LFP/Scalograms Analysis

Filename Definitions:

**Amplifier.dat** – raw data containing signal with {potentially} spikes and [monopolar] lfps. Acquired at 20kHz so the file size is quite large. Note – Intan records the amplifier.data as amplifier channels 0-63. MATLAB brings these channels in as 1-64. Located in ‘rawdata\sessionDate\ephys’ folder

**\_lfp.mat** – file generated from the calculate\_monopolar\_LFPs function using information recorded from the electrophysiology of each behavior session (generated mostly from the amplifier.dat file but requires all e-phys data to work). This file calculates monopolar LFPs (to separate lfps from any spikes present) and decimates the signal to a 500Hz ‘sampling’ frequency to reduce file size. Note – Intan records the amplifier.data as amplifier channels 0-63. MATLAB brings these channels in as 1-64. Thus, the lfp.mat file is from 1-64. Located in ‘processed’ folder.

**\_ordered\_lfp.mat –** file contains the lfps ordered in a manner consistent with probe\_type mapping. To order the probe sites, use the lfp\_by\_probe\_site function. If present, file is located in the processed folder (function is quick so no need to use space so if present, the file was used for troubleshooting purposes of other code/functions).

**\_monopolarpower.mat** – file generated from script\_extract\_power\_spectra to generate the power spectral density analysis of the lfp files (script uses several dependencies, check below for more thorough notes). Should bring in the ‘ordered\_lfps’ but does so within the code without saving an \_ordered\_lfp.mat file. Uses pwelch to calculate the power spectra. Located in ‘processed’ folder.

**\_diffpower.mat –** file generated from script\_extract\_power\_spectra to generate the differentials of the lfp files (script uses several dependencies, check below for more thorough notes). The probe sites must be geographically mapped to ensure you’re taking the differentials between neighboring probe sites (e.g. using the ordered\_lfps file). Located in ‘processed’ folder.

\*New files saved go here (name might change)

**eventFieldname\*\_sessionScalos\_ch\*.mat** – calculates the scalograms from the monopolar **lfps** – note lfp data needs to be organized by site number before processing (e.g. using the lfp\_by\_probe\_site function). Calculate the scalos by trial type (correctGo, correctRight, correctLeft, etc) and event type (noseIn, cueOn, noseOut, etc). Filename will likely include ratID, sessionDate, trial type and event type. Currently working on code to generate these files. \* JM 20220929

LFP Analysis – more thorough description of steps used to generate probe\_type mapping and rationale.

1. Map channels for each specific probe paradigm (e.g. Neuronexus vs Cambridge), to the representative amplifier channel number. Use this to write a probe mapping file in MATLAB
   1. CODE – util for LFP analysis
      1. intan\_to\_site\_map = probe\_site\_mapping(probe\_type)
         1. Neuronexus H64LP 8x8, use ‘NN8x8’ as the probe\_type
         2. Cambridge 2 shank, 4 columns with omnetics connector ‘ASSY156’
         3. Cambridge 2 shank 4 columns with Molex connector ‘ASSY236’
      2. To create a caption for plots, use –
         1. naming\_convention; % this is a script to create a workspace variable for labeling captions in figures that use multiple probe sites in one figure e.g. when plotting from ‘monopolarpower’ files.
         2. naming\_conventinon\_diffs; % this is a script to create a workspace variable for the diffs – e.g. when plotting from ‘diffpower’ files.
2. Review Histology according to channel number to verify location of each site
   1. Include AP, ML, DV coordinates
   2. Verify atlas locations of each site (e.g. VM, VA, etc)
   3. Create an excel sheet that is MATLAB friendly (e.g. no notes). The notes are helpful to have though for future training purposes.
   4. Include a column that references the specific amplifier channel associated with the respective channels (mapped from 1 above)
      1. Note: Intan Amplifier channels are labelled 0-63 and the Intan system records data from 0-63 but MATLAB reads in the data as 1-64. Keep this in mind when checking data.
3. Review amplifier.dat data using Neuroscope
   1. Create a file or tab within excel data sheets to specify good or bad channels (include each date as a separate column; channels might be good one day and bad the next)
      1. Artifacts like high or low voltage can be accommodated within the code itself
         1. Note: we have not done this yet as of 9/23/2022 JM
      2. Current ‘codes’ for individual good vs bad **lfp** channels are:
         1. 0 = good channel
         2. 1 = bad channel
         3. 2 = has some issues but also good spots, verify the data post analysis
      3. For the column header, the dates are highlighted to show whether the file could be analyzed for spikes (though by eye these spikes may be very small at best for NeuroNexus probes tested from 2020 to 2021 by JM)
         1. Green >80% of the file is good (minus a few bad individual channels etc)
         2. Yellow 50-80% of the file should be good. SPI cable was attached for at least half the file from the beginning of the recording.
         3. Red <50% of the file had solid SPI connection, the SPI cable came off before 30 min of recording, or the file started without a refresh of the software (thus had a lot of Intan issues with recording a line as solely that line without interference).
4. Create LFP.mat files
   1. Script\_analyze\_choice\_task\_LFPs
      1. This file will CD through the directory and calculate the monopolar LFPs (calculate\_monopolar\_LFPs) – see next code line
      2. This file does NOT filter through ‘test’ electrophysiology files (e.g. files not associated with behavior but run to troubleshoot the Intan system, different SPI cables, spikes, etc)
   2. [lfp\_data, actual\_lfpFs] = calculate\_monopolar\_LFPs(intan\_folder, target\_Fs)
      1. This file does not need to be in probe\_type order. It calculates the LFPs and generates \_lfp.mat files.
      2. This is the SINGLE file (if you want to analyze a specific date)
      3. Data located in RatID\processed\sessionDate
   3. [ordered\_lfp, intan\_site\_order, NNsite\_order] = lfp\_by\_probe\_site\_ALL(lfp\_data, probe\_type)
      1. This file re-orders the \_LFP.mat file generated from calculate\_monopolar\_LFPs (single files) and the script to run all the files (from a and b) and orders the file according to probe\_type

For Power Spectra/Differential analysis – Use generated lfp files to calculate power spectra density (pwelch) and diffpower (differentials) data.

1. Script\_extract\_power\_spectra (run on ordered\_lfps AFTER you run ‘d’ for single files or ‘c’ for all folders)
   1. This file generates diff (\_diffpower.mat) and monopolar LFPs (\_monopolarpower.mat) as a full script through the directories. Requires probe\_type
      1. Dependencies (monopolarpower)
         1. [ordered\_lfp, intan\_site\_order, site\_order] = lfp\_by\_probe\_site\_ALL(lfp\_data, probe\_type); % Orders the lfps by probe site mapping; for single files
         2. [power\_lfps, f] = extract\_power(ordered\_lfp,Fs); % This one does one single file
      2. Dependencies (diffpower, NNsite)
         1. lfp\_NNsite\_diff = diff\_probe\_site\_mapping(lfp\_data, probe\_type);
         2. [power\_lfps\_diff, f] = extract\_power(lfp\_NNsite\_diff,Fs);
      3. Dependencies (diffpower, Cambridge probes)
         1. diff\_lfps = diff\_probe\_site\_mapping\_CAMBRIDGE(lfp\_data, probe\_type);
         2. [power\_lfps\_diff, f] = extract\_power(diff\_lfps,Fs);
2. Differentials (diffpower) - Analyze the differences between neighboring probe sites/verifying data.
   1. E.g. for a NeuroNexus 8x8 probe, analyze site 1 vs site 8 on shank 1. Include in the code specifications to NOT analyze the dorsal site from one shank and the ventral site from the next shank (e.g. data that happen to be functionally near each other in the ‘\*.mat’ files.)
      1. All probe designs are here:
         1. C:\Users\magnusje\Dropbox (University of Michigan)\MED-LeventhalLab\Protocols\electrode\_pinouts\ProbeDesigns
      2. NeuroNexus Probe Design (NeuroNexus\_ProbeDesign)
         1. C:\Users\magnusje\Dropbox (University of Michigan)\MED-LeventhalLab\Protocols\lesioning protocols\NeuroNexus\_ProbeDesign.pdf
      3. Cambridge Omnetics Connector Probe Design (Cambridge\_ProbeDesign\_Assy-156\_H6\_map OR ASSY-156-H6\_map)
         1. C:\Users\magnusje\Dropbox (University of Michigan)\MED-LeventhalLab\Protocols\lesioning protocols\ Cambridge\_ProbeDesign\_ASSY-156-H6-map.pdf
         2. C:\Users\magnusje\Dropbox (University of Michigan)\MED-LeventhalLab\Protocols\Cambridge\_neurotech\ ASSY-156-H6-map.pdf
      4. Cambridge Molex Connector Probe Design (ASSY-236\_h6-map)
         1. C:\Users\magnusje\Dropbox (University of Michigan)\MED-LeventhalLab\Protocols\Cambridge\_neurotech\ ASSY-236-H6-map.pdf
   2. Use the intan\_to\_site\_map function to help with coding in the site mapping for the Neuronexus probe
      1. Main lfp generation file does not need a probe\_type. All other files (e.g. for creating monopolar, diffs, and plots need a probe\_type)
   3. lfp\_NNsite\_diff = diff\_probe\_site\_mapping(lfp\_fname, probe\_type) can be used to assess the differences based on this site mapping.
   4. Verify the math for a few channels - LFP\_check\_diffs.m

FOR SCALOGRAMS – Use generated lfp files to generate mean scalograms for an individual session, individual trial type (correctgo) and event type (noseIn).

1. Extract trials by type – at least to gather/sort trials by type for later analysis. Will be helpful information when trying to extract lfps by correct trial type, etc.
   1. Run the Choice\_task\_intan\_workflow to generate the trials structure
   2. Run getTrialEventParams.m to select for which trials match the type of event you are interested
      1. E.g. ‘correct go’
   3. Run extractTrials.m to get an index of the trials with the event type you’re interested (from getTrialEventParams.m in step b).
   4. Run periEventTrialTs.m for desired eventFieldnames (e.g. cueOn or centerIn).
      1. This file uses the trials structure from the Choice\_task\_intan\_workflow.
      2. Start with a tWindow of ~2.5 seconds. A tWindow of 1s is good to get data of interest from either the original (reorangized for NNsite) or the lfp\_diff (differentials), but need a little extra data around the tWindow to account for data processing issues.
      3. Use trIdx in the function execution to only pick out the trials of interest (from running extractTrials.m in step c)
         1. trialRanges = periEventTrialTs(trials(trIdx),[-2 2],eventFieldnames)
            1. Be sure there is an eventFieldnames workspaces variable

eventFieldnames = {‘cueOn’};

This gives a trial structure array m x n x 2.

Currently this function is not needed but available. Use trial\_ts = extract\_trial\_ts(trials, eventFieldnames)instead.

* + - 1. trial\_ts = extract\_trial\_ts(trials(trIdx), eventFieldnames)
         1. Still need eventFieldnames
         2. Creates a vector (trial\_ts) of timestamps of when (cueOn) a particular type of trial (correctgo) occurs
         3. Use trial\_ts for now (not trialRanges from above)
      2. The trials struct timestamps will show you the relevant eventFieldnames associated with that particular trial -- e.g. if selecting ‘correctgo’ from getTrialEventParams, the trIdx will show you which actual trials from the trials structure are ‘correctgo’. Within the trials structure, find that particular trial number and click on ‘timestamps’. This will give timestamps for that particular trial number including the relevant eventFieldnames. This can include: cueOn, centerIn, centerOut, sideIn, sideOut, etc.
         1. eventFieldnames = {‘cueOn’};

1. event\_triggered\_lfps = extract\_LFP\_around\_timestamps(ordered\_lfp, trial\_ts, t\_win, Fs)
   1. This file extracts LFP data around the timestamps you selected (from the above directions). It uses the workspace variable ordered\_lfp from the function [ordered\_lfp, intan\_site\_order, site\_order] = lfp\_by\_probe\_site\_ALL(lfp\_data, probe\_type); (this file reorders the original \_lfp.mat file)
      1. Time domain of graph
         1. Use time = linspace(-2,2,2001); % time (x-axis)
         2. If you’re pulling out 2 seconds around your point of interest, the data is pulled at a Fs of 500 so it will grab 2001 data points.
      2. Frequency domain of the graph
         1. f = flip(linspace(0,60,81))';% frequency (y-axis); writing it this way allows for the high frequencies to actually plot correctly
   2. Goal is to write a script to pull out across all events of a single channel, calculate the CWT and average all of the trials desired
2. event\_triggered\_lfps = extract\_event\_related\_LFPs(ordered\_lfp, trials, eventname, varargin)
   1. This file uses event\_triggered\_lfps = extract\_LFP\_around\_timestamps(ordered\_lfp, trial\_ts, t\_win, Fs)
3. calculate\_cwt\_3D\_matrix\_testing.m
   1. This file runs through the specified event\_triggered\_lfps and calculates the CWT (continuous wavelet transform). This will allow you to generate scalograms (heat maps) of the data.
4. % script\_calculate\_perievent\_scalograms
   1. This file might be helpful for plotting within a loop (pulling in the ChoiceTask\_Intan\_workflow data)
   2. Not currently in use JM 20220923

Notes: Run the event\_triggered\_lfps on ‘ordered’ data so you can pull from that info to plot the data. At some point it should be ordered by site and it seems easier to order the original \_lfp.mat file then run through the required scripts. The trIdx should be the same regardless of \_lfp.mat order. Maybe save an \_ordered\_lfp.mat file so as not to have to create it each time you run the sequence to verify?

Cheatsheet/sample lines of code – Keeping this in the file for now so I can re-use the information to double check files for data accuracy

myData = mean(event\_triggered\_lfps, 1);  
myData2 = squeeze(myData);

fb = cwtfilterbank('Wavelet', 'amor');

% C = reshape(event\_triggered\_lfps\_ordered,[],size(event\_triggered\_lfps\_ordered,3),1);

%

% szA = size(event\_triggered\_lfps\_ordered, 1);

% szB = size(event\_triggered\_lfps\_ordered, 2);

% szC = size(event\_triggered\_lfps\_ordered, 3);

% D = szA \* szB; % using this value to reshape the data into a vector array instead of a 3D Matrix

% result\_test = reshape(event\_triggered\_lfps\_ordered,D,[]); % lines 4 through 9 are doing what line 2 is doing.

test\_data = event\_triggered\_lfps(1,1,:); % pulls out event 1 of site 1 all data (all LFPs across event 1 of site 1).

test\_data1 = squeeze(test\_data); %squeezes the data so that it can be graphed.

Also works as test\_data = squeeze(event\_triggered\_lfps(1,1,:));