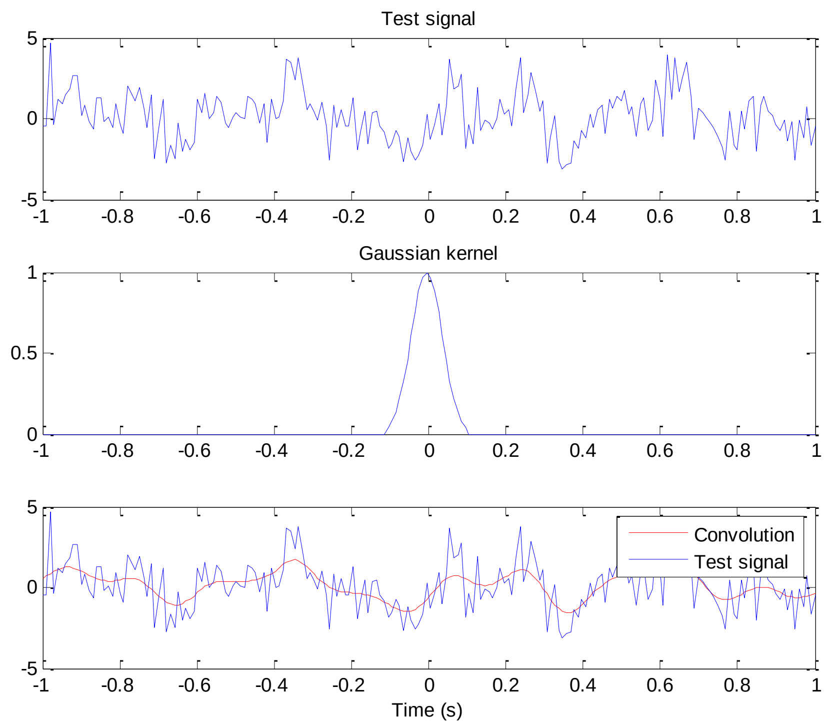
BME 295: Analysis of Neural Time Series Data
Homework 2: Dot product, convolution, and FFT

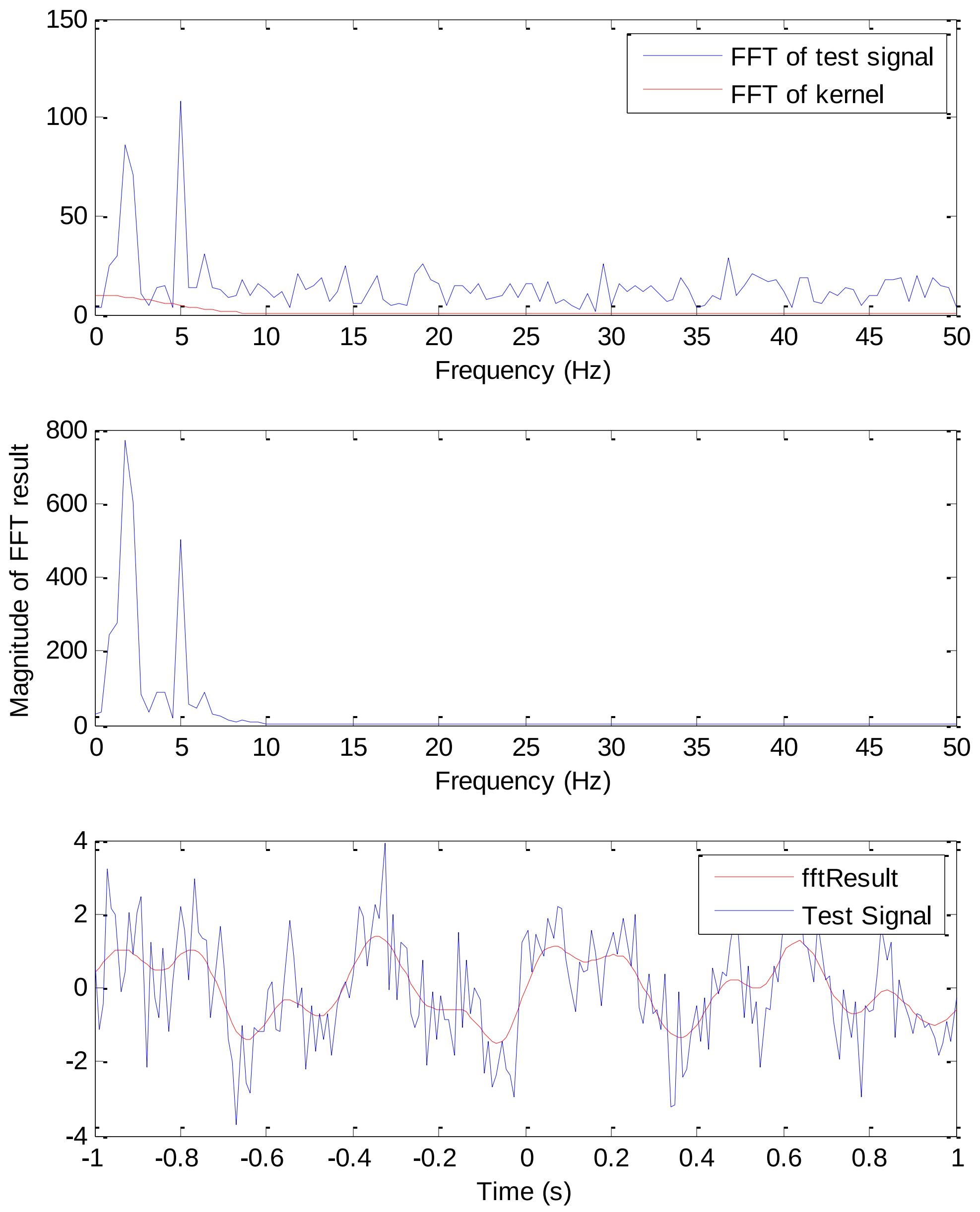
For this homework assignment, we will again analyze EEG data from a P300 experiment performed by Ulrich Hoffman, Jean-Marc Vesin, Touradj Ebrahimi, and Karen Diserens in Lausanne, Switzerland.

