The Chronux Manual

Peter Andrews
Hemant Bokil
Sumanjit Kaur
Catherine Loader
Hiren Maniar
Samar Mehta
Partha Mitra
Hariharan Nalatore
Ramesh Yadav
Ravi Shukla

August 16, 2008

Contents

Chapter 1

Chronux Overview

1.1 Introduction

Neuroscientists are increasingly gathering large time series data sets in the form of multichannel electrophysiological recordings, EEG, MEG, fMRI and optical image time series. The availability of such data has brought with it new challenges for analysis, and has created a pressing need for the development of software tools for storing and analyzing neural signals. In fact, while sophisticated methods for analyzing multichannel time series have been developed over the past several decades in statistics and signal processing, the lack of a unified, user-friendly, platform that implements these methods is a critical bottleneck in mining large neuroscientific datasets.

Chronux is an open source software platform that aims to fill this lacuna by providing a comprehensive software platform for the analysis of neural signals. It is a collaborative research effort currently based at Cold Spring Harbor Laboratory that draws on a number of previous research projects [1, 2, 3, 4, 5, 7, 8, 9]. The current version of Chronux includes a Matlab toolbox for signal processing of neural time series data, several specialized mini-packages for spike sorting, local regression, audio segmentation and other tasks. It also includes a graphical user interface (GUI). The current version of the GUI contains a number of features specialised to the analysis of electroencephalography (EEG) data. The eventual aim is to provide domain specific user interfaces (UIs) for each experimental modality, along with corresponding data management tools. In particular, we expect Chronux to grow to support analysis of time series data from most of the

standard data acquisition modalities in use in neuroscience. We also expect it to grow in the types of analyses it implements. Chronux is supported by grant R01MH071744 from the NIH to Partha P. Mitra.

1.2 Installation and Setup

The Chronux website at `http://chronux.org/` is the central location for information about the current and all previous releases of Chronux. The home page contains links to pages for downloads, people, recent news, tutorials, various files, documentation and our discussion forum. Most of the code is written in the Matlab scripting language, with some exceptions as compiled C code integrated using Matlab mex functions. Chronux has been tested and runs under Matlab releases R13 to the current R2008a under the Windows, Macintosh and Linux operating systems. Extensive online and within-Matlab help is available. The code is available as a smaller zip file (without data for testing) and a larger zip file that contains testing data.

To install Chronux, first unzip the zip archive into any location on your computer. Then the the Matlab path must be set to include the Chronux directory and all subdirectories (recursively) contained within. All Chronux functions and help information are then available from the Matlab command line.

Besides Matlab itself, Chronux requires the Matlab Signal Processing Toolbox for proper operation. The specscope utility depends upon the Matlab Data Acquisition Toolbox as well. The Locfit and spikesort subpackages utilize Matlab mex functions, which are pre-compiled and included for the Windows and Linux platform. For the Mac platform recompilation of the locfit and spikesort subpackages is currently necessary.

As an open source project released under the GNU Public License GPL v2, we welcome development, code contributions, bug reports, and discussion from the community. To date, Chronux has been downloaded over 10000 times. Questions or comments about can be posted on our discussion forum at `http://chronux.org/forum/` (after account registration). Announcements are made through the google group chronux-announce.

1.3 Documentation and Help

Documentation and help material for the Chronux package are provided in a few different formats. The latest and most complete type of documentation is the document you are currently reading. It contains this section on Chronux usage in Matlab, and a cross-referenced comprehensive function reference at the end. This document is expected to be the starting point for help for new Chronux users.

Once Chronux is installed on your computer you will also be able to access help material for each Chronux function from within the Matlab environment using the Matlab help command. The help material for any Chronux function can be accessed by typing `help function-name`. For example, to obtain help for the `mtspectrumc` Chronux function, type `help mtspectrumc` at the Matlab prompt. The output will typically contain the purpose and expected usage of the function, with the format and allowed values for input arguments explained in detail. The output (if any) of the function is also described. Note that this material is only available through the Matlab command line using the help command, and will not show up in searches in the GUI-based help system in Matlab.

The Chronux function reference is also available online in hyperlinked html format at `http://chronux.org/documentation/`. This reference material displays the purpose, usage, help text, source code and cross-referencing information for all Chronux functions. It is created directly from the Chronux source code using the Matlab utility m2html. For each Chronux subdirectory a dependency graph is shown which helps users to understand the relationship between Chronux functions in a given subdirectory. This function reference is also included with the Chronux distribution in the documentation subdirectory.

Finally, several tutorials for using Chronux are available at

`http://chronux.org/tutorials/`,

organized by the tutorial creator. Tutorials are provided which cover signal processing and spectral analysis theory, Chronux usage for spectral analysis and local regression. Other tutorials go into detail on spectral analysis and SVDs for neural time series. Another tutorial covers the practical aspects of detrending neural data and line (or noise) removal. The final tutorial covers Image processing using SVDs and spectral analysis. Most of these tutorials consist of a PDF or powerpoint presentation plus actual code which you can execute as you read the presentation.

1.4 Chronux Organization

On installing chronux, you should have a folder named chronux on your computer. Since the main components of Chronux are the spectral analysis toolbox, the local regression and likelihood toolbox and the spike-sorting toolbox, these are subdirectories of the chronux folder. In addition, the chronux folder also contains subdirectories for a specscope and wavebrowser and Chronux GUI. Finally, it contains script called testscript in the test directory that runs a test on each of the spectral analysis routines and a subdirectory called tutorials that contains the tutorials mentioned above. As the code and teaching material evolves, these directories will be get updated, as will this manual.

Chapter 2

Spectral Analysis Toolbox

The spectral analysis toolbox is the heart of chronux and is perhaps its most widely used component. It computes the spectrum of one or more time series data as well as the coherence between two simultaneously measured time series. In addition, it computes multivariate measures such as the cross-spectral matrix between multiple simultaneously measured time series. It also enables computation of spectral derivatives. Finally, it allows the user to perform harmonic analysis (identification and extraction of periodic components) on time series data. All these computations are performed using the multi-taper spectral estimation method. Where possible, Chronux provides confidence intervals on estimated quantities using both asymptotic formulae based on appropriate probabilistic models, as well as nonparametric bands based on the Jackknife method. Chronux includes various statistical tests such as the two-group tests for the spectrum and coherence and a test for nonstationarity based on quadratic inverse theory.

2.1 Data Formats and Parameters

Chronux spectral analysis routines are supplied to operate on three basic types of input data: continuous valued data and point process data supplied as individual times or as binned counts. Accordingly, for each type of analysis there will generally be three separate functions available, depending upon the input data type. For convenience, each such function is identified by an appropriate suffix: c for continuous, pt for point times and pb for binned point processes. For example, the multitaper spectrum function is available

in the three forms: `mtspectrumc`, `mtspectrumpt` and `mtspectrumpb`. For the functions where two data inputs of different format are used, two suffixes indicate the accepted data formats. For example, `cohgramcpb` is the multitaper time-frequency coherence between continuous and binned point data.

Continuous data is simply any data stream where the measurements are provided at evenly sampled intervals. This type of data must be input to Chronux routines in matrix form where the first dimension is time, and the second dimension is channels or trials. Binned point process data is supplied to the routines in the same matrix format as continuous data, but in this case the values for each element are interpreted as counts. Point process data supplied as times, on the other hand, must be input to Chronux routines as a structure array of spike times (with field name 'times'), with dimension of channels or trials. The difference in format stems from the fact that each channel or trial for spike time data will generally have a different length because the number of spikes recorded will vary. This makes a standard Matlab matrix inappropriate for the spike time data format. For single-channel point time data an ordinary column vector can be used instead. Note that point process data is usually derived from a continuous data stream in hardware or software.

A number of frequently used and important spectral analysis parameters to Chronux functions are passed for convenience in the Matlab structure called params. The fields in the params structure are all optional, and are named tapers, Fs, fpass, pad, err and trialave. We will discuss each of these in detail in the following paragraphs.

- The tapers field of the params structure describes the scale and number of tapers used in the multitaper calculation. This field is given as a Matlab array `[NW K]`, where NW is the time-bandwidth product and K is the number of leading tapers to use. The default value here is `[3 5]`. You should generally set K to be $2NW - 1$, as using more tapers will include tapers with poor concentration in the specified frequency bandwidth. The user may also specify this field as a 3 element array `[W T p]`. In this case, W is interpreted as the bandwidth, T the duration over which the tapers are to be computed and p is an integer such that $K = 2TW - p$ tapers are used. Note that T and W have to be consistent with other quantities such as the field Fs.

- The Fs field of the params structure describes the sampling frequency

of the input data, and controls the units used in the output. By default, an Fs of 1 indicates a sampling frequency of 1 sample per second. Take care that the value of Fs and the fpass and movingwin parameters (described below) are given in consistent units.

- The fpass field of the params structure controls the range of frequencies supplied in the output. This is given as a two element array with the lower and upper bounds of frequency output. By default, the value of fpass is `[0 Fs/2]`. These values, if supplied, must be given in units consistent with the Fs parameter.

- The pad field of the params structure controls the amount of padding used by the fast fourier transform (FFT) routine. This field is an integer from -1 and up. The value -1 results in no padding, the value 0 results in padding the data length to the next power of 2, the value 1 goes a further power of 2, etc. Padding the data to a length of a power of 2 improves the efficiency of the FFT algorithm, and increases the number of frequency bins of the result. Although this procedure does not affect the result calculation in any way, it does produce a more finely interpolated output which may assist with visualization and the precise identification of spectral lines. The default value for this parameter is 0.

- The err field of the params structure controls the type of error bars calculated (if any) for the output. It is supplied as a two element matrix where the first element gives the type of error calculation and the second element gives the p value used for the calculation. The supported error calculation types are 0 for no error bar calculations, 1 for theoretical error bars and 2 for jackknife error bars. The default value for the error type is 0 for no error bars. Note that requesting error bars (where available) will result in an additional output result from the function.

- The trialave field of the params structure governs whether or not trial or channel averaging is performed on the quantity of interest. If trialave is set to 0, no trial averaging is done, and the function will output independent results for each trial or channel passed as input data. If trialave is set to 1, the results passed to the user are averaged over trials or channels. The default for trialave is 0, or no averaging. Note

that setting trialave to 1 will result in a lower-dimensional output to the user.

Another common parameter encountered in Chronux routines, although not in the params structure, is the movingwin parameter. The movingwin parameter is used for the time-frequency version of the spectral analysis routines where the quantity of interest is calculated as a function of time. Instead of calculating a single spectrum of a dataset, the evolution of the spectrum versus time can be determined by calculating the spectrum over many small time windows. The result is frequently plotted as an image or a 3-dimensional contour or surface plot. The movingwin parameter is given as a two element array where the first element is the size of the moving window and the second is the step size to advance the window. Both of these values must be given in units consistent with params.Fs.

2.2 Examples

Space constraints preclude covering all of spectral analysis here, but the functions generally have a uniform function calling signature. We illustrate three canonical routines below.

