

```

close all; clear all; clc;

% Parameters
Fs= 500e6; % Sample rate (Hz) > 2*100 Hz, my highest frequency
T=20e-6; %Signal duration (seconds)
N= round (Fs*T); %Number of samples
t=(0:N-1).'/Fs; % Time Vector (column)

% Tones between 0-100MHz
f=[10, 37, 85]*10^6; %Hz
A= [1.0,0.7,0.5]; % amplitude
phi= 2*pi*rand(size(f)); % random phase

% Real band-limited signal (sum of cosines)
x=A(1)*cos(2*pi*f(1)*t+phi(1))+A(2)*cos(2*pi*f(2)*t+phi(2))
+A(3)*cos(2*pi*f(3)*t+phi(3));

% We are adding random noise to x (SNR)
% SNR is ratio of avg power of signal (Px) to avg power of the noise (Pn)

% Target SNR in dB relative to highest power
snr=20;
Px=mean(x.^2); % signal power
Pn=Px/10^(snr/10); %noise power
noise=sqrt(Pn)*rand(size(x)); % AWGN (real)

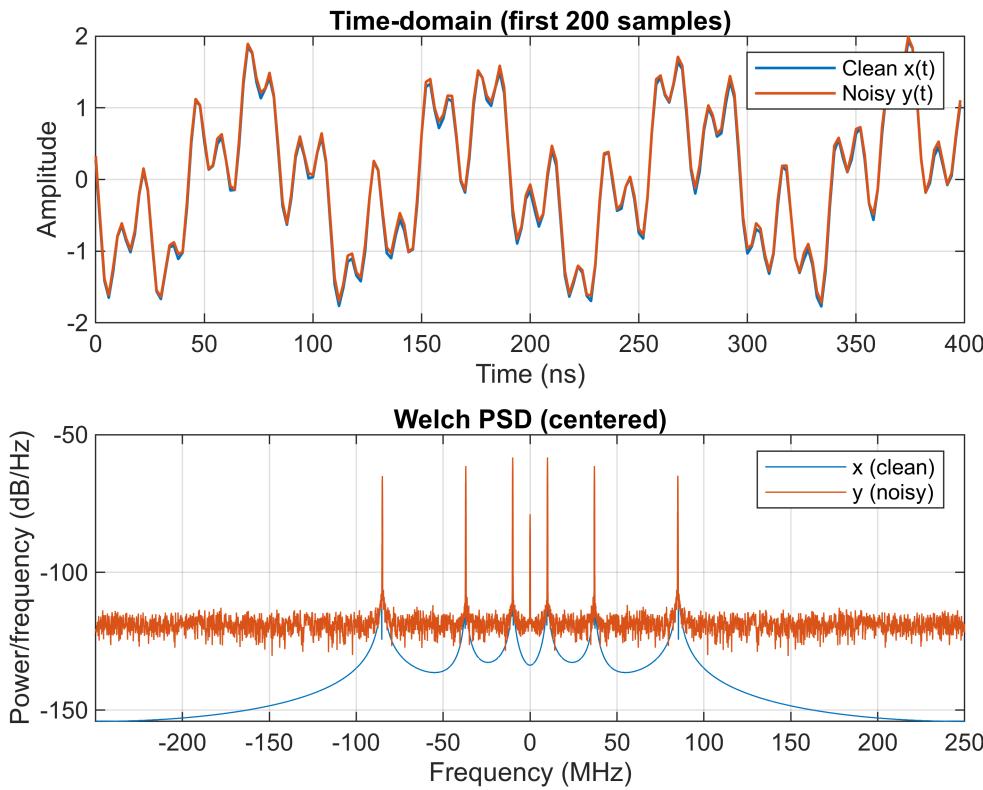
y=x+noise;

%% Quick looks
figure;
subplot(2,1,1);
plot(t(1:200)/1e-9, x(1:200), 'LineWidth',1); hold on;
plot(t(1:200)/1e-9, y(1:200), 'LineWidth',1);
xlabel('Time (ns)'); ylabel('Amplitude');
legend('Clean x(t)', 'Noisy y(t)'), grid on; title('Time-domain (first 200 samples)');

%% Using pwelch that estimates Power Spectral Density (PSD) of a signal
% This is how a signal's power is distributed across frequency

subplot(2,1,2);
pwelch(x,4096,2048,4096, Fs,'centered'); hold on;
pwelch(y,4096,2048,4096, Fs,'centered');
legend('x (clean)', 'y (noisy)'), title('Welch PSD (centered)'), grid on;

```



```

%% What happens to your tones (10, 37, 85 MHz):
% Multiply by cos(10MHz)

%Dif: |f-flo|: 0, 27, 75 MHz
% Sum: |f+flo|: 20, 47, 95 MHz (each gets 1/2 ampl. unless you scale by 2
% for unity conversion gain

flo=10e6; % LO=10MHz local oscillator

%Real mixing (double-sideband)
y_RF= 2*y.*cos(2*pi*flo*t); % 2x for the ~0 dB conversion gain
figure;
%pwelch(y,4096,2048,4096, Fs,'centered'); hold on;
%pwelch(y_RF,4096,2048,4096, Fs,'centered');

% I altered the above two lines so that I can more easily show the LO-value
% on the plot.

[PSD_y, fvec]      = pwelch(y,        4096, 2048, 4096, Fs, 'centered');
[PSD_yRF, fvecRF] = pwelch(y_RF,    4096, 2048, 4096, Fs, 'centered');

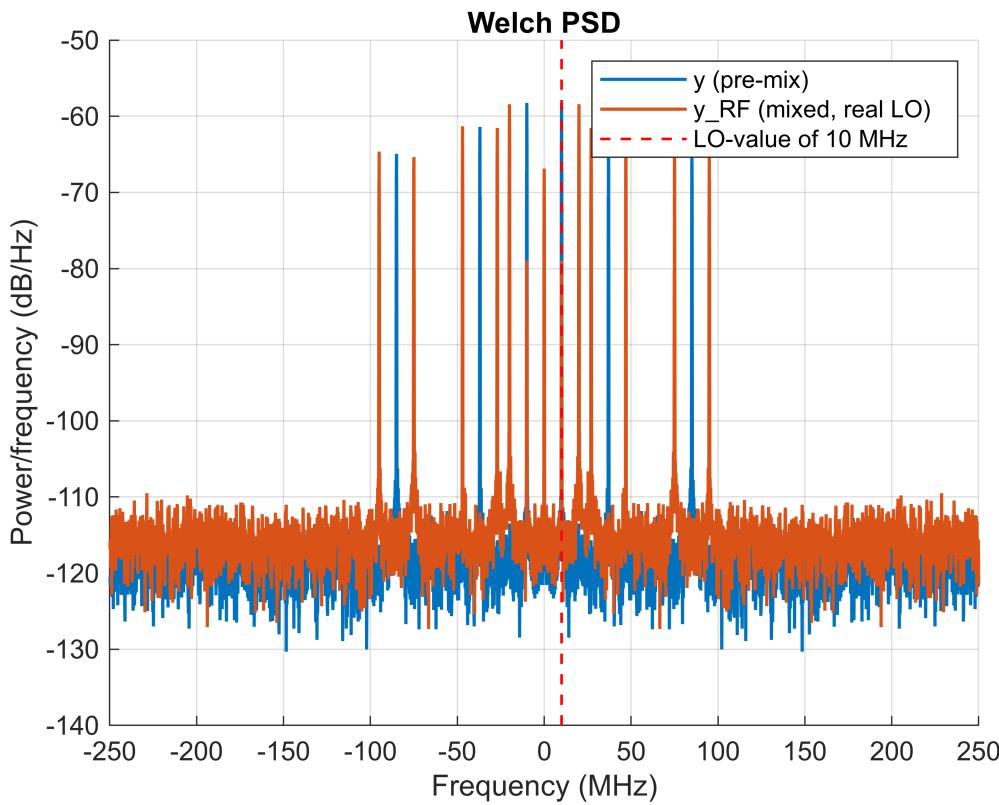
figure; hold on; grid on;
plot(fvec/1e6, 10*log10(PSD_y), 'LineWidth', 1.2);
plot(fvecRF/1e6, 10*log10(PSD_yRF), 'LineWidth', 1.2);
plot([flo/1e6 flo/1e6], ylim, 'r--', 'LineWidth',1.0); % Shows the LO frequency

```

```

xlabel('Frequency (MHz)'); ylabel('Power/frequency (dB/Hz)')
xlim([-250 250]);
legend('y (pre-mix)', 'y_RF (mixed, real LO)', sprintf('LO-value of %d MHz', flo/
1e6));
title('Welch PSD'); grid on;

```



```

%% 2MHz Low Pass Filter that should erase everything beyond 2MHz
Fc=2e6;
z_lp=lowpass(y_RF,Fc,Fs);

%% Downsample by M
% Bandwidth is narrowed by factor of M
M=50;
Fs_ds= Fs/M;
z_ds= downsample(z_lp,M);
t_ds = (0:numel(z_ds)-1).'/Fs_ds;

% QUICK LOOK
figure;
subplot(2,1,1);
ns=min(5000, numel(z_ds));
plot(t_ds(1:ns)/1e-6, z_ds(1:ns), 'LineWidth',1);
xlabel('Time (\mu s)'); ylabel('Amplitude'); grid on;

```

```

title('Baseband (real mix) after LPF + downsample');

subplot(2,1,2);
pwelch(y,[],[],[], Fs, 'centered'); hold on;
pwelch(z_lp, [], [], [], Fs, 'centered');
pwelch(z_ds, [], [], [], Fs_ds, 'centered');
legend('Input y', 'After LPF', 'After LPF+DS'); title('PSD progression'); grid on;

```