*** Please label the axes of all plots and use appropriate units, i.e. Hertz, seconds, etc. ***

1. **The convolution theorem.** In class, we learned that time-domain convolution can act as a low-pass filter. Here, you will illustrate this by creating a plot similar to Figures 11.11 and 11.12 in the book (these are also shown on slides 25 and 26 of lecture 4). You should follow this general procedure:
   a. Create a test signal that is a sum of multiple sine waves (at least one low frequency and one high frequency) and Gaussian noise. Plot this signal versus time (in seconds).
   b. Create a Gaussian kernel using the "gausswin" function in MATLAB. Plot the kernel versus time (in seconds).
   c. Use the function "conv" to convolve the two signals, and use the extra options of the function to automatically trim the result. Scale (divide) the result of the convolution by the sum of the kernel. Plot this result and overlay the original test signal in a different color.



   d. What effect did the convolution have on the frequency content of your test signal? The convolution with a Gaussian kernel acts like a low-pass filter.
   e. Take the FFT of each signal, multiply the resulting spectra, and take the inverse FFT. Plot all three steps, and overlay the original test signal on top of the IFFT result.
      HINT: You will need to choose the length of the FFT to match the convolution. This is a bit tricky. See Section 11.11 in your book for help.



f. If your code is correct, the results of parts (c) and (d) should look the same. What theorem does this demonstrate? The convolution theorem

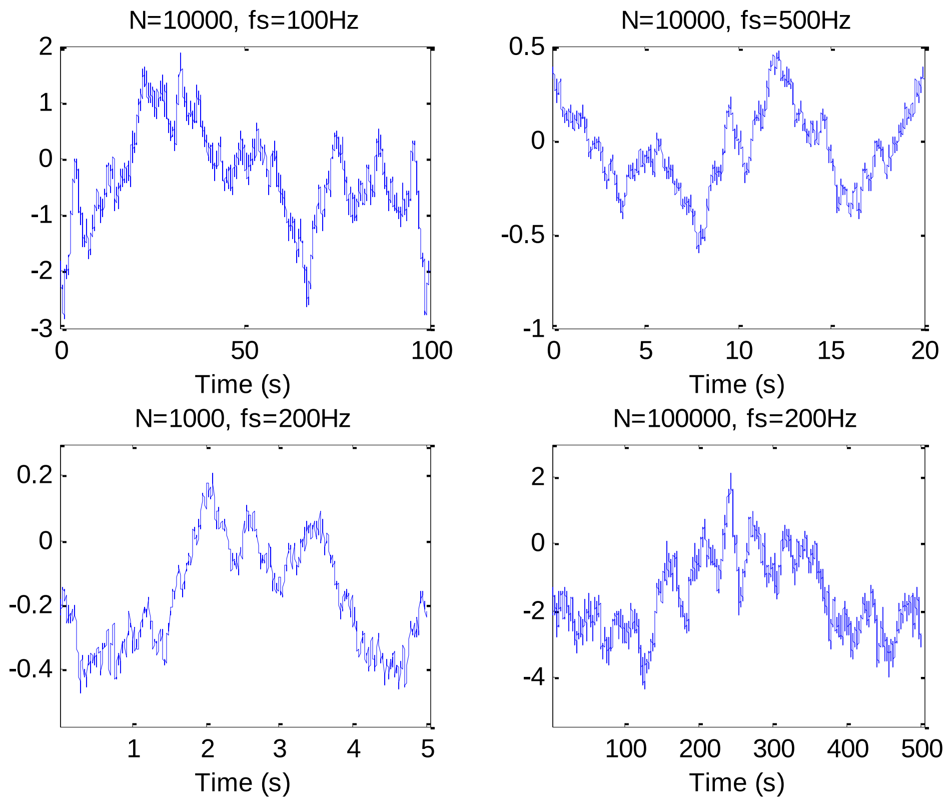
2. **Creating simulated EEG data**. Write a function to create simulated EEG data with a 1/f power spectrum:

   Inputs: the desired length of the signal and the sampling frequency

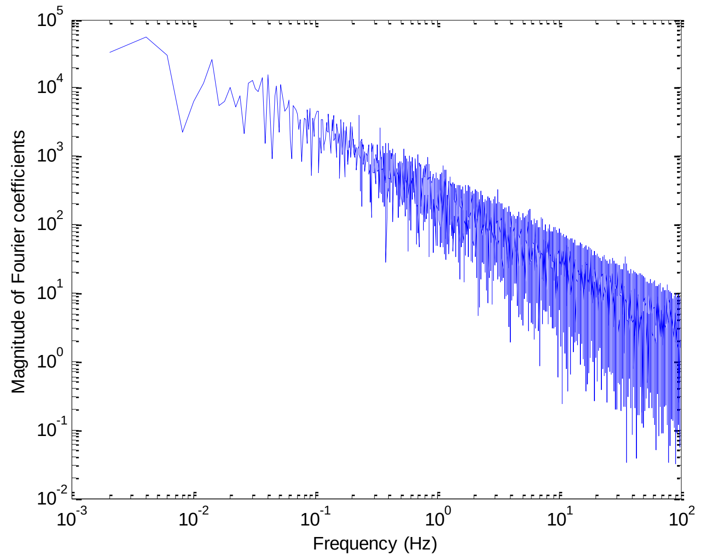
   Outputs: a time vector for plotting and the simulated EEG signal

The process is very similar to what you did in #1(e). You may want to use the related lecture slides as a guide.

a. Plot several examples of simulated EEG at different lengths and sampling frequencies.



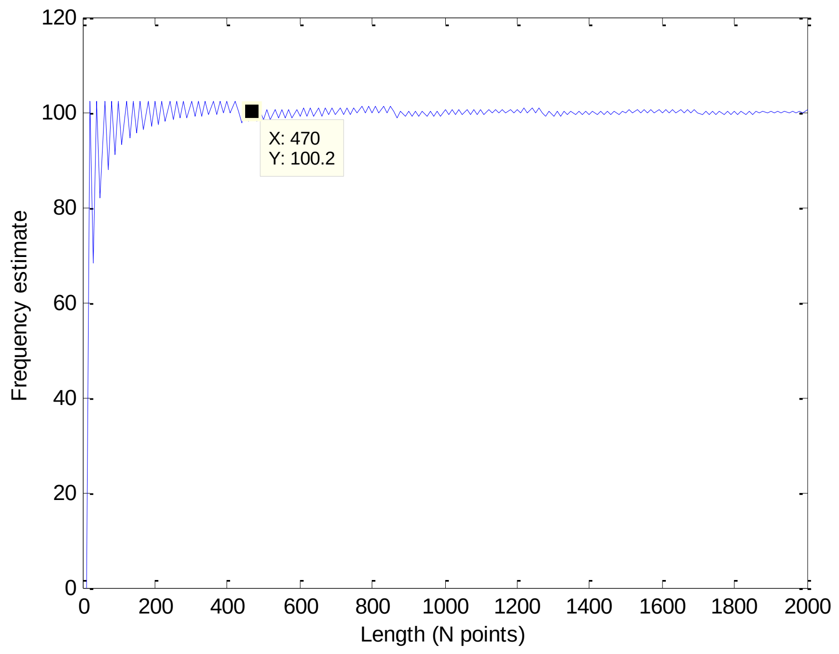
b. Plot the Fourier coefficients of one of the signals on a log-log plot and verify that it looks approximately linear.