2.2.1 mtspectrumc

As a first example, we show how to estimate the spectrum of a single trial local field potential measured from macaque during a working memory task. Figure 2.1 shows the spectrum estimated by the Chronux function `mtspectrumc`. For comparison we also display the ordinary periodogram. In this case, `mtspectrumc` was called with `params.tapers=[5 9]`. The

The Matlab calling signature of the `mtspectrumc` function is as follows:

```
[S,f,Serr] = mtspectrumc( data, params );
```

The first argument is the data matrix in the form of times * trials or channels, while the second argument *params* is a Matlab structure defining the sampling frequency[1], the time-bandwidth product used to compute the

[1]The current version of Chronux assumes continuous valued data to be uniformly sampled

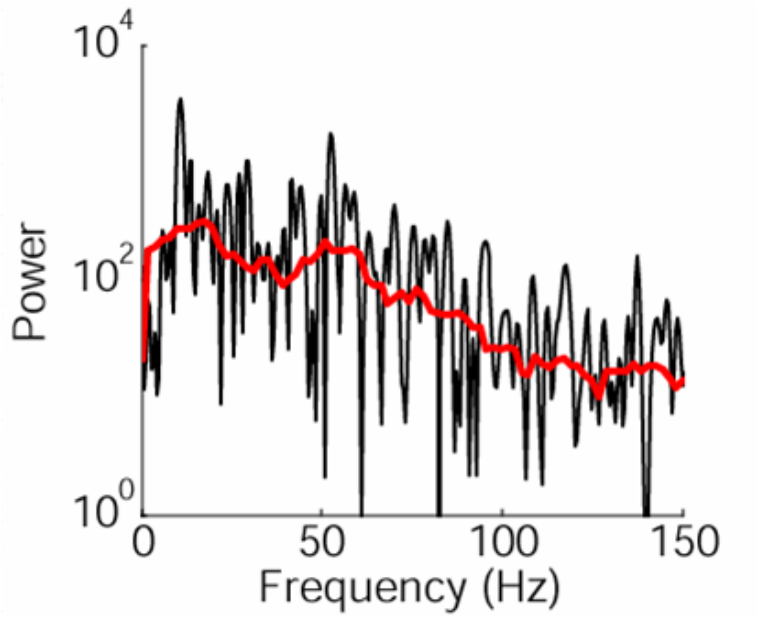

Figure 2.1: Comparison of a periodogram (black) and multitaper estimate (red or light) of a single trial local field potential measurement from macaque during a working memory task. This estimate used 9 tapers.

tapers, and the amount of zero padding to use. It also contains flags controlling the averaging over trials and the error computation. In this example, params.tapers was set to be `[5 9]`, thus giving an estimate with a time bandwidth product 5, using 9 tapers (For more details on this argument, see below).

The three variables returned by `mtspectrumc` are, in order, the estimated spectrum, the frequencies of estimation, and the confidence bands. The spectrum is in general two-dimensional, with the first dimension being the power as a function of frequency and the second dimension being the trial or channel. The second dimension is 1 when the user requests a spectrum that is averaged over the trials or channels. The confidence bands are provided as a lower and upper confidence band at a p value set by the user. As indicated by the last letter c in its name, the routine `mtspectrumc` is applicable to continuous valued data such as the local field potential or the EEG. The corresponding routines for point processes are `mtspectrumpt` and `mtspectrumpb`, applicable to point processes stored as a sequence of times

and binned point processes, respectively. In addition to these routines, it is also sometimes useful to compute the spectrum by breaking the data into short segments and averaging the estimates over those segments. Such a computation is performed by the routines `mtspectrumsegc`, `mtspectrumsegpb` and `mtspectrumsegpt`.

The rountines for point processes have one or two extra input arguments compared to those for continous processes. One of these arguments is `fscorr` which takes a value 0 if the user wants to compute Jackknife confidence bands with a correction taking into account the number of spikes. This optional argument is available for all point process rountines where Jackknife confidence bands are computed. Another argument, `t` is an input argument only for point processes stored as spike times. This optional argument may contain the grid over which a point process fourier transform is to be computed. This is useful because for a point process stored as times, simply knowing the grid spacing is not enough to compute the tapers. One also needs to know the width of the window. For example, given a sequence of spike times $0.2, 0.6, 0.8$ seconds, the spectrum for the case when the trial was from 1 second long is different from that in the case where the trial was 2 seconds long. This difference may be entered by specifying `t`.

2.2.2 mtspecgramc

The second example is a moving window version of `mtspectrumc` called `mtspecgramc`. This function, and `mtspecgrampt` and `mtspecgrampb`, calculate the multitaper spectrum over a moving window with user adjustable time width and step size. The calling signature of this function is:

```
[S,t,f,Serr] =
mtspecgramc( data, movingwin, params );
```

Note that the only differences from the `mtspectrumc` function signature are in the function name, the additional `movingwin` argument and the addition of a return value `t` which contains the centers of the moving windows. The `movingwin` argument is given as `[winsize winstep]` in units consistent with the sampling frequency. The returned spectrum here is in general three dimensional: times * frequency * channel or trial.

The variable params.tapers controls the computation of the tapers used in the multitaper estimate. params.tapers is a two-dimensional vector whose

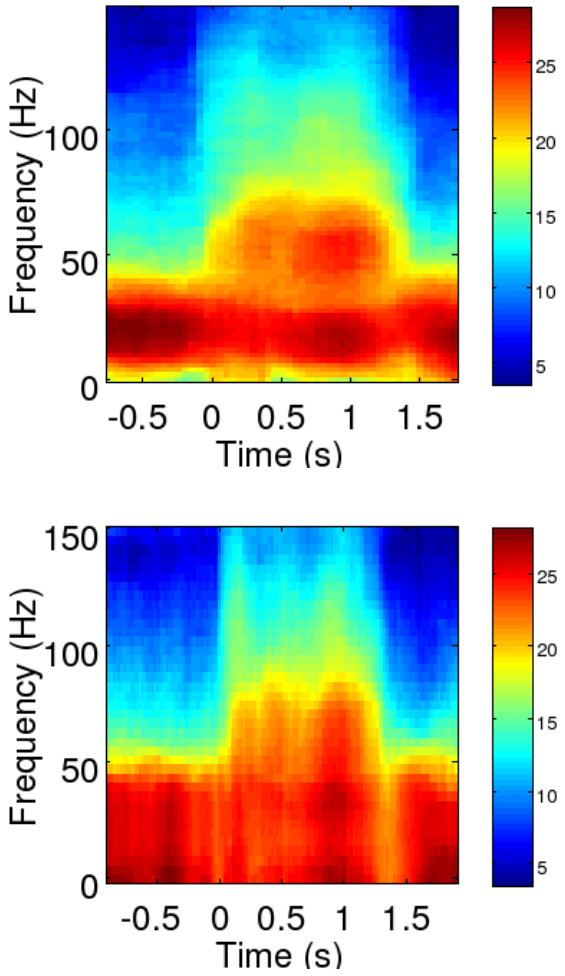

Figure 2.2: The effect of changing the components of the time-bandwidth product TW. a) T = 0.5s, W = 10Hz. b) T = 0.2s, W = 25Hz. Data from macaque monkey performing a working memory task. Sharp enhancement in high frequency power occurs during the memory period.

first element, TW, is the time-bandwidth product, where T is the duration and W is the desired bandwidth. The second element of params.tapers is the number of tapers to be used. For a given TW, the latter can be at most $2TW - 1$. Higher order taper functions will not be sufficiently concentrated in frequency and may lead to increased broadband bias if used. As discussed

above, params.tapers may also be supplied as a 3 element vector `[W T p]`. In this case, there are no defaults and it is the user's responsibility to ensure consistency between these inputs and any other relevant ones. For example, when computing a spectrogram, T has to equal the duration of the moving window given by `movingwin(1)`. Figure 2.2 shows the effect on the spectrogram of changing the time-bandwidth product. The data again consists of local field potentials recorded from macaque during a working memory task.

2.2.3 coherencycpt

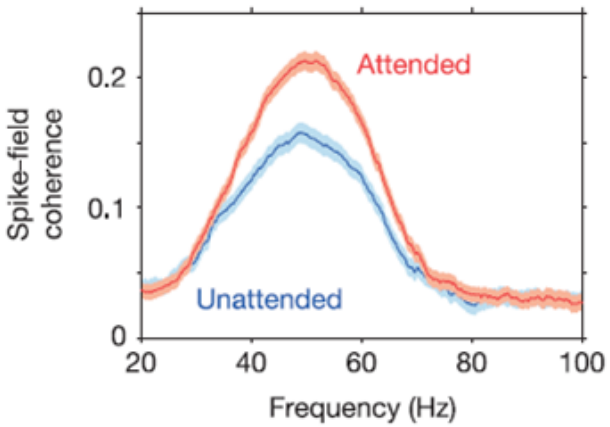

Figure 2.3: The spike-field coherence recorded from visual cortex of monkey, showing significant differences between the attended and unattended conditions. In addition to the coherence in the two conditions, we also show the 95% confidence bands computed using the Jackknife. We thank Pascal Fries for permission to use this figure.

Figure 2.3 [10] shows significant differences in the spike-field coherence recorded from the primary visual cortex of monkeys during an attention modulation task. The coherences were computed using `coherencycpt`. This function is called with two timeseries as arguments: the continuous LFP data and the corresponding spikes which are stored as event times. It returns not only the magnitude and phase of the coherency, but the cross-spectrum and individual spectra from which the coherence is computed. As with the spectra, confidence intervals on the magnitude and phase of the coherency may also be obtained. The calling signature here is

```
[C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=
   coherencycpt(data1,data2,params,fscorr,t)
```

`confC` contains a confidence level based on the assumption of zero population coherence, `phistd` contains the standard deviation of the phase estimate and `Cerr` contains Jackknife estimates of the confidence bands around C if they are asked for by the user. The output argument zerosp contains 1 for trials in which there were no spikes and 0 otherwise.

Chapter 3

Locfit

Local regression and likelihood provide a powerful set of methods of fitting functions to data. Local regression refers to regression using moving windows and local likelihood refers to a generalization of this to allow for non-Gaussian errors. The Locfit package by Catherine Loader [6] is included in Chronux. Locfit can be used for local regression, local likelihood estimation, local smoothing, density estimation, conditional hazard rate estimation, classification and censored likelihood models. Locfit is written in C and is available in Chronux through a mex interface to Matlab. From a neuroscience user's standpoint, the most important routines in Locfit are

- The rountine locfit.m computes the fits, whether they be for regression or density estimation.
- The rountine lfband.m computes confidence bands.
- The routine lfplot.m generate plots of the fit.

In addtion, Locfit also has routines to compute cross-validation scores based on the likelihood, the generalized cross-validation scores and the Akaike Information Criterion scores.

3.1 Data Format and Parameters

Locfit is a fairly extensive and involved package and full discussion of its capabilities can be found in the book by Loader [6]. We restrict our discussion to illustrating local regression based smoothing and local likelihood based

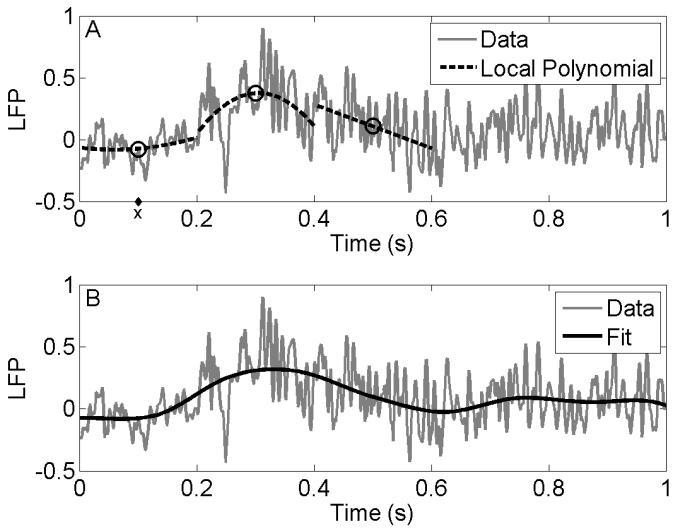

Figure 3.1: Schematic depiction of local regression. The data is a voltage segment recorded from area LIP of macaque. A. In a window of duration 0.2 s centered on the point x, a second order polynomial is fitted to the data using weighted least squares. The constant term of the polynomial fit centered on at x is taken to be the estimate $\hat{\mu}(x)$. We show local ploynomial fits (degree two) in three windows as examples. As the window slides along the data, a smooth fit is generated. B. Local regression fit generated by the procedure described above.

rate estimation using Locfit. Finally, we illustrate the use of cross-validation based assesment of the appropriate bandwidth.

