The PACCalculator uses methods described both in Tort et al (2010), Lafon et al (2017), Staresina et al (2015) and Hemptinne et al (2015).

**Calculating PAC Index**

calculatePAC - The method calculates the PAC Index either for two sets of data (e.g. recorded simultaneously from two brain regions) or for a single set of data (e.g. for quantifying interaction between oscillations of different frequencies on the same electrode). I will refer to the data set from which the phase is calculated as phase data and to the data set from which the amplitude is calculated as amplitude data, but as mentioned it is possible for some uses that the phase dataset and the amplitude dataset will be the same.

The method can also return preferred phase - the phase for which the amplitude is maximal (elaborated on later).

The method performs the following steps:

* Filters the phase data using a bandpass filter at the required frequencies range (a property of the class, can also be provided as input to the method).
* Filters the amplitude dataset using a bandpass filter at the required frequencies range (a property of the class, can also be provided as input to the method).
* Extracts the instantaneous phase from the filtered phase dataset using Hilbert transform (preserving NaN sample points by turning them to 0 before the transform and turning them back to NaN after).
* Extracts the instantaneous amplitude from the filtered amplitude dataset using Hilbert transform (preserving NaN sample points by turning them to 0 before the transform and turning them back to NaN after)

**Note:** Lafon use Morlet wavelet for filtering and finding instantaneous phase & amplitude.

* Removes outlier cycles using the method removeOutlierCycles (described later on)
* Optional: removes spikes (IIS) from the data. Spike times can be provided as input to the method (either a single array of indices if the phase dataset and the amplitude dataset are the same, or a cell array containing two arrays). IIS are removed by setting the points around the IIS to NaN (the duration of the window around the IIS that is set to NaN is a property of the class).
* Checks whether there is enough data left after the removal of the outliers and IIS. The parameter that sets the minimal duration of data required for PAC calculation is minSecToCalculatePacOn. If there isn’t enough data the PAC index will be NaN.
* Calculates PAC index using one of two methods (method type is a property): MI using the method calcMI, mean vector using the method calcMeanVector (described later on). These methods can also calculate the preferred phase if required.

calcMI - The index implemented in this method is described in Tort et al (page 2). It is calculated using the following steps:

* Calculates the average amplitude per discretized phase value (phase values are discretized to N bins, as set by the property numBins), using the method buildAmpDistPerPhase. The average amplitudes are then normalized to resemble a distribution.
* Calculates the KL distance between this “distribution” and the uniform distribution. The MI index is the KL distance / log(N) (N - the number of bins, the purpose of dividing by logN is that the index will be between 0 to 1).

calcMeanVector - The index implemented in this method is decribed by Tort et al (page 4) for the non-equalized version, and by Lafon et al (page 11) for the equalized version (that is also the index used by Lafon).

* The non equalized version is simply the amplitude of the average vector of the series Amp(t)\*exp(i\*phase(t)). In the equalized version we first calculate the average amplitude per discretized phase (using buildAmpDistPerPhase) and then calculate the amplitude of the average vector of Amp(bin index)\*exp(i\*Phase(bin index)).

removeOutlierCycles - Outlier removal is generally based on what is described in Lafon (page 11), the exact way to set the outlier thresholds is different in different articles, thus the class allows several options.

A single cycle is the stretch of data between instantaneous phase -pi to instantaneous phase pi (i.e. - what in the filtered slow wave data is a single oscillation from one trough to the next trough).

There are two types of outliers that are removed: cycles in which the filtered amplitude data (i.e. fast frequency data) is abnormally large and cycles in which the filtered phase data (i.e. slow wave data) is abnormally small. The thresholds to define what is considered abnormally large/small can be either predefined or set relatively to the distribution of all the cycles in the current dataset. The predefined threshold can be set using “human logic” (as they do in Lafon for the slow waves) or relative to the distribution of all the cycles in a larger data set by using first the methods setThreshAmpFast and setThreshAmpSlow (the PAC index is usually calculated on a small stretch of time, which is why it’s useful to calculate the thresholds first on a larger stretch from the same electrode and only then perform PAC calculations).

Outlier removal steps (Note: amplitude data refers to the data on which the instantaneous amplitude is calculated, after the bandpass filter but before the hilbert transform. Similarly phase data refers to the data on which the instantaneous phase is calculated, after the bandpass filter but before the hilbert transform):

* Divides the data to cycles (by finding points of instantaneous phase change from pi to -pi).
* For each cycle - calculates the mean squared amplitude over the cycle for the amplitude data.
* For each cycle - calculates the maximal value over the cycle of the phase data (note: absolute value at max, not peak minus trough)
* Cycles in which the mean squared amplitude of the amplitude data is above a threshold are marked as outliers. The threshold is either predefined (if the property thresholdForFastAmpOutlier is not NaN) or calculated on the current dataset as nSigmasForFastAmpOutlier above mean.
* Cycles in which the max value of the phase data is below a threshold are marked as outliers. The threshold is either predefined using prior considerations (as in Lafon, if the property usePredefinedThresholdForSlowAmpOutlier is true), predefined using calculation on a larger dataset (if the property usePredefinedThresholdForSlowAmpOutlier is false and thresholdForSlowAmpOutlier is not NaN) or calculated on the current dataset as the low percentile defined by percentileForSlowAmpOutlier (i.e. will leave only the largest X% of slow waves, where X is set by percentileForSlowAmpOutlier).

setThreshAmpFast & setThreshAmpSlow - These methods should be used in advance to set the low and high thresholds for outlier detection as described above – for the fast waves the threshold is set as nSigmasForFastAmpOutlier above the mean, for the slow waves the threshold is set as the low percentile defined by percentileForSlowAmpOutlier. The input should be a larger set of data from the same electrodes (for setting the low threshold the only input data required is the one from which the phase will be extracted, for setting the high threshold the input should be both data from which the phase is extracted and the data from which the amplitude is extracted. If only one dataset is provided the method assumes that the phase and amplitude are both extracted from this dataset).

**Finding Preferred Phase**

The preferred phase is the phase for which the amplitude is maximal, it is an output of calculatePAC if the class property findPreferredPhase is set to true. The algorithm for finding the peak (i.e. fitting a sinus function) is not based on any of the papers. The method which performs the actual calculation is buildAmpDistPerPhase, in the following way:

* The method calculates the average amplitude per discretized phase value (phase values are discretized to N bins, as set by the property numBins).
* It fits a cosinus model to the result using the method createCosFitModel and returns the estimated phase at the peak of the cosinus function (i.e., the fitted model is a\*cos(x+b)+c and the output is b).
* If the property plotDist is set to true the method will also plot the graph of the mean amplitude vs the phase (on which the preferred phase, and the PAC indices, are calculated).

**Calculating Comodulogram**

calculateComodulogram - The comodulogram is a concept described in Tort et al (page 8-9). It’s a matrix of the PAC indices calculated for various combinations of frequency ranges used for filtering the phase data and for filtering the amplitude data.

The properties relevant to the calculated comodulogram:

For the phase data filtering: phaseFrequencyBandwidth (band width of each frequency band), phaseFrequencyStepSize (step size between centers of frequency bands), maxFreqToCheckPhase (maximal frequency that will be the center of a frequency band), minFreqToCheckPhase (minimal frequency that will be the center of a frequency band). Similarly for the amplitude data filtering: ampFrequencyBandwidth, ampFrequencyStepSize, maxFreqToCheckAmp.

For each combination of frequency bands the PAC index is calculated and stored in the output matrix. The method will also plot the comodulogram using scaled colors (unless the parameter toPlot is false).

calculateMeanPacOnComodulogram – The method calls calculateComodulogram using the class default properties and then returns the average of the resulting matrix. The output is thus the mean PAC index for a range of slow and fast frequency ranges, which is what Hemptinne et al use in their paper.

**Calculating significance of a PAC index**

The significance calculation is based on Lafon et al (pages 11-12). The method receives the PAC index and the input datasets (and IIS times if removal is required) and returns the p-value by finding the distribution of PAC indices on the same dataset but with randomly shifted phases for each cycle. The steps are:

* Calculates instantaneous phase and amplitude as in calculatePAC (including IIS removal and outliers removal).
* Divides the data to cycles (A single cycle is the stretch of data between instantaneous phase -pi to instantaneous phase pi).
* Performs N iterations (the number is set by nItersPerPValue). In each iteration:
  + For each cycle the instantaneous phase is shifted by a random number between 0 to pi, chosen once for each cycle (i.e. for each t in the cycle, new\_phase(t) = old\_phase(t)+random\_number(cycle)).
  + The PAC index is calculated on the jumbled phase data and the (unchanged) amplitude data.
* The p-value is calculated relative to the “random” PAC indices distribution built by jumbling the phase data.

**Test method - effect of spikes on PAC**

plotEffectOfNSpikesOnPAC - The method creates simulated signals with a changing number of spikes added to them and plots the effect of the number of spikes on the calculated PAC.

* The signal is created using simulateSignal (similar to synthetic examples in Tort, fig 2, fig 6). The strength of the correlation (i.e. the size of the PAC index) is controlled by the input parameter multConst. For each number of spikes n in the tested spike number range nItersPerSpikes iterations are performed, in each iteration a simulated signal is created and n randomly located spikes are planted into it. In each iteration the PAC index is calculated for the signal with the spikes with and without spikes removal. Simulated signal:
  + The simulated phase signal is cos(phaseFreq\*t), the simulated amplitude signal is cos(ampFreq\*t)\*(0.5+multConst\*phaseSignal). phaseFreq and ampFreq are class properties, multConst is an input parameter.
  + The spike shape was previously extracted from real data (only one shape is used), n randomly located spikes are planted into the data (assuring that the data remains continuous).
* The mean PAC index (with and without spikes removal) is calculated over the iterations for each number of spike, and the results are plotted.

**Synchronization Index Calculation**

calcPreferredAnglesSI – The method calculates the angles of the synchronization index over the slow waves, based on Staresina et al 2015. The method receives the phase data and the amplitude data, the slow wave times (either as a vector of trough times, or as an array of start and end times for each slow wave – as is the output of the SlowWaveDetector). As usual the amplitude data can be left empty and will be set to be equal to the phase data. The calculation steps are:

* Filters the phase data using a bandpass filter at the required frequencies range (a property of the class, can also be provided as input to the method).
* Filters the amplitude dataset using a bandpass filter at the required frequencies range (a property of the class, can also be provided as input to the method).
* Extracts the instantaneous phase from the filtered phase dataset using Hilbert transform (preserving NaN sample points by turning them to 0 before the transform and turning them back to NaN after).
* Extracts the instantaneous amplitude from the filtered amplitude dataset using Hilbert transform (preserving NaN sample points by turning them to 0 before the transform and turning them back to NaN after).
* Extracts the instantaneous phase of the instantaneous amplitude of the amplitude data, by performing Hilbert transform on the result of the previous step (after normalization using zscore).
* For every slow wave:
  + Find the trough index (if not provided as input) by finding the minimal point on the filtered low-range data.
  + The current slow wave epoch indices are: timeAroundSlowWave seconds before and after the trough (1 second by default).
  + Calculate the Synchronization Index on the instantaneous phase of the low range data ( and the instantaneous phase of the instantaneous amplitude of the high range data (: , where N is the number of sample points in the epoch.
  + Find the angle of the synchronization index.
* Plot and return the angles of the SIs over all slow waves.

**Plotting Average TFR around events**

plotAvgSpecDiff plots the average time frequency representation over all the events of specific type (e.g. slow waves, spindles) in the data, as in Staresina et al 2015. The method receives the data and the event indices (for slow waves - either as a vector of trough indices, as an array of start and end times for each slow wave – as is the output of the SlowWaveDetector, otherwise as an array of indices).

The method calculates the TFR in the epoch around each event – timeBeforeAfterEvent before and after the event (by default 1 second), and subtracts from it a baseline TFR. The baseline spectrogram is the average spectrogram in the timeForBaseline seconds before the epoch (by default 1 second). The spectrogram is calculated using the FieldTrip function ft\_specest\_mtmconcol (as in Staresina et al).

Output:

meanSpec – the average TFR around the events

meanEpochs – the average of the data around the event

allSpec – all the TFRs (matrix of size number of epochs \* freq range \* time range)

nEpochs – how many events are included in the average

**Test Method – Artificial Data**

The method tests the output of the PAC calculation and the SI angles calculation on artificial data with set properties. The artificial data is as follows: a pure cosinus wave (representing the slow waves) with a slow frequency (by default 0.5 Hz), on which a spindle cut from real data is added. The spindle is received as input and is then normalized such that its absolute amplitude is at max maxSpindleAmp (set by default to 0.3). There are three sets of artificial data created:

* 1. Troughs – the spindle is always added at the troughs of the slow wave.
  2. Peaks - the spindle is always added at the peaks of the slow wave.
  3. Random – the spindle is added at random locations.

The artificial data is then used as input to PAC calculation and SI angles calculation. We would expect the preferred phase to be pi for condition A, 0 for condition B and random for condition C.

**Appendix**:

Simulated data and effect of spikes on PAC index calculation:

Upper trace shows the original signal, lower trace after band-pass filter for the high frequency range with the signal’s envelope. On the right – the average amplitude per phase (normalized so that the sum equals 1). The PAC index type used is MI, segLength is 40 seconds, the slow wave frequency is 1 Hz, the fast waves frequency is 15 Hz.

|  |  |
| --- | --- |
|  |  |
| MultConst = 0, PAC index = 1.3\*10^(-6)  MultConst = 0.1, PAC index = 0.0032 |  |
| MultConst = 0.2, PAC index = 0.0128 |  |
| MultConst = 0.3, PAC index = 0.0294 |  |

PAC Index as a function of multConst for the simulations (recreation of Tort et al Figure 2):



simulateSignal also adds spikes artificially to the data. The spike shape used is:

Examples of artificial data with implanted spikes (for multConst = 0):





The effect of the number of spikes in the data on the PAC index is examined in the method plotEffectOfNSpikesOnData (documented above):

Each row is for a different multConst parameter (i.e. strength of coupling), the left panel shows the PAC index as a function of the number of added spikes without spikes removel, the right panel shows the PAC index as a function of the number of added spikes with spikes removal. Note the scales for the PAC index on the left and right panel are different. The simulated segments length is 40 seconds.

|  |
| --- |
|  |
|  |
|  |
|  |

It can be seen that spikes cause the calculated PAC index to be smaller and a lot more variable than PAC index of the data without spikes, and that their removal solves the issue, but the variability of the PAC index still becomes larger as the number of removed spikes grows larger (note the scales are different on the left and right panel, so still the variability is much smaller after the removal).

Calculating preferred phase using PAC vs using SI angles

The class allows two options for calculating the preferred phase of the spindle (high frequency) vs the slow waves (low frequency):

* 1. By calculating PAC according to Tort et al and finding the preferred phase, this is done using the method calculatePAC, which can return as its second output the preferred phase if the class property findPreferredPhase is set to true.
  2. By calculating the SI per slow wave using the method calcPreferredAnglesSI and then finding the mean of the output (i.e. the mean preferred phase over all slow waves).

The two methods correspond well, the next figures show the preferred phase as calculated using SI vs the preferred phase as calculated using PAC for two patients – p487 and p489 (note that preferred phase is in radians, i.e. -pi and pi are the same angle thus the two seeming outliers for p489 are not actual outliers):



Some examples:

P487 – sleep\_data\_cell{6}, preferred PAC phase = -1.508 (273.6 degrees), preferred SI phase = -1.56



P487 – sleep\_data\_cell{1}, preferred PAC phase = 1.9464 (111.5 degrees), preferred SI phase = 1.8



P489 - sleep\_data\_cell{11}, preferred PAC phase = 3.13 (179 degrees), preferred SI phase = -3.046

 

P489 - sleep\_data\_cell{7}, preferred PAC phase = 2.25 (129 degrees), preferred SI phase = 2.68

 