3. Effect of length N on the Fourier transform

a. We learned in class that the frequency resolution of the Fourier transform depends on the length of the signal, N. This means that, for very short segments of data, the Fourier transform will provide an inaccurate estimate of the signal frequency. Test this using the following procedure:

1. Create 100 Hz sine waves of varying length, N. Use a sampling frequency of 2048 Hz to match the EEG data we have been analyzing.
2. For each value of N, take the Fourier transform of the sine wave and identify the frequency at which the magnitude of the Fourier coefficients reach their maximum value (this is the best estimate of the signal frequency).
3. Create a plot of the estimated frequency (y-axis) versus N (x-axis).



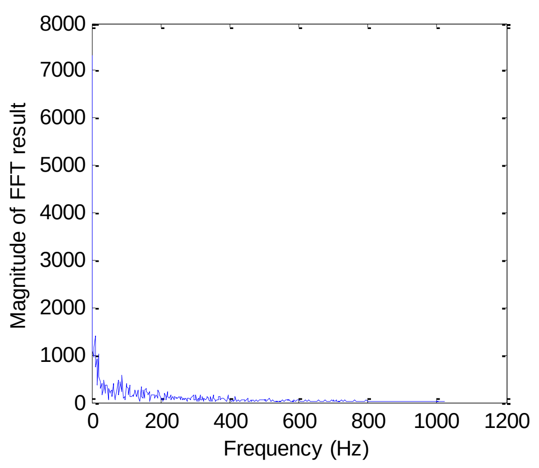
b. At what signal length, N, is the estimate of frequency reliably within +/- 2Hz of the actual frequency (100 Hz)? What is this length in seconds? N = 470 = 0.229 seconds

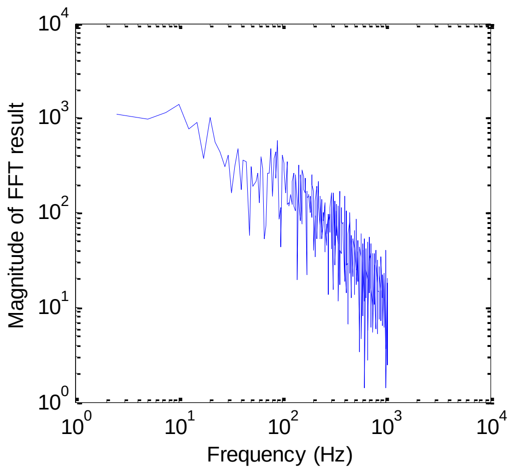
c. Are the trials in the P300 data set long enough to provide an accurate estimate of the power spectrum using the Fourier transform? Yes, the trials are 0.4 seconds

4. **Analysis of P300 dataset.** Now we will use the Fourier transform to analyze the frequency characteristics of the P300 EEG dataset. Parts (a) through (d) will all use dataset "sub8_sess4_1" and data from channel 1.

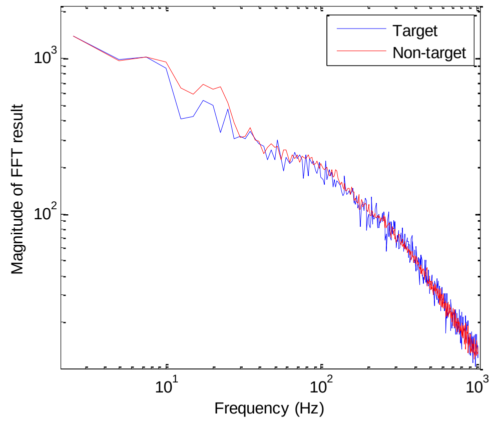
a. Load dataset "sub8_sess4_1.mat." Use your "extractAllTrials" function to extract the trials from channel 1 starting at the time of the stimulus (t=0) and ending 400ms later. Take the Fourier transform of each trial using NFFT = the length of the trial. Note that "fft" can take a matrix input, so you don't need to use a for loop.