3.1.1 Local Regression

Figure 3.1 shows a local regression based fit to a voltage segment recorded from area LIP of macaque. The top panel shows third order polynomial fits of the data in non-overlapping windows of 0.2 second duration. The bottom panel shows the complete fit. In this case, the independent variable was time and the dependent variable was the voltage and the calling sequence was

```
fit=locfit(t,V, 'h',0.2)
```

The first and second argument is the dependent. Here, the name is 'h', which is the fixed bandwidth chosen to be equal to 0.2. Note that the units of h have to be consistent with the units of the independent variable. Locfit may also be used with a bandwidth that depends on the number of points in the vicinity, as illustrated by the next example.

3.1.2 Local Likelihood

Figure 3.2 is an example of Locfit estimate of the rate given a sequence of spike times. In this case, there is just one variable, the spike times and Locfit was used to estimate the rate based on a local Poisson likelihood. The local rate was modeled as a cubic polynomial. Also shown are the Locfit computed 95% local confidence bands around the smoothed rate estimate. For comparison a histogram is also shown. In this case, the calling sequence was

```
fit=locfit(t,'deg', 3, 'nn',0.35)
```

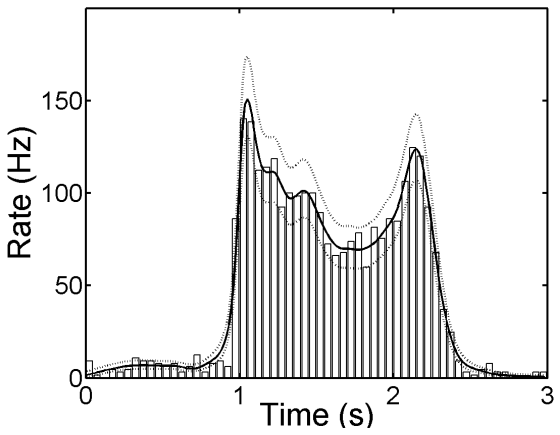

Figure 3.2: A traditional binned histogram and a Locfit smoothed estimate of the same data set. The Locfit estimate shown uses a nearest-neighbor fraction of 35% when calculating the distribution.

Here, the names are 'deg' which controls the polynomial degree (the default is 2) and 'nn' which denotes a nearest-neighbour bandwidth. With the name, value pair 'nn', 0.35, the local window is chosen to include 35% of the total

number of points. Thus, the window size decreases with increasing density of points. Locfit also allows both variable and fixed bandwidths in the form either of name,value pairs as above, or as a name,value pair involving a 2 element vector 'alpha'. In this case, the first argument of 'alpha' is taken to be the variable bandwidth and the second is taken to be the fixed bandwidth. Note also that the larger of the two is used in the fit.

3.1.3 Model Selection

The essential idea behind model selection in Locfit is to use cross-validation. One drops one point from the data in turn and estimates (by some measure) how well the fit with the remaining points fits the dropped data point. Locfit provides lilelihood based cross-validation (LCV) scores and an approximation to it known as the generalized cross-validation (GCV) score which is easier to compute. Figure 3.3 shows the GCV score for the data in Figure 3.1. Note that it is conventional to plot the score vs. the degrees of freedom rather than the bandwidth. The reason for this is that the meaning of the bandwidth depends on the smoothing method and the degrees of freedom is a more unambiguous measure. In our case, using about $12 - 15$ degrees of freedom seems reasonable. This corresponds to a bandwidth of around $200 - 250$ samples. Figure 3.1 was computed with a bandwidth of 200 samples. The calling sequence in this case was

```
gcvplot(alpha,t,V)
```

where *alpha* was a column vector of fixed bandwidths from 0.2 to 0.6. Variable bandwidths may also be specified in a second column of α.

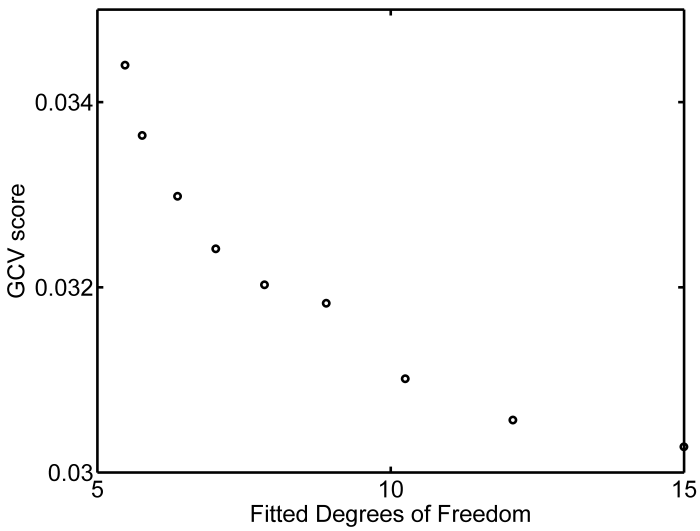

Figure 3.3: GCV score vs. Degrees of Freedom for the data in Figure 3.1.

Chapter 4

Function Reference

4.1 CrossSpecMatc

Purpose:
Multi-taper cross-spectral matrix - another routine, allows for multiple trials
and channels

Synopsis:
`function [Sc,Cmat,Ctot,Cvec,Cent,f]=CrossSpecMatc(data,win,params)`

Comments:

```
Multi-taper cross-spectral matrix - another routine, allows for multiple trials and channels
Does not do confidence intervals. Also this routine always averages over trials - continuous process

Usage:

[Sc,Cmat,Ctot,Cvec,Cent,f]=CrossSpecMatc(data,win,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (in form samples x channels x trials)
      win  (duration of non-overlapping window)
      params: structure with fields tapers, pad, Fs, fpass
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
```

```
                    tapers to be used (less than or equal to
                    2TW-1).
                (2) A numeric vector [W T p] where W is the
                    bandwidth, T is the duration of the data and p
                    is an integer such that 2TW-p tapers are used. In
                    this form there is no default i.e. to specify
                    the bandwidth, you have to specify T and p as
                    well. Note that the units of W and T have to be
                    consistent: if W is in Hz, T must be in seconds
                    and vice versa. Note that these units must also
                    be consistent with the units of params.Fs: W can
                    be in Hz if and only if params.Fs is in Hz.
                    The default is to use form 1 with TW=3 and K=5

         pad            (padding factor for the FFT) - optional. Defaults to 0.
                    e.g. For N = 500, if PAD = 0, we pad the FFT
                    to 512 points; if PAD = 2, we pad the FFT
                    to 2048 points, etc.
        Fs   (sampling frequency) - optional. Default 1.
        fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
Output:
      Sc (cross spectral matrix frequency x channels x channels)
      Cmat Coherence matrix frequency x channels x channels
      Ctot Total coherence: SV(1)^2/sum(SV^2) (frequency)
      Cvec leading Eigenvector (frequency x channels)
      Cent A different measure of total coherence: GM/AM of SV^2s
      f (frequencies)
```

This function calls:

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

This function is called by:

none No functions call this function

4.2 CrossSpecMatpb

Synopsis:
`function [Sc,Cmat,Ctot,Cvec,Cent,f]=CrossSpecMatpb(data,win,params)`

Comments:

```
Multi-taper cross-spectral matrix - another routine, this one allows for multiple trials and channels
Does not do confidence intervals.
Also this routine always averages over trials - binned point process

Usage:

[Sc,Cmat,Ctot,Cvec,Cent,f]=CrossSpecMatpb(data,win,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (in form samples x channels x trials)
      win  (duration of non-overlapping window)
      params: structure with fields tapers, pad, Fs, fpass
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
Output:
      Sc (cross spectral matrix frequency x channels x channels)
```

```
Cmat Coherence matrix frequency x channels x channels
Ctot Total coherence: SV(1)^2/sum(SV^2) (frequency)
Cvec leading Eigenvector (frequency x channels)
Cent A different measure of total coherence: GM/AM of SV^2s
f (frequencies)
```

This function calls:

mtfftpb (Section 4.56) Multi-taper fourier transform - binned point process data

This function is called by:

none No functions call this function

4.3 CrossSpecMatpt

Synopsis:
`function [Sc,Cmat,Ctot,Cvec,Cent,f]=CrossSpecMatpt(data,win,T,params)`

Comments:

```
Multi-taper cross-spectral matrix - another routine, this one allows for multiple trials and channels
but does not do confidence intervals. Also this routine always averages
over trials - point process as times

Usage:

[Sc,Cmat,Ctot,Cvec,Cent,f]=CrossSpecMatpt(data,win,T,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (as a struct array with dimensions channels x trials) - note
      that times of measurement have to be consistent, we assume all
      times are specified relative to the start time of the trials which
      are taken to be zero.
      win  (duration of non-overlapping window)
      trialduration (since it is not possible to infer trial duration
              from spike times, this is an optional argument. If not specified
              the routine uses the minimum and maximum spike time (across all
              channels and trials) as the window of calculation.) -
              optional
      params: structure with fields tapers, pad, Fs, fpass
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
```

```
                    to 512 points, if pad=1, we pad to 1024 points etc.
                    Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
Output:
      Sc (cross spectral matrix frequency x channels x channels)
      Cmat Coherence matrix frequency x channels x channels
      Ctot Total coherence: SV(1)^2/sum(SV^2) (frequency)
      Cvec leading Eigenvector (frequency x channels)
      Cent A different measure of total coherence: GM/AM of SV^2s
      f (frequencies)
```

This function calls:

extractdatapt (Section 4.38) Extract segements of spike times between t(1) and t(2)

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

mtfftpt (Section 4.57) Multi-taper fourier transform for point process given as times

This function is called by:

none No functions call this function

4.4 binspikes

Purpose:
bin spikes at a specified frequency sampling i.e. sampling rate 1/sampling

Synopsis:
`function [dN,t]=binspikes(data,Fs,t)`

Comments:

```
bin spikes at a specified frequency sampling i.e. sampling rate 1/sampling
eg: 1ms accuracy use sampling = 1000
Usage: [dN,t]=binspikes(data,Fs,t)
Inputs:
data   (data as a structure array of spike times; or as a single
       vector of spike times)
Fs     (binning frequency)
t      (the minimum and maximum times to be used to form the bins - [mint maxt]
           - optional. Default use the spike times themselves to
             determine the location of the bins.
Note: the times in data can be in any units. However, it is important
that all units are chosen consistently. So, if spike times are in secs,
Fs and t (if present) have to be in Hz and secs respectively. If spike
times are in number of samples, Fs has to be 1, and t has to be in number
of samples.
Outputs:
dN     (output binned spike counts as a matrix defined on bins starting with the
        earliest spike across all channels and ending with the latest spike)
t      (lower limit of each bin)
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.5 change_row_to_column

Purpose:
Helper routine to transform 1d arrays into column vectors that are needed

Synopsis:
`function data=change_row_to_column(data)`

Comments:

```
Helper routine to transform 1d arrays into column vectors that are needed
by other routines in Chronux

Usage: data=change_row_to_column(data)

Inputs:
data -- required. If data is a matrix, it is assumed that it is of the
form samples x channels/trials and it is returned without change. If it
is a vector, it is transformed to a column vector. If it is a struct
array of dimension 1, it is again returned as a column vector. If it is a
struct array with multiple dimensions, it is returned without change
Note that the routine only looks at the first field of a struct array.

