Jacky Au

HW 2

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).

**Here are 4 separate sine waves on the left and summed with noise on right**

 

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. 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?

**It got rid of all the high-frequency information**

e. Take the FFT of the test signal from part (a) and the Gaussian kernel from part (b), multiply the two spectra together, 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 (e) should look the same. What theorem does this demonstrate?

**The convolution theorem, which states that multiplication in the frequency domain is the same as convolution in the time domain. This would explain why multiplying the spectra together gives me the same result as using the conv command in 1c.**

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. Use the MATLAB function loglog to plot the Fourier coefficients of one of the signals on a log-log plot and verify that it looks approximately linear.

**I verify that it looks approximately linear**

****

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?

**I tested every length, n, between 100 and 3000. Visually, it looks like you would need just over 500 (~600) samples in order to achieve the desired reliability. With a sampling rate of 2048, this translates to 600/2048 seconds (293 milliseconds).**

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, because each trial is 400 milliseconds long**

4.

a. Load dataset “sub8\_sess4\_1.mat.” Use the “extractAllTrials” function to extract the trials from channel 1 starting at the time of the stimulus (t=0) and ending 400ms later. The correct solution for the “extractAllTrials” function has been posted to the course website; it is very strongly recommended that you download and use this function, in case there are errors in the one you wrote.

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?

**Below are first the linear plot and then the log-log plot. On first impression, it does appear to have a 1/f distribution. However, if you only look at the biologically plausible range (loosely defined) below 100 Hz, then I don’t think it does anymore.**





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?

**I see power spectra differences mostly below 25 Hz, in favor of nontarget trials having greater power. The greatest difference is at around 22.5 Hz, after excluding the frequency of 0 Hz, which I think just captures the DC offset (or something like that?), or else other irrelevant information.**



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?

**Again, most of the differences seem to exist below 25 Hz, so the range is pretty similar. The greatest differences are observed first at 0 Hz, and then at 1 Hz, but I’m not sure either of those frequencies are meaningful? The third greatest difference is at 6 Hz, this time in favor of Target trials.**

****

**Matlab Script**

close all

clear

%Question 1

%Create sine waves with arbitrary parameters

amp=[2 3 3 4];

sr=1000;

time=0:1/sr:2;

freq=[2 5 10 17];

phase=[pi/2 pi/4 pi/8 pi/16]

for i=1:4

sines(i,:)=amp(i).\*sin(2\*pi\*freq(i).\*time + phase(i));

subplot(4,1,i)

plot(time,sines(i,:))

end

figure

ComSines=sum(sines+randn(size(sines)));%sum together and add noise

plot(time,ComSines)

%1b - Create Gaussian Kernel

kernel=gausswin(101); kernel=kernel'

figure

subplot(2,1,1);plot(time,ComSines);title('summed sine waves');

subplot(2,1,2);plot((1:length(kernel))./1000,kernel);xlabel('time(s)');axis([0 2 0 1]);title('kernel'); %plot kernel below summed sines

%1c - convolve signals

figure

convsig=conv(ComSines,kernel,'same'); convsig=convsig/sum(kernel);

plot(time,convsig,'LineWidth',5);;hold on;plot(time,ComSines,'r','linewidth',.5);legend(gca,'Convolved','Original')

%1e - take inverse fft

FFTlength=length(ComSines)+length(kernel)-1;

fft\_ComSines=fft(ComSines,FFTlength);

fft\_kernel=fft(kernel,FFTlength);

fft\_mult=fft\_ComSines.\*fft\_kernel;

inverse=ifft(fft\_mult); inverse=inverse/sum(kernel);

inverse=inverse(ceil(length(kernel)/2):end-floor(length(kernel)/2));

subplot(221);plot(fft\_ComSines);title('FFT of test signal');

subplot(222);plot(fft\_kernel);title('FFT of kernel');

subplot(223);plot(fft\_mult);title('multiply spectra together');

subplot(224);plot(time,inverse);title('inverse fourier');

%Question 2- Plot simulated EEGs

[time1, EEG1]=SimEEG(5000,500);[time2,EEG2]=SimEEG(1000,250);[time3,EEG3]=SimEEG(3000,1000);[time4,EEG4]=SimEEG(5000,2048);

subplot(2,2,1);plot(time1,EEG1)

subplot(2,2,2);plot(time2,EEG2)

subplot(2,2,3);plot(time3,EEG3)

subplot(2,2,4);plot(time4,EEG4)

fft\_EEG1=fft(EEG1); loglog(abs(fft\_EEG1(1:end/2+1))) %plot fourier coefficients on loglog scale

%3a - create sine waves of 100 hz with variable n

amp=1;

sr=2048;

n=[100:3000];

freq=100;

sines=cell(1,length(n));

ffts=cell(1,length(n));

fVecs=cell(1,length(n));

times=cell(1,length(n));

for i=1:length(n)

fVecs{i}=linspace(0,sr/2,round(n(i)/2+1));

times{i}=1/sr:1/sr:n(i)/sr;

sines{i}=amp\*sin(2\*pi\*freq.\*times{i});

%subplot(4,1,i)

plot(times{i},sines{i})

ffts{i}=fft(sines{i}); ffts{i}=abs(ffts{i}(1:round(n(i)/2+1)));

EstFreq(i)=find(ffts{i}==max(ffts{i})); EstFreq(i)=fVecs{i}(EstFreq(i));

end

figure;plot(n,EstFreq);xlabel('Number Samples');ylabel('Freq(Hz)');

%Question 4

clear

load sub8\_sess4\_1.mat

before=0;after=.4;

fs=2048;

nfft=ceil(after\*fs-before\*fs);

Nyquist=fs/2;

NumFreqs=round(nfft/2)+1;

fVec=linspace(0,Nyquist,NumFreqs);

[trials time]=extractAllTrials(data,events,1,before,after);

fft\_t1=fft(trials(1,:),nfft);

plot(fVec,abs(fft\_t1(1:NumFreqs)));xlabel('Freq(Hz)');ylabel('Power');axis([0 500 0 2000]);

loglog(fVec,abs(fft\_t1(1:NumFreqs)));xlabel('Freq(Hz)');ylabel('Power');

%4b

TargetTrials=stimuli==6;

NonTargetTrials=~TargetTrials;

MeanTargets=mean(trials(TargetTrials,:));

MeanNonTargets=mean(trials(NonTargetTrials,:));

fft\_Targets=fft(MeanTargets,nfft);

fft\_NonTargets=fft(MeanNonTargets,nfft);

subplot (211)

loglog(fVec,abs(fft\_Targets(1:NumFreqs)));xlabel('Freq(Hz)');ylabel('Power');

hold on

loglog(fVec,abs(fft\_NonTargets(1:NumFreqs)),'r');xlabel('Freq(Hz)');ylabel('Power');

subplot(212)

Difference=abs(fft\_Targets(1:NumFreqs))- abs(fft\_NonTargets(1:NumFreqs));

plot(fVec,Difference);xlabel('Freq(Hz)');ylabel('Power');axis([0 500 -100 500]);

MaxDiff=find(Difference==max(Difference));

%4c - rerun above code with new nfft

nfft=2048;

**Function for Question 2**

function [time, EEG] = SimEEG(n, sr)

%initialize parameters

Nyquist=sr/2;

NumFreqs=n/2; %for some reason, I'm not including 0?

fVec=linspace(1,Nyquist,NumFreqs);

time=1/sr:1/sr:n/sr;

sig=randn(1,n);

fft\_s=fft(sig);

newCoeff=fft\_s(1:NumFreqs).\*(1./fVec);

newCoeff=[newCoeff, fliplr(newCoeff)];

EEG=abs(ifft(newCoeff));

plot(time,EEG);xlabel('time(s)');ylabel('Power');title('Simulated EEG');

end