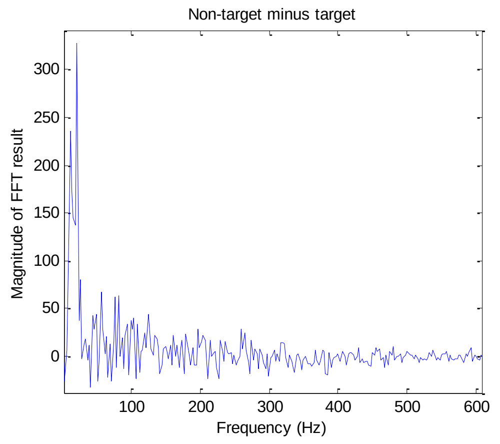
Plot the power spectrum of trial 1 on both linear and log-log plots. Does this EEG data have an approximately 1/f distribution? Yes:





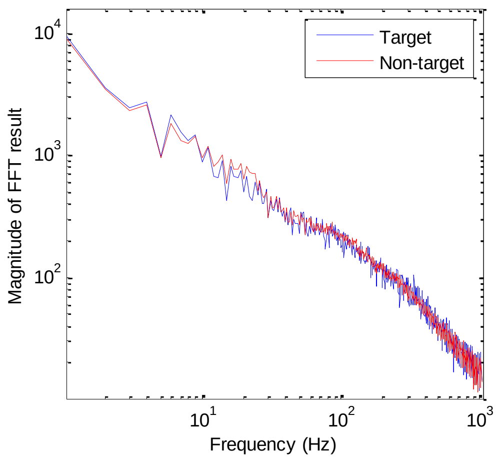
b. Divide the trials into "targets" and "non-targets," and calculate the mean power spectrum of each group. Plot both mean power spectra on the same log-log plot. In a second subplot, plot the difference between the power spectra as a function of frequency. Over what frequency range do you see differences in the power spectrum between target and non-target trials? At which frequency does the largest difference occur? The largest difference occurs at 22.48 Hz, and over the range 10-23Hz:

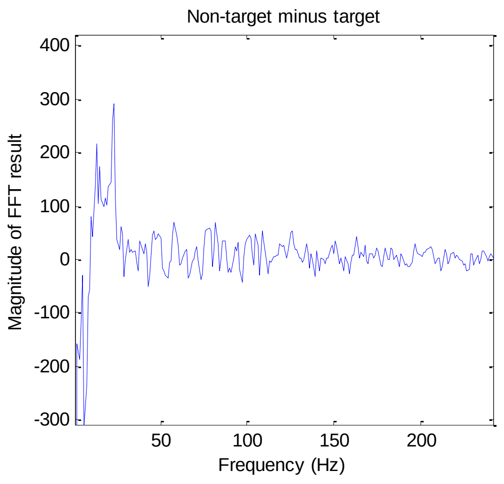




We expect that if the FFT is working perfectly, every frequency will have the same maximum Fourier coefficient. You should see that this is the case, with the exception of 0 and 1024Hz.

c. Recall that zero-padding the data can both increase the frequency resolution of the FFT and reduce computation time. Use NFFT = 2048 and repeat your FFT analysis from part (b). Again, plot the power spectra for targets and non-targets, and then plot the difference between the two. Over what range do you see a difference in the power spectra, and at what frequency does the maximum difference occur?
The zero-padding increases the *resolution* of the FFT, but does not really change the results. We see the largest difference in power over 10-23 Hz, with the maximum difference occurring at 23 Hz.


Target
Non-target
10^4
10^3
10^2
10^1
10^2
10^3
Magnitude of FFT result
Frequency (Hz)


