FIELDTRIP TOOLBOX TUTORIAL REPORT

PRADEEP KUMAR YADAV | BM18MTECH11005

TUTORIAL 3-

Sensor-level ERF, TFR and connectivity analyses

Introduction

We will start by looking at the event-related field (ERF) surrounding visual stimulus onset.

Next, we will examine the induced oscillatory activity during the visual stimulation. Finally, we will investigate the connectivity between the cortex and the muscle, by computing MEG-EMG coherence.

Reading in the data

```
load subjectK
```

data_left, containing the trials where the subjects had to respond with the left wrist; and data_right, where the right wrist was cued.

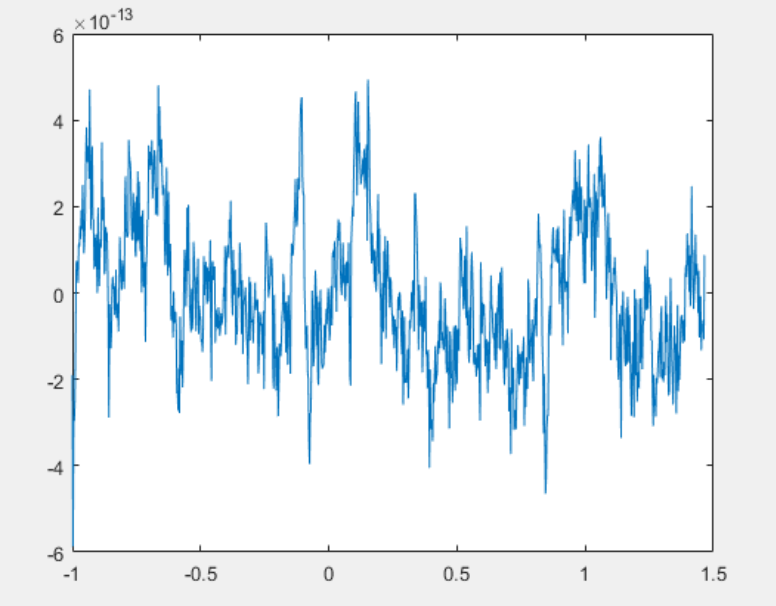
Look data structure

```
data_left =

       hdr: [1x1 struct]
     label: {153x1 cell}
      time: {1x140 cell}
     trial: {1x140 cell}
   fsample: 400
      grad: [1x1 struct]
       cfg: [1x1 struct]
sampleinfo: [140x2 double]
```

The trial field contains the data for each trial as channel X time points matrices. The time field contains the time axis for each trial, and the label field contains the names of the channels in the data.

Plot the data for the first trial, 130th channel

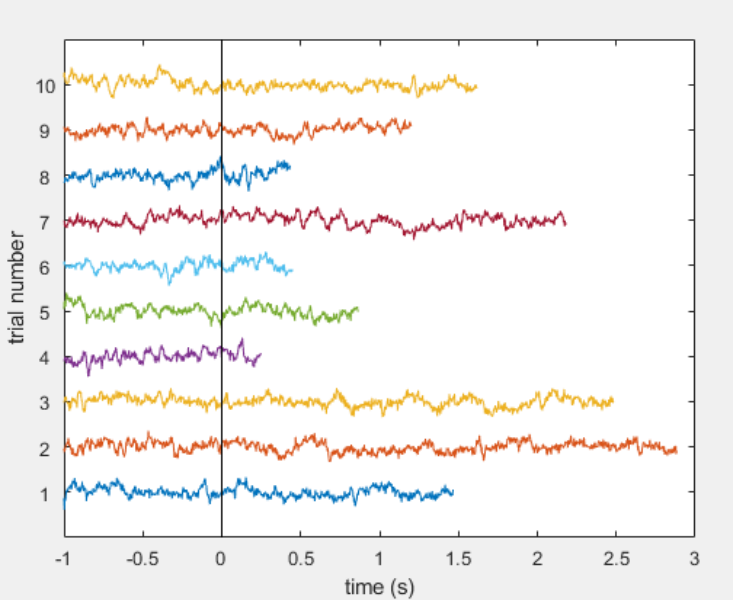
```
plot(data_left.time{1}, data_left.trial{1}(130,:));
```



Time point 0 in all trials corresponds to the onset of the visual stimulation.

The visual stimulus speed change happened at an unpredictable time after t=0, so not all trials are the same length

```
for k = 1:10
plot(data_left.time{k}, data_left.trial{k}(130,:)+k*1.5e-12);
hold on;
end
plot([0 0], [0 1], 'k');
ylim([0 11*1.5e-12]);
set(gca, 'ytick', (1:10).*1.5e-12);
set(gca, 'yticklabel', 1:10);
ylabel('trial number');
xlabel('time (s)');
```



Some example trials, with time t=0 marked.

Before measurement there is also recording which is called offset or error or artifacts.

Event-related analysis

due to intrinsic and extrinsic noise in the signals - which in single trials is often higher than the signal evoked by the brain - it is typically required to average data from several trials to increase the signal-to-noise ratio(SNR)

. Timelock analysis can be used to calculate ERPs/ERFs by **ft_timelockanalysis**.

 we are interested in the ERF locked to visual stimulation. Since visual stimulation was the same in both the response-right and response-left conditions, we can combine the two data sets using **ft_appenddata*

```
cfg  = [];
data = ft_appenddata(cfg, data_left, data_right);
```

concatenating over the "rpt" dimension

the call to "ft_selectdata" took 1 seconds

the call to "ft_appenddata" took 5 seconds

proceed to compute the ER

```
cfg                 = [];
cfg.channel         = 'MEG';
```

```
tl                  = ft_timelockanalysis(cfg, data);
```

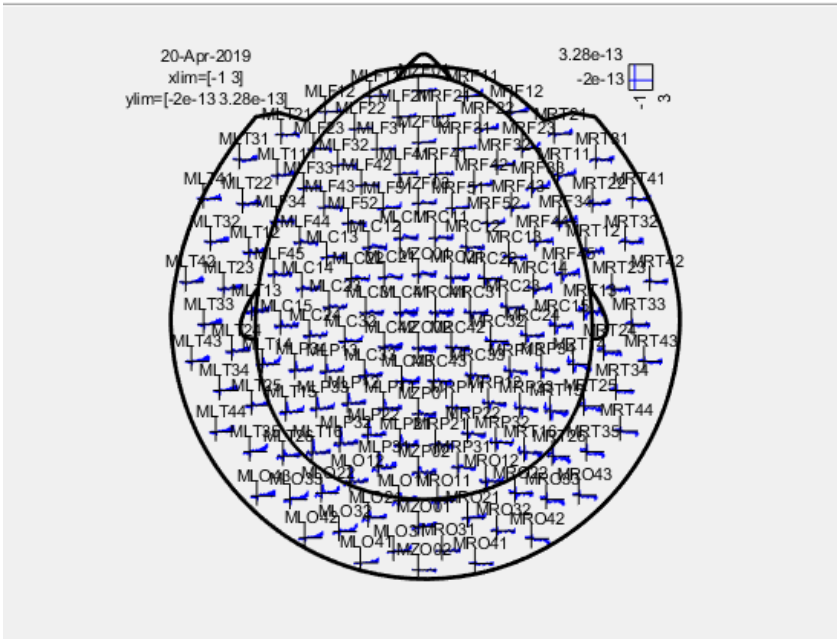
the input is raw data with 153 channels and 281 trials

Plotting the results

FieldTrip provides several options for visualizing the results of event-related analyses: **ft_singleplotER**, **ft_multiplotER**, and **ft_topoplotER.**