Ouputs:
data (in the form samples x channels/trials)
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.6 check_consistency

Purpose:
Helper routine to check consistency of data dimensions

Synopsis:
`function [N,C]=check_consistency(data1,data2,sp)`

Comments:

```
Helper routine to check consistency of data dimensions
Usage: [N,C]=check_consistency(data1,data2,sp)
Inputs:
data1 - first dataset
data2 - second dataset
sp - optional argument to be input as 1 when one of the two data sets is
spikes times stored as a 1d array.
Outputs:
Dimensions of the datasets - data1 or data2 (note that
   routine stops with an error message if dimensions don't match - [N,C]
   N left empty for structure arrays
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.7 chronux

Purpose:
This library performs time-frequency analysis (mostly using the

Synopsis:
`function chronux`

Comments:

```
This library performs time-frequency analysis (mostly using the
multi-taper spectral estimation method) of univariate and multivariate
data, both for continuous processes such as LFP/EEG and for point
processes such as spike times. Point process can either be stored as
times or as a binned process of counts. The routines in this library
are named differently for the three cases. For calculations
that can be performed for each of the three data types, we use suffixes
c, pb, or pt to refer to continuous, point process binned counts, or
point process times. For example, the spectrum calculation is performed
mtspectrumc for continuous processes, mtspectrumpb for a binned point
process, and mtspectrumpt for a point process consisting of times. There
are also routines for calculating hybrid quantities involving one continuous
and one point process. These are suffixed in a similar manner. For
example, coherencycpb calculates the coherency between a binned point process
and a continuous process.

Certain variables are used repeatedly in this library.

DATA
data in most cases can be univariate or multivariate, and either point process,
or continuous.

     Continuous data: Continuous data is assumed to be a matrix with
                      dimensions samples x channels/trials.

     Point Process: A single time series of spike times can be in the form of
                    a column vector.
                    Multichannel/trial spike time data is not amenable to this
                    storage format, since there are generally different
                    number of spikes in each channel/trial. Instead,
                    multichannel/trial spike data is stored in a structure
                    array. A structure is a matlab data object with various
                    fields. These fields contain the elements
                      e.g. The command data=struct('times',[]); creates an empty
                           structure with field 'times'. Similarly, the command
                           data=struct('times',[1 2 3]); creates the structure with
                           the field 'times' containing integers 1, 2, and 3.

                    We can also have a structure array (or an array of structures)
                    defined for example, by
                    data(1)=struct('times',rand(1,100)); and
                    data(2)=struct('times',rand(1,200));
```

```
                    This is a 2 dimensional structure array where the
                    first field is a 100 dimensional random vector, and
                    the second field is a 200 dimensional random vector.
                    This format allows storage of multichannel point
                    process times in a single variable data.

                    The above holds for point processes stored as times.
                    If instead, the point processes are binned, then one
                    can use a matrix to represent them


     Summary: data - array of continuous data with dimensions time x channels
                     structural array of spike times with dimensions
                              equal to the number of channels
                     1d array of spike times as a column vector
                     array of binned spike counts with dimensions time x channels

PARAMETERS:
These are various parameters used in the spectral calculations. Since
these parameters are used by most routines in Chronux, they are stored in
a single structure params. The fields of params are

tapers : precalculated tapers from dpss or in the one of the following
         forms:
         (1) A numeric vector [TW K] where TW is the
             time-bandwidth product and K is the number of
             tapers to be used (less than or equal to
             2TW-1).
         (2) A numeric vector [W T p] where W is the
             bandwidth, T is the duration of the data and p
             is an integer such that 2TW-p tapers are used. In
             this form there is no default i.e. to specify
             the bandwidth, you have to specify T and p as
             well. Note that the units of W and T have to be
             consistent: if W is in Hz, T must be in seconds
             and vice versa. Note that these units must also
             be consistent with the units of params.Fs: W can
             be in Hz if and only if params.Fs is in Hz.
             The default is to use form 1 with TW=3 and K=5


pad:   (padding factor for the FFT) - optional (can take values -1,0,1,2...).
        -1 corresponds to no padding, 0 corresponds to padding
        to the next highest power of 2 etc.
             e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                  to 512 points, if pad=1, we pad to 1024 points etc.
                  Defaults to 0.

Fs:sampling frequency.optional (default 1)


fpass: frequencies in an fft calculation can range from 0 to Fs/2 where
       Fs is the sampling frequency. Sometimes it may be useful to
       compute fourier transforms (and resulting quantities like the
```

```
       spectrum over a smaller range of frequencies). This is specified
       by fpass, which can be in the form [fmin fmax] where fmin &gt;=0 and
       fmax&lt;=Fs/2. optional (default [0 Fs/2])

err=[errtype p] calculates theoretical error bars (confidence levels)
                when errtype=1 and jackknife error bars when errchk=2. In each case, the
                error is calculated at a p value specified by p. -
                optional (default 0)

trialave: trialave controls whether or not to average over channels/trials for
          multichannel/trial analyses. trialave=0 (default) implies no trial
          averaging, trialave=1 implies that the quantity of interest is averaged
          over channels/trials. optional (default 0)

Other parameters are discussed in individual routines as and when they
are used.
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.8 coherencyc

Purpose:
Multi-taper coherency,cross-spectrum and individual spectra - continuous process

Synopsis:
`function [C,phi,S12,S1,S2,f,confC,phistd,Cerr]=coherencyc(data1,data2,params)`

Comments:

```
Multi-taper coherency,cross-spectrum and individual spectra - continuous process

Usage:
[C,phi,S12,S1,S2,f,confC,phistd,Cerr]=coherencyc(data1,data2,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data1 (in form samples x trials) -- required
      data2 (in form samples x trials) -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad             (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
```

```
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
Output:
      C (magnitude of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      phi (phase of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S12 (cross spectrum -  frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S1 (spectrum 1 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S2 (spectrum 2 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      f (frequencies)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi.
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

This function is called by:

coherencysegc (Section 4.14) Multi-taper coherency, cross-spectrum and individual spectra with segmenting - continuous process

cohgramc (Section 4.20) Multi-taper time-frequency coherence,cross-spectrum and individual spectra - continuous processes

4.9 coherencyc_unequal_length_trials

Purpose:
This routine computes the average multi-taper coherence for a given set of unequal length segments. It is

Synopsis:
```
function [Cmn,Phimn,Smn,Smm,f,ConfC,PhiStd,Cerr] = coherencyc_unequal_length_trials(
data, movingwin, params, sMarkers )
```

Comments:

```
This routine computes the average multi-taper coherence for a given set of unequal length segments. It is
based on modifications to the Chronux routines. The segments are continuously structured in the
data matrix, with the segment boundaries given by markers. Below,
movingwin is used in a non-overlaping way to partition each segment into
various windows. Th coherence is evaluated for each window, and then the
window coherence estimates averaged. Further averaging is conducted by
repeating the process for each segment.

Inputs:

  data = data( samples, channels )- here segments must be stacked
  as explained in the email
  movingwin = [window winstep] i.e length of moving
              window and step size. Note that units here have
              to be consistent with units of Fs. If Fs=1 (ie normalized)
              then [window winstep]should be in samples, or else if Fs is
              unnormalized then they should be in time (secs).
  sMarkers = N x 2 array of segment start &amp;amp; stop marks. sMarkers(n, 1) = start
          sample index; sMarkers(n,2) = stop sample index for the nth segment
  params = see Chronux help on mtspecgramc

Output:
      Cmn     magnitude of coherency - frequencies x iChPairs
      Phimn   phase of coherency - frequencies x iChPairs
      Smn     cross spectrum -  frequencies x iChPairs
      Smm     spectrum m - frequencies x channels
      f       frequencies x 1
      ConfC   1 x iChPairs; confidence level for Cmn at 1-p % - only for err(1)&amp;gt;=1
      PhiStd  frequency x iChPairs; error bars for phimn - only for err(1)&amp;gt;=1
      Cerr    2 x frequency x iChPairs; Jackknife error bars for Cmn - use only for Jackknife - err(1)=2

      Here iChPairs = indices corresponding to the off-diagonal terms of the
      lower half matrix. iChPairs = 1 : nChannels*(nChannels-1)/2. So,
      iChPairs=1,2,3,4,...correspond to C(2,1), C(3,1), C(3,2), C(4,1), etc.
      The mapping can be obtained as follows:

      C(i,j) = Cmn(:,k) where k = j + (1/2)*(i-1)*(i-2)

      The above also applies to phimn, Smn
```

```
Note:
  segment length &gt;= NW/2 where NW = half bandwidth parameter (see dpss). So the power spectrum will
  be computed only for those segments whose length &gt; NW/2. For that reason, the routine returns the
  indices for segments for which the spectra is computed. This check is
  done here since pSpecgramAvg calls it.
```

This function calls:

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

This function is called by:

none No functions call this function

4.10 coherencycpb

Purpose:
Multi-taper coherency,cross-spectrum and individual spectra - continuous and binned point process data

Synopsis:
`function [C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencycpb(data1,data2,params,fscorr)`

Comments:

```
Multi-taper coherency,cross-spectrum and individual spectra - continuous and binned point process data

Usage:

[C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencycpb(data1,data2,params,fscorr)
Inputs:
      data1 (continuous data in form samples x trials) -- required
      data2 (binned spike data in form samples x trials) -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
```

```
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or 1
             (use finite size corrections) - optional (available only for spikes). Defaults 0.
Outputs:
      C (magnitude of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      phi (phase of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S12 (cross spectrum -  frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S1 (spectrum 1 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S2 (spectrum 2 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      f (frequencies)
      zerosp (1 for trials where no spikes were found, 0 otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife -  err(1)=2)
```

This function calls:

none This function calls no functions

This function is called by:

coherencysegcpb (Section 4.15) Multi-taper coherency,cross-spectrum and individual spectra with segmenting

cohgramcpb (Section 4.21) Multi-taper time-frequency coherence,cross-spectrum and individual spectra

4.11 coherencycpt

Purpose:
Multi-taper coherency,cross-spectrum and individual spectra -continuous data
and point process as times

Synopsis:
```
function [C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencycpt(data1,data2,params,fscorr,t
```

Comments:

```
Multi-taper coherency,cross-spectrum and individual spectra -continuous data and point process as times

Usage:

[C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencycpt(data1,data2,params,fscorr,t)
Input:
      data1        (continuous data in time x trials form) -- required
      data2        (structure array of spike times with dimension trials;
                    also accepts 1d array of spike times) -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
```

```
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
                                      1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
      t        (time grid over which the tapers are to be calculated:
                     this argument is useful when calling the spectrum
                     calculation routine from a moving window spectrogram
                     calculation routine). If left empty, the spike times
                     are used to define the grid.
Output:
      C (magnitude of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      phi (phase of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S12 (cross spectrum -  frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S1 (spectrum 1 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S2 (spectrum 2 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      f (frequencies)
      zerosp (1 for trials where no spikes were found, 0 otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

none This function calls no functions

This function is called by:

coherencysegcpt (Section 4.16) Multi-taper coherency,cross-spectrum and individual spectra computed by segmenting

cohgramcpt (Section 4.22) Multi-taper time-frequency coherence,cross-spectrum and individual spectra

4.12 coherencypb

Purpose:
Multi-taper coherency,cross-spectrum and individual spectra - binned point process

Synopsis:
```
function [C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencypb(data1,data2,params,fscorr)
```

Comments:

```
Multi-taper coherency,cross-spectrum and individual spectra - binned point process

Usage:

[C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencypb(data1,data2,params,fscorr)
Input:
      data1 (in form samples x trials) -- required
      data2 (in form samples x trials) -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                 Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
```

```
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
                1 (use finite size corrections) - optional
                (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      phi (phase of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S12 (cross spectrum -  frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S1 (spectrum 1 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S2 (spectrum 2 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      f (frequencies)
      zerosp (1 for trials in either channel where spikes were absent, zero otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - jackknife/theoretical standard deviation for phi.  Note that
               phi + 2 phistd and phi -2 phistd will give 95% confidence bands
               for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

mtfftpb (Section 4.56) Multi-taper fourier transform - binned point process data

This function is called by:

coherencysegpb (Section 4.17) Multi-taper coherency,cross-spectrum and individual spectra computed by segmenting

cohgrampb (Section 4.23) Multi-taper time-frequency coherence,cross-spectrum and individual spectra - two binned point processes

4.13 coherencypt

Purpose:
Multi-taper coherency - point process times

Synopsis:
`function [C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencypt(data1,data2,params,fscorr,t)`

Comments:

```
Multi-taper coherency - point process times

Usage:

[C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencypt(data1,data2,params,fscorr,t)
Input:
      data1  (structure array of spike times with dimension trials; also accepts 1d array of spike times) --
      data2  (structure array of spike times with dimension trials; also accepts 1d array of spike times) --
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
```

```
      fscorr   (finite size corrections, 0 (don't use finite size corrections)
               or 1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
      t        (time grid over which the tapers are to be calculated:
                     this argument is useful when calling the spectrum
                     calculation routine from a moving window spectrogram
                     calculation routine). If left empty, the spike times
                     are used to define the grid.
Output:
      C (magnitude of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      phi (phase of coherency - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S12 (cross spectrum -  frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S1 (spectrum 1 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      S2 (spectrum 2 - frequencies x trials if trialave=0; dimension frequencies if trialave=1)
      f (frequencies)
      zerosp (1 for trials where no spikes were found, 0 otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

mtfftpt (Section 4.57) Multi-taper fourier transform for point process given as times

This function is called by:

coherencysegpt (Section 4.18) Multi-taper coherency computed by segmenting two univariate point processes into chunks

cohgrampt (Section 4.24) Multi-taper time-frequency coherence - two point processes given as times

4.14 coherencysegc

Purpose:
Multi-taper coherency, cross-spectrum and individual spectra with segmenting - continuous process

Synopsis:
`function [C,phi,S12,S1,S2,f,confC,phistd,Cerr]=coherencysegc(data1,data2,win,params)`

Comments:

```
Multi-taper coherency, cross-spectrum and individual spectra with segmenting - continuous process
  computed by segmenting two univariate time series into chunks

Usage:
[C,phi,S12,S1,S2,f,confC,phistd,Cerr]=coherencysegc(data1,data2,win,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data1 (column vector) -- required
      data2 (column vector) -- required
      win   (length of segments) - required
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
```

```
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
Output:
      C (magnitude of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      phi (phase of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S12 (cross spectrum -  frequencies x segments if segave=0; dimension frequencies if segave=1)
      S1 (spectrum 1 - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S2 (spectrum 2 - frequencies x segments if segave=0; dimension frequencies if segave=1)
      f (frequencies)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

coherencyc (Section 4.8) Multi-taper coherency,cross-spectrum and individual spectra - continuous process

createdatamatc (Section 4.30) Helper function to create an event triggered matrix from univariate

This function is called by:

none No functions call this function

4.15 coherencysegcpb

Purpose:
Multi-taper coherency,cross-spectrum and individual spectra with segmenting

Synopsis:
```
function [C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencysegcpb(data1,data2,win,params,s
```

Comments:

```
Multi-taper coherency,cross-spectrum and individual spectra with segmenting
  computed by segmenting two univariate time series into chunks - continuous and binned point process

Usage:
[C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencysegcpb(data1,data2,win,params,segave,fscorr)
Input:
Note units have to be consistent. See chronux.m for more information.
      data1 (column vector, continuous data) -- required
      data2 (column vector, binned point process data) -- required
      win   (length of segments) - required
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
```

```
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
      segave (average over segments for 1, don't average for 0)
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      phi (phase of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S12 (cross spectrum -  frequencies x segments if segave=0; dimension frequencies if segave=1)
      S1 (spectrum 1 - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S2 (spectrum 2 - frequencies x segments if segave=0; dimension frequencies if segave=1)
      f (frequencies)
      zerosp (1 for trials where no spikes were found, 0 otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

coherencycpb (Section 4.10) Multi-taper coherency,cross-spectrum and individual spectra - continuous and binned point process data

This function is called by:

none No functions call this function

4.16 coherencysegcpt

Purpose:
Multi-taper coherency,cross-spectrum and individual spectra computed by segmenting

Synopsis:
```
function [C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencysegcpt(data1,data2,win,params,s
```

Comments:

```
Multi-taper coherency,cross-spectrum and individual spectra computed by segmenting
  two univariate time series into chunks - continuous and point process stored as times

Usage:
[C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencysegcpt(data1,data2,win,params,segave,fscorr)
Input:
Note units have to be consistent. See chronux.m for more information.
      data1 (column vector, continuous data) -- required
      data2 (1d structure array of spike times; also accepts 1d array of spike times) -- required
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
```

```
                              [0 p] or 0 - no error bars) - optional. Default 0.
      segave (average over segments for 1, don't average for 0)- optional. Default 1
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      phi (phase of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S12 (cross spectrum -  frequencies x segments if segave=0; dimension frequencies if segave=1)
      S1 (spectrum 1 - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S2 (spectrum 2 - frequencies x segments if segave=0; dimension frequencies if segave=1)
      f (frequencies)
      zerosp (1 for trials where no spikes were found, 0 otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

coherencycpt (Section 4.11) Multi-taper coherency,cross-spectrum and individual spectra -continuous data and point process as times

This function is called by:

none No functions call this function

4.17 coherencysegpb

Purpose:
Multi-taper coherency,cross-spectrum and individual spectra computed by segmenting

Synopsis:
```
function [C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencysegpb(data1,data2,win,params,se
```

Comments:

```
Multi-taper coherency,cross-spectrum and individual spectra computed by segmenting
  two univariate binned point processes into chunks

Usage:
[C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencysegpb(data1,data2,win,params,segave,fscorr)
Input:
Note units have to be consistent. See chronux.m for more information.
      data1 (column vector, binned point process data) -- required
      data2 (column vector, binned point process data) -- required
      win   (length of segments) - required
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
```

```
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
      segave (average over segments for 1, don't average for 0)
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      phi (phase of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S12 (cross spectrum -  frequencies x segments if segave=0; dimension frequencies if segave=1)
      S1 (spectrum 1 - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S2 (spectrum 2 - frequencies x segments if segave=0; dimension
      frequencies if segave=1)
      f (frequencies)
      zerosp (1 for segments where no spikes were found, 0 otherwise)
      confC (confidence level for C at 1-p %)
      phistd - jackknife/theoretical standard deviation for phi - Note that
               phi + 2 phistd and phi -2 phistd will give 95% confidence bands for phi -
               only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife)
```

This function calls:

coherencypb (Section 4.12) Multi-taper coherency,cross-spectrum and individual spectra - binned point process

createdatamatpb (Section 4.31)

This function is called by:

none No functions call this function

4.18 coherencysegpt

Purpose:
Multi-taper coherency computed by segmenting two univariate point processes into chunks

Synopsis:
```
function [C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencysegpt(data1,data2,win,params,se
```

Comments:

```
Multi-taper coherency computed by segmenting two univariate point processes into chunks

Usage:
[C,phi,S12,S1,S2,f,zerosp,confC,phistd,Cerr]=coherencysegpt(data1,data2,win,params,segave,fscorr)
Input:
Note units have to be consistent. See chronux.m for more information.
      data1 (1d structure array of spike times; also accepts 1d array of spike times) -- required
      data2 (1d structure array of spike times; also accepts 1d array of spike times) -- required
      win   (length of segments) - required
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
```

```
                                [0 p] or 0 - no error bars) - optional. Default 0.
      segave - optional 0 for don't average over segments, 1 for average - default
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      phi (phase of coherency - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S12 (cross spectrum -  frequencies x segments if segave=0; dimension frequencies if segave=1)
      S1 (spectrum 1 - frequencies x segments if segave=0; dimension frequencies if segave=1)
      S2 (spectrum 2 - frequencies x segments if segave=0; dimension frequencies if segave=1)
      f (frequencies)
      zerosp (1 for segments where no spikes were found, 0 otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

coherencypt (Section 4.13) Multi-taper coherency - point process times

createdatamatpt (Section 4.32) Helper function to create an event triggered matrix from a single

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

This function is called by:

none No functions call this function

4.19 coherr

Purpose:
Function to compute lower and upper confidence intervals on the coherency

Synopsis:
`function [confC,phistd,Cerr]=coherr(C,J1,J2,err,trialave,numsp1,numsp2)`

Comments:

```
Function to compute lower and upper confidence intervals on the coherency
given the tapered fourier transforms, errchk, trialave.

Usage: [confC,phistd,Cerr]=coherr(C,J1,J2,err,trialave,numsp1,numsp2)
Inputs:
C     - coherence
J1,J2 - tapered fourier transforms
err - [errtype p] (errtype=1 - asymptotic estimates; errchk=2 - Jackknife estimates;
                   p - p value for error estimates)
trialave - 0: no averaging over trials/channels
           1 : perform trial averaging
numsp1    - number of spikes for data1. supply only if finite size corrections are required
numsp2    - number of spikes for data2. supply only if finite size corrections are required

Outputs:
         confC - confidence level for C - only for err(1)&gt;=1
         phistd - theoretical or jackknife standard deviation for phi for err(1)=1 and err(1)=2
                  respectively. returns zero if coherence is 1
         Cerr  - Jacknife error bars for C  - only for err(1)=2
```

This function calls:

none This function calls no functions

This function is called by:

cohmathelper (Section 4.25) Helper function called by coherency matrix computations.

4.20 cohgramc

Purpose:
Multi-taper time-frequency coherence,cross-spectrum and individual spectra
- continuous processes

Synopsis:
`function [C,phi,S12,S1,S2,t,f,confC,phistd,Cerr]=cohgramc(data1,data2,movingwin,params)`

Comments:

```
Multi-taper time-frequency coherence,cross-spectrum and individual spectra - continuous processes

Usage:

[C,phi,S12,S1,S2,t,f,confC,phistd,Cerr]=cohgramc(data1,data2,movingwin,params)
Input:
Note units have to be consistent. Thus, if movingwin is in seconds, Fs
has to be in Hz. see chronux.m for more information.

      data1 (in form samples x trials) -- required
      data2 (in form samples x trials) -- required
      movingwin (in the form [window winstep] -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5
                    Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
           Fs   (sampling frequency) - optional. Default 1.
```

```
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
Output:
      C (magnitude of coherency time x frequencies x trials for trialave=0;
            time x frequency for trialave=1)
      phi (phase of coherency time x frequencies x trials for no trial averaging;
            time x frequency for trialave=1)
      S12 (cross spectrum - time x frequencies x trials for no trial averaging;
            time x frequency for trialave=1)
      S1 (spectrum 1 - time x frequencies x trials for no trial averaging;
            time x frequency for trialave=1)
      S2 (spectrum 2 - time x frequencies x trials for no trial averaging;
            time x frequency for trialave=1)
      t (time)
      f (frequencies)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

coherencyc (Section 4.8) Multi-taper coherency,cross-spectrum and individual spectra - continuous process

This function is called by:

none No functions call this function

4.21 cohgramcpb

Purpose:
Multi-taper time-frequency coherence,cross-spectrum and individual spectra

Synopsis:
`function [C,phi,S12,S1,S2,t,f,zerosp,confC,phistd,Cerr]=cohgramcpb(data1,data2,movingwin,param`

Comments:

```
Multi-taper time-frequency coherence,cross-spectrum and individual spectra
continuous process and binned point process

Usage:

[C,phi,S12,S1,S2,t,f,zerosp,confC,phistd,Cerr]=cohgramcpb(data1,data2,movingwin,params,fscorr)
Input:
Note units have to be consistent. Thus, if movingwin is in seconds, Fs
has to be in Hz. see chronux.m for more information.

      data1 (continuous data in form samples x trials) -- required
      data2 (binned point process data in form samples x trials) -- required
      movingwin (in the form [window winstep] -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5
                   Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
            Fs   (sampling frequency) - optional. Default 1.
```

```
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
                1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency time x frequencies x trials for trialave=0;
          time x frequency for trialave=1)
      phi (phase of coherency time x frequencies x trials for no trial averaging;
          time x frequency for trialave=1)
      S12 (cross spectrum - time x frequencies x trials for no trial averaging;
          time x frequency for trialave=1)
      S1 (spectrum 1 - time x frequencies x trials for no trial averaging;
          time x frequency for trialave=1)
      S2 (spectrum 2 - time x frequencies x trials for no trial averaging;
          time x frequency for trialave=1)
      t (time)
      f (frequencies)
      zerosp (1 for windows and trials where no spikes were found, 0 otherwise: dimensions time x trials)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

coherencycpb (Section 4.10) Multi-taper coherency,cross-spectrum and individual spectra - continuous and binned point process data