Non-target minus target
400
300
200
100
0
-100
-200
-300
50
100
150
200
Magnitude of FFT result
Frequency (Hz)

```
function [time, simEEG] = simulateEEG(N,fs)
% Create simulated EEG data with a 1/f frequency distribution.
%
% INPUTS:
%    N = length of simulated EEG signal
%    fs = sampling frequency
%
% OUTPUT:
%    simEEG = simulated EEG data

time = (1:N)/fs;  % Define time vector
x = randn(N,1);  % signal starts as Gaussian noise
xfft = fft(x);     % take FFT
fVec = linspace(0, fs/2, N/2+1);  % Define frequency vector
fVec = fVec(2:end);  % Remove zero because we divide by fVec
xfft2 = xfft(1:(N/2))./fVec';   % Multiply power spectrum by 1/f
xf_filtered = [xfft2; flipud(xfft2)];  % Add coefficients for negative
frequencies
simEEG = real(ifft(xf_filtered'));  % simEEG is real part of IFFT
```

```
% Homework 2
% Beth A. Lopour
%% #1 - The convolution theorem
% 1A
clear all
fs = 100;
N = 200;
t = linspace(-1,1,N);  % time in seconds
x = sin(2*pi*2*t) + sin(2*pi*5*t) + randn(size(t)); % test signal
figure; subplot(311); plot(t,x)
xlabel('Time (s)')

% 1B
kernel = gausswin(21)';  % gaussian kernel, length = 21
kernelToPlot = [zeros(1,89) kernel zeros(1,90)];
subplot(312); plot(t,kernelToPlot)
xlabel('Time (s)')

% 1C
convResult = conv(x,kernel,'same')/sum(kernel);
subplot(313); plot(t,convResult,'r')
hold on; plot(t,x)
legend('Convolution', 'Test signal')
xlabel('Time (s)')

% 1E
NFFT = length(x)+length(kernel)-1;  % Choose NFFT to match convolution
xfft = fft(x, NFFT);
kfft = fft(kernel, NFFT);
fVec = linspace(0, fs/2, NFFT/2+1);
figure; subplot(311);
plot(fVec, abs(xfft(1:(NFFT/2+1))))
hold on; plot(fVec, abs(kfft(1:(NFFT/2+1))), 'r')
xlabel('Frequency (Hz)'); ylabel('Magnitude of FFT result')

subplot(312);
plot(fVec, abs(xfft(1:(NFFT/2+1))).*abs(kfft(1:(NFFT/2+1))))
xlabel('Frequency (Hz)'); ylabel('Magnitude of FFT result')

fftResult = ifft(xfft.*kfft);
fftResult = fftResult(11:210);
subplot(313);
plot(t, fftResult/sum(kernel),'r')
hold on; plot(t, x)
legend('fftResult','Test Signal')
xlabel('Time (s)')

%% #3 Effect of length N on the Fourier transform

N = 10:10:2000;  % length of signal
fs = 2048; % sampling frequency
F = 100; % Hz
fAtMax = zeros(1,length(N));

for i=1:length(N)
```

```matlab
    test = sin(2*pi*F*(1:N(i))/fs);
    testfft = fft(test);
    [~, maxInd] = max(abs(testfft(1:(N(i)/2+1))));
    fVec = linspace(0,fs/2,N(i)/2+1);
    fAtMax(i) = fVec(maxInd);
end

figure; plot(N, fAtMax)
xlabel('Length (N points)'); ylabel('Frequency estimate (Hz)')

%% #4 Analysis of the P300 dataset
% #4A
clear all
load('sub8_sess4_1.mat')
fs = 2048;

[trials, time] = extractAllTrials(data, events, 1, 0, 0.4);
N = size(trials,2);
datafft = fft(trials');
fVec = linspace(0,fs/2, N/2+1);

figure; subplot(121); plot(fVec, abs(datafft(1:(N/2+1),1)))
xlabel('Frequency (Hz)'); ylabel('Magnitude of FFT result')
subplot(122); loglog(fVec, abs(datafft(1:(N/2+1),1)))
xlabel('Frequency (Hz)'); ylabel('Magnitude of FFT result')

% #4B
targetBool = (stimuli==target);  % make a boolean where "1" indicates target
trial
targetFFT = abs(datafft(1:(N/2+1),targetBool));  % target trials are in
columns
nontargetFFT = abs(datafft(1:(N/2+1),~targetBool));  % non-target trials
% figure; subplot(121); loglog(fVec, targetFFT)
% subplot(122); loglog(fVec, nontargetFFT)
figure; subplot(121); loglog(fVec, mean(targetFFT,2));
xlabel('Frequency (Hz)'); ylabel('Magnitude of FFT result')
hold on; loglog(fVec, mean(nontargetFFT,2),'r')
subplot(122); plot(fVec,mean(nontargetFFT,2) - mean(targetFFT,2));
xlabel('Frequency (Hz)'); ylabel('Magnitude of FFT result')
title('Non-target minus target')

% #4C
NFFT = 2048;
datafft2 = fft(trials',NFFT);
fVec2 = linspace(0,fs/2, NFFT/2+1);

targetFFT2 = abs(datafft2(1:(NFFT/2+1),targetBool));  % target trials in cols
nontargetFFT2 = abs(datafft2(1:(NFFT/2+1),~targetBool));  % non-target trials
figure; subplot(121); loglog(fVec2, mean(targetFFT2,2));
xlabel('Frequency (Hz)'); ylabel('Magnitude of FFT result')
hold on; loglog(fVec2, mean(nontargetFFT2,2),'r')
legend('Target','Non-target')
subplot(122); plot(fVec2,mean(nontargetFFT2,2) - mean(targetFFT2,2));
xlabel('Frequency (Hz)'); ylabel('Magnitude of FFT result')
title('Non-target minus target')
```