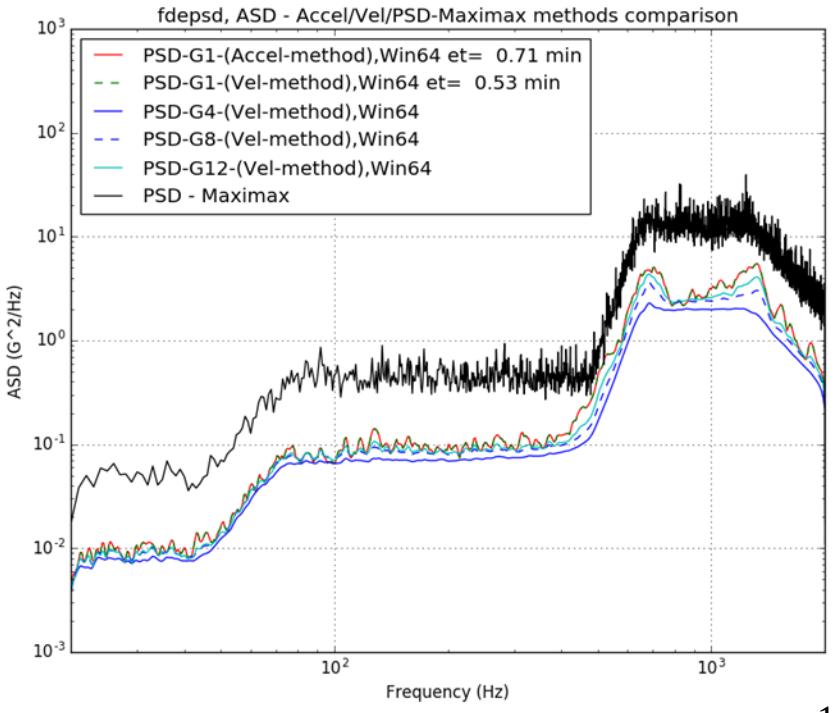




Fatigue Damage Equivalent – mixed signal comparison

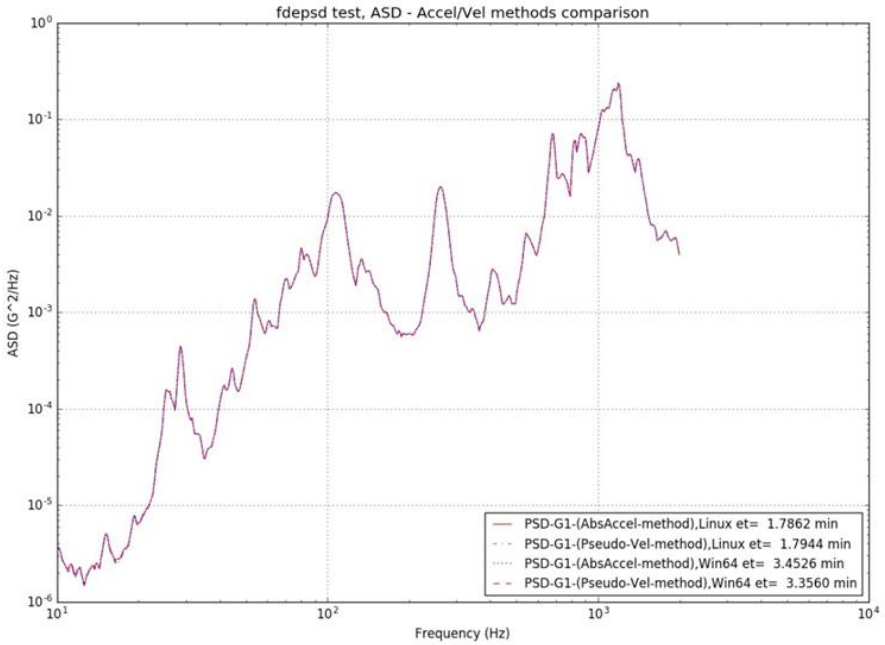
- Fatigue Damage Equivalent (fdepsd) G1-G12 – Acceleration, Velocity methods compared to Maximax.
- Synthetic stationary Accel-Time signal injected with multiple non-stationary transient events approx. 200ms in length. fdepsd provides a better method to evaluate both combined steady-state and transient events in a composite signal.





