Appendix A.

In this appendix we provide our MATLAB script for simulating data and calculating phase-amplitude coupling.

% Script for simulating EEG data for testing cross-frequency

% phase-amplitude coupling methods

% defining center frequencies, for which to calculate coupling

freq4phs = 9; % in Hz

freq4amp = 60; % in Hz

% amount of simulations (e. g. 100 for coupling or 10000 for no coupling)

nrpts = 100;

% define parameters used for modifying the simulated signal

trial\_length = [400,2500,5000]; % in msec

sampling\_rate = [500 1000]; % in Hz

noise = [0.9,1,1.1]; % additional noise

modulation\_strength = [0.9,1,1.1];

mw\_max\_dur = (1000/((freq\_ph\_min+freq\_ph\_max)/2))/4;

modulation\_width = [0.9\*mw\_max\_dur,mw\_max\_dur,1.1\*mw\_max\_dur]; % in ms

phasic = {'mono','bi'};

ntrial = 30; % amount of trials extracted from the simulated data

niter = 1000; % amount of iterations for permutation testing

% defining frequency band borders accoring to the center frequencies

freq\_ph\_min = freq4phs-1; % in Hz

freq\_ph\_max = freq4phs+1; % in Hz

freq\_amp\_min = freq4amp-freq\_ph\_max; % in Hz

freq\_amp\_max = freq4amp+freq\_ph\_max; % in Hz

% defining data length according to trial length and amount of trials

datalength = zeros(size(trial\_length));

for n=1:length(trial\_length)

datalength(n) = (trial\_length(n)/1000 \* ntrial) + ntrial;

end

% initialising phase-amplitude plot

nbin = 18;

winsize = 2\*pi/nbin;

position = zeros(1,nbin); % in radians

for j=1:nbin

position(j) = -pi+(j-1)\*winsize;

end

% initialise empty cell matrix for PAC values

PAC\_values = cell(nrpts,length(datalength),length(sampling\_rate), ...

length(noise),length(modulation\_strength),length(modulation\_width), ...

length(phasic));

% shuffling random number generator

rng('shuffle'); % creates a different seed for the rand-nr-generator

rng\_settings = rng; % saves current settings of the generator

rng\_example = rand(1,5); % saves 5 examples of the generated random nrs

eeglab;

cd 'C:\EEGLAB\eeglab14\_1\_2b\plugins\noise generation' % function for ...

% noise generation downloaded from ...

% https://www.mathworks.com/matlabcentral/fileexchange/42919-pink-red- ...

% blue-and-violet-noise-generation-with-matlab

runIdx = 1;

for iteri=1:nrpts

% simulating EEG structure for eegfilt

EEG.srate = 1000; % initial sampling rate

EEG.pnts = max(datalength)\*EEG.srate;

EEG.nbchan = 1;

EEG.trials = 1;

EEG.data = rednoise(EEG.pnts);

EEG.times = 0:1/EEG.srate:max(datalength)-1/EEG.srate;

rawEEG = EEG;

for dli=1:length(datalength)

for sri=1:length(sampling\_rate)

for noi=1:length(noise)

for msi=1:length(modulation\_strength)

for mwi=1:length(modulation\_width)

for phi=1:length(phasic)

EEG = rawEEG;

% change original data to defined length

EEG.data = EEG.data(1:datalength(dli)\*EEG.srate);

EEG.pnts = length(EEG.data);

EEG.times = 0:1/EEG.srate:datalength(dli)-1/EEG.srate;

EEG.xmax = (datalength(dli)-1/EEG.srate);

% resample original data

EEG = pop\_resample(EEG, sampling\_rate(sri));

% initialise random time points for permuted phase time series in data

% points

numpoints\_trial = (datalength(dli) - ntrial) / ntrial \* sampling\_rate(sri);

% bandpass filtering phase\_signal

sig\_ph = pop\_eegfiltnew(EEG, freq\_ph\_min, freq\_ph\_max, [], 0, [], 0);

sig\_ph = sig\_ph.data; % lower frequency signal for phase

% bandpass filtering amplitude\_signal

sig\_amp = pop\_eegfiltnew(EEG,freq\_amp\_min, freq\_amp\_max, [], 0, [], 0);

sig\_amp = sig\_amp.data; % higher frequency signal for power

% modulate amplitude signal, if there is modulation, i.e. I~=0;

if modulation\_strength(msi)~=0

% creation of a Hanning window with "uneven" length, in order to be

% able to place it evenly around the extreme values

if rem(ceil((modulation\_width(mwi)/1000\*EEG.srate)),2)

myHanning = modulation\_strength(msi) .\* ...

hann(ceil(modulation\_width(mwi)/1000\*EEG.srate))';

else % the data length of the Hanning window is even;

% therefore one data point is subtracted

myHanning = modulation\_strength(msi) .\* ...

hann(ceil((modulation\_width(mwi)/1000\*EEG.srate)-1))';

end

modulation\_width\_final = length(myHanning) \* 1000/EEG.srate;

% find peak indexes

IdxMaxima = findpeaks(sig\_ph);

if strcmp(phasic(phi),'mono') % modulation only at peaks

% Looping through all peaks and multiply amplitude with hanning

% window

for exi=1:length(IdxMaxima.loc)

StartIdx = IdxMaxima.loc(exi)-floor(length(myHanning)/2);

EndIdx = IdxMaxima.loc(exi)+floor(length(myHanning)/2);

if StartIdx < 1

y = 1-StartIdx;

shortenedHann = myHanning(y+1:end);

StartIdx = 1;

sig\_amp(1,StartIdx:EndIdx) = sig\_amp(1,StartIdx:EndIdx).\*(1+shortenedHann);

elseif EndIdx > length(sig\_amp)

y = EndIdx - length(sig\_amp);

shortenedHann = myHanning(1:end-y);

EndIdx = length(sig\_amp);

sig\_amp(1,StartIdx:EndIdx) = sig\_amp(1,StartIdx:EndIdx).\*(1+shortenedHann);

else

sig\_amp(1,StartIdx:EndIdx) = sig\_amp(1,StartIdx:EndIdx).\*(1+myHanning);

end

end

elseif strcmp(phasic(phi),'bi') % modulation at peaks and throughs

% find peaks indexes: already done

% find through indexes

DataInv = 1.01\*max(sig\_ph) - sig\_ph;

IdxMinima = findpeaks(DataInv);

IdxExtrms(1:length(IdxMaxima.loc)) = IdxMaxima.loc;

IdxExtrms(length ...

(IdxMaxima.loc)+1:length(IdxMaxima.loc)+length(IdxMinima.loc)) = ...

IdxMinima.loc;

% Looping through all peaks and throughs and multiply amplitude with

% Hanning window

for exi=1:length(IdxExtrms)

StartIdx = IdxExtrms(exi)-floor(length(myHanning)/2);

EndIdx = IdxExtrms(exi)+floor(length(myHanning)/2);

if StartIdx < 1

y = 1-StartIdx;

shortenedHann = myHanning(y+1:end);

StartIdx = 1;

sig\_amp(1,StartIdx:EndIdx) = sig\_amp(1,StartIdx:EndIdx).\*(1+shortenedHann);

elseif EndIdx > length(sig\_amp)

y = EndIdx - length(sig\_amp);

shortenedHann = myHanning(1:end-y);

EndIdx = length(sig\_amp);

sig\_amp(1,StartIdx:EndIdx) = sig\_amp(1,StartIdx:EndIdx).\*(1+shortenedHann);

else

sig\_amp(1,StartIdx:EndIdx) = sig\_amp(1,StartIdx:EndIdx).\*(1+myHanning);

end

end

end

else

% if there is no modulation, the above procedure

% does not need to be carried out

modulation\_width\_final = 0;

end

% generate additional noise

EEG.data = rednoise(EEG.pnts);

noise\_ph = pop\_eegfiltnew(EEG, freq\_ph\_min, freq\_ph\_max,[],0,[],0);

noise\_ph = noise\_ph.data;

noise\_amp = pop\_eegfiltnew(EEG,freq\_amp\_min,freq\_amp\_max,[],0,[],0);

noise\_amp = noise\_amp.data;

% final synthetic signal

sig\_ph = sig\_ph + noise(noi) .\* noise\_ph;

sig\_amp = sig\_amp + noise(noi) .\* noise\_amp;

% extract amplitude and phase from simulated continiuous data

phase\_ts\_cont = angle(hilbert(sig\_ph));

amplt\_ts\_cont = abs(hilbert(sig\_amp));

ph\_ap\_ts\_cont = angle(hilbert(abs(hilbert(sig\_amp))));

% epoch data (as often real data does conatin data inconsistencies)

triallength = (datalength(dli) - ntrial) / ntrial; % in sec

marker\_dist = 1+triallength;

event\_latency = ...

EEG.srate/2:marker\_dist\*EEG.srate:marker\_dist\*ntrial\*EEG.srate;

start = 1;

phase\_ts = zeros(1,triallength\*EEG.srate\*ntrial);

amplt\_ts = zeros(1,triallength\*EEG.srate\*ntrial);

ph\_ap\_ts = zeros(1,triallength\*EEG.srate\*ntrial);

for eli=1:ntrial

phase\_ts(1,start:eli\*triallength\*EEG.srate) = ...

phase\_ts\_cont(event\_latency(eli):event\_latency(eli)+ ...

(triallength\*EEG.srate-1));

amplt\_ts(1,start:eli\*triallength\*EEG.srate) = ...

amplt\_ts\_cont(event\_latency(eli):event\_latency(eli)+ ...

(triallength\*EEG.srate-1));

ph\_ap\_ts(1,start:eli\*triallength\*EEG.srate) = ...

ph\_ap\_ts\_cont(event\_latency(eli):event\_latency(eli)+ ...

(triallength\*EEG.srate-1));

start = start + triallength\*EEG.srate;

end

% calculate mean amplitude in phase bins for MI

MeanAmp = zeros(1,nbin);

for j=1:nbin

MeanAmp(j) = mean(amplt\_ts(find ...

(phase\_ts >= position(j) & phase\_ts < position(j)+winsize)));

end

% calculate observed MI (complex-valued composite signal)

NormAmp = MeanAmp/sum(MeanAmp);

ShannonEntropy = -sum(NormAmp.\*log(NormAmp));

KL\_Dist = log(nbin)-ShannonEntropy;

obs\_MI = KL\_Dist / log(nbin);

% calculate observed MVL (complex-valued composite signal)

obs\_MVL = mean(amplt\_ts(:).\*exp(1i\*phase\_ts(:)));

% calculate observed PLV (complex-valued composite signal)

obs\_PLV = mean(exp(1i\*(phase\_ts(:) - ph\_ap\_ts(:))));

% calculate observed GLM

[obs\_GLM,obs\_GLM\_CI,NrCtrlPts] = glmfun(phase\_ts,amplt\_ts,10,'AIC');

% The glmfun.m provided by Kramer & Eden (2013), ...

% was modified, such that it takes the phase time series and amplitude

% time series instead of the band pass filtered signal.

% split continious data into trials

phase\_ts = reshape(phase\_ts,triallength\*EEG.srate,ntrial);

amplt\_ts = reshape(amplt\_ts,triallength\*EEG.srate,ntrial);

ph\_ap\_ts = reshape(ph\_ap\_ts,triallength\*EEG.srate,ntrial);

EEG.trials = ntrial;

% calculate surrogate values

permuted\_phase = zeros(length(phase\_ts),ntrial);

permuted\_MI = zeros(1,niter);

permuted\_MVL = zeros(1,niter);

permuted\_PLV = zeros(1,niter);

permuted\_GLM = zeros(1,niter);

for s=1:niter

skip = randsample ...

(round(numpoints\_trial\*.8),ntrial) + round(numpoints\_trial\*.1);

for ti=1:ntrial

permuted\_phase(:,ti) = phase\_ts([skip(ti):end 1:skip(ti)-1],ti);

end

MeanAmpShuffeld = zeros(1,nbin);

for j=1:nbin

MeanAmpShuffeld(j) = mean(amplt\_ts(find ...

(permuted\_phase>=position(j)&permuted\_phase<position(j)+winsize)));

end

permuted\_MI(s) = (log(nbin)- ...

(-sum((MeanAmpShuffeld/sum(MeanAmpShuffeld)) .\* ...

log((MeanAmpShuffeld/sum(MeanAmpShuffeld)))))) / log(nbin);

permuted\_MVL(s) = abs(mean( amplt\_ts(:).\*exp(1i\*permuted\_phase(:)) ));

permuted\_PLV(s) = abs(mean(exp(1i\*(permuted\_phase(:) - ph\_ap\_ts(:)))));

[permuted\_GLM(s),~,~] =glmfun(permuted\_phase(:),amplt\_ts(:),NrCtrlPts);

end

% calculate z-standardized PAC values

MIz = ( obs\_MI - mean(permuted\_MI)) / std(permuted\_MI);

MVLz = (abs(obs\_MVL) - mean(permuted\_MVL)) / std(permuted\_MVL);

PLVz = (abs(obs\_PLV) - mean(permuted\_PLV)) / std(permuted\_PLV);

GLMz = ( obs\_GLM - mean(permuted\_GLM)) / std(permuted\_GLM);

MVLphase = angle(obs\_MVL);

PLVphase = angle(obs\_PLV);

% save PAC values

parameter\_name = ['freq' num2str(freq4phs) 'to' num2str(freq4amp) '\_' ...

num2str(datalength(dli)) 'DL\_' ...

num2str(EEG.srate) 'SR\_' ...

num2str(noise(noi)\*100) 'NO\_' ...

num2str(modulation\_strength(msi)\*10) 'MS\_' ...

num2str(modulation\_width\_final) 'MW\_' char(phasic(phi))];

toSave = {parameter\_name,obs\_MI,MIz,obs\_MVL,MVLz,MVLphase, ...

obs\_PLV,PLVz,PLVphase,obs\_GLM,obs\_GLM\_CI,NrCtrlPts,GLMz};

PAC\_values{iteri,dli,sri,noi,msi,mwi,phi} = toSave;

% clear variables because of varying data length

EEG.data=[]; EEG.srate=[]; EEG.times=[]; EEG.pnts=[]; EEG.nbchan=[];

EEG.trials=[];

clear sig\_amp noise\_amp amplt\_ts amplt\_ts\_cont sig\_ph noise\_ph

clear phase\_ts phase\_ts\_cont permuted\_phase ph\_ap\_ts ph\_ap\_ts\_cont

clear MeanAmp MeanAmpShuffeld NormAmp event\_latency IdxMaxima myHanning

clear bedname toSave

end

end

end

end

end

end

save('tmp\_results','PAC\_values','rng\_settings')

end

save('results','PAC\_values','rng\_settings','rng\_example')

Zhivomirov, H. (2013). Pink, Red, Blue and Violet Noise Generation with Matlab Implementation. Accessed April 17, 2016, http://www.mathworks.com/matlabcentral/fileexchange/42919-pink--red--blue-and-violet-noise-generation-with-matlab-implementation/content/rednoise.m

Kramer, M. A., and Eden, U. T. (2013). Assessment of cross-frequency coupling with confidence using generalized linear models. Journal of Neuroscience Methods 220, 64–74. doi: 10.1016/j.jneumeth.2013.08.006