This function is called by:

none No functions call this function

4.22 cohgramcpt

Purpose:
Multi-taper time-frequency coherence,cross-spectrum and individual spectra

Synopsis:
`function [C,phi,S12,S1,S2,t,f,zerosp,confC,phistd,Cerr]=cohgramcpt(data1,data2,movingwin,param`

Comments:

```
Multi-taper time-frequency coherence,cross-spectrum and individual spectra
  continuous process and point process times

Usage:

[C,phi,S12,S1,S2,t,f,zerosp,confC,phistd,Cerr]=cohgramcpt(data1,data2,movingwin,params,fscorr)
Input:
Note units have to be consistent. Thus, if movingwin is in seconds, Fs
has to be in Hz. see chronux.m for more information.

      data1 (continuous data in form samples x trials) -- required
      data2 (structure array of spike times with dimension trials;
             also accepts 1d array of spike times) -- required
      movingwin (in the form [window winstep] -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5
                   Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
```

```
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
                1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency time x frequencies x trials for trialave=0;
             time x frequency for trialave=1)
      phi (phase of coherency time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      S12 (cross spectrum - time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      S1 (spectrum 1 - time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      S2 (spectrum 2 - time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      t (time)
      f (frequencies)
      zerosp (1 for windows where no spikes were found, 0 otherwise;
              dimensions time x trials if no trial averaging)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

coherencycpt (Section 4.11) Multi-taper coherency,cross-spectrum and individual spectra -continuous data and point process as times

This function is called by:

none No functions call this function

4.23 cohgrampb

Purpose:
Multi-taper time-frequency coherence,cross-spectrum and individual spectra
- two binned point processes

Synopsis:
```
function [C,phi,S12,S1,S2,t,f,zerosp,confC,phistd,Cerr]=cohgrampb(data1,data2,movingwin,params
```

Comments:

```
Multi-taper time-frequency coherence,cross-spectrum and individual spectra - two binned point processes

Usage:

[C,phi,S12,S1,S2,t,f,zerosp,confC,phistd,Cerr]=cohgrampb(data1,data2,movingwin,params,fscorr)
Input:
Note units have to be consistent. Thus, if movingwin is in seconds, Fs
has to be in Hz. see chronux.m for more information.

      data1 (binned point process data in form samples x trials) -- required
      data2 (binned point process data in form samples x trials) -- required
      movingwin (in the form [window winstep] -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5
                  Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
```

```
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials when 1, don't average when 0) -
          optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
                1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency time x frequencies x trials for trialave=0;
             time x frequency for trialave=1)
      phi (phase of coherency time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      S12 (cross spectrum - time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      S1 (spectrum 1 - time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      S2 (spectrum 2 - time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      t (time)
      f (frequencies)
      zerosp (1 for windows and trials where spikes were absent (in either channel),zero otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - jackknife/theoretical standard deviation for phi - Note that
               phi + 2 phistd and phi -2 phistd will give 95% confidence bands for phi
                - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

coherencypb (Section 4.12) Multi-taper coherency,cross-spectrum and individual spectra - binned point process

This function is called by:

none No functions call this function

4.24 cohgrampt

Purpose:
Multi-taper time-frequency coherence - two point processes given as times

Synopsis:
`function [C,phi,S12,S1,S2,t,f,zerosp,confC,phistd,Cerr]=cohgrampt(data1,data2,movingwin,params`

Comments:

```
Multi-taper time-frequency coherence - two point processes given as times

Usage:

[C,phi,S12,S1,S2,t,f,zerosp,confC,phistd,Cerr]=cohgrampt(data1,data2,movingwin,params,fscorr)
Input:
Note units have to be consistent. Thus, if movingwin is in seconds, Fs
has to be in Hz. see chronux.m for more information.

      data1  (structure array of spike times with dimension trials; also accepts 1d array of spike times) --
      data2  (structure array of spike times with dimension trials; also accepts 1d array of spike times) --
      movingwin (in the form [window winstep] -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5
                   Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
      Fs   (sampling frequency) - optional. Default 1.
      fpass    (frequency band to be used in the calculation in the form
```

```
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
                1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency time x frequencies x trials for trialave=0;
             time x frequency for trialave=1)
      phi (phase of coherency time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      S12 (cross spectrum - time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      S1 (spectrum 1 - time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      S2 (spectrum 2 - time x frequencies x trials for no trial averaging;
             time x frequency for trialave=1)
      t (time)
      f (frequencies)
      zerosp (1 for windows and trials where spikes were absent (in either channel),zero otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

coherencypt (Section 4.13) Multi-taper coherency - point process times

extractdatapt (Section 4.38) Extract segements of spike times between t(1) and t(2)

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

This function is called by:

none No functions call this function

4.25 cohmathelper

Purpose:
Helper function called by coherency matrix computations.

Synopsis:
`function [C,phi,S12,confC,phierr,Cerr]=cohmathelper(J,err,Nsp)`

Comments:

```
Helper function called by coherency matrix computations.

Usage: [C,phi,S12,confC,phierr,Cerr]=cohmathelper(J,err,Nsp)
Inputs:
J   : Fourier transforms of data
err : [0 p] or 0 for no errors; [1 p] for theoretical confidence level,
      [2 p] for Jackknife (p - p value)
Nsp : pass the number of spikes in each channel if finite size corrections are desired

Outputs:

C   : coherence
phi : phase of coherency
S12 : cross spectral matrix
confC : confidence level for coherency - only for err(1)&gt;=1
      phierr - standard deviation for phi (note that the routine gives phierr as phierr(1,...)
               and phierr(2,...) in order to incorporate Jackknife (eventually).
               Currently phierr(1,...)=phierr(2,...). Note that phi + 2 phierr(1,...) and phi -2
               phierr(2,...) will give 95% confidence bands for phi - only for err(1)&gt;=1
Cerr  : error bars for coherency (only for Jackknife estimates)-only for err(1)=2
```

This function calls:

coherr (Section 4.19) Function to compute lower and upper confidence intervals on the coherency

This function is called by:

none No functions call this function

4.26 cohmatrixc

Purpose:
Multi-taper coherency,cross-spectral matrix - continuous process

Synopsis:
`function [C,phi,S12,f,confC,phistd,Cerr]=cohmatrixc(data,params)`

Comments:

```
Multi-taper coherency,cross-spectral matrix - continuous process

Usage:

[C,phi,S12,f,confC,phistd,Cerr]=cohmatrixc(data,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (in form samples x channels) -- required
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
Output:
```

```
C (magnitude of coherency frequency x channels x channels)
phi (phase of coherency frequency x channels x channels)
S12 (cross-spectral matrix frequency x channels x channels)
f (frequencies)
confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
         Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
         bands for phi - only for err(1)&gt;=1
Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

This function is called by:

none No functions call this function

4.27 cohmatrixpb

Purpose:
Multi-taper coherency matrix - binned point process

Synopsis:
`function [C,phi,S12,f,zerosp,confC,phistd,Cerr]=cohmatrixpb(data,params,fscorr)`

Comments:

```
Multi-taper coherency matrix - binned point process

Usage:

[C,phi,S12,f,zerosp,confC,phistd,Cerr]=cohmatrixpb(data,params,fscorr)
Input:
      data (in form samples x channels) -- required
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
```

```
               (available only for spikes). Defaults 0.
 Output:
       C (magnitude of coherency frequency x channels x channels)
       phi (phase of coherency frequency x channels x channels)
       S12 (cross-spectral matrix frequency x channels x channels)
       f (frequencies)
       zerosp (1 for channels where no spikes were found, zero otherwise)
       confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
       phistd - jackknife/theoretical standard deviation for phi - Note that
                phi + 2 phistd and phi -2 phistd will give 95% confidence bands for phi
                - only for err(1)&gt;=1
       Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

mtfftpb (Section 4.56) Multi-taper fourier transform - binned point process data

This function is called by:

none No functions call this function

4.28 cohmatrixpt

Purpose:
Multi-taper coherency matrix - point process times

Synopsis:
`function [C,phi,S12,f,zerosp,confC,phistd,Cerr]=cohmatrixpt(data,params,fscorr)`

Comments:

```
Multi-taper coherency matrix - point process times

Usage:

[C,phi,S12,f,zerosp,confC,phistd,Cerr]=cohmatrixpt(data,params,fscorr)
Input:
      data    (structure array of spike times with dimension channels) - required
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
```

```
               (available only for spikes). Defaults 0.
Output:
      C (magnitude of coherency frequency x channels x channels)
      phi (phase of coherency frequency x channels x channels)
      S12 (cross-spectral matrix frequency x channels x channels)
      f (frequencies)
      zerosp (1 for channels where no spikes were found, zero otherwise)
      confC (confidence level for C at 1-p %) - only for err(1)&gt;=1
      phistd - theoretical/jackknife (depending on err(1)=1/err(1)=2) standard deviation for phi
               Note that phi + 2 phistd and phi - 2 phistd will give 95% confidence
               bands for phi - only for err(1)&gt;=1
      Cerr  (Jackknife error bars for C - use only for Jackknife - err(1)=2)
```

This function calls:

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

mtfftpt (Section 4.57) Multi-taper fourier transform for point process given as times

This function is called by:

none No functions call this function

4.29 countsig

Purpose:
Give the program two spike data sets and one

Synopsis:
```
function[H,P,M1,M2,N1,N2] = countsig(data1,data2,T1,T2,parametric,p,quiet)
```

Comments:

```
Give the program two spike data sets and one
or two time intervals and it will decide if
the counts are significantly different.
this is either with a non-parametric method
or with a sqrt transformation followed by a
t-test
Usage: [H,P,M1,M2,N1,N2] = countsig(data1,data2,T1,T2,parametric,p,quiet)

Input:
Note that all times have to be consistent. If data
is in seconds, so must be sig and t. If data is in
samples, so must sig and t. The default is seconds.

data1      - structure array of spike times (required)
data2      - structure array of spike times (required)
T1         - time interval (default all)
T2         - time interval (default T1)
parametric - 0 = non-parametric (Wilcoxon)
           - 1 = ttest on sqrt of counts
           - 2 = Poisson assumption
             (default = 0)
p          - significance level (0.05)
quiet      - 1 = no display 0 = display

Output:

H          - 1 if different 0 if not
P          - prob of result if same
M1         - mean count for data1
M2         - mean count for data2
N1         - counts for data1
N2         - counts for data2
```

This function calls:

padNaN (Section 4.77) Creates a padded data matrix from input structural array of spike times

This function is called by:

none No functions call this function

4.30 createdatamatc

Purpose:
Helper function to create an event triggered matrix from univariate

Synopsis:
```
function data=createdatamatc(data,E,Fs,win)
```

Comments:

```
Helper function to create an event triggered matrix from univariate
continuous data
Usage: data=createdatamatc(data,E,Fs,win)
Inputs:
data   (input time series as a column vector) - required
E      (events to use as triggers) - required
Fs     (sampling frequency of data) - required
win    (window around triggers to use data matrix -[winl winr]) - required
         e.g [1 1] uses a window starting 1 * Fs samples before E and
             ending 1*Fs samples after E.
Note that E, Fs, and win must have consistent units

Outputs:
data      (event triggered data)
```

This function calls:

none This function calls no functions

This function is called by:

coherencysegc (Section 4.14) Multi-taper coherency, cross-spectrum and individual spectra with segmenting - continuous process

mtspecgramtrigc (Section 4.62) Multi-taper event triggered time-frequency spectrum - continuous process

mtspectrum_of_spectrumc (Section 4.65) Multi-taper segmented, second spectrum (spectrum of the log spectrum) for a continuous process

mtspectrumsegc (Section 4.70) Multi-taper segmented spectrum for a univariate continuous process

mtspectrumtrigc (Section 4.73) Multi-taper event triggered time-frequency spectrum - continuous process

4.31 createdatamatpb

Synopsis:
`function data=createdatamatpb(data,E,Fs,win)`

Comments:

```
Helper function to create an event triggered matrix from a single
channel of data.
Usage: data=createdatamatpb(data,E,Fs,win)
Inputs:
data   (input time series as a single vector) - required
E      (events to use as triggers) - required
Fs     (sampling frequency of data) - required
win    (window around triggers to use data matrix -[winl winr]) - required
         e.g [1 1] uses a window starting 1 sec before E and
             ending 1 sec after E if E is in secs
Note that E, Fs, and win must have consistent units

Outputs:
data      (event triggered data)
```

This function calls:

none This function calls no functions

This function is called by:

coherencysegpb (Section 4.17) Multi-taper coherency,cross-spectrum and individual spectra computed by segmenting

mtspecgramtrigpb (Section 4.63) Multi-taper event triggered time-frequency spectrum - binned point process

mtspectrumsegpb (Section 4.71) Multi-taper segmented spectrum for a univariate binned point process

mtspectrumtrigpb (Section 4.74) Multi-taper event triggered time-frequency spectrum - binned point process

4.32 createdatamatpt

Purpose:
Helper function to create an event triggered matrix from a single

Synopsis:
`function data=createdatamatpt(data,E,win)`

Comments:

```
Helper function to create an event triggered matrix from a single
channel of spike times.
Usage:  data=createdatamatpt(data,E,win)
Inputs:
data   (input spike times as a structural array or as a column vector) - required
E      (events to use as triggers) - required
win    (window around triggers to use data matrix -[winl winr]) - required
         e.g [1 1] uses a window starting 1 sec before E and
             ending 1 sec after E if E and data are in secs.
Note that E, win and data must have consistent units
Outputs:
data      (event triggered data as a structural array - times are stored
relative to the E-winl
```

This function calls:

none This function calls no functions

This function is called by:

coherencysegpt (Section 4.18) Multi-taper coherency computed by segmenting two univariate point processes into chunks

mtspecgramtrigpt (Section 4.64) Multi-taper event triggered time-frequency spectrum - point process times

mtspectrumsegpt (Section 4.72) Multi-taper segmented spectrum for a univariate binned point process

mtspectrumtrigpt (Section 4.75) Multi-taper time-frequency spectrum - point process times

4.33 den_jack

Purpose:
Function to compute smooth estimates of the mean of x using locfit,

Synopsis:
`function [m,ll,ul,llj,ulj]=den_jack(X,family,varargin)`

Comments:

```
Function to compute smooth estimates of the mean of x using locfit,
the corresponding confidence intervals, and jackknife estimates of
the confidence intervals
Usage: [m,ll,ul,llj,ulj]=den_jack(x)

Inputs:
X: data in the form samples x trials
family: 'density' or 'reg' for regression
       If the family is density, the entire input matrix X is considered
       as data. If the family is regression then the first column of X is
       taken to be the independent variable and the remaining columns are
       regressed on this variable (for example, the first column may be
       the centers of the bins for binned spike count data)
varargin is the set of arguments used by locfit to perform the smoothing

Outputs:
m : smoothed estimate of the mean
ll : estimate of the lower confidence level
ul : estimate of the upper confidence level
llj : jackknife estimate of the lower confidence level (+2\sigma
      where sigma is the jackknife variance)
llu : jackknife estimate of the upper confidence level (-2\sigma
      where sigma is the jackknife variance)
```

This function calls:

jackknife (Section 4.45) Compute jackknife estimates of the mean and standard deviation of input data x

This function is called by:

none No functions call this function

4.34 dpsschk

Purpose:
Helper function to calculate tapers and, if precalculated tapers are supplied,

Synopsis:
`function [tapers,eigs]=dpsschk(tapers,N,Fs)`

Comments:

```
Helper function to calculate tapers and, if precalculated tapers are supplied,
to check that they (the precalculated tapers) the same length in time as
the time series being studied. The length of the time series is specified
as the second input argument N. Thus if precalculated tapers have
dimensions [N1 K], we require that N1=N.
Usage: tapers=dpsschk(tapers,N,Fs)
Inputs:
tapers        (tapers in the form of:
                                (i) precalculated tapers or,
                                (ii) [NW K] - time-bandwidth product, number of tapers)

N             (number of samples)
Fs            (sampling frequency - this is required for nomalization of
                                  tapers: we need tapers to be such
                                  that integral of the square of each taper equals 1
                                  dpss computes tapers such that the
                                  SUM of squares equals 1 - so we need
                                  to multiply the dpss computed tapers
                                  by sqrt(Fs) to get the right
                                  normalization)
Outputs:
tapers        (calculated or precalculated tapers)
eigs          (eigenvalues)
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.35 evoked

Purpose:
Function to calculate the evoked response given continuous data in the

Synopsis:
```
function [V,t,Err] = evoked(data,Fs,win,width,plt,err)
```

Comments:

```
Function to calculate the evoked response given continuous data in the
form time x channels
Usage [V,t,Err] = evoked(data,Fs,win,width,plt,err)

Inputs
  Note that all times can be in arbitrary units. But the units have to be
  consistent. So, if win is in secs, width is in secs and Fs has to be Hz.
  If win is in samples, so is width and Fs=1.

   data(times, channels/trials or a single vector)      (required)
   Fs  sampling frequency            (required)
   win   subsection of data to be used. Default all available data
   width (s) of smoothing kernel. Default 50 samples
   plt plot 'n' for no plot, otherwise plot color. Default blue colored lines.
   err = 0/1. Default 1=calculate bootstrap errorbars.

Outputs
   V = evoked potential
   t = times of evaluation
   Err = bootstrap statdard deviation
```

This function calls:

locsmooth (Section 4.47) Running line fit (using local linear regression) - 1d only, continuous

This function is called by:

none No functions call this function

4.36 extractdatac

Purpose:
Extract segements of continuous data between t(1) and t(2)

Synopsis:
`function data=extractdatac(data,Fs,t)`

Comments:

```
Extract segements of continuous data between t(1) and t(2)
Usage: data=extractdatac(data,Fs,t)

Input:
data: continous data in the form samples x channels or a single vector
Fs: sampling frequency
t   : time as a 2d vector [t(1) t(2)]
Note that sampling frequency and t have to be in consistent units
Output:
data: data between t(1) and t(2)
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.37 extractdatapb

Purpose:
Extract segements of binned point process data between t(1) and t(2)

Synopsis:
`function data=extractdatapb(data,Fs,t)`

Comments:

```
Extract segements of binned point process data between t(1) and t(2)
Usage: data=extractdatapb(data,Fs,t)

Input:
data: binned point process data in the form samples x channels or single
vector
Fs: sampling frequency
t   : time as a 2d vector [t(1) t(2)]
Note that sampling frequency and t have to be in consistent units
Output:
data: data between t(1) and t(2)
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.38 extractdatapt

Purpose:
Extract segements of spike times between t(1) and t(2)

Synopsis:
`function data=extractdatapt(data,t,offset)`

Comments:

```
Extract segements of spike times between t(1) and t(2)
Usage: data=extractdatapt(data,t,offset)

Input:
data: structural array of spike times for each channel/trial or a single
      array of spike times
t   : time as a 2d vector [t(1) t(2)]
offset: 0/1 - if 1, store the spike times relative to start of window i.e. t(1)
        if 0, don't reset the times. Default 0.
Note that all times can be in arbitrary units. But the units have to be
consistent. So, if E is in secs, win, t have to be in secs, and Fs has to
be Hz. If E is in samples, so are win and t, and Fs=1. In case of spike
times, the units have to be consistent with the units of data as well.

Output:
data: spike times between t(1) and t(2)
```

This function calls:

none This function calls no functions

This function is called by:

CrossSpecMatpt (Section 4.3)

cohgrampt (Section 4.24) Multi-taper time-frequency coherence - two point processes given as times

mtdspecgrampt (Section 4.51) Multi-taper derivative time-frequency spectrum - point process times

mtspecgrampt (Section 4.61) Multi-taper time-frequency spectrum - point process times

4.39 findpeaks

Purpose:
Helper function to find peaks in a given continuous valued time series x

Synopsis:
`function xmax=findpeaks(data,threshold)`

Comments:

```
Helper function to find peaks in a given continuous valued time series x
Usage: xmax=findpeaks(data,threshold)
Input:
     data     (data in time x channels/trials form or a single vector)
     threshold (if specified returns locations of peaks at which data exceeds threshold) - optional
Output:
     xmax     (locations of local maxima of data in a structure array of dimensions channels/trials)
```

This function calls:

none This function calls no functions

This function is called by:

fitlinesc (Section 4.40) fits significant sine waves to data (continuous data).

4.40 fitlinesc

Purpose:
fits significant sine waves to data (continuous data).

Synopsis:
`function [datafit,Amps,freqs,Fval,sig]=fitlinesc(data,params,p,plt,f0)`

Comments:

```
 fits significant sine waves to data (continuous data).

 Usage: [datafit,Amps,freqs,Fval,sig]=fitlinesc(data,params,p,plt,f0)

  Inputs:
 Note that units of Fs, fpass have to be consistent.
       data        (data in [N,C] i.e. time x channels/trials or a single
       vector) - required.
       params      structure containing parameters - params has the
       following fields: tapers, Fs, fpass, pad
           tapers : precalculated tapers from dpss or in the one of the following
                    forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

            Fs              (sampling frequency) -- optional. Defaults to 1.
               fpass        (frequency band to be used in the calculation in the form
                                    [fmin fmax])- optional.
                                    Default all frequencies between 0 and Fs/2
            pad             (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                    -1 corresponds to no padding, 0 corresponds to padding
                    to the next highest power of 2 etc.
                       e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                       to 512 points, if pad=1, we pad to 1024 points etc.
                       Defaults to 0.
        p             (P-value to calculate error bars for) - optional.
                            Defaults to 0.05/N where N is data length.
      plt         (y/n for plot and no plot respectively) - plots the
```

```
      Fratio at all frequencies if y
      f0          frequencies at which you want to remove the
                  lines - if unspecified the program
                  will compute the significant lines


Outputs:
      datafit        (linear superposition of fitted sine waves)
      Amps           (amplitudes at significant frequencies)
      freqs          (significant frequencies)
      Fval           (Fstatistic at all frequencies)
      sig            (significance level for F distribution p value of p)
```

This function calls:

findpeaks (Section 4.39) Helper function to find peaks in a given continuous valued time series x

ftestc (Section 4.41) computes the F-statistic for sine wave in locally-white noise (continuous data).

This function is called by:

rmlinesc (Section 4.85) removes significant sine waves from data (continuous data).

rmlinesmovingwinc (Section 4.86) fits significant sine waves to data (continuous data) using overlapping windows.

4.41 ftestc

Purpose:
computes the F-statistic for sine wave in locally-white noise (continuous data).