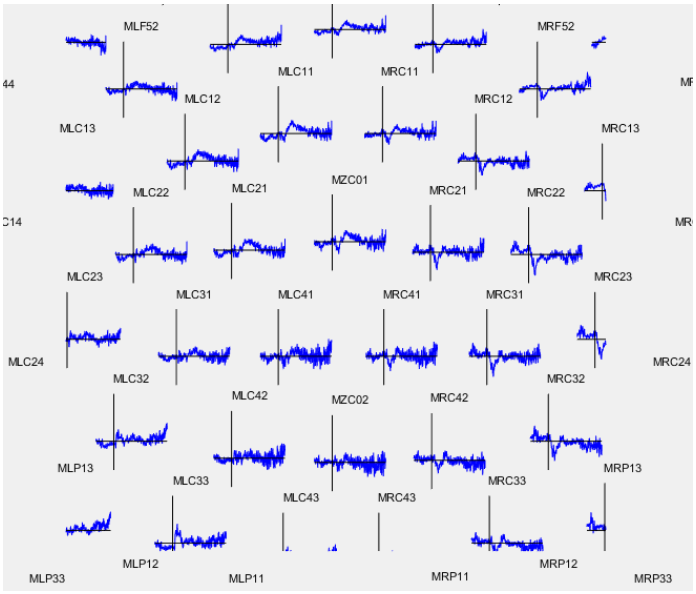
Here is one example for ft_multiplot which is generally used

```
cfg                 = [];
cfg.showlabels      = 'yes';
cfg.showoutline     = 'yes';
cfg.layout          = 'CTF151_helmet.mat';
ft_multiplotER(cfg, tl);
```



event-related field for each MEG sensor.

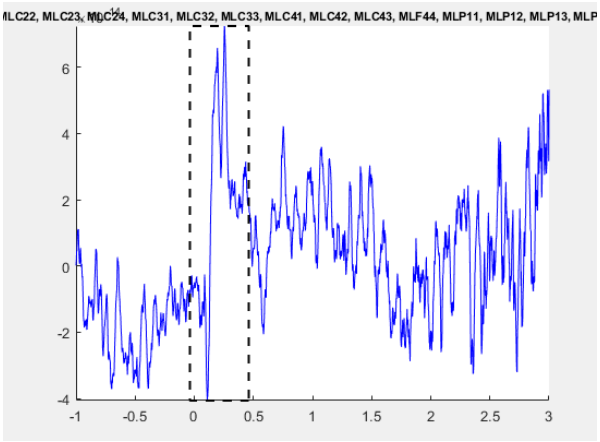
Each channel recording-



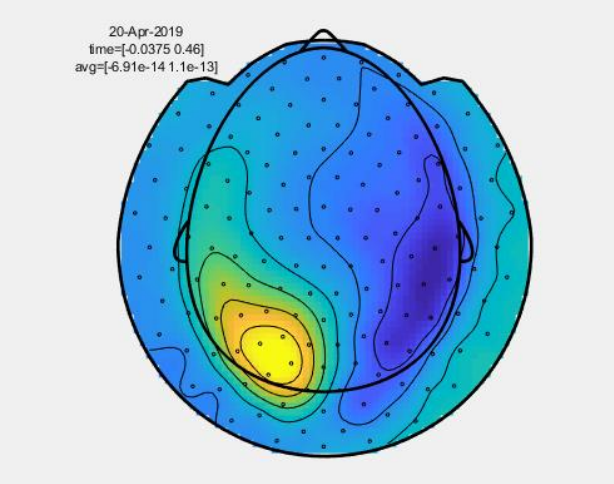
it is possible to select sensors and view an average plot (corresponding to an **ft_singleplotER**) of those sensors.

box to show the average of the selected sensors.

average ERF for some left posterior sensors.



The below topography indicates artifacts(bright yellow) at some particular interval of time.



The planar gradient

```
cfg                 = [];
cfg.method          = 'template';
cfg.template        = 'CTF151_neighb.mat';
neighbours          = ft_prepare_neighbours(cfg, data);

cfg                 = [];
cfg.method          = 'sincos';
cfg.neighbours      = neighbours;
data_planar         = ft_megplanar(cfg, data);

cfg                 = [];
cfg.channel         = 'MEG';
tl_planar           = ft_timelockanalysis(cfg, data_planar);

cfg                 = [];
tl_plancmb          = ft_combineplanar(cfg, tl_planar);
```

first-order axial gradiometer sensors that measure the gradient of the magnetic field in the radial direction, i.e. orthogonal to the scalp.

advantage of the planar gradient transformation is that the signal amplitude typically is largest directly above a source, whereas with axial gradient the signal amplitude is largest away from the source.

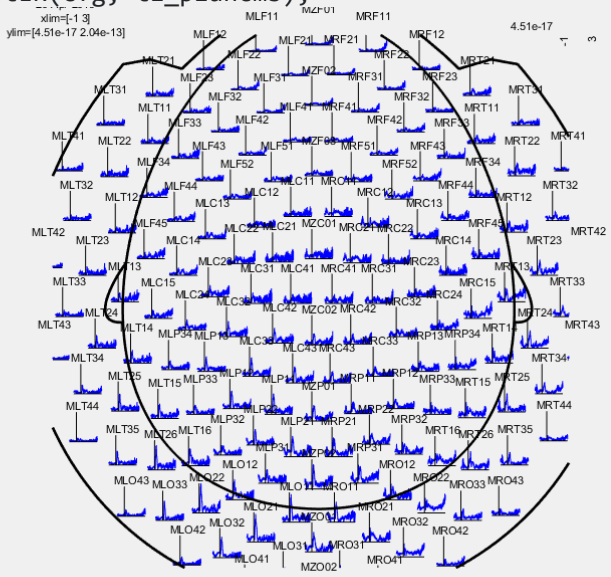
We can compute the planar magnetic gradient using **ft_megplanar**, which gives us the planar gradient in the vertical and horizontal orientations. For visualization, and many subsequent analysis steps, these components need to be combined, which is implemented by **ft_combineplanar**.

compute the planar gradient and recompute the ERFs on this data

```
cfg                 = [];
cfg.method          = 'template';
cfg.template        = 'CTF151_neighb.mat';
neighbours          = ft_prepare_neighbours(cfg, data);

cfg                 = [];
cfg.method          = 'sincos';
cfg.neighbours      = neighbours;
data_planar         = ft_megplanar(cfg, data);

cfg                 = [];
cfg.channel         = 'MEG';
tl_planar           = ft_timelockanalysis(cfg, data_planar);

cfg                 = [];
tl_plancmb          = ft_combineplanar(cfg, tl_planar);
```

Note that we create a 'neighbours' structure before calling **ft_megplanar**. This is required by **ft_megplanar** because the 'sincos' algorithm needs to know which channels are adjacent to one another. Plot the results again

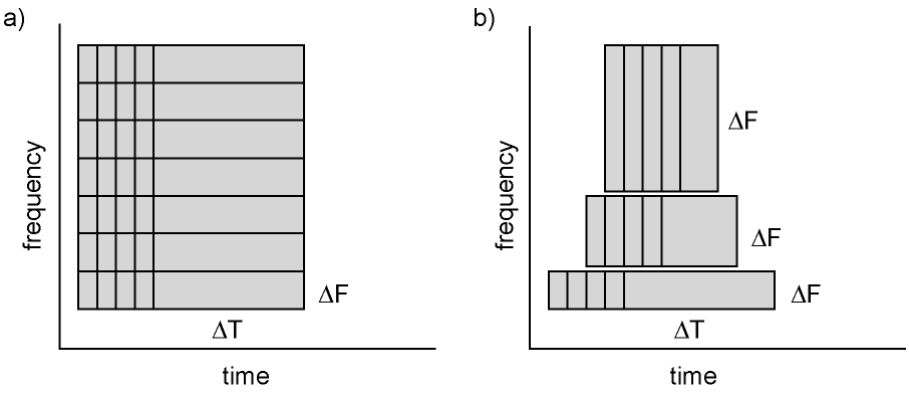
```
cfg                 = [];
cfg.showlabels      = 'yes';
cfg.showoutline     = 'yes';
cfg.layout          = 'CTF151_helmet.mat';
ft_multiplotER(cfg, tl_plancmb);
```



be advised that this might result in unexpected and undesirable effects due to different number of trials and/or due to baselining effects. In general we recommend to not use combined planar gradients for ERFs, unless you know what you are doing.

Time-frequency analysis

Background



Time and frequency smoothing. (a) For a fixed length time window the time and frequency smoothing remains fixed. (b) For time windows that decrease with frequency, the temporal smoothing decreases and the frequency smoothing increases.

Oscillatory components contained in the ongoing EEG or MEG signal often show power changes relative to experimental events. These signals are not necessarily phase-locked to the event and will not be represented in event related fields and potential Oscillatory gamma activity in humans and its role in object representation.

The goal of this section is to compute and visualize event related changes by calculating time-frequency representations (TFRs) of power. This will be done using analysis based on Fourier analysis and wavelets. The Fourier analysis will include the application of multitapers.

Time-frequency representations using Hanning tapers

When choosing for a fixed window length procedure the frequency resolution is defined according to the length of the time window (delta T). The frequency resolution (delta f in figure 1) = 1/length of time window in sec.

thus a 500 ms time window results in a 2 Hz frequency resolution (1/0.5 sec= 2 Hz) meaning that power can be calculated for 2 Hz, 4 Hz, 6 Hz etc. An integer number of cycles must fit in the time window.

We need a part of the baseline interval and a part of the stimulation interval, so we choose the interval from -0.8s to 1.0s.

TFR-

```
cfg                 = [];
cfg.toilim          = [-0.8 1];
```

```
cfg.minlength       = 'maxperlen'; % this ensures all resulting trials are equal
length
data_small          = ft_redefinetrial(cfg, data_planar);
```

using **ft_freqanalysis** to compute our TFR using a 0.2s window size:

```
cfg                 = [];
cfg.method          = 'mtmconvol';
cfg.taper           = 'hanning';
cfg.channel         = 'MEG';

% set the frequencies of interest
cfg.foi             = 20:5:100;

% set the timepoints of interest: from -0.8 to 1.1 in steps of 100ms
cfg.toi             = -0.8:0.1:1;

% set the time window for TFR analysis: constant length of 200ms
cfg.t_ftimwin       = 0.2 * ones(length(cfg.foi), 1);

% average over trials
cfg.keeptrials      = 'no';

% pad trials to integer number of seconds, this speeds up the analysis
% and results in a neatly spaced frequency axis
cfg.pad             = 2;
freq                = ft_freqanalysis(cfg, data_small);
```

trial 180, frequency 17 (100.00 Hz), 1 tapers

Again we have to combine the two components of the planar gradient

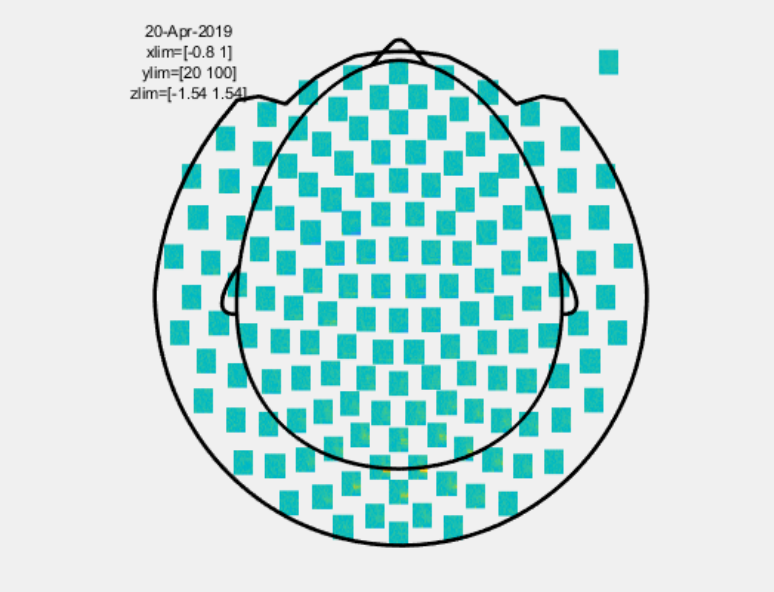
```
cfg                 = [];
freq                = ft_combineplanar(cfg, freq);
```

plot the graph with the help of ft_multiplotTFR function.

```
cfg                 = [];
cfg.interactive     = 'yes';
cfg.showoutline     = 'yes';
cfg.layout          = 'CTF151_helmet.mat';
cfg.baseline        = [-0.8 0];
cfg.baselinetype    = 'relchange';
cfg.zlim            = 'maxabs';
ft_multiplotTFR(cfg, freq);
```

the input is freq data with 302 channels, 17 frequencybins and 19 timebins

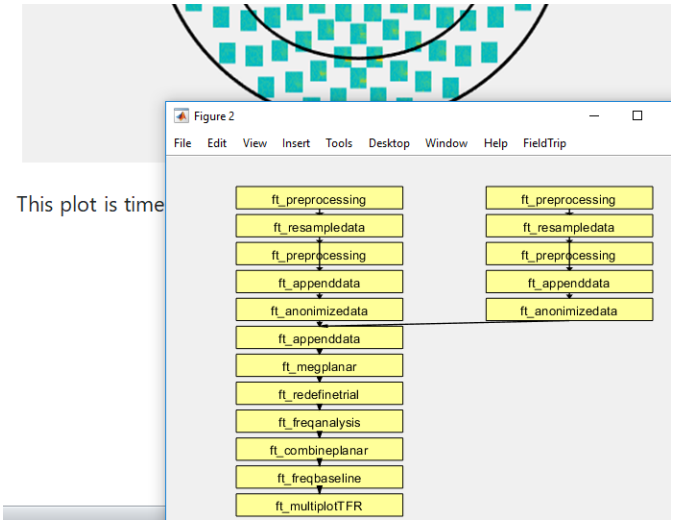
we want to plot the relative change of our power data with respect to the interval between -0.8s and 0s (corresponding to the no-stimulation baseline interval in the experimental design).



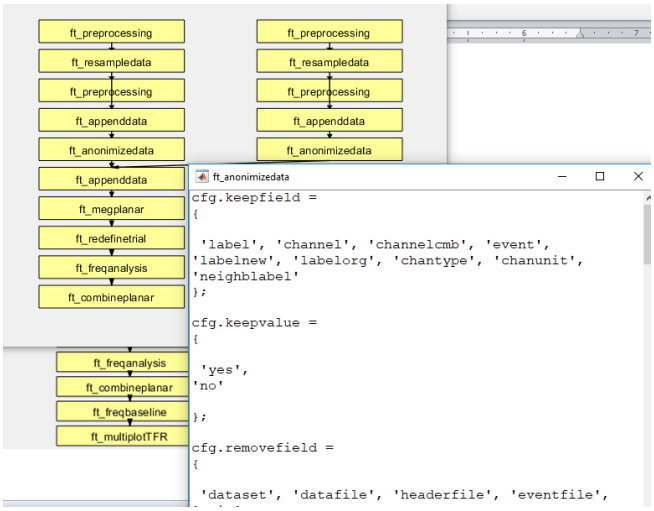
This plot is time frequency presentation with fixed window length.

The below block performed the processing at each stage or sequence of the fuction used in the algorithms.

```
cfg = [];
ft_analysispipeline(cfg, freq);
```



By clicking on one of the boxes a new figure will appear that shows all cfg-options that were used to in the respective function.



Cortico-muscular coherence

we have computed MEG data but there is also EMG data left and right wrist muscles.

These signals provide excellent reference signals to investigate the connectivity between the cortex and the muscle.

we will compute the coherence between the MEG signal and the left and right EMG signals.

Coherence is computed in the frequency domain by normalizing the magnitude of the summed cross-spectral density between two signals by their respective power. For each frequency bin the coherence value is a number between 0 and 1. The coherence values reflect the consistency of the phase difference between the two signals at a given frequency.

we will now look at corticomuscular coherence irrespective of the response cue, and simply pool the two conditions. The cleanest time window in which to estimate this effect is the baseline window, with no visual stimulation present. Therefore, we subselect the baseline data to subject to our coherence analysis.

```
cfg                 = [];
cfg.toilim          = [-1 -0.0025];
cfg.minlength       = 'maxperlen'; % this ensures all resulting trials are equal
length
data_stim           = ft_redefinetrial(cfg, data);
```

removing 0 trials in which no data was selected

removing 57 trials that are too short

the call to "ft_redefinetrial" took 8 seconds

coherence is one of the metrics which can be computed by **ft_connectivityanalysis**. To compute coherence, this function needs the cross-spectral density matrix as input. This can be computed by **ft_freqanalysis**, the same function we used earlier to compute TFRs. By default **ft_freqanalysis** only outputs power values, and not the cross-spectral density. To change this, we have to specify cfg.output = 'powandcsd'.

```
cfg                 = [];
cfg.output          = 'powandcsd';
cfg.method          = 'mtmfft';
cfg.taper           = 'dpss';
cfg.tapsmofrq       = 5;
cfg.foilim          = [5 100];
cfg.keeptrials      = 'yes';
cfg.channel         = {'MEG' 'EMGlft' 'EMGrgt'};
cfg.channelcmb      = {'MEG' 'EMGlft'; 'MEG' 'EMGrgt'};
freq_csd            = ft_freqanalysis(cfg, data_stim);
```

he input is raw data with 153 channels and 224 trials

the call to "ft_selectdata" took 0 seconds

Default cfg.pad='maxperlen' can run slowly. Consider using cfg.pad='nextpow2' for more efficient FFT computation.

processing trials ,processing trial 224/224 nfft: 400 samples, datalength: 400 samples, 9 tapers

cfg.keeptrials = 'yes' because phase estimates are required for each individual trial (i.e., averaging phase over trials is generally not a good idea).]

using multitapers ('dpss') this time, to have a good control over our spectral smoothing.

cfg.channel now also includes the two EMG channels, and that cfg.channelcmb is a 2x2 cell-array specifying that we want to compute the cross-spectral density between the MEG and the left EMG, and the MEG and the right EMG.

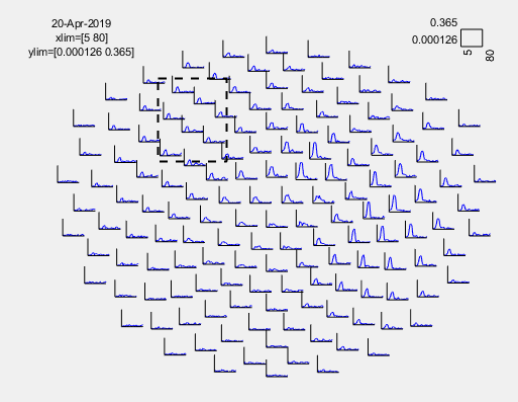
After computing the cross-spectral density, we can invoke **ft_connectivityanalysis** to compute the coherence

```
cfg                 = [];
cfg.method          = 'coh';
cfg.channelcmb      = {'MEG' 'EMG'};
conn                = ft_connectivityanalysis(cfg, freq_csd);
```

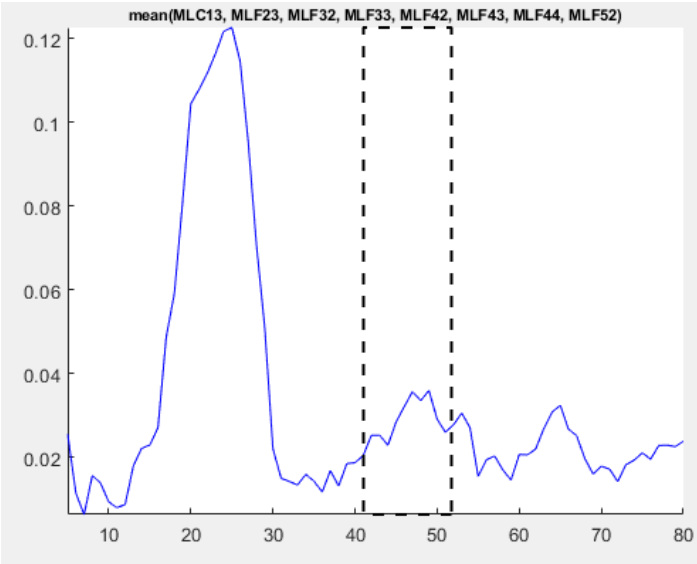
the above code signify that coherency between MEG and EMG signal.

The plotting functions plot the coherence of one channel X to typically all MEG channels. The channel X is determined by the parameter cfg.refchanne

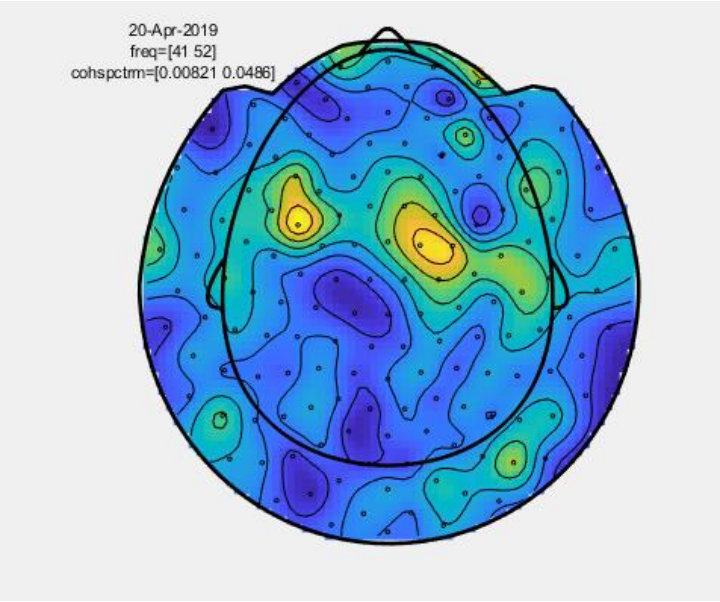
```
cfg                 = [];
cfg.parameter       = 'cohspctrm';
cfg.xlim            = [5 80];
cfg.refchannel      = 'EMGlft';
cfg.layout          = 'CTF151_helmet.mat';
cfg.showlabels      = 'no';
cfg.interactive     = 'yes';
figure;
ft_multiplotER(cfg, conn);
```



Now select the area on graph and click on it.



Again select the area shown in the graph.



This nice overview of the exact spectrum and the topography of the peak (which is in the beta frequency range).

results of sensor-level analysis of corticomuscular coherence. Reference channel was the left EMG.

Tutorial 4-

Analysis of corticomuscular coherence

Introduction

cortico-muscular coherence reflects functional connectivity between primary motor cortex and a contralateral effector muscle during isometric contraction.

Changing the forces on wrist manuallyand monitored by strain gauge and data is recorded.

MEG signals were recorded with a 151 sensor CTF Omega System (Port Coquitlam, Canada). In addition, the EOG was recorded to later discard trials contaminated by eye movements and blinks. The ongoing MEG and EOG signals were lowpass filtered at 300 Hz, digitized at 1200 Hz and stored for off-line analysis.

Magnetic resonance images (MRIs) were obtained from a 1.5 T Siemens system. During the MRI scan, ear molds containing small containers filled with vitamin E marked the same landmarks. This allows us, together with the anatomical landmarks, to align source estimates of the MEG with the MRI.

Background

First we will explore the coherence between the EMG signal and all MEG channels. Secondly, we will investigate how the coherence estimate is influenced by the number of trials, and by the degree of spectral smoothing using multitaper spectral analysis.

Procedure

To compute the coherence between the MEG and EMG signals for the example dataset we will perform the following step

- Read the data into MATLAB using **ft_preprocessing**
- Compute the power spectra and cross-spectral densities using the function **ft_freqanalysis** and subsequently compute the coherence using **ft_connectivityanalysis**
- Visualize the results using **ft_singleplotER**, **ft_multiplotER**, and **ft_topoplotER**
- Subsequently it is possible to localise the neuronal sources coherent with the EMG, using **ft_sourceanalysis.**

Preprocessing

We will calculate the coherence between the MEG and the EMG when the subject extended her LEFT wrist, while keeping the right forearm muscle relaxed.

Then after preprocess the MEG data and identify the artifacts and apply the automatic artifacts rejection.

Take the 10 sec recorded data .

```
% find the interesting epochs of data
cfg = [];
cfg.trialfun                = 'trialfun_left';
cfg.dataset                 = 'SubjectCMC.ds';
cfg = ft_definetrial(cfg);

% detect EOG artifacts in the MEG data
cfg.continuous              = 'yes';
cfg.artfctdef.eog.padding   = 0;
cfg.artfctdef.eog.bpfilter  = 'no';
cfg.artfctdef.eog.detrend   = 'yes';
cfg.artfctdef.eog.hilbert   = 'no';
cfg.artfctdef.eog.rectify   = 'yes';
cfg.artfctdef.eog.cutoff    = 2.5;
cfg.artfctdef.eog.interactive = 'no';
cfg = ft_artifact_eog(cfg);

% detect jump artifacts in the MEG data
cfg.artfctdef.jump.interactive = 'no';
cfg.padding                    = 5;
cfg = ft_artifact_jump(cfg);

% detect muscle artifacts in the MEG data
cfg.artfctdef.muscle.cutoff      = 8;
cfg.artfctdef.muscle.interactive = 'no';
cfg = ft_artifact_muscle(cfg);

% reject the epochs that contain artifacts
cfg.artfctdef.reject          = 'complete';
cfg = ft_rejectartifact(cfg);
```

```
% preprocess the MEG data
cfg.demean                     = 'yes';
cfg.dftfilter                  = 'yes';
cfg.channel                    = {'MEG'};
cfg.continuous                 = 'yes';
meg = ft_preprocessing(cfg);
```

```
evaluating trialfunction 'trialfun_left'


readCTFds: You are reading CTF data for use with a software-application tool
        that is not manufactured by VSM MedTech Ltd. and has not received marketing
        clearance for clinical applications.  If CTF MEG data are processed by this tool,
        they should not be later employed for clinical and/or diagnostic purposes.


found 0 events
created 192 trials
the call to "ft_definetrial" took 24 seconds
Warning: use cfg.trlpadding instead of cfg.padding
searching for artifacts in 1 channels
searching in trial 192 from 192
detected 9 artifacts
the call to "ft_artifact_zvalue" took 12 seconds
searching for artifacts in 151 channels
searching in trial 192 from 192
detected 0 artifacts
the call to "ft_artifact_zvalue" took 153 seconds
searching for artifacts in 151 channels
searching in trial 192 from 192
```

detected 1 artifacts

the call to "ft_artifact_zvalue" took 92 seconds

detected 9 eog artifacts

detected 0 jump artifacts

detected 1 muscle artifacts

rejected 9 trials completely

rejected 0 trials partially

filled parts of 0 trials with nans

filled parts of 0 trials with the specified value

resulting 183 trials

the call to "ft_rejectartifact" took 5 seconds

processing channel { 'MLC11' 'MLC12' 'MLC13' 'MLC14' 'MLC15' 'MLC21' 'MLC22' 'MLC23' 'MLC24' 'MLC31' 'MLC32' 'MLC33' 'MLC41' 'MLC42' 'MLC43' 'MLF11' 'MLF12' 'MLF21' 'MLF22' 'MLF23' 'MLF31' 'MLF32' 'MLF33' 'MLF34' 'MLF41' 'MLF42' 'MLF43' 'MLF44' 'MLF45' 'MLF51' 'MLF52' 'MLO11' 'MLO12' 'MLO21' 'MLO22' 'MLO31' 'MLO32' 'MLO33' 'MLO41' 'MLO42' 'MLO43' 'MLP11' 'MLP12' 'MLP13' 'MLP21' 'MLP22' 'MLP31' 'MLP32' 'MLP33' 'MLP34' 'MLT11' 'MLT12' 'MLT13' 'MLT14' 'MLT15' 'MLT16' 'MLT21' 'MLT22' 'MLT23' 'MLT24' 'MLT25' 'MLT26' 'MLT31' 'MLT32' 'MLT33' 'MLT34' 'MLT35' 'MLT41' 'MLT42' 'MLT43' 'MLT44' 'MRC11' 'MRC12' 'MRC13' 'MRC14' 'MRC15' 'MRC21' 'MRC22' 'MRC23' 'MRC24' 'MRC31' 'MRC32' 'MRC33' 'MRC41' 'MRC42' 'MRC43' 'MRF11' 'MRF12' 'MRF21' 'MRF22' 'MRF23' 'MRF31' 'MRF32' 'MRF33' 'MRF34' 'MRF41' 'MRF42' 'MRF43' 'MRF44' 'MRF45' 'MRF51' 'MRF52' 'MRO11' 'MRO12' 'MRO21' 'MRO22' 'MRO31' 'MRO32' 'MRO33' 'MRO41' 'MRO42' 'MRO43' 'MRP11' 'MRP12' 'MRP13' 'MRP21' 'MRP22' 'MRP31' 'MRP32' 'MRP33' 'MRP34' 'MRT11' 'MRT12' 'MRT13' 'MRT14' 'MRT15' 'MRT16' 'MRT21' 'MRT22' 'MRT23' 'MRT24' 'MRT25' 'MRT26' 'MRT31' 'MRT32' 'MRT33' 'MRT34' 'MRT35' 'MRT41' 'MRT42' 'MRT43' 'MRT44' 'MZC01' 'MZC02' 'MZF01' 'MZF02' 'MZF03' 'MZO01' 'MZO02' 'MZP01' 'MZP02' }

reading and preprocessing

reading and preprocessing trial 183 from 183

the call to "ft_preprocessing" took 72 seconds

read the left and right EMG data. Note that the settings are different for the EMG and MEG data. Most importantly, the EMG data are highpass filtered and rectified

```
cfg              = [];
cfg.dataset      = meg.cfg.dataset;
cfg.trl          = meg.cfg.trl;
cfg.continuous   = 'yes';
cfg.demean       = 'yes';
cfg.dftfilter    = 'yes';
```

```
cfg.channel      = {'EMGlft' 'EMGrgt'};
cfg.hpfilter     = 'yes';
cfg.hpfreq       = 10;
cfg.rectify      = 'yes';
emg = ft_preprocessing(cfg);
```

processing channel { 'EMGlft' 'EMGrgt' }

combine the EMG and MEG trials to a common data structure:

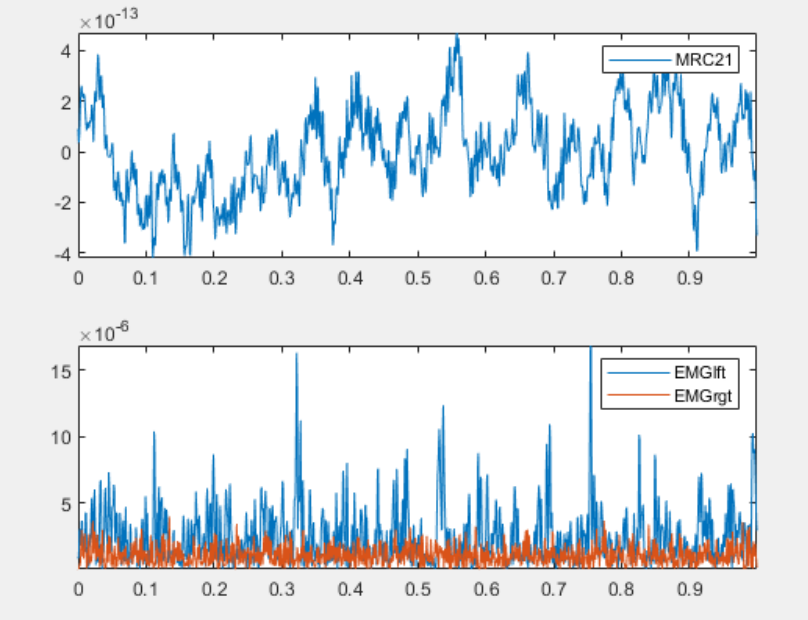
```
data = ft_appenddata([], meg, emg);
```

plot the data-

plot a trial from a sensor overlying the left motor-cortex (MRC21) and the left and right EMG-signals, by selecting the first trial from the data:

```
figure
subplot(2,1,1);
plot(data.time{1},data.trial{1}(77,:));
axis tight;
legend(data.label(77));

subplot(2,1,2);
plot(data.time{1},data.trial{1}(152:153,:));
axis tight;
legend(data.label(152:153));
```

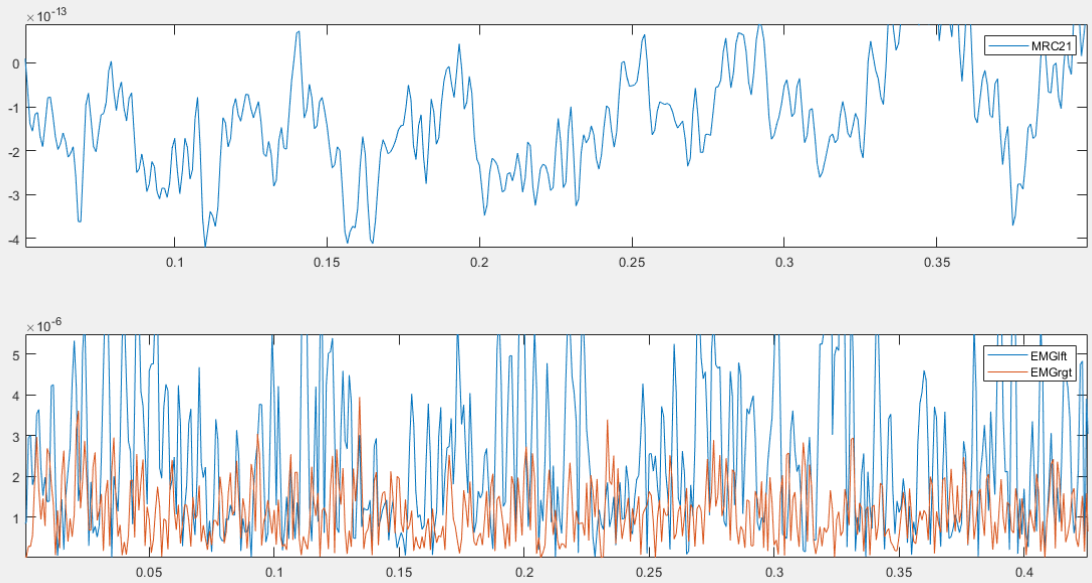
magnitude verses frequency plot-



This figure shows an example of the raw MEG data from sensor MLC21 (upper frame) and the EMG data (lower frame). The signals are from the output of ***ft_preprocessing*** *and plotted using the MATLAB plot function. Note that the signal strength of the left EMG is bigger than that of the right EMG.*

Exercise 1

Explore the MEG and EMG in figure 1, e.g. by zooming in. How are the signals different from one another?



By the figure we can analyse that EMGlft is correlated with EMGrgt . becouse phase difference betwenn two signal is constant. The channel signal and EMG signals are also coherence .

Computing the coherence

Using **ft_freqanalysis**, the characteristics in the frequency domain will be computed. This step requires the preprocessed MEG and EMG data.

After the computation of the frequency domain representation **ft_connectivityanalysis** will be used to compute the coherence.

There are two way to represent the coherence.

Method1-

In this 'method' we will use **ft_freqanalysis** for the computation of the fourier spectra, which is the 'bare' frequency domain representation of the signal, where both amplitude and phase information of the oscillations are represented in a complex number for each frequency.

```
cfg            = [];
cfg.output     = 'fourier';
cfg.method     = 'mtmfft';
cfg.foilim     = [5 100];
cfg.tapsmofrq  = 5;
cfg.keeptrials = 'yes';
cfg.channel    = {'MEG' 'EMGlft' 'EMGrgt'};
freqfourier    = ft_freqanalysis(cfg, data);
```

the input is raw data with 153 channels and 183 trials

the call to "ft_selectdata" took 1 seconds

Default cfg.pad='maxperlen' can run slowly. Consider using cfg.pad='nextpow2' for more efficient FFT computation.

processing trials

processing trial 183/183 nfft: 1200 samples, datalength: 1200 samples, 9 tapers

The FFT-algorithm will be used to compute the Fourier representation of each signal. To optimize the estimation, spectral smoothing using 'multitapers' will be applied

Method 2

In this 'method' we will use **ft_freqanalysis** for the computation of the cross- and power spectra, which are mathematically constructed from the multiplication of a complex-valued fourier spectrum with the complex conjugate of another fourier spectrum.

cross spectrum is called the auto spectrum and this is exactly the same as the power spectrum.

the phase in the cross spectra represent the *phase difference* between the oscillations of a specific channel pair.

```
cfg            = [];
cfg.output     = 'powandcsd';
cfg.method     = 'mtmfft';
cfg.foilim     = [5 100];
cfg.tapsmofrq  = 5;
cfg.keeptrials = 'yes';
cfg.channel    = {'MEG' 'EMGlft' 'EMGrgt'};
cfg.channelcmb = {'MEG' 'EMGlft'; 'MEG' 'EMGrgt'};
freq           = ft_freqanalysis(cfg, data);
```

To calculate the coherence between the EMG and the MEG signals from the fourier spectra, or from the power- and cross spectra use the following function. This function does not care whether the input data contains fourier spectra, or power/cross spectra.

```
fg             = [];
cfg.method     = 'coh';
cfg.channelcmb = {'MEG' 'EMG'};
fd             = ft_connectivityanalysis(cfg, freq);
fdfourier      = ft_connectivityanalysis(cfg, freqfourier)
```

fdfourier =

struct with

elec: [1×1 struct]

grad: [1×1 struct]

labelcmb: {302×2 cell}

dimord: 'chancmb_freq'

cohspctrm: [302×96 double]

freq: [1×96 double]

dof: 1647

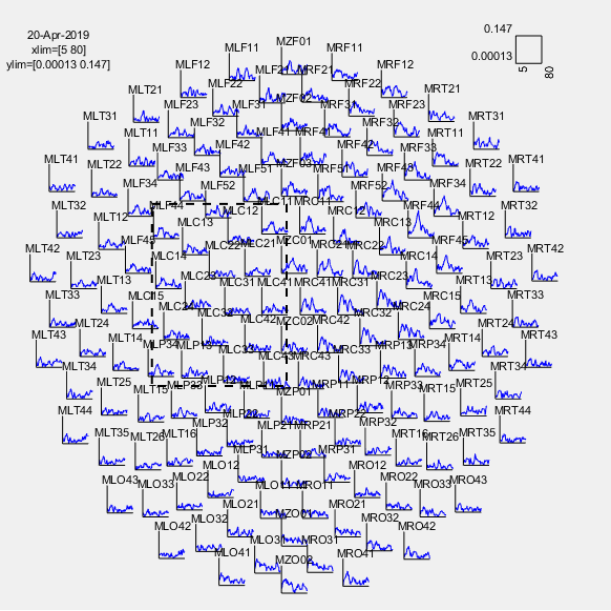
cfg: [1×1 struct]

now contains the coherence for all MEG sensors with respect to the EMG signals.

Displaying the coherence

Visualize the coherence between the EMG and all the MEG sensor

```
cfg                 = [];
cfg.parameter       = 'cohspctrm';
cfg.xlim            = [5 80];
cfg.refchannel      = 'EMGlft';
cfg.layout          = 'CTF151_helmet.mat';
cfg.showlabels      = 'yes';
figure; ft_multiplotER(cfg, fd)
```



The coherence between the left EMG and all the MEG sensors calculated using ft_freqanalysis and ft_connectivityanalysis. Plotting was done with ft_multiplotER.

Plot for MRC21 sensor

```
cfg.channel = 'MRC21';
figure; ft_singleplotER(cfg, fd);
```

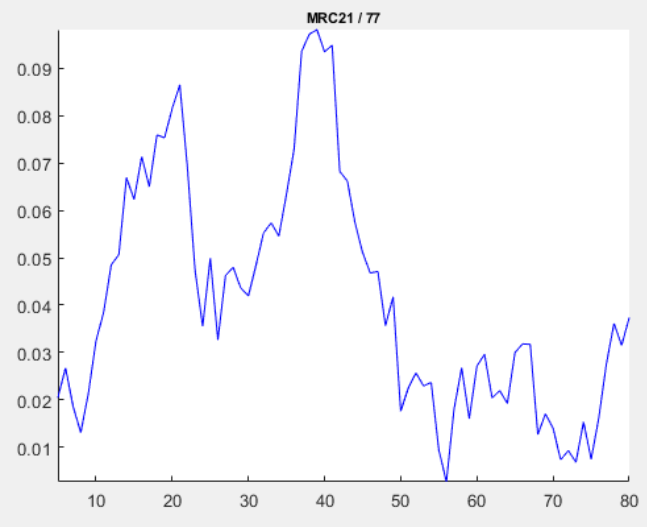
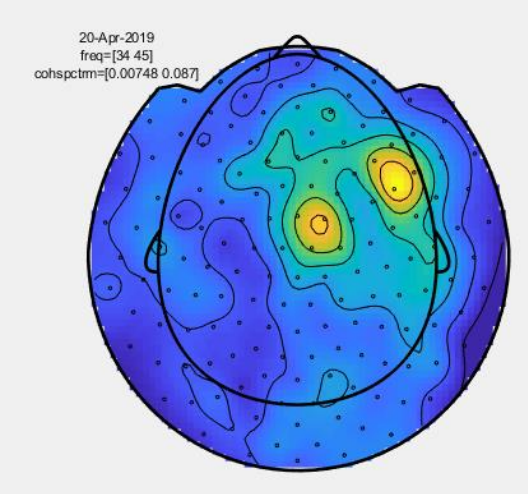


Figure shows the coherence spectrum between the EMG and sensor MRC21



Exercise 2

a) What determines the frequency resolution of the spectrum, as displayed in figure 3? How can it be increased or decreased? Answer the same question for smoothing.

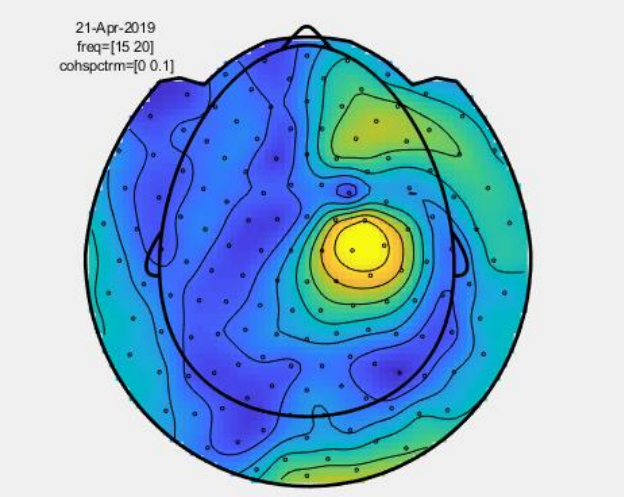
b) Plot a topographical distribution of the coherence in the beta band. The variable cfg.xlim defines the edges of the frequency band.

A)frequency resolution can be increased or decreased by sampling frequency and fft size.

Frequency resolution can be increased or decreased by time resolution there are inversely proportional to each other.

b)

```
cfg                  = [];
cfg.parameter        = 'cohspctrm';
cfg.xlim             = [15 20];
cfg.zlim             = [0 0.1];
cfg.refchannel       = 'EMGlft';
cfg.layout           = 'CTF151_helmet.mat';
figure; ft_topoplotER(cfg, fd)
```

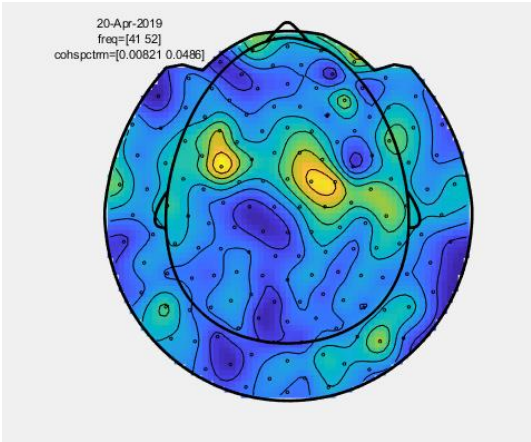


A topographic representation of the coherence between the left EMG and the sensors. The plot was created with ft_topoplotER.

Exercise 3

Explain the pattern of activation in Figure 4.

Plot the topographic representation for other frequencies that might be of interest.

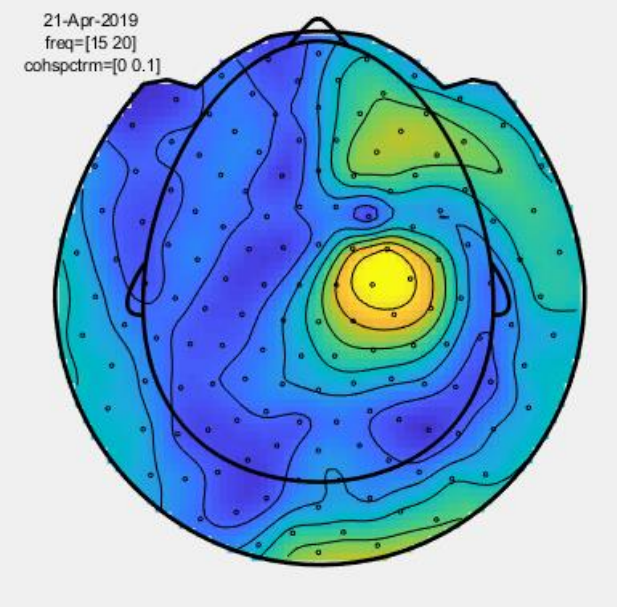


Exercise 4

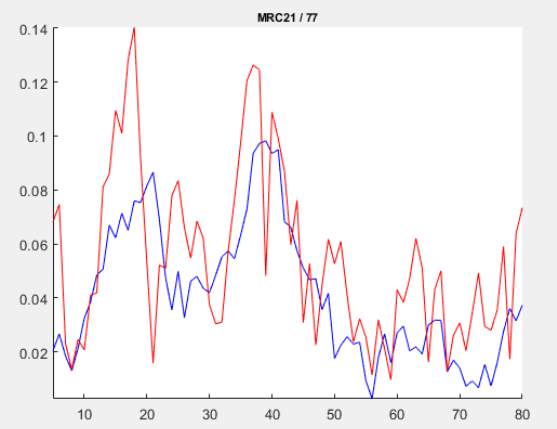
Explore the consequence of changing the smoothing in the frequency domain. Do this by recomputing the cortico-muscular coherence between the EMG signal and MEG sensor MRC21 for different degrees of smoothing. Compute the powerspectra and the cross-spectra, and the corresponding coherence using different degrees of smoothing.

a) 2 Hz smoothing (cfg.tapsmofrq = 2 Hz)

```
cfg            = [];
cfg.output     = 'powandcsd';
cfg.method     = 'mtmfft';
cfg.foilim     = [5 100];
cfg.tapsmofrq  = 2;
cfg.keeptrials = 'yes';
cfg.channel    = {'MEG' 'EMGlft'};
cfg.channelcmb = {'MEG' 'EMGlft'};
freq2          = ft_freqanalysis(cfg,data);

cfg            = [];
cfg.method     = 'coh';
cfg.channelcmb = {'MEG' 'EMG'};
fd2            = ft_connectivityanalysis(cfg,freq2);
```



Plot the results of the 5 and 2Hz smoothing:

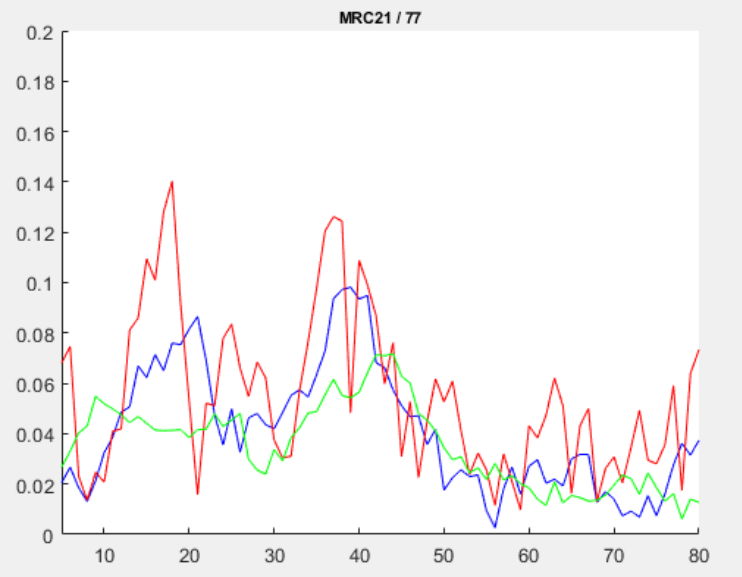


b) 10 Hz smoothing (e.g. cfg.tapsmofrq = 10 Hz)

```
cfg            = [];
cfg.output     = 'powandcsd';
cfg.method     = 'mtmfft';
cfg.foilim     = [5 100];
cfg.keeptrials = 'yes';
cfg.channel    = {'MEG' 'EMGlft'};
cfg.channelcmb = {'MEG' 'EMGlft'};
cfg.tapsmofrq = 10;
freq10        = ft_freqanalysis(cfg,data);

cfg            = [];
cfg.method     = 'coh';
cfg.channelcmb = {'MEG' 'EMG'};
fd10          = ft_connectivityanalysis(cfg,freq10);
```

Plot the results of the 5, 2, and 10 Hz smoothing

```
cfg               = [];
cfg.parameter     = 'cohspctrm';
cfg.xlim          = [5 80];
cfg.ylim          = [0 0.2];
cfg.refchannel    = 'EMGlft';
cfg.channel       = 'MRC21';
figure;ft_singleplotER(cfg, fd, fd2, fd10);
```



Which degree of smoothing do you consider optimal in the calculations above??

Ans-

I will choose 10hz optimal value smoothing .

Exercise 5

Another question pertains to how the estimate of coherence is affected by the number of trials. We will compare the cortico-muscular coherence at two MEG sensors for different amount of data.

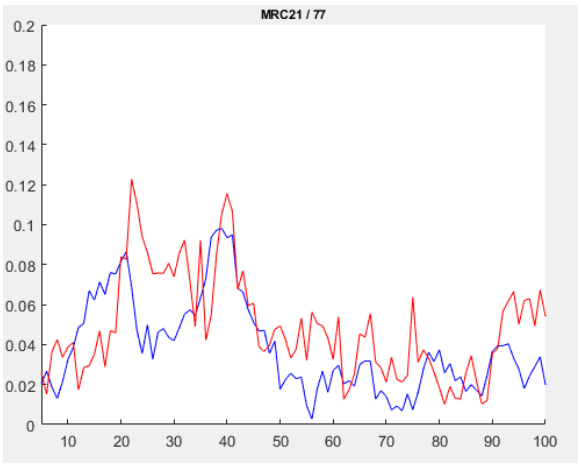
Create the following configuration, and compute the coherence.

```
cfg            = [];
cfg.output     = 'powandcsd';
cfg.method     = 'mtmfft';
cfg.foilim     = [5 100];
cfg.tapsmofrq  = 5;
cfg.keeptrials = 'yes';
cfg.channel    = {'MEG' 'EMGlft'};
cfg.channelcmb = {'MEG' 'EMGlft'};
cfg.trials     = 1:50;
freq50         = ft_freqanalysis(cfg,data);

cfg            = [];
cfg.method     = 'coh';
cfg.channelcmb = {'MEG' 'EMG'};
fd50           = ft_connectivityanalysis(cfg,freq50);
```

now plot the data-

Plot the result

```
cfg                 = [];
cfg.parameter       = 'cohspctrm';
cfg.xlim            = [5 100];
cfg.ylim            = [0 0.2];
cfg.refchannel      = 'EMGlft';
cfg.channel         = 'MRC21';
figure; ft_singleplotER(cfg, fd, fd50);
```



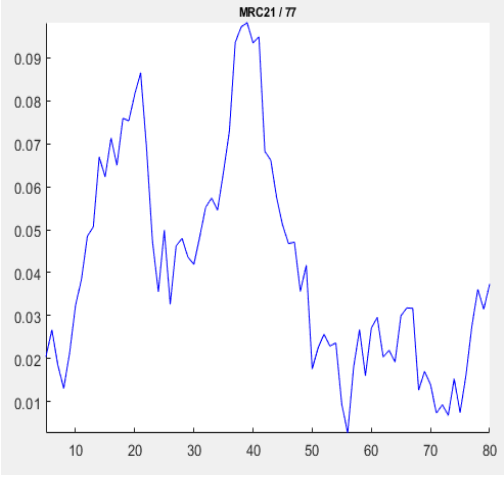


figure3

Compare the results with figure 3. Pay special attention to the noise bias.

We can clearly say that spectrum are coherent means they are at same phase difference.

Taking about noise figure 3 have more artifacts than current figure.