Performance considerations

- For previous Fatigue Damage Equivalent processing which used Matlab like software, the average time to process the results for a signal > 300sec illustrated below was over 45 minutes compared to 1.8 minutes for a Linux system. Note: Absolute Acceleration and Pseudo-Velocity methods for ASD computation correlate very closely.
- Total CPU time spent – processing fdepsd using Absolute Accel/Pseudo-Velocity methods on both Linux_64 and Windows-x64. Both systems set to 8-cpus, parallel processing.
- CPU time spent on Linux 64bit is very efficient and almost half that of Windows_x64. Most parallel CPU cycles on Linux consumed by user process with very little system/idle time.



Conclusions

- In summary, Damage Potential or Fatigue Damage Equivalent provides a powerful method for characterizing both the Stationary and Non-Stationary nature of a random vibration signal. Both the Amplitude and Fatigue characteristics of a signal are quantified based on characteristics of the system such as damping and stress cycles. Details for the damage indicators are summarized below:
  - G_1: A 60 sec equivalent PSD input that ensures the test response of a component envelopes the maximum amplitude response of a component in flight
  - G_4, G_8 , G_{12} : A 60 sec equivalent PSD input that ensures the fatigue damage of a component in test envelopes the corresponding fatigue damage in flight for a fatigue exponent of b=4,8,12.
- Pseudo-Velocity approach provides a more efficient method to apply fatigue damage exponents.
- The current Python implementation provides efficient Open Source software which is platform independent and can compute the results in a reasonable amount of time.
- Damage Potential Analysis provides a way to characterize the differences in energy imparted to a system during both Test and Flight and ensure that ground testing will envelope both the Amplitude and Fatigue qualification requirements of a component in Flight.

References

1. "Analysis of Nonstationary Vibroacoustic Flight Data Using a Damage-Potential Basis", S.J.DiMaggio, B.H.Sako, S.Rubin, Journal of Spacecraft an Rockets, Vol. 40, No. 5, September - October 2003.
2. "Implementing the Fatigue Damage Spectrum and Fatigue Damage Equivalent Vibration Testing", Scot I. McNeill, Ph.D., Shock & Vibration Symposium, 10/26-10/30-2008, Orlando Fl.
3. International Standard, ISO 18431-4:2007(E), Mechanical Vibration and Shock – Signal Processing – Part-4: Shock-response spectrum analysis, 1st edition 2007-02-01.
4. "Standard practices for Cycle Counting in Fatigue Analysis", ASTM International E 1049-85 (Reapproved 2005).
5. "Random Vibration Data, Analysis and Measurement Procedures", Julius S. Bendat and Allan G. Piersol, 4th edition, John Wiley and Sons, Inc.

Appendix A – Absolute Acceleration Method

- Fatigue Damage Equivalent – FDE – Acceleration approach

  - The assumption can be made that the distribution of the peak values in a narrowband random vibration response follows a Rayleigh distribution for a given SDOF system, lightly damped stationary Gaussian input.
  - The Rayleigh probability density function for amplitude A, reference[1,5] is given by:

$$p(A) = \left(\frac{A}{\sigma^2}\right) e^{\left(-\frac{A^2}{2\sigma^2}\right)} \qquad A \geq 0 \qquad eq(1)$$

- If we define the normalized and maximum amplitudes, see S.J.DiMaggio reference [1]:

$\overline{A}^2_{max}$ Normalized max Amplitude response

A^2_{max} Maximum Amplitude response

- The maximum amplitude response, A^2_{max} is calculated from the Shock Response Spectrum (SRS) where the Acceleration, A^2_{max} is recovered from the Response Acceleration-Time history of the SRS. $\overline{A}^2_{max} = A^2_{max}/\sigma^2$ where $\sigma =$ (1-sigma) standard Deviation Accel(rms) response.