Synopsis:
```
function [Fval,A,f,sig,sd] = ftestc(data,params,p,plt)
```

Comments:

```
 computes the F-statistic for sine wave in locally-white noise (continuous data).

 [Fval,A,f,sig,sd] = ftestc(data,params,p,plt)

  Inputs:
       data        (data in [N,C] i.e. time x channels/trials or a single
       vector) - required.
       params      structure containing parameters - params has the
       following fields: tapers, Fs, fpass, pad
           tapers : precalculated tapers from dpss or in the one of the following
                    forms:
                    (1) A numeric vector [TW K] where TW is the
                        time-bandwidth product and K is the number of
                        tapers to be used (less than or equal to
                        2TW-1).
                    (2) A numeric vector [W T p] where W is the
                        bandwidth, T is the duration of the data and p
                        is an integer such that 2TW-p tapers are used. In
                        this form there is no default i.e. to specify
                        the bandwidth, you have to specify T and p as
                        well. Note that the units of W and T have to be
                        consistent: if W is in Hz, T must be in seconds
                        and vice versa. Note that these units must also
                        be consistent with the units of params.Fs: W can
                        be in Hz if and only if params.Fs is in Hz.
                        The default is to use form 1 with TW=3 and K=5

            Fs             (sampling frequency) -- optional. Defaults to 1.
           fpass       (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
            pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                    -1 corresponds to no padding, 0 corresponds to padding
                    to the next highest power of 2 etc.
                       e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                       to 512 points, if pad=1, we pad to 1024 points etc.
                       Defaults to 0.
             p            (P-value to calculate error bars for) - optional.
                               Defaults to 0.05/N where N is the number of samples which
                          corresponds to a false detect probability of approximately 0.05.
           plt         (y/n for plot and no plot respectively)
```

```
Outputs:
     Fval        (F-statistic in frequency x channels/trials form)
        A            (Line amplitude for X in frequency x channels/trials form)
      f            (frequencies of evaluation)
     sig         (F distribution (1-p)% confidence level)
     sd          (standard deviation of the amplitude C)
```

This function calls:

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

mtspectrumc (Section 4.66) Multi-taper spectrum - continuous process

This function is called by:

fitlinesc (Section 4.40) fits significant sine waves to data (continuous data).

4.42 getfgrid

Purpose:
Helper function that gets the frequency grid associated with a given fft based computation

Synopsis:
`function [f,findx]=getfgrid(Fs,nfft,fpass)`

Comments:

```
Helper function that gets the frequency grid associated with a given fft based computation
Called by spectral estimation routines to generate the frequency axes
Usage: [f,findx]=getfgrid(Fs,nfft,fpass)
Inputs:
Fs        (sampling frequency associated with the data)-required
nfft      (number of points in fft)-required
fpass     (band of frequencies at which the fft is being calculated [fmin fmax] in Hz)-required
Outputs:
f         (frequencies)
findx     (index of the frequencies in the full frequency grid). e.g.: If
Fs=1000, and nfft=1048, an fft calculation generates 512 frequencies
between 0 and 500 (i.e. Fs/2) Hz. Now if fpass=[0 100], findx will
contain the indices in the frequency grid corresponding to frequencies &lt;
100 Hz. In the case fpass=[0 500], findx=[1 512].
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.43 getparams

Purpose:
Helper function to convert structure params to variables used by the

Synopsis:
`function [tapers,pad,Fs,fpass,err,trialave,params]=getparams(params)`

Comments:

```
Helper function to convert structure params to variables used by the
various routines - also performs checks to ensure that parameters are
defined; returns default values if they are not defined.

Usage: [tapers,pad,Fs,fpass,err,trialave,params]=getparams(params)

Inputs:
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
          - optional
            tapers : precalculated tapers from dpss or in the one of the following
                     forms:
                     (1) A numeric vector [TW K] where TW is the
                         time-bandwidth product and K is the number of
                         tapers to be used (less than or equal to
                         2TW-1).
                     (2) A numeric vector [W T p] where W is the
                         bandwidth, T is the duration of the data and p
                         is an integer such that 2TW-p tapers are used. In
                         this form there is no default i.e. to specify
                         the bandwidth, you have to specify T and p as
                         well. Note that the units of W and T have to be
                          consistent: if W is in Hz, T must be in seconds
                           and vice versa. Note that these units must also
                          be consistent with the units of params.Fs: W can
                          be in Hz if and only if params.Fs is in Hz.
                         The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials when 1, don't average when 0) - optional. Default 0
Outputs:
```

```
The fields listed above as well as the struct params. The fields are used
by some routines and the struct is used by others. Though returning both
involves overhead, it is a safer, simpler thing to do.
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.44 isi

Purpose:
Calculate the inter-spike-interval histogram

Synopsis:
`function[N,B,E] = isi(data,T,err,Nbins,plt)`

Comments:

```
Calculate the inter-spike-interval histogram
   Usage: [N,B,E] = isi(data,T,err,Nbins,plt)

Input:
Note that all times have to be consistent.

data   - structure array of spike times  (required)
T      - time interval of interest (default all)
err    - 0 for no error bars, 1 for jackknife errors

Nbins  - number of bins in the isi

Output:

N      - count in bins
B      - bin centres
E      - errorbar (this is 2 sig deviation
         calculated using a jackknife over trials)
```

This function calls:

padNaN (Section 4.77) Creates a padded data matrix from input structural array of spike times

This function is called by:

none No functions call this function

4.45 jackknife

Purpose:
Compute jackknife estimates of the mean and standard deviation of input data x

Synopsis:
`function [m,jsd]=jackknife(x)`

Comments:

```
Compute jackknife estimates of the mean and standard deviation of input data x
Usage: [m,jsd]=jackknife(x)

Inputs:
x : data in the form samples x trials

Outputs:
m : estimate of the mean (across trials)
jsd: jackknife estimate of the standard deviation (across trials)
```

This function calls:

none This function calls no functions

This function is called by:

den_jack (Section 4.33) Function to compute smooth estimates of the mean of x using locfit,

4.46 locdetrend

Purpose:
Remove running line fit (using local linear regression)-continuous

Synopsis:
`function data=locdetrend(data,Fs,movingwin)`

Comments:

```
Remove running line fit (using local linear regression)-continuous
processes
Usage: data=locdetrend(data,Fs,movingwin)
Inputs:
Note that units of Fs, movinwin have to be consistent.
data         (data as a matrix times x channels or a single vector)
Fs           (sampling frequency) - optional. Default 1
movingwin    (length of moving window, and stepsize) [window winstep] - optional.
                Default. window=full length of data (global detrend).
                winstep=window -- global detrend

Output:
data:        (locally detrended data)
```

This function calls:

runline (Section 4.87) Running line fit (local linear regression)

This function is called by:

none No functions call this function

4.47 locsmooth

Purpose:
Running line fit (using local linear regression) - 1d only, continuous

Synopsis:
```
function data=locsmooth(data,Fs,Tw,Ts)
```

Comments:

```
 Running line fit (using local linear regression) - 1d only, continuous
 processes
 Usage: data=locsmooth(data,Fs,Tw,Ts)
 Inputs:
Note that units of Fs, movinwin have to be consistent.
 data  (single vector)
 Fs    (sampling frequency) - optional. Default 1
 Tw    (length of moving window) - optional.  Default. full length of data (global detrend)
 Ts    (step size) - optional. Default Tw/2.

Output:
data   (locally smoothed data).
```

This function calls:

runline (Section 4.87) Running line fit (local linear regression)

This function is called by:

evoked (Section 4.35) Function to calculate the evoked response given continuous data in the

4.48 minmaxsptimes

Purpose:
Find the minimum and maximum of the spike times in each channel

Synopsis:
```
function [mintime, maxtime]=minmaxsptimes(data)
```

Comments:

```
Find the minimum and maximum of the spike times in each channel
Usage: [mintime, maxtime]=minmaxsptimes(data)
Input:
data  (spike times as a structural array of multiple dimensions e.g. channels; channels x trials;
       can also accept a 1d matrix of spike times)
Output:
mintime       (minimum of the spike time across channels)
maxtime       (maximum of the spike time across channels)
```

This function calls:

none This function calls no functions

This function is called by:

CrossSpecMatpt (Section 4.3)

coherencypt (Section 4.13) Multi-taper coherency - point process times

coherencysegpt (Section 4.18) Multi-taper coherency computed by segmenting two univariate point processes into chunks

cohgrampt (Section 4.24) Multi-taper time-frequency coherence - two point processes given as times

cohmatrixpt (Section 4.28) Multi-taper coherency matrix - point process times

mtdspecgrampt (Section 4.51) Multi-taper derivative time-frequency spectrum - point process times

mtdspectrumpt (Section 4.54) Multi-taper spectral derivative - point process times

mtspecgrampt (Section 4.61) Multi-taper time-frequency spectrum - point process times

mtspectrumpt (Section 4.69) Multi-taper spectrum - point process times

mtspectrumsegpt (Section 4.72) Multi-taper segmented spectrum for a univariate binned point process

4.49 mtdspecgramc

Purpose:
Multi-taper derivative of the time-frequency spectrum - continuous process

Synopsis:
`function [dS,t,f]=mtdspecgramc(data,movingwin,phi,params)`

Comments:

```
Multi-taper derivative of the time-frequency spectrum - continuous process

Usage:

[dS,t,f]=mtdspecgramc(data,movingwin,phi,params)
Input:
  Note that all times can be in arbitrary units. But the units have to be
  consistent. So, if E is in secs, win, t have to be in secs, and Fs has to
  be Hz. If E is in samples, so are win and t, and Fs=1. In case of spike
  times, the units have to be consistent with the units of data as well.

      data        (in form samples x channels/trials or a single vector) -- required
      movingwin         (in the form [window winstep] i.e length of moving
                                          window and step size.
                                          Note that units here have
                                          to be consistent with
                                          units of Fs - required
      phi         (angle for evaluation of derivative) -- required
                     e.g. phi=[0,pi/2] giving the time and frequency
                     derivatives
      params: structure with fields tapers, pad, Fs, fpass, trialave
      -optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5
                    Note that T has to be equal to movingwin(1).

       pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
```

```
                -1 corresponds to no padding, 0 corresponds to padding
                to the next highest power of 2 etc.
                   e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                   to 512 points, if pad=1, we pad to 1024 points etc.
                   Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          trialave - (average over trials/channels when 1, don't average when 0) - optional. Default 0
Output:
      dS      (spectral derivative in form phi x time x frequency x channels/trials if trialave=0;
              in form phi x time x frequency if trialave=1)
      t       (times)
      f       (frequencies)
```

This function calls:

mtdspectrumc (Section 4.52) Multi-taper frequency derivative of the spectrum - continuous process

This function is called by:

none No functions call this function

4.50 mtdspecgrampb

Purpose:
Multi-taper derivatives of time-frequency spectrum - binned point process

Synopsis:
`function [dS,t,f]=mtdspecgrampb(data,movingwin,phi,params)`

Comments:

```
Multi-taper derivatives of time-frequency spectrum - binned point process

Usage:

[dS,t,f]=mtdspecgrampb(data,movingwin,phi,params)
Input:
  Note that all times can be in arbitrary units. But the units have to be
  consistent. So, if E is in secs, win, t have to be in secs, and Fs has to
  be Hz. If E is in samples, so are win and t, and Fs=1. In case of spike
  times, the units have to be consistent with the units of data as well.
      data        (in form samples x channels/trials or a single vector) -- required
      movingwin         (in the form [window winstep] i.e length of moving
                                            window and step size.
                                            Note that units here have
                                            to be consistent with
                                            units of Fs
      phi         (angle for evaluation of derivative) -- required.
                     e.g. phi=[0,pi/2] giving the time and frequency
                     derivatives
      params: structure with fields tapers, pad, Fs, fpass,trialave
      -optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5
                   Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
```

```
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                 [fmin fmax])- optional.
                                 Default all frequencies between 0 and
                                 Fs/2
          trialave (average over trials when 1, don't average when 0) -
          optional. Default 0
Output:
      dS      (spectral derivative in form phi x time x frequency x channels/trials if trialave=0;
              phi x time x frequency if trialave=1)
      t       (times)
      f       (frequencies)
```

This function calls:

mtdspectrumpb (Section 4.53) Multi-taper spectral derivative - binned point process

This function is called by:

none No functions call this function

4.51 mtdspecgrampt

Purpose:
Multi-taper derivative time-frequency spectrum - point process times

Synopsis:
`function [dS,t,f]=mtdspecgrampt(data,movingwin,phi,params)`

Comments:

```
Multi-taper derivative time-frequency spectrum - point process times

Usage:

[dS,t,f]=mtdspecgrampt(data,movingwin,phi,params)
Input:
  Note that all times can be in arbitrary units. But the units have to be
  consistent. So, if E is in secs, