Converting Laser PSD to Lineshape

Cyrus Young, June 5, 2024

Introduction to PSD, lineshape, and goals of report:

The Frequency PSD (Power Spectral Density) is a spectrum that measures the power of a signal as a function of frequency. Often, the laser will have frequency noise sources associated with it, which can be characterized with a PSD, thus describing the intensity of the noise source as a function of frequency. The lineshape of a laser essentially describes the frequency spectrum of the laser output (i.e shows the intensity of the output laser as a function of frequency) and thus is a good metric to measure the quality of the laser. The goal of this report is to use a method to find the lineshape of a laser only given the Frequency Noise PSD of the laser and to test the accuracy of the method on a variety of different Frequency Noise PSDs.

Introduction to first paper:

I) General Algorithm:

For pages 1→ 20 of this document, the work refers to the following paper, labeled *Simple approach to the relation between laser frequency noise and laser line shape*. The paper describes an approach for which, given the noise spectral density of an arbitrary laser, one can calculate the laser lineshape. The general approach is as follows; given a noise spectral density as well as the laser light field, we can directly get the autocorrelation function. The laser lineshape is simply the fourier transform of the autocorrelation function.

In terms of the mathematical details, imagine we are given the frequency noise spectral density $S_{\delta\nu}(f)$ of the laser light field $E(t) = E_0 exp[i(2\pi\nu_0 + \phi(t))]$. We can then calculate the autocorrelation function, $\Gamma_E(\tau) = E^*(t)E(t + \tau)$ which is equation (1) in the paper, as follows:

$$\Gamma_E(\tau) = E_0^2 e^{i2\pi\nu_0\tau} exp(-2 * \int_0^\infty S_{\delta\nu}(f) * \frac{sin^2(\pi f\tau)}{f^2} df)$$

Where $\delta\nu = \nu - \nu_0$ is the laser frequency deviation about its average value ν_0. Using the Wiener–Khintchine theorem, we can find the laser lineshape $S_E(\nu)$ which is equation (2) in the paper, by taking the fourier transform of the autocorrelation function:

$$S_E(\nu) = 2\int_{-\infty}^{\infty} e^{-i2\pi\nu\tau}\Gamma_E(\tau)d\tau$$

Unfortunately, in the majority of cases where the noise is not pure white noise, the integral above cannot be solved analytically so we often have to resort to software to solve for the lineshape.

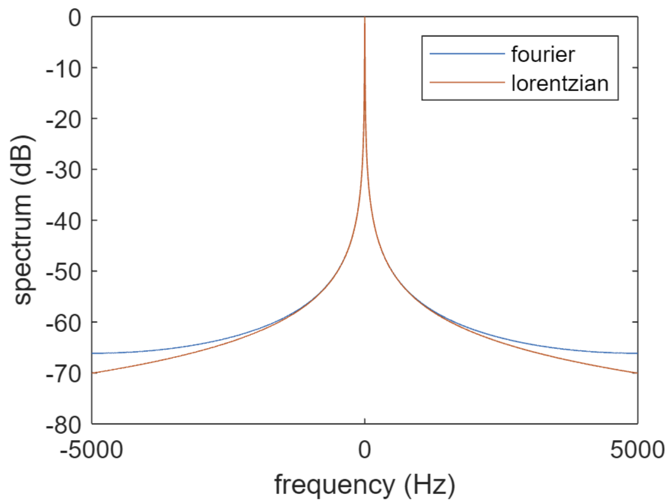
II) Error of lineshape estimate for white noise:

Using the paper's approach above, we can use the following MATLAB program to calculate the laser lineshape of white noise (as it is the easiest to calculate). We do this by finding the line shape $S_E(\nu)$ of pure white noise represented by $S_{\delta\nu}(\nu) = h_0$ for all values of ν where in our case, we choose $h_0 = 1$. An important fact to know is that if we integrate $\int_0^{\infty} S_{\delta\nu}(f) * \frac{sin^2(\pi f\tau)}{f^2}df$ where $S_{\delta\nu}(\nu) = h_0$ then the result, `integ`, will be equal to $\pi^2|\tau|h_0/2$. Note that the central frequency and the electric field magnitude are $\nu_0 = 0$ and $E_0 = 1$ respectively.

```
n=100000;
warning('off','MATLAB:integral:MaxIntervalCountReached')

global t hwhite;  % h_0
hwhite=1;

autocorr=zeros(1,n);

fs=10000; % Sampling Frequency
Ts=1/fs;

for x =0:n-1
    t=x*Ts;
    integ=pi^2*t*hwhite/2;
    autocorr(1,x+1)=exp(-2*integ);
end

y=cat(2,flip(autocorr),autocorr(2:end));

plot(10*log10(y));

absfft=abs(fftshift(fft(y)));
```

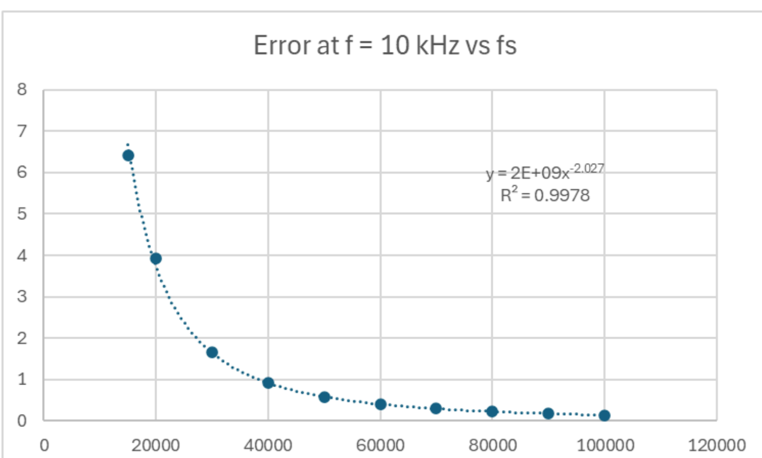
```
spectrum=transpose(absfft/max(absfft));

f=transpose(linspace(-fs/2,fs/2,2*n-2));
spectrum=spectrum(2:n*2-1);
plot(f,10*log10(spectrum))
xlabel('frequency (Hz)')
ylabel('spectrum (dB)')

hold on
plot(f,10*log10(lorentz(f)))

legend('fourier','lorentzian')
hold off

desired_frequency = 1e4;  % 10^4 Hz
[~, idx] = min(abs(f - desired_frequency));  % Find the closest frequency index

% Calculate dB values for spectrum and Lorentzian function at the index
dB_spectrum = 10 * log10(spectrum(idx));
dB_lorentz = 10 * log10(lorentz(desired_frequency));

% Calculate the difference in dB
dB_difference = dB_spectrum - dB_lorentz;

% Display the result
disp(['Difference at f = 10^4 Hz: ', num2str(dB_difference), ' dB']);

function y = lorentz(f)
    global hwhite
    y=(pi*hwhite/2)^2./((f).^2+(pi*hwhite/2)^2);
end
```

If we graph the results below and compare our calculated result, named “fourier” to a perfect lorentzian called “lorentzian” (note that the laser lineshape of pure white noise is a lorentzian), we see that the fit is very good for about the middle 35% of the graph. However, as the magnitude of the frequency increases, the accuracy of our approximation decreases. Fortunately, there is a partial fix for this problem.

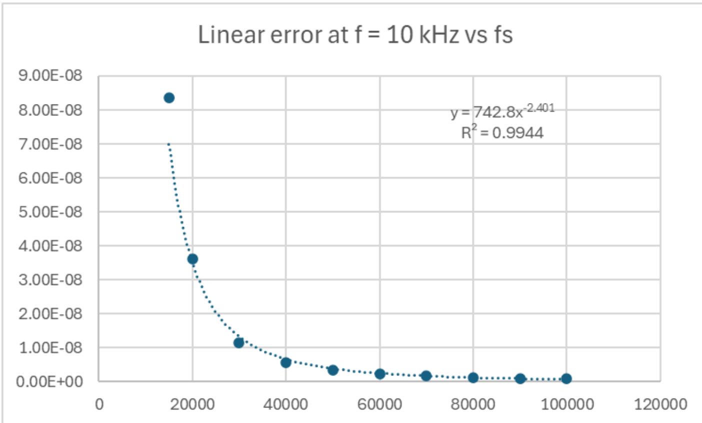


II.A) Analyzing Error for Pure White Noise:

If we increase the sampling frequency, `fs`, of our program, then we can decrease the error of our lineshape estimate for any chosen frequency. The error is calculated as $fourier(f_0) - lorentzian(f_0)$ where f_0 is a chosen frequency (10 kHz in our case). A plot showing the error at $f_0 = 10\ kHz$ as a function of sampling frequency is shown below:

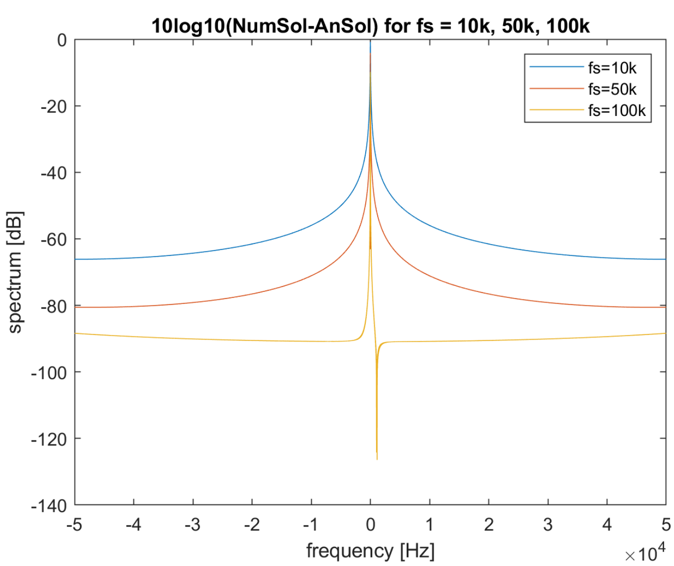


If we want to know how the *linear* error (i.e. the error without taking $10log_{10}$ of the the lorentzian and the spectrum) behaves as we change the sampling frequency, f_s, the graph is shown below:



This is important since we now have a straightforward method of reducing the error of the lineshape estimate for any chosen frequency. Certain tasks may require us to know the lineshape of a laser at a given frequency. If we want an accurate estimation of the lineshape of an arbitrary laser given only its frequency noise spectrum, then all we have to do is apply the program above. In order to increase the accuracy of the estimation, then by simply increasing the sampling frequency of the program, we can produce an exponential decrease in the error of our approximation.

For our error analysis above we checked the error at a fixed frequency, f_0, as a function of sampling frequency, `fs`. The next step is to analyze the error of our program as a function of *output frequency* rather than the sampling frequency of our program. The following graph displays $10log_{10}$ of `Spectrum-Lorentz` (which are the numerical and exact solutions, respectively) for a fixed sampling frequency of 100000. As expected the error increases slightly as the magnitude of our output frequency increases:



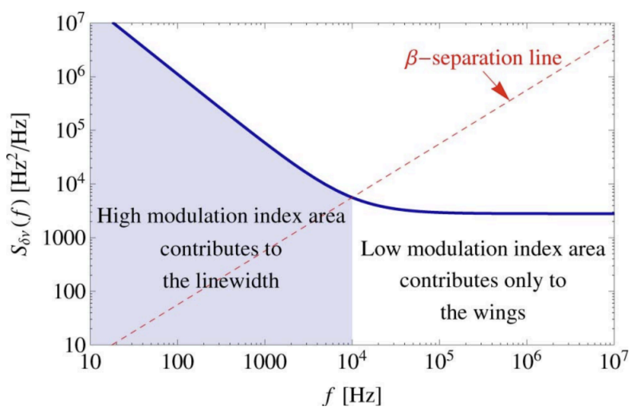
III) Flicker Noise:

Using the same approach as before, I derived the lineshape $S_E(\nu)$ of pure flicker noise (which is found in the section of the paper labeled "**Application 1: Laser Spectrum in the Case of Flicker Frequency Noise"**) which has the following frequency noise spectrum:

$$S_{\delta\nu} = \alpha f^{-\alpha}$$

Using this flicker noise model, I derived the Full-Width Half-Maximum (FWHM), represented by equations (12) and (13) in the paper, as shown below. Note that f_m is the frequency at which our noise intersects the β-separation line. The dashed line in the image below is given by the equation:

$$S_{\delta\nu} = \frac{8ln(2)f}{\pi^2}$$



$$\text{FWHM} = f_m \frac{8\ln(2)}{\pi}[\ln(f_m T_o)]^{1/2}, \qquad (12)$$

and for $\alpha > 1$,

$$\text{FWHM} = f_m \frac{8\ln(2)}{\pi}\left[\frac{(f_m T_o)^{\alpha-1} - 1}{\alpha - 1}\right]^{1/2}. \qquad (13)$$

Using the lineshape method that was introduced at the beginning of this document as well as the following equations from the paper:

$$\text{FWHM} = (8\ln(2)A)^{1/2}, \qquad (9)$$

$$A = \int_{1/T_o}^{\infty} H(S_{\delta\nu}(f) - 8\ln(2)f/\pi^2)S_{\delta\nu}(f)\mathrm{d}f, \qquad (10)$$

We can derive the FWHM of pure flicker noise, as represented by equations (12) and (13) above. The derivations are shown in the following screenshot:

<u>Flicker Noise:</u> $S_{\delta\nu}(f) = \alpha f^{-\alpha}$

<u>For β-line barrier:</u> $S_{\delta\nu}(f) = \alpha f^{-\alpha} = 8\ln(2) f_m / \pi^2 \Rightarrow \alpha = \frac{8\ln(2) f_m^{\alpha+1}}{\pi^2}$

<u>Calculate A (surface of high modulation index area):</u>

$$A = \int_{1/T_o}^{\infty} H\left[\alpha/f^{\alpha} - 8\ln(2) f/\pi^2\right] \alpha f^{-\alpha} \Rightarrow \left\{\alpha/f^{\alpha} \geq 8\ln(2) f/\pi^2 \Rightarrow \frac{\pi^2\alpha}{8\ln(2)} \geq f^{\alpha+1}\right\}$$

$$A = \int_{1/T_o}^{\left(\frac{\pi^2\alpha}{8\ln(2)}\right)^{\frac{1}{\alpha+1}}} \alpha f^{-\alpha}\, df = \int_{1/T_o}^{f_m} \alpha f^{-\alpha}\, df = \begin{cases} \alpha \ln(f)\Big|_{1/T_o}^{f_m} = \frac{8\ln(2) f_m^2}{\pi^2} \ln(f_m T_o), & \alpha = 1 \\ \frac{\alpha f^{1-\alpha}}{1-\alpha}\Big|_{T_o}^{f_m} = \frac{8\ln(2)}{\pi^2} f_m^2 \left[\frac{(T_o f_m)^{\alpha-1} - 1}{\alpha - 1}\right], & \alpha > 1 \end{cases}$$

<u>Using eq (9) in paper:</u>

$$FWHM = \left[8\ln(2) A\right]^{1/2} = \begin{cases} \frac{8\ln(2)}{\pi} f_m \left[\ln(f_m T_o)\right]^{1/2}, & \alpha = 1 \\ \frac{8\ln(2)}{\pi} f_m \left[\frac{(T_o f_m)^{\alpha-1} - 1}{\alpha - 1}\right]^{1/2}, & \alpha > 1 \end{cases}$$

However, the above process was merely to confirm the exact theoretical equations given in the paper. The next step is to use the paper's approach to calculate the lineshape of the flicker noise model and see if we can reproduce it. For a given value of f_m (in our case the arbitrary value of $f_m = 1$ is used) , we can use the given MATLAB program below to calculate $FWHM/f_m$ with T_o and α as our inputs.

```
function y = FWHMOverfm(To, alpha)
   % Parameters
   fm = 1; % Reference freq where noise intersects beta-separation line
   frequencies = linspace(1/To, 100*fm, 100000); % Frequency range for integration
   tau_values = linspace(-5, 5, 1000); % Range of tau values for autocorrelation
   a = (8 * log(2) * fm^(alpha + 1)) / pi^2; % Flicker noise coefficient

   % Frequency noise spectral density
   S_delta_nu = @(f) a * f.^(-alpha);

   % Autocorrelation function
   Gamma_E = @(tau) exp(-2 * integral(@(f) S_delta_nu(f) .* (sin(pi * f * tau) ./
f).^2, 1/To, inf, 'ArrayValued', true));
   autocorrelation = arrayfun(Gamma_E, tau_values);
```

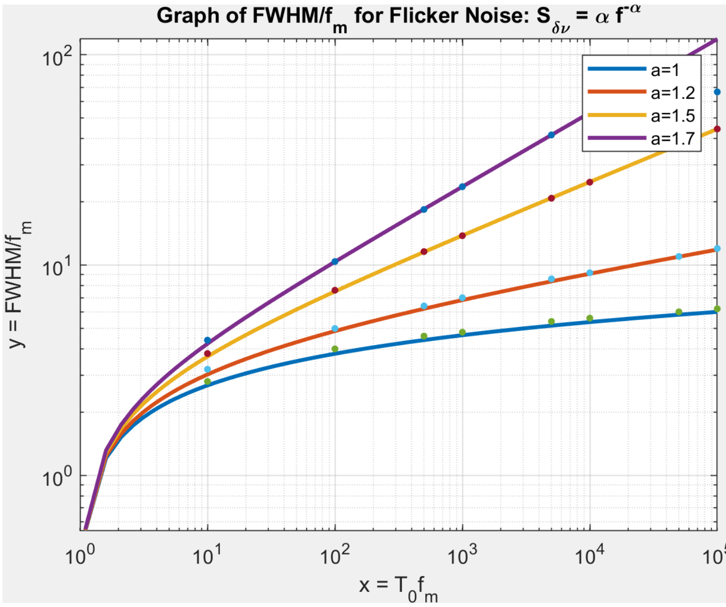
```
    % Compute laser lineshape using FFT of autocorrelation function
    lineshape = abs(fftshift(fft(ifftshift(autocorrelation))));

    % Freqs corresponding to lineshape
    n = length(tau_values);
    delta_tau = tau_values(2) - tau_values(1);
    frequencies_lineshape = (-n/2:n/2-1) / (n * delta_tau);

    % Calculate FWHM
    half_max = max(lineshape) / 2;
    indices_above_half = find(lineshape > half_max);
    fwhm = frequencies_lineshape(indices_above_half(end)) -
frequencies_lineshape(indices_above_half(1));

    y = fwhm/fm;
end
```

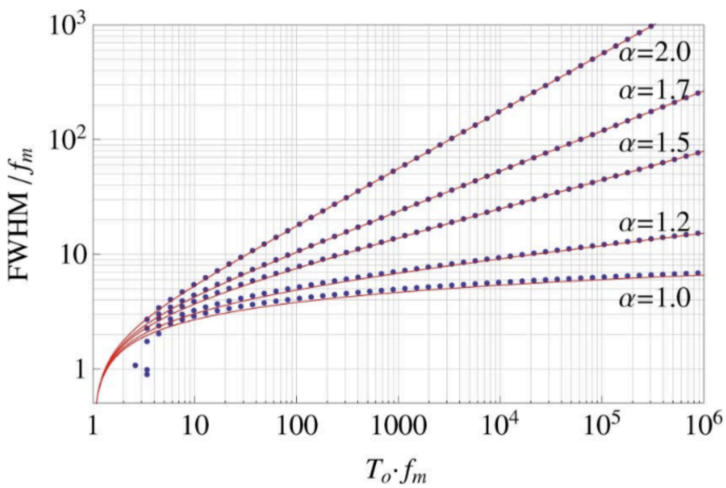
If we actually want to graph our results, we need to graph $FWHM/f_m$ vs $T_o f_m$ for different values of α. The following MATLAB code to do so is shown below:

```
in_arr = [10, 100, 500, 1000, 5000, 10000, 50000, 100000];
out_arr10 = [0,0,0,0,0,0,0,0];
out_arr12 = [0,0,0,0,0,0,0,0];
out_arr15 = [0,0,0,0,0,0,0,0];
out_arr17 = [0,0,0,0,0,0,0,0];
L = length(in_arr);
% alpha = 1
for i = 1:L
    out_arr10(i) = FWHMOverfm(in_arr(i), 1);
end
% alpha = 1.2
for i = 1:L
    out_arr12(i) = FWHMOverfm(in_arr(i), 1.2);
end
% alpha = 1.5
for i = 1:L
    out_arr15(i) = FWHMOverfm(in_arr(i), 1.5);
end
% alpha = 1.7
for i = 1:L
    out_arr17(i) = FWHMOverfm(in_arr(i), 1.7);
end
% Constants
a1 = 1.2;
a2 = 1.5;
a3 = 1.7;
C = 8 * log(2) / pi;
% Define the range of x
```

```
x = linspace(0.1, 100000, 200000); % Starting from 0.01 to avoid division by zero
or log of zero
% Calculate the function
y0 = C * sqrt(log(x));
y1 = C * sqrt((x.^(a1-1) - 1) / (a1-1));
y2 = C * sqrt((x.^(a2-1) - 1) / (a2-1));
y3 = C * sqrt((x.^(a3-1) - 1) / (a3-1));
% Plot
figure;
loglog(x, y0, 'LineWidth', 2);
hold on
loglog(x, y1, 'LineWidth', 2);
hold on
loglog(x, y2, 'LineWidth', 2)
hold on
loglog(x, y3, 'LineWidth', 2)
hold on
loglog(in_arr, out_arr10, '.', 'MarkerSize', 12);
hold on
loglog(in_arr, out_arr12, '.', 'MarkerSize', 12);
hold on
loglog(in_arr, out_arr15, '.', 'MarkerSize', 12);
hold on
loglog(in_arr, out_arr17, '.', 'MarkerSize', 12);
hold on
title('Plot of the function y = C[(x^{(a-1)}-1)/(a-1)]^{1/2}');
xlabel('x = T_{0}f_{m}');
ylabel('y = FWHM/f_{m}');
hold off
legend('a=1','a=1.2','a=1.5','a=1.7')
grid on;
```

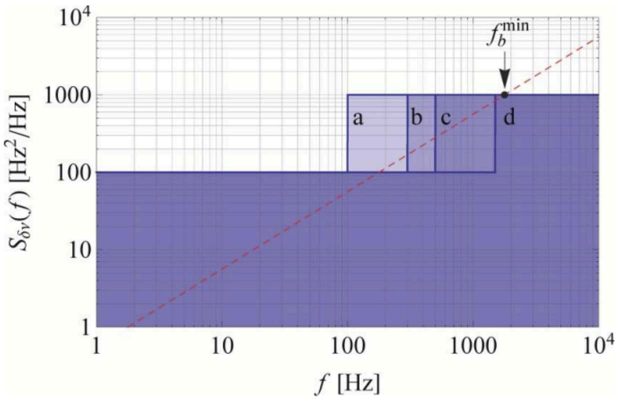
The MATLAB graph is shown below. We see that it matches figure 5 of the paper (shown below the MATLAB graph):





IV) Step Function Noise:

The next form of laser noise that will be analyzed is step function noise, as shown below. This noise is also generated due to a presence of a servo bump (which causes the sudden jump in the noise level).



We can represent general step function noise as follows: $S_{\delta\nu} = h_a$ for $f < f_b$, $S_{\delta\nu} = h_b$ for $f \geq f_b$ where f_b is the bandwidth frequency of the servo loop. The autocorrelation function of this noise is shown below:

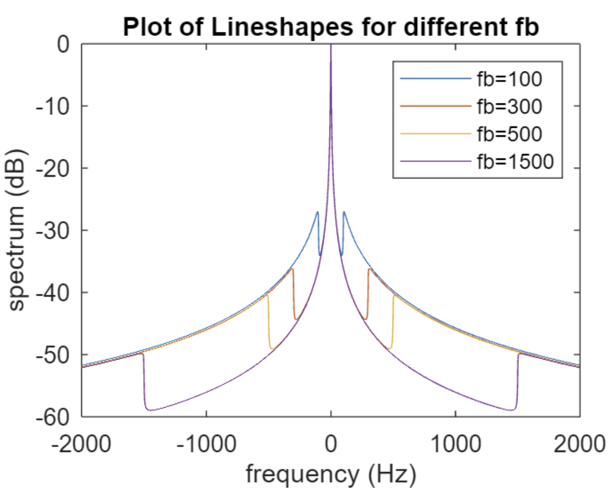
$$\Gamma_E(\tau) = E_0^2 e^{i2\pi\nu_0\tau} e^{-h_b\pi^2|\tau| - \frac{h_a - h_b}{f_b}\left(\omega_b\tau\mathrm{Si}(\omega_b\tau) - 2\sin^2\left(\frac{\omega_b\tau}{2}\right)\right)}$$

Where $\omega_b = 2\pi f_b$ and Si is the sine integral function. The final step is to find the lineshape of the laser with the noise input as above. Below is a MATLAB program that takes in a Frequency Noise input as above and plots the lineshape. We have $h_a = 1$, $h_b = 10$, $f_b = 1500\ Hz$:

```
n=100000;
warning('off','MATLAB:integral:MaxIntervalCountReached')
global t ha hb fb;
ha=1;
hb=10;
fb=1500;
wb = 2*pi*fb;
autocorr=zeros(1,n);
fs=100000;
Ts=1/fs;
for x = 0:n-1
   t=x*Ts;
```

```
   autocorr(1,x+1) = exp(-hb * pi^2 * abs(t)) * exp(-(ha - hb) / fb * (wb * t *
sinint(wb * t) - 2 * sin(wb * t / 2)^2));
end
y=cat(2,flip(autocorr),autocorr(2:end));
plot(10*log10(y));
absfft=abs(fftshift(fft(y)));
spectrum=transpose(absfft/max(absfft));
freq=transpose(linspace(-fs/2,fs/2,2*n-2));
spectrum=spectrum(2:n*2-1);
plot(freq,10*log10(spectrum))
xlim([-2000,2000])
xlabel('frequency (Hz)')
ylabel('spectrum (dB)')
plot(freq,10*log10(spectrum));
```

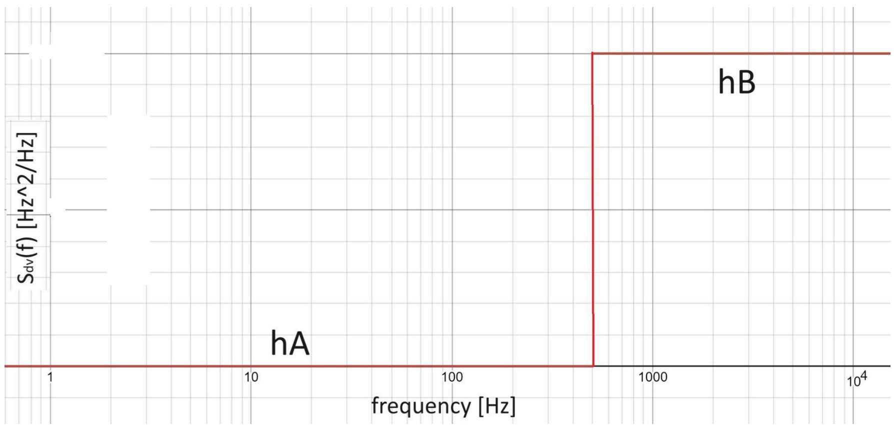
If we graph the line shape for 4 values $f_b = 100\ Hz,\ 300\ Hz,\ 500\ Hz,\ 1500\ Hz$, we get the final plot as shown below:



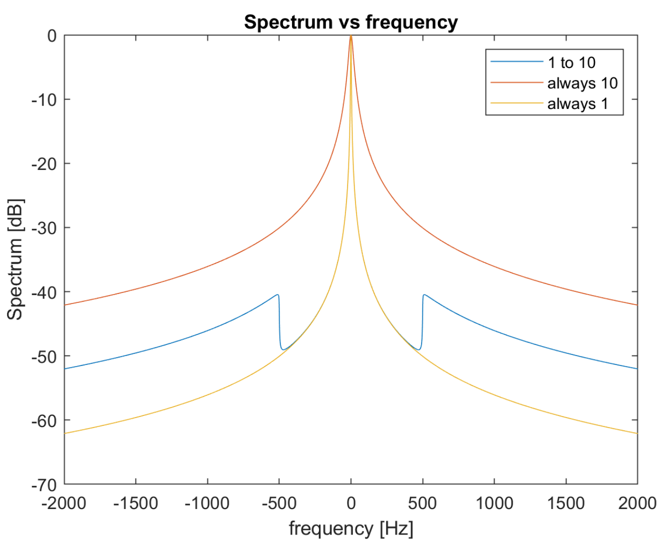
The sharp increases in both the left and right side of the lineshape are the servo bumps. Notice how as we increase f_b, the magnitude of the frequencies corresponding to the servo bump increases.

IV.A) Comparing step noise to white noise:

It is important to compare the step noise function to pure white noise. We are given a step noise function that follows this relation: $1\ Hz,\ f\ <\ 500\ Hz,\ 10\ Hz,\ f\ \geq\ 500\ Hz$ as shown in the graph below ($h_A\ =\ 1\ Hz$, $h_B\ =\ 10\ Hz$). We can call this relation "1 to 10":



If we were to graph the lineshape of "1 to 10" versus two lineshapes of pure white noise for two values ($h_1\ =\ 1\ Hz$, $h_2\ =\ 10\ Hz$ called "always 1" and "always 10" respectively), we get the following:

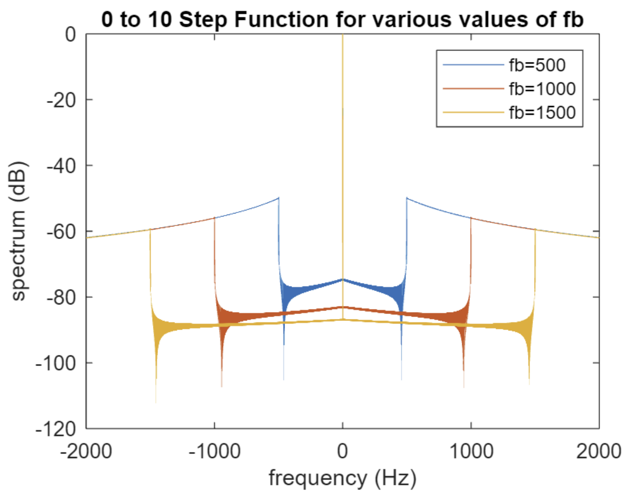


As we can see, the "always 1" white noise is basically identical to the "1 to 10" step noise as long as $|f|\ <\ 500\ Hz$.

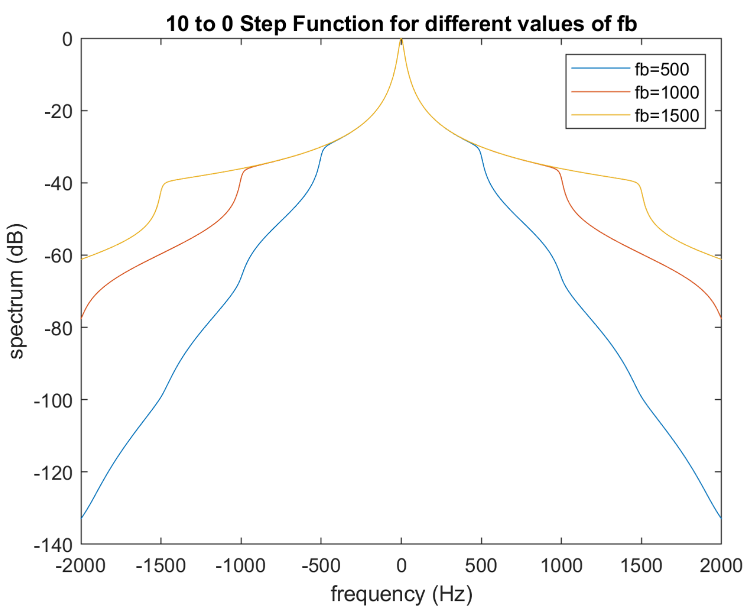
IV.B) Analyzing h_A and h_B separately for different values of f_b:

The next step is to analyze the hypothetical situation in which we focus on two separate situations; 1) $h_A = 0\ Hz$, $h_B = 10\ Hz$ and 2) $h_A = 10\ Hz$, $h_B = 0\ Hz$ for a constant bandwidth frequencies $f_b = 500\ Hz,\ 1000\ Hz,\ 1500\ Hz$.

For $h_A = 0\ Hz$, $h_B = 10\ Hz$, the graph is shown below:



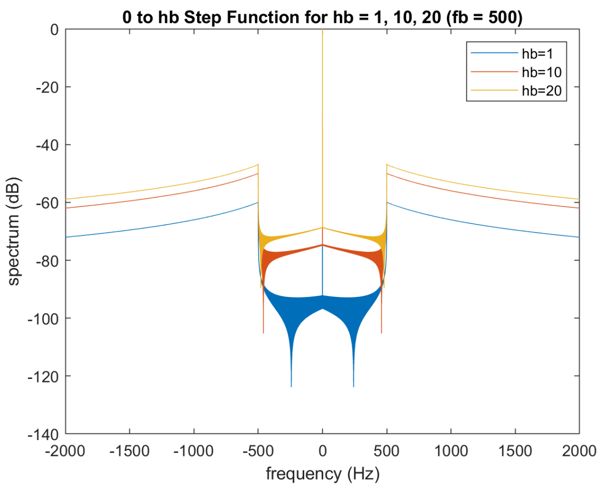
For $h_A = 0\ Hz$, $h_B = 10\ Hz$, the graph is shown below:



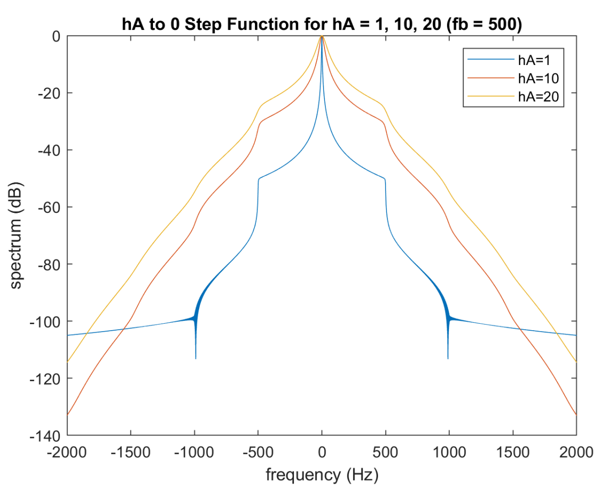
IV.C) Analyzing different values of h_A and h_B for $f_b = 500\ Hz$:

It is important to also focus on two more separate situations: 1) $h_A = 0\ Hz$, $h_B = 1,\ 10,\ 20\ Hz$ and 2) $h_A = 1,\ 10,\ 20\ Hz$, $h_B = 0\ Hz$ for a constant bandwidth frequency of $f_b = 500\ Hz$.

For $h_A = 0\ Hz$, $h_B = 1,\ 10,\ 20\ Hz$ for $f_b = 500\ Hz$, the graph is shown below:



For $h_A = 1,\ 10,\ 20\ Hz$, $h_B = 0\ Hz$ for $f_b = 500\ Hz$, the graph is shown below:

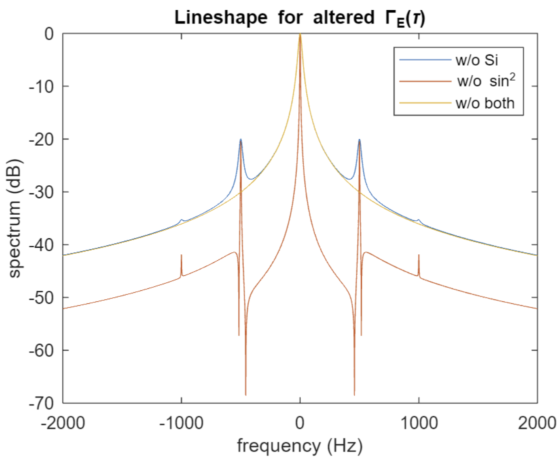


IV.D) Analyzing each component of the autocorrelation function:

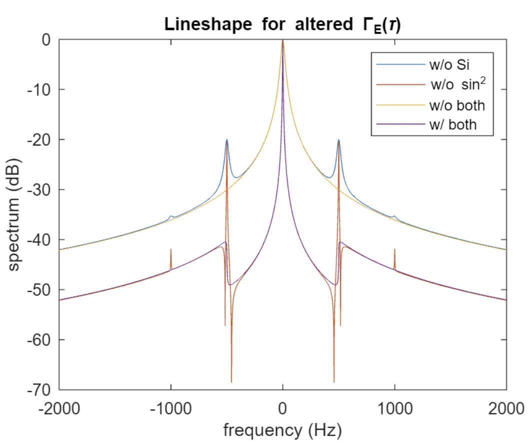
Recall the autocorrelation function for step noise:

$$\Gamma_E(\tau) = E_0^2 e^{i2\pi\nu_0\tau} e^{-h_b\pi^2|\tau| - \frac{h_a - h_b}{f_b}\left(\omega_b\tau \mathrm{Si}(\omega_b\tau) - 2\sin^2\left(\frac{\omega_b\tau}{2}\right)\right)}$$

If we alter our autocorrelation function so that there is (1) no $\omega_b\tau Si(\omega_b\tau)$ and (2) no $2sin^2(\frac{\omega_b\tau}{2})$ term in the exponential and we run the same program as before, we generate the following graph:



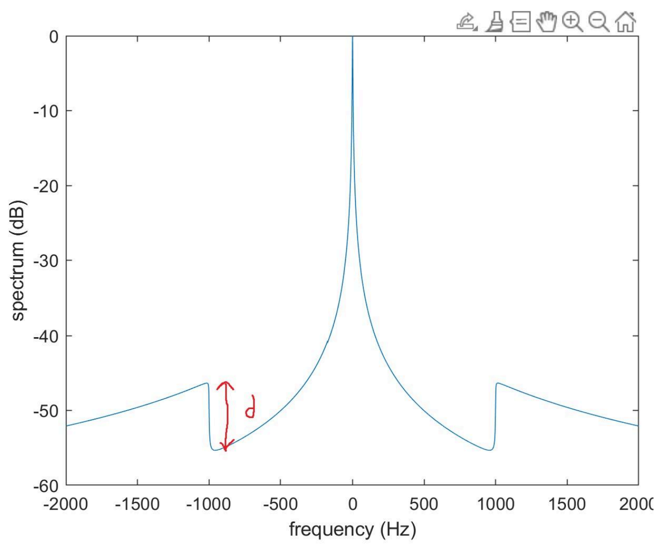
If we also graph our actual autocorrelation function (labeled “w/ both”) along with the three plots above, we get the following:



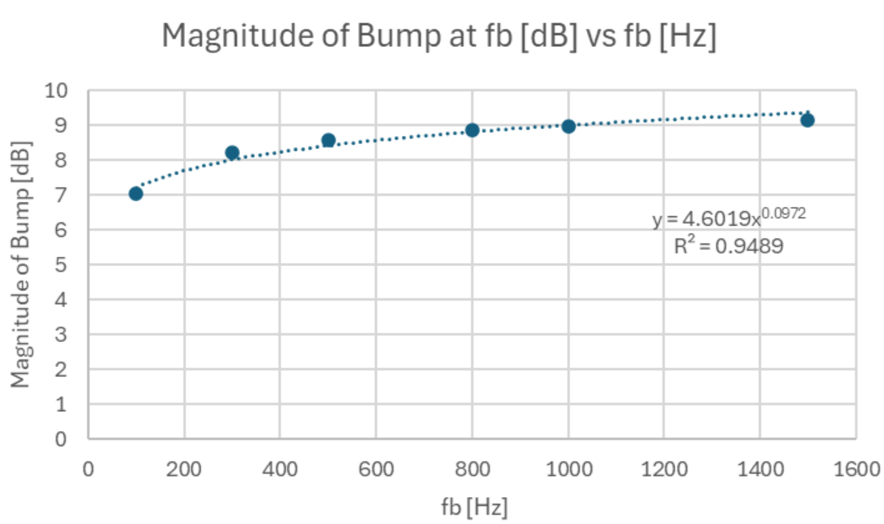
As we can see, the Si function contributes entirely to the drop in magnitude and partially to the bump while the sin^2 contributes to the bump only.

IV.E) Plotting Servo Bump height as a function of f_b:

Imagine we have a step function noise source that has a value of $h_A = 1\ Hz$ for $f < f_b$ and a value of $h_B = 10\ Hz$ for $f \geq f_b$. A characteristic of the lineshape of a step function noise is that there are servo bumps at the output frequency values of $|f| = f_b$. These bumps can be characterized with a rough height, *d*, as shown below:

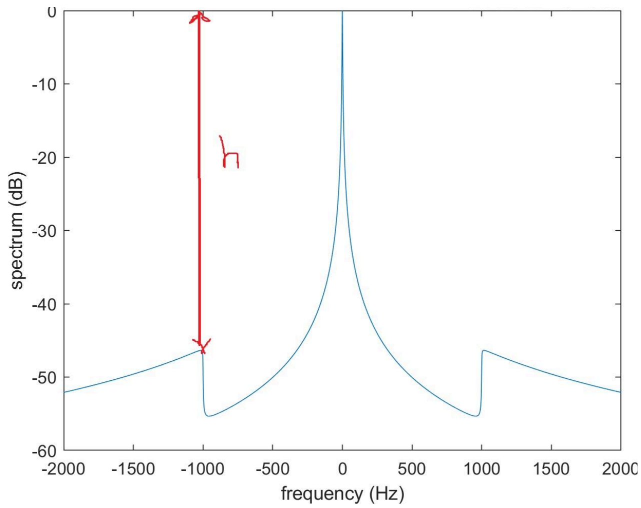


How does changing f_b effect *d* ? The graph below shows the relationship between the magnitude of *d* and f_b. As we can see, increasing f_b indeed does increase *d*.

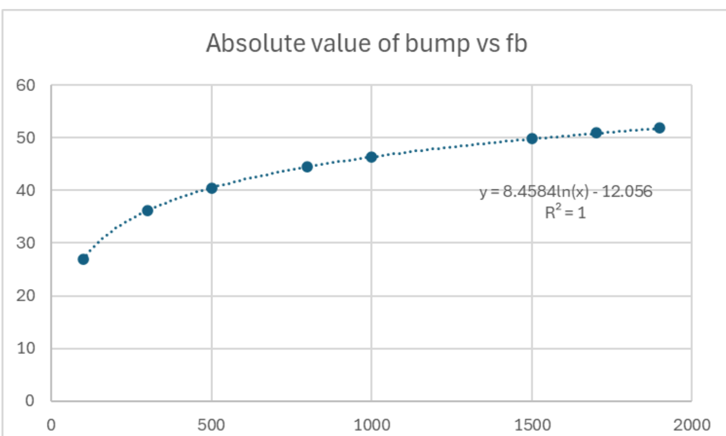


IV.F) Plotting Spectrum height of servo bump as a function of f_b:

Instead of plotting the height of each servo bump relative to the lineshape right before the bump as in the previous section, what if we plotted the spectrum level, called *h*, of the servo bump as a function of f_b. Such an image is shown below:

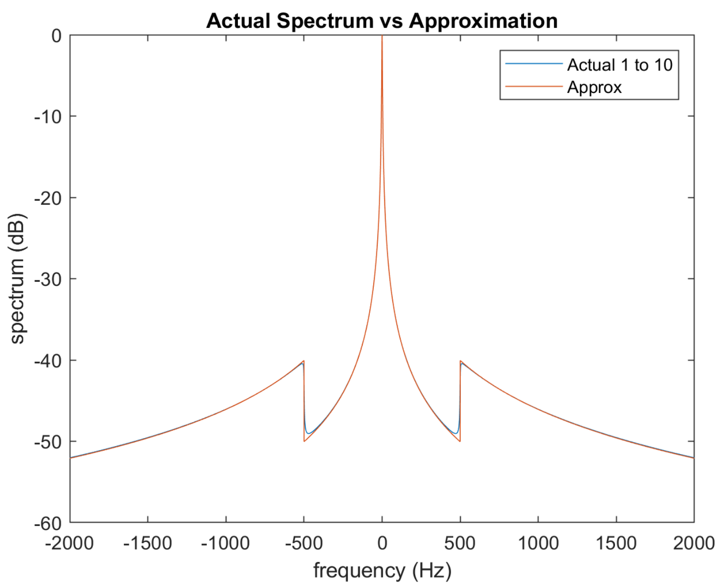


If we were to plot the value of *h* as a function of f_b, we obtain the following graph:

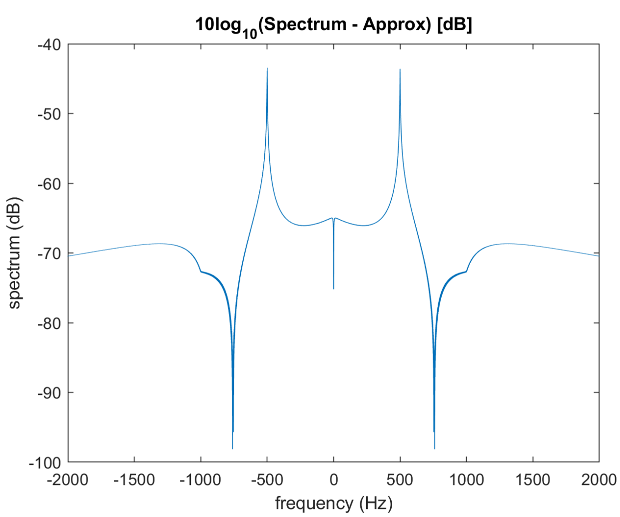


IV.G) Approximating lineshape for step function noise:

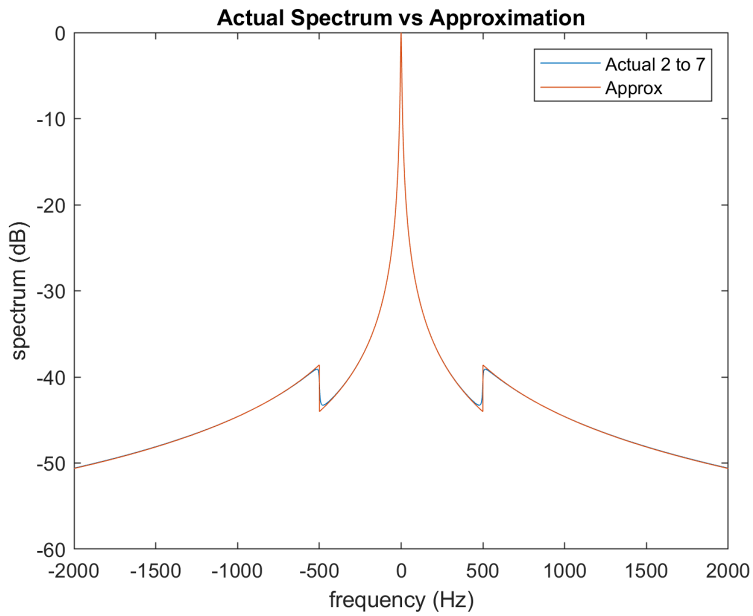
Imagine we have a step noise PSD that starts out at $h_A = 1\ Hz$ and then jumps to $h_B = 10\ Hz$ at a bandwidth frequency of $f_b = 500\ Hz$. We can approximate the lineshape of the PSD as follows. Find the lorentzian corresponding to a constant white noise value of $h_A = 1\ Hz$. then multiply said lorentzian by a factor of 10 (or h_B/h_A) for frequencies, $|f| > f_b$. An example of this process is shown below:



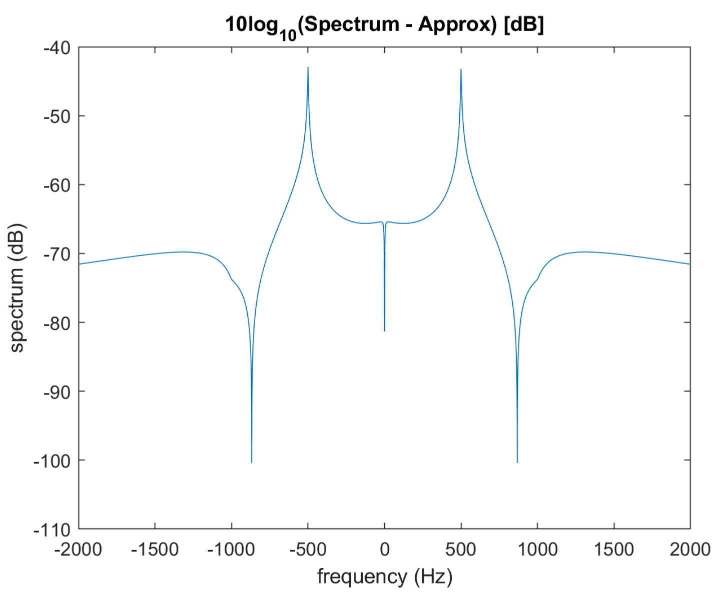
If we graph the error by taking $10log_{10}$ of the actual analytical spectrum (called "Spectrum") and subtract from it our approximation (called "Approx") we get the following:



If we try the same algorithm but with a step function that starts out at $h_A = 2\ Hz$ and then jumps to $h_B = 7\ Hz$ at a bandwidth frequency of $f_b = 500\ Hz$, we get the following result:



The corresponding error becomes:



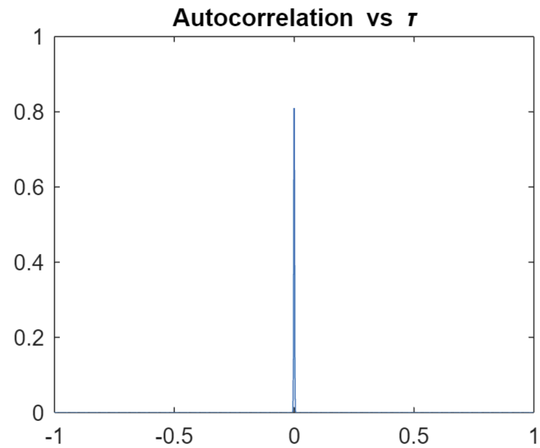
Analyzing PSDs of 2nd paper:

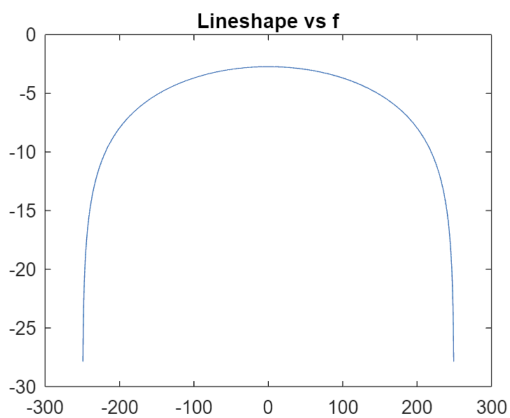
The next paper we will work with is: [Modulation-free laser stabilization technique using integrated cavity-coupled MachZehnder interferometer](). The purpose of the next task is to calculate the lineshape of experimental data in Fig 5 (b, c, d) of the paper.

Using the code below:

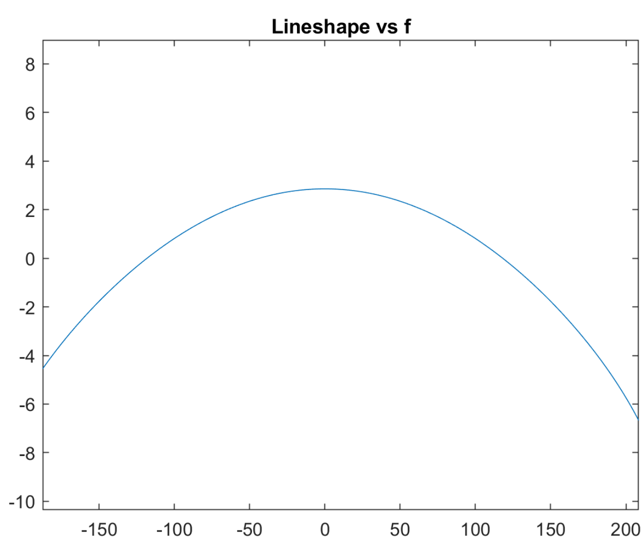
```
% ClosedLoop is already defined
freqs = ClosedLoop(:, 1);
S_dv = ClosedLoop(:, 2);
tau_vals = linspace(-1, 1, 1000);

S_dv_in = @(f) interp1(freqs, S_dv, f, 'linear');

Gamma_E = @(T) exp(-2 * integral(@(f) S_dv_in(f) .* (sin(pi * f * T) ./ f).^2, 1,
10000, 'ArrayValued', true));
autocorrelation = arrayfun(Gamma_E, tau_vals);

plot(tau_vals, autocorrelation);
title("Autocorrelation vs \tau");

lineshape = abs(fftshift(fft(ifftshift(autocorrelation))));

% Freqs corresponding to lineshape
n = length(tau_vals);
delta_tau = tau_vals(2) - tau_vals(1);
freqs_lineshape = (-n/2:n/2-1) / (n * delta_tau);


plot(freqs_lineshape, 10*log10(lineshape));
title("Lineshape vs f");
```

We get the following graphs:





If we zoom in on the final graph:



Analyzing the code section by section:

Let's analyze the code section by section.

The first section involves creating the frequency input variable, as well as the corresponding noise spectral density (which is a function of frequency). The `tau_vals` variable is used for the autocorrelation function and covers the range for which the autocorrelation function, $\Gamma_E(\tau)$ will be defined.

```
freqs = ClosedLoop(:, 1);
S_dv = ClosedLoop(:, 2);
tau_vals = linspace(-1, 1, 1000);
```

The next line is important as what it does is it takes the raw data of $S_{\delta\nu}$ vs f and then what it does is perform a linear interpolation of $S_{\delta\nu}$ in between points. The next thing that it does is to make $S_{\delta\nu}$ *a function* of f. What this allows us to do is to find the value of $S_{\delta\nu}$ for *any* value of f, something that will come in handy later:

```
S_dv_in = @(f) interp1(freqs, S_dv, f, 'linear');
```

The code referenced below allows us to create the autocorrelation function, $\Gamma_E(\tau)$, as a function of τ. As a reminder the autocorrelation function is shown below:

$$\Gamma_E(\tau) \;=\; E_0^2 e^{i2\pi\nu_0\tau} exp(-\,2 * \int_0^\infty S_{\delta\nu}(f) * \frac{sin^2(\pi f\tau)}{f^2} df)$$

The code below evaluates the function above as a function of τ, which I symbolize in MATLAB as 'T'. to avoid messy integration, I integrate from 1 to 20.

```
Gamma_E = @(T) exp(-2 * integral(@(f) S_dv_in(f) .* (sin(pi * f * T) ./ f).^2, 1,
10000, 'ArrayValued', true));
```

The next few lines are important as it defines the autocorrelation function using an array,and then plots it. The first line of code in the segment below defines $\Gamma_E(\tau)$ (represented by `Gamma_E` in the segment) by evaluating our function `Gamma_E` for every value of τ defined by the variable `tau_vals`.

```
autocorrelation = arrayfun(Gamma_E, tau_vals);

plot(tau_vals, autocorrelation);
title("Autocorrelation vs \tau");
```

The next line is necessary to calculate the lineshape $S_E(\nu)$ by taking the fourier transform of the autocorrelation function $\Gamma_E(\tau)$. The fftshift is merely to have the lineshape symmetric about the vertical axis:

```
lineshape = abs(fftshift(fft(ifftshift(autocorrelation))));
```

We now have to define the frequency axis for our lineshape. Lastly, we plot the line shape:

```
% Freqs corresponding to lineshape
n = length(tau_vals);
delta_tau = tau_vals(2) - tau_vals(1);
freqs_lineshape = (-n/2:n/2-1) / (n * delta_tau);


plot(freqs_lineshape, 10*log10(lineshape));
title("Lineshape vs f");
```