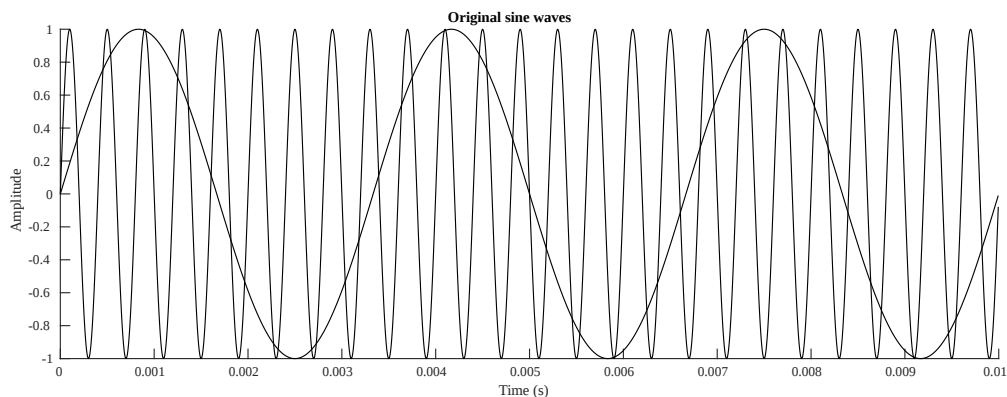


Filtering in MATLAB

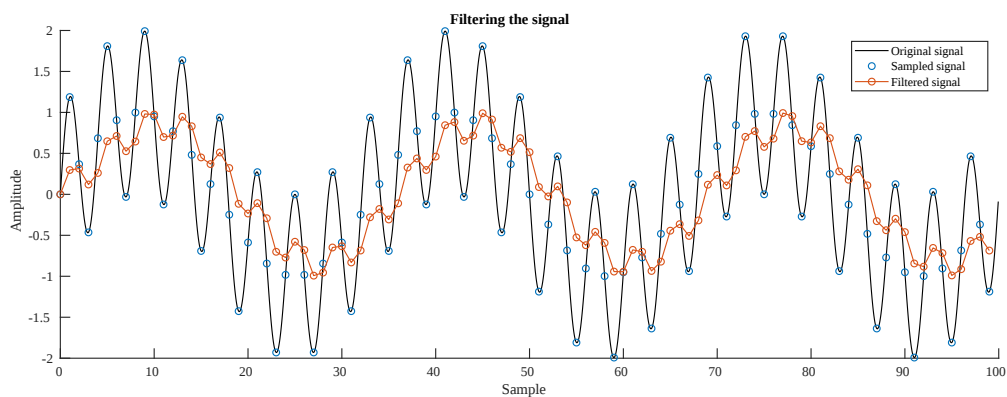
Pieter P

We can use MATLAB to visualize the effects of the filter. The scripts used can be found at the bottom of the page.

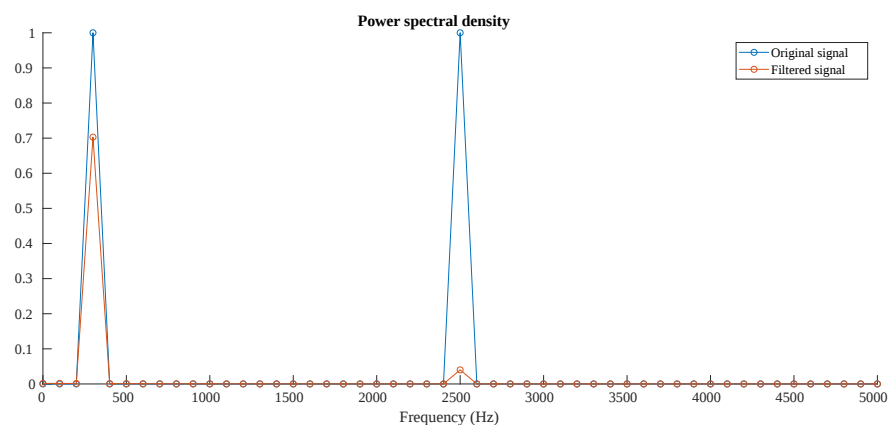
First, we generate a test signal that consists of two sine waves.



Then we apply the filter to it and plot the result. You can clearly see how the high-frequency sine wave is attenuated. Also note the phase shift between the original and the filtered signal: the red curve is delayed slightly, it is shifted to the right.



Finally, we can apply a fast fourier transform to inspect the frequency content.



Attenuation of first sine wave (30 Hz) = -1.53 dB
Attenuation of second sine wave (250 Hz) = -13.97 dB

You can hear the difference for yourself:

Audio

It can be used on music as well:

Code

Sine Wave Code

```
1 %% Visualization
2
3 close all; % Close all open figures
4
5 alpha = 0.25; % Filter factor of 1/4
6
7 f_s = 10000; % 10 kHz sample frequency
8 f_1 = 300; % First sine wave with a frequency of 300 Hz
9 f_2 = 2500; % Second sine wave with a frequency of 2.5 kHz
10
11 samples = 100; % Calculate/plot 100 samples
12 n = linspace(0,samples-1,samples); % Generate a vector with sample numbers
13 t = n / f_s; % Generate a vector with time
14
15 sine_1 = sin(2*pi*f_1*t); % Calculate the (sampled) sine waves
16 sine_2 = sin(2*pi*f_2*t);
17 signal = (sine_1 + sine_2); % Mix the two sine waves together
18
19 b = alpha; % Coefficients of the numerator of the transfer function
20 a = [1,-(1-alpha)]; % Coefficients of the denominator of the transfer function
21 filtered = filter(b,a,signal); % Filter the signal
22
23 oversample_continuous = 20; % Create a version with ten times more samples
24 % to display the smooth, continuous signal
25 samples_continuous = oversample_continuous * samples;
26 n_continuous = linspace(0, samples_continuous-1,samples_continuous) / oversample_continuous;
27 t_continuous = n_continuous / f_s;
28 sine_1_continuous = sin(2*pi*f_1*t_continuous);
29 sine_2_continuous = sin(2*pi*f_2*t_continuous);
30 signal_continuous = (sine_1_continuous + sine_2_continuous);
31
32 % Plot the two original sine waves
33 figure('pos',[0,0,1200,400]);
34 hold on;
35 plot(t_continuous, sine_1_continuous, 'k');
36 plot(t_continuous, sine_2_continuous, 'k');
37 title('Original sine waves');
38 xlabel('Time (s)');
39 ylabel('Amplitude');
40
41 % Plot the continuous signal, the sampled version and the filtered output
42 figure('pos',[0,0,1200,400]);
43 hold on;
44 plot(n_continuous, signal_continuous, 'k');
45 plot(n, signal, 'o');
46 plot(n, filtered, '-o');
47 title('Filtering the signal');
48 xlabel('Sample');
49 ylabel('Amplitude');
50 legend('Original signal','Sampled signal','Filtered signal');
51
52 % Apply a fast fourier transform and plot the spectra of the
53 % original signal and of the filtered output
54 figure('pos',[0,0,1000,400]);
55 hold on;
56 f = linspace(0,samples-1,samples)*f_s/samples;
57 original_spectrum = (abs(fft(signal))*2/samples).^2;
58 filtered_spectrum = (abs(fft(filtered))*2/samples).^2;
59 plot(f(1:1+samples/2),original_spectrum(1:1+samples/2),'-o');
60 plot(f(1:1+samples/2),filtered_spectrum(1:1+samples/2),'-o');
61 title('Power spectral density');
62 xlabel('Frequency (Hz)');
63 legend('Original signal','Filtered signal');
64
65 % Calculate the attenuation of the two sine waves
66 f_1_index = f_1*samples/f_s+1;
67 A_1 = filtered_spectrum(f_1_index) / original_spectrum(f_1_index);
68 A_1_dB = 10*log10(A_1);
69 fprintf('Attenuation of first sine wave (%.0f Hz) = %.02f dB\n', f_1, A_1_dB);
70
71 f_2_index = f_2*samples/f_s+1;
72 A_2 = filtered_spectrum(f_2_index) / original_spectrum(f_2_index);
73 A_2_dB = 10*log10(A_2);
74 fprintf('Attenuation of second sine wave (%.0f Hz) = %.02f dB\n', f_2, A_2_dB);
75
76 % Open the filter visualization tool
77 fvtool(b,a,'Fs',f_s);
78
79 %% WAV export
80
81 samples = f_s*2; % 2 seconds of audio
82 n = linspace(0,samples-1,samples); % Generate a vector with sample numbers
83 t = n / f_s; % Generate a vector with time
84
85 sine_1 = sin(2*pi*f_1*t); % Calculate the (sampled) sine waves
86 sine_2 = sin(2*pi*f_2*t);
87 signal = (sine_1 + sine_2)/2; % Mix the two sine waves together
88
89 filtered = filter(alpha,[1,-(1-alpha)],signal); % Filter the signal
90
91 audiowrite('original.wav',signal,f_s); % Export as audio
92 audiowrite('filtered.wav',filtered,f_s);
```

Audio Code

```
1 [signal,f_s] = audioread('telegraph_road_original.wav');
2
3 alpha = 0.25;    % Filter factor of 1/4
4
5 b = alpha;        % Coefficients of the numerator of the transfer function
6 a = [1,-(1-alpha)]; % Coefficients of the denominator of the transfer function
7 filtered = filter(b,a,signal); % Filter the signal
8
9 audiowrite('telegraph_road_filtered.wav',filtered,f_s);
```