Neurophotonics HW1

Shani Sela  
316602598

* Link to Github repo:

<https://github.com/shanisela/Neurophotonics-HW1>

1. Writing a function that calculates ΔHbR and ΔHbO over time

**Main function :**

clc;clear;close all;

% Loading the intensity meadata

dataFileList = {'FN\_031\_V2\_Postdose2\_Nback.mat','FN\_032\_V1\_Postdose1\_Nback.mat'};

SDS = 3; %Sourse-Detector Separation distance [cm]

relDPF = "RelativeDPFCoefficients.csv";

DPFperTissue = "DPFperTissue.txt"; %Relative DPF according to wavelength

extinctionCoefficients = "ExtinctionCoefficientsData.csv"; %.csv file with the following columns : wavelength, Water, HbO2, HHb, FatSoybean

tissueType = 'adult\_head'; %Options: 'adult\_forearm' \ 'baby\_head' \ 'adult\_head' \ 'adult\_leg'

plotChannelIdx = [1,2]; %Vector indicating channels to plot.

for i = 1:length(dataFileList)

[ dHbR , dHbO, fig ] = CalcNIRS(dataFileList{i}, SDS, tissueType, plotChannelIdx, extinctionCoefficients , DPFperTissue, relDPF );

% Set unique figure identifier to the file used

end

**dHbR & dHbO Calc function**

function [ dHbR , dHbO, fig ] = CalcNIRS(dataFile, SDS, tissueType, plotChannelIdx, extinctionCoefficientsFile, DPFperTissueFile, relDPFfile )

%% Verifying input validity

% Check if dataFile is a .mat file and exists

if ~ischar(dataFile) || ~exist(dataFile, 'file') || ~strcmp(dataFile(end-3:end), '.mat')

error('dataFile must be a valid .mat file.');

end

% Check if SDS is a numeric value

if ~isnumeric(SDS) || ~isscalar(SDS)

error('SDS must be a numeric scalar.');

end

% Check if tissueType is a string

if ~ischar(tissueType)

error('tissueType must be a string.');

end

%% Set default values

if nargin < 7 || isempty(relDPFfile)

relDPFfile = '.\RelativeDPFCoefficients.csv';

end

if nargin < 6 || isempty(DPFperTissueFile)

DPFperTissueFile = '.\DPFperTissue.txt';

end

if nargin < 5 || isempty(extinctionCoefficientsFile)

extinctionCoefficientsFile = '.\ExtinctionCoefficientsData.csv';

end

if nargin < 4 || isempty(plotChannelIdx)

plotChannelIdx = [];

end

%% Extract relevant data from the input dataFile

% Load data from .mat file and validate its contents

data = load(dataFile);

if ~isfield(data, 'SD') || ~isfield(data.SD, 'Lambda') || ~isfield(data, 't') || ~isfield(data, 'd')

error('dataFile must contain the fields: DS.Lambda, t, and d.');

end

if size(data.d, 2) ~= 40

error('Intensity data (d) must have 40 rows.');

end

wavelengths =data.SD.Lambda; %Two wavelengths [nm]

time = data.t; %Time vector [sec]

intensities = data.d; %Intensity measurements

%% Load extinction coefficients and DPF data

extinctionCoefficients = readtable(extinctionCoefficientsFile); %Absorption coefficient (epsilon) [L/mol\*cm]

DPFperTissue = readtable(DPFperTissueFile);

relDPF = readtable(relDPFfile);

%% Calculate DPF for the given tissue type and wavelengths

DPF807nm = DPFperTissue.DPF(strcmp(DPFperTissue.Tissue, tissueType)); %Finding the DPF at 807 for the required tissue type

relDPFCoeff = interp1(relDPF.wavelength, relDPF.relDPFcoeff, wavelengths); %Find the relative DPF for the required wl

DPF = DPF807nm .\* relDPFCoeff; %calculate final DPF

pathlength = (SDS.\*DPF);

%% Calculate optical densities

I0 = intensities(1,:);

OD = log10(I0 ./ intensities);

%% Calculate extinction coefficients for the given wavelengths

epsilonHbR = interp1(extinctionCoefficients.wavelength, extinctionCoefficients.HHb, wavelengths);

epsilonHbO = interp1(extinctionCoefficients.wavelength, extinctionCoefficients.HbO2, wavelengths);

epsilonMat = [epsilonHbO.',epsilonHbR.'];

%% Output

dHbO = zeros(size(OD(:,1:20)));

dHbR = zeros(size(OD(:,1:20)));

for i = 1:20

% Extract data per channel

A = [OD(:, i), OD(:, i+20)].';

% modifies Beer-Lambert equation

conc\_changes = (1./pathlength) .\* inv(epsilonMat) \* (A);

% Store results

dHbO(:, i) = conc\_changes(1,:).';

dHbR(:, i) = conc\_changes(2,:).';

end

%% Plot the specified channels

fig = [];

if ~isempty(plotChannelIdx) && isvector(plotChannelIdx)

for ch = plotChannelIdx

f = figure;

plot(time, dHbR(:, ch), 'b');

hold on;

plot(time, dHbO(:, ch), 'r');

hold off;

title(sprintf("File name: %s, Channel %d", strrep(dataFile, '\_',''),ch));

xlabel('Time (s)');

ylabel('Concentration Change');

legend('dHbR', 'dHbO', 'Location', 'best');

fig = [fig, f]; % Store the figure handle

end

elseif isempty(plotChannelIdx)

disp('No channels specified for plotting.');

else

disp('Invalid input for plotChannelIdx. Please provide a vector with values in the range [1-20].');

end

1. First 2 channels plots for “FN\_031\_V2\_Postdose2\_Nback.mat” :





First 2 channels plots for “ FN\_032\_V1\_Postdose1\_Nback.mat” :



1. FFT of the intensity of the first channel from the first file:

%% Question 2

close all;

FirstFile = load(dataFileList{1});

FirstChannel = FirstFile.d(:,1);

L = length(FirstChannel);

Fs = 7.8; % Sampling frequency in Hz

Y = abs(fft(FirstChannel)/L);% Normalized Fourier Transform

f = (0:L-1) \* (Fs/L); % Frequency axis

% Only take the first half of the spectrum

Y = Y(1:floor(L/2));

f = f(1:floor(L/2));

% Define frequency range for signal and noise

noiseFrequencyStart = 2.5; % Noise frequency starts at 2.5 Hz

HeartBeatFreq = [1 2];

fMaxVal = 3.9;

% Find the indices for the relevant frequency ranges

pulseIndices = find(f >= HeartBeatFreq(1) & f <= HeartBeatFreq(2));

noiseIndices = find(f >= noiseFrequencyStart & f <= fMaxVal);

% Calculate the signal strength at the pulse frequency

[~, HRpulsePeakIDX] = max(Y(pulseIndices));

HRpulsePeakfft = Y(pulseIndices(HRpulsePeakIDX));

% Calculate the noise as the mean FFT values in the noise range

noiseStrength = mean(Y(noiseIndices));

% Calculate the Signal-to-Noise Ratio (SNR)

SNR = HRpulsePeakfft / noiseStrength;

% Plot the Fourier Transform

figure;

semilogy(f, Y.');

hold on

plot(f(pulseIndices(HRpulsePeakIDX)),HRpulsePeakfft,'o')

hold off

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('Fourier Transform of First Channel');

legend('fft signal', 'HR peak')

% Display the SNR

fprintf('The Signal-to-Noise Ratio (SNR) is: %.2f\n', SNR);



The Signal-to-Noise Ratio (SNR) is: 8.03