- SRS calculation
  - The SRS is based on the absolute acceleration transfer function response:

$$G(S) = \frac{a_2(S)}{a_1(S)} = \frac{\frac{\omega_n S}{Q} + \omega_n^2}{S^2 + \frac{\omega_n S}{Q} + \omega_n^2} \qquad \text{eq(2)}$$

and the corresponding Frequency Response Function (FRF) Digital filter is:

$$H(z) = \frac{\beta_0 + \beta_1 z^{-1} + \beta_2 z^{-2}}{1 + \alpha_1 z^{-1} + \alpha_2 z^{-2}} \qquad \text{eq(3)}$$

  - The terms in equation(3) as defined in ISO 18431-4:2007(E) reference[4], represent a ramp invariant method or digital filter for solving the equations of motion for the SRS, refer to reference[4] for definition of FRF coefficients and more information.
  - Note: The coefficients in eq(3) are a function of : sampling freq. f_s ,time interval T=1/f_s, natural frequency f_n, natural angular frequency ω_n and resonance gain Q.
  - From equations (22 & 5), S.J.DiMaggio reference[2] :

$$\bar{A}_{max}^2 = A_{max}^2/\sigma^2 = 2\ln(f_n\, T_0) \qquad \text{eq(4)}$$

$$\sigma^2 \cong \frac{\pi}{2} G * f_n * Q \qquad \text{Miles Eq.} \qquad \text{eq(5)}$$

Where: $\sigma =$ (1-sigma) stand. Dev. Accel(rms) response, G = Accel. Spectral Density (PSD), f_n = natural freq., $T_0 =$ Ref. 60sec test time, Q = damping factor, $\frac{1}{2\zeta}$, (zeta=damp. ratio)

Graphical representations, A^2_{max}/Q , G1 & G2

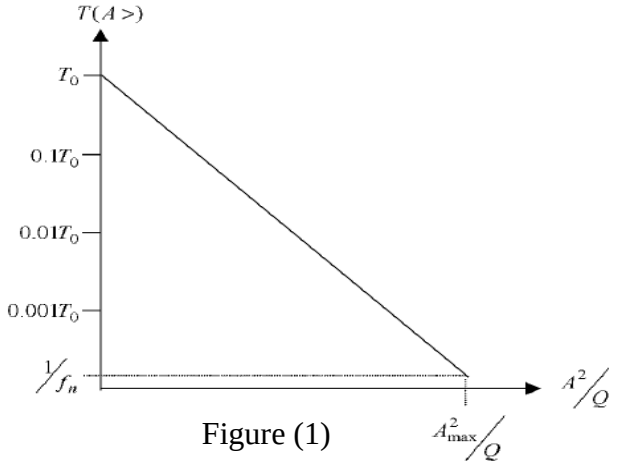


Figure (1)

Figure (1) reference[1] , represents a linear plot for the logarithm of time where response cycles exceed A^2/Q. For T(A>) decreasing as a function of T_0, the function converges to $1/f_n$ and the resulting acceleration $\frac{A^2}{Q}$ is the maximum acceleration, $\frac{A^2_{max}}{Q}$.

Plot of G1 and G2, Freq = 664.3 Hz, T_0= 382.4sec, A^2_{max}/Q=563.4

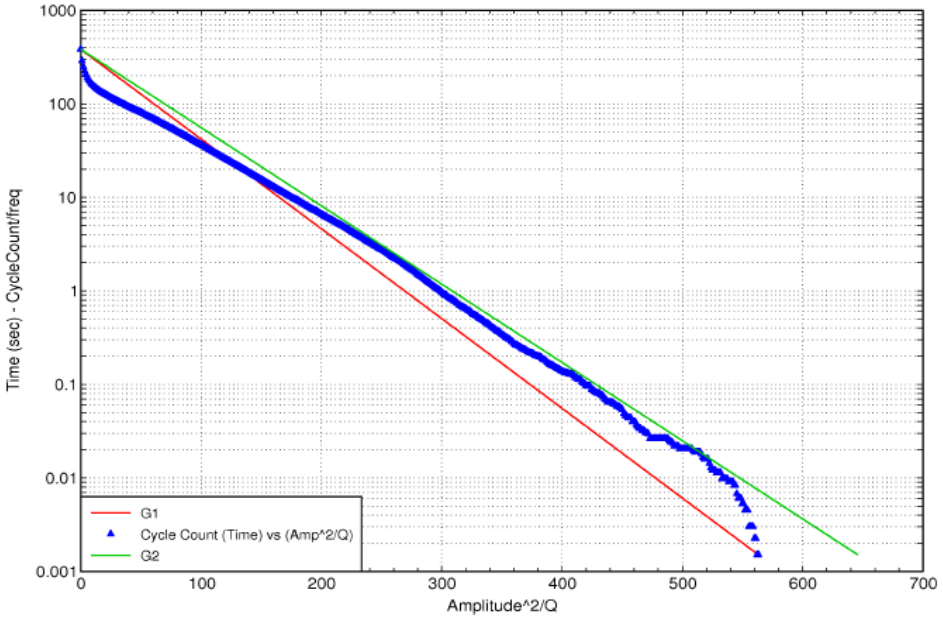


Figure (2)

Fatigue Damage Indicators: G4,G8,G12

- In order to derive G4, we start with the definition of the Flight Damage Indicator D_F as defined in S.J. DiMaggio reference [1], equation [14]:

$$D_F = Q^{b/2} \sum_i \left(\frac{A_i^2}{Q}\right)^{b/2} T(A_i) \qquad \text{eq(7)}$$

For b = 4: $D_F = Q^2 \sum_i \left(\frac{A_i^2}{Q}\right)^2 T(A_i)$

Cancel terms: $D_F^{b=4} = \sum_i (A_i)^4 T(A_i)$ eq(8)

- In order to calculate the flight damage indicator in eq(8) for fatigue duration analysis, a rainflow process is used to count the cycles and place results into bins to account for the lower and upper bounds of the environment for each frequency, f_n. Refer to reference[4], Standard Practices for Cycle Counting in Fatigue Analysis.
- Next, we need to derive the Test damage indicator D_T. We start with the definition, from S.J. DiMaggio - reference[1], eq(18) :

$$D_T = \frac{T_0}{2\,\sigma^2} \int_{A^2_{min}}^{A^2_{max}} (A^2)^{\frac{b}{2}}\; e^{\left(\frac{-A^2}{2\sigma^2}\right)}\; \mathrm{d}(A^2) \qquad \text{eq(9)}$$

- We can solve eq(9) by applying a series of substitutions and using multiple applications of integration by parts. The end result is:

$$D_T^{b=4} = T_0\; \sigma^4 [8 - e^{\left(\frac{-\overline{A}^2_{max}}{2}\right)} (\overline{A}^4_{max} + 4 * \overline{A}^2_{max} + 8)] \qquad \text{eq(10)}$$

Derivation of damage indicators

- If we equate $D_F = D_T$ for b=4 and solve for σ^2, we get :

$$\sum_i (A_i)^4 T(A_i) = T_0 \sigma^4 [8 - e^{\left(\frac{-\overline{A}_{max}^2}{2}\right)} (\overline{A}_{max}^4 + 4 * \overline{A}_{max}^2 + 8)]$$

$$\sigma^2 = \sqrt{\frac{\sum_i (A_i)^4 T(A_i)}{T_0 [8 - e^{\left(\frac{-\overline{A}_{max}^2}{2}\right)} (\overline{A}_{max}^4 + 4 * \overline{A}_{max}^2 + 8)]}} \cong \frac{\pi}{2} G\, f_n\, Q \qquad \text{eq(11)}$$

- Solve for G = G4:

$$G4 = \frac{2}{\pi\, f_n\, Q} \sqrt{\frac{\sum_i (A_i)^4 T(A_i)}{T_0 [8 - e^{\left(\frac{-\overline{A}_{max}^2}{2}\right)} (\overline{A}_{max}^4 + 4 * \overline{A}_{max}^2 + 8)]}} \quad \text{or} \quad G4 = \frac{2}{\pi\, f_n\, Q} \sqrt{\frac{D_F^{b=4}}{D_T^{b=4}}} \qquad \text{eq(12)}$$

G4 = APSD environment for fatigue damage indicator with exponent b=4.

T_0 = Time Slice − constant, default 60sec.

$\overline{A}_{max}{}^2 = A_{max}^2/\sigma^2 = 2 \ln (f_n T_0)$ or $A_{max} = \overline{A}_{max}\, \sigma$

$\overline{A}_{max}$ is the normalized max accel., (vector quantity)

f_n = Natural Frequency (Hz), evaluated for a series of SDOF systems spanning a specific Frequency range.

$\overline{A}_{max}{}^{4} = (\overline{A}_{max}{}^{2})^2 = (2 * f_n * T_{0)}{}^{2}$ (vector quantity)

$T(A_i) =$ Rainflow count of the Response Accel-Time History, recovered response from SRS of the input accel-time history, (vector) – Time Domain. See detailed counting of peaks and rainflow cycles in reference[4].

A_i = Rainflow count of the Response Acceleration-Time History, SDOF base-drive response from SRS of the input accel-time history, (vector) – Time Domain.

- In a similar manner,

$$G8 = \frac{2}{\pi f_n Q}^{1/4} \sqrt{\frac{\sum_i (A_i)^8 T(A_i)}{T_0 \left[384 - e^{\left(\frac{-\overline{A}_{max}^2}{2}\right)} (\overline{A}_{max}^8 + 8\overline{A}_{max}^6 + 48\overline{A}_{max}^4 + 192\overline{A}_{max}^2 + 384)\right]}} \quad \text{eq(13)}$$

$$G12 = \frac{2}{\pi f_n Q} * \sqrt[1/6]{\frac{\sum_i (A_i)^{12} T(A_i)}{T_0 \left[384 - e^{\left(\frac{-\overline{A}_{max}^2}{2}\right)} \left(\overline{A}_{max}^{12} + 12\overline{A}_{max}^{10} + 120\overline{A}_{max}^8 + 960\overline{A}_{max}^6 + 5760\overline{A}_{max}^4 + 23040\overline{A}_{max}^2 + 46080\right)\right]}} \quad \text{eq(14)}$$

Appendix B, Pseudo-Velocity Method

Pseudo-Velocity Method

- The relationship between axial velocity and stress for a long thin rod is $\sigma_{max} = \rho c v_{max}$ where v_{max} is the max. axial velocity.
- It can be shown that the maximum stress σ_s is proportional to pseudo-velocity, σ_s = k σ_{pv} where σ_{pv} is the oscillator pseudo-velocity RMS. The corresponding pseudo-velocity transfer function is:

$$G(S) = \frac{a_2(S)}{a_1(S)} = \frac{-\omega_n}{S^2 + \frac{\omega_n S}{Q} + \omega_n^2} \quad ; \text{ (pseudo-velocity)} \quad \text{eq(15)}$$

and the corresponding Frequency Response Function (FRF) Digital filter is:

$$H(z) = \frac{\beta_0 + \beta_1 * z^{-1} + \beta_2 * z^{-2}}{1 + \alpha_1 * z^{-1} + \alpha_2 * z^{-2}} \quad ; \text{ (pseudo-velocity)} \quad \text{eq(16)}$$

- For the Pseudo-Velocity approach, in a similar manner to the acceleration method, we define the Probability Density Function :

$$P(PV) = \frac{PV}{\sigma_{pv}^2} e^{\frac{-PV^2}{2\sigma_{pv}^2}} \quad \text{eq(17)}$$

where PV is the pseudo-velocity oscillator response.

- Integrating from arbitrary amplitude PV to infinity results in the probability that cycles have amplitude > PV or P(PV >). If the stationary environment duration is T_0, the cumulative duration is T(PV >) :

$$\text{T(PV >)} = T_0 \, e^{\frac{-PV^2}{2\sigma_{pv}^2}} \quad \text{eq(18)}$$

Pseudo-Velocity Approach - continued

- Take the natural log of each side for eq(16), simplify terms where T(PV >) = $\frac{1}{f_n}$; as the lower limit is reached for $\frac{PV_{max}^2}{Q}$ where PV_{max}^2 is the maximum pseudo-velocity amplitude :

$$\ln[f_n T_0] = \frac{PV_{max}^2}{2\sigma_{pv}^2} \qquad \text{eq(19)}$$

- Recall from Scott I. McNeill paper, equation(15), reference[2], derived from Example 6.3, Force-Input/Disp.-Output system in reference[5] where the input is white noise :

$$\sigma_{pv} = \sqrt{\frac{P_a(f_n)Q}{8\pi f_n}} \qquad \text{where } P_a(f_n) \text{ is the PSD term, } G_1. \qquad \text{eq(20)}$$

- Substitute σ_{pv} eq(20) into eq(19), rearrange terms and we have:

$$G_1 = \frac{PV_{max}^2 4\pi f_n}{Q \ln[f_n T_0]} \qquad \text{eq(21)}$$

- The resulting eq(21) represents the power spectral density (PSD), which is a function of the max pseudo-velocity response at each natural frequency represented by a Single Degree of Freedom (SDOF) system. This result is similar to eq(6).
- Calculate G4, G8 & G12, Fatigue Damage Spectrum – Frequency Domain, for the damage due to stress, eel.(11), reference[2], we have :

$$D = \frac{V_m^+ T}{c} \int_0^\infty p(S) S^{+b} dS \qquad \text{where : } \quad p(S) = \frac{S}{\sigma_s^2} e^{\frac{-S^2}{2\sigma_s^2}} \qquad \text{eq(22)}$$

  - where P(S) in eq(22) is the probability density function of the stress maxima, T is the total time of exposure to the stress environment, V_m^+ is the number of positive cycles per unit time in the stress history, b is the fatigue exponent, c is a proportionality constant and S is the stress value of the peaks, see reference [2].

Pseudo-Velocity Approach - continued

- Since maxima occur every $(1/f_n)$ seconds for a lightly damped oscillator response as illustrated in Figure(1), $V_m^+ = f_n$. We can now rewrite equation (22) as :

$$D = \frac{f_n T}{\sigma_s^2 c} \int_0^\infty S^{(1+b)} e^{\frac{-S^2}{2\sigma_s^2}} dS$$

- Let u = S^2 ; $S = \sqrt{u}$; $du = 2sds$; $K = \frac{f_n T}{\sigma_s^2 C}$, also, for a property of the Gamma Function :

$\int_0^\infty t^r e^{-at} dt = \frac{\Gamma(r+1)}{a^{r+1}}$; $let\ a = \frac{1}{2\sigma_s^2}$; perform integration, make substitutions and distribute sq.-root :

$$D = \frac{f_n T}{C} k^b (2\sigma_{pv}^2)^{b/2} \Gamma\left(\frac{b}{2}+1\right) \qquad \text{eq(22)}$$

- Now substitute eq(20) into eq(22), Isolate σ_{pv} and multiply by (2/b), cancel constants where c/ $k^b\ are\ (1)$, solve for $P_a(f_n)$ with b=4. Substitute $G_4 = P_a(f_n)$ and $\Gamma\left(\frac{4}{2}+1\right) = 2$, $D = D_F$, T=T_0:

$$G_4 = \frac{4\pi f_n}{Q} \sqrt{\frac{D_F}{2 f_n T_0}} \qquad \text{eq(23)}$$

  - Where D_F is the number of fatigue damage cycles from flight data, counted using rainflow method in a series of Amplitude Bins similar to the method used in the acceleration approach.

- Similarly for G_8 & G_{12} where b = 8 & 12 and $\Gamma\left(\frac{b}{2}+1\right)$ = 24 & 720:

$$G_8 = \frac{4\pi f_n}{Q} \sqrt[\frac{1}{4}]{\frac{D_F}{24 f_n T_0}} \qquad \text{eq(24)}$$

$$G_{12} = \frac{4\pi f_n}{Q} \sqrt[\frac{1}{6}]{\frac{D_F}{720 f_n T_0}} \qquad \text{eq(25)}$$

Appendix – C, Python Code - Options

Fatigue Damage Equivalent – fdepsd python code

pyyeti.fdepsd.**fdepsd**(*sig*, *sr*, *freq*, *Q*, *resp='absacce'*, *hpfilter=5.0*, *nbins=300*, *T0=60.0*, *rolloff='lanczos'*, *ppc=12*, *parallel='auto'*, *maxcpu=14*, *verbose=False*)

Compute a fatigue damage equivalent PSD from a signal.

Parameters:

- **sig** (*1d array_like*) – Base acceleration signal.
- **sr** (*scalar*) – Sample rate.
- **freq** (*array_like*) – Frequency vector in Hz. This defines the single DOF (SDOF) systems to use.
- **Q** (*scalar > 0.5*) – Dynamic amplification factor $Q = 1/(2\zeta)$ where ζ is the fraction of critical damping.
- **resp** (*string; optional*) – The type of response to base the damage calculations on:

resp	Damage is based on
'absacce'	absolute acceleration [1]
'pvelo'	pseudo velocity [2]

- **hpfilter** (*scalar or None; optional*) – High pass filter frequency; if None, no filtering is done. If filtering is done, it is a 3rd order butterworth via **`scipy.signal.filtfilt()`**.
- **nbins** (*integer; optional*) – The number of amplitude levels at which to count cycles
- **T0** (*scalar; optional*) – Specifies test duration in seconds
- **rolloff** (*string or function or None; optional*) – Indicate which method to use to account for the SRS roll off when the minimum *ppc* value is not met. Either 'fft' or 'lanczos' seem the best. If a string, it must be one of these values:

rolloff	Notes
'fft'	Use FFT to upsample data as needed. See **`scipy.signal.resample()`**.

Fatigue Damage Equivalent – fdepsd python code - continued

'lanczos'	Use Lanczos resampling to upsample as needed. See **`dsp.resample()`**.
'prefilter'	Apply a high freq. gain filter to account for the SRS roll-off. See **`srs.preroll()`** for more information. This option ignores *ppc*.
'linear'	Use linear interpolation to increase the points per cycle (this is not recommended; method; it's only here as a test case).
'none'	Don't do anything to enforce the minimum *ppc*. Note error bounds listed above.
None	Same as 'none'.

- If a function, the call signature is: `sig_new, sr_new = rollfunc(sig, sr, ppc, frq)`. Here, *sig* is 1d, len(time). The last three inputs are scalars. For example, the 'fft' function is (trimmed of documentation):
- **ppc** (*scalar; optional*) – Specifies the minimum points per cycle for SRS calculations. See also *rolloff*. Min. recommended ppc=10 which produces Max. errror at highest freq. of 8.14%, ppc=20, 2.05%.
- **parallel** (*string; optional*) – Controls the parallelization of the calculations:

parallel	**Notes**
'auto'	Routine determines whether or not to run parallel.
'no'	Do not use parallel processing.
'yes'	Use parallel processing. Beware, depending on the particular problem, using parallel processing can be slower than not using it. On Windows, be sure the **`fdepsd()`** call is contained within: `if __name__ == "__main__":`

- **maxcpu** (*integer or None; optional*) – Specifies maximum number of CPUs to use. If None, it is internally set to 4/5 of available CPUs (as determined from **`multiprocessing.cpu_count()`**.
- **verbose** (*bool; optional*) – If True, routine will print some status information.

Fatigue Damage Equivalent – fdepsd python code - continued

- **maxcpu** (*integer or None; optional*) – Specifies maximum number of CPUs to use. If None, it is internally set to 4/5 of available CPUs (as determined from **`multiprocessing.cpu_count()`**.
- **verbose** (*bool; optional*) – If True, routine will print some status information.
- *A record (SimpleNamespace class) with the members*
- **freq** (*1d ndarray*) – Same as input *freq*.

Values Returned :

- **psd** (2d ndarray; `len(freq) x 5`) – The five columns are: [G1, G2, G4, G8, G12]:

Name	Description
G1	The “G1” PSD (Mile’s or similar equivalent from SRS); uses the maximum cycle amplitude instead of the raw SRS peak for each frequency. G1 is not a damage-based PSD.
G2	The “G2” PSD of reference [1]; G2 >= G1 by bounding lower amplitude counts down to 1/3 of the maximum cycle amplitude. G2 is not a damage-based PSD.
G4, G8, G12	The damage-based PSDs with fatigue exponents of 4, 8, and 12

- **amp** (2d ndarray; `len(freq) x 5`) – The five columns correspond to the columns in *psd*. They are the Mile’s equation (or similar if using `resp='pvelo'`) SDOF (single DOF oscillator) peak response using the peak factor `sqrt(2*log(f*T0))`. Note that the first column is, by definition, the maximum cycle amplitude for each SDOF from the rainflow count (G1 was calculated from this). Typically, this should be very close to the raw SRS peaks but a little lower since SRS just grabs peaks without consideration of the opposite peak.
- **binamps** (2d ndarray; `len(freq) x nbins`) – Each row (for a specific frequency SDOF

in *binamps* contains the lower amplitude boundary of each bin. Spacing of the bins is linear. The next column for this matrix would be `amp[:, 0]`.

- **count** (2d ndarray; `len(freq) x nbins`) – Summary matrix of the rainflow cycle counts. Size corresponds with *binamps* and the count is cumulative; that is, the count in each entry includes cycles at the *binamps* amplitude and above. Therefore, first column has total cycles for the SDOF.
- **srs** (1d ndarray; length = `len(freq)`) – The raw SRS peaks version of the first column in *amp*. See *amp*.
- **var** (1d ndarray; length = `len(freq)`) – Vector of the SDOF response variances.
- **parallel** (*string*) – Either ‘yes’ or ‘no’ depending on whether parallel processing was used or not.
- **ncpu** (*integer*) – Specifies the number of CPUs used.

resp (*string*) – Same as the input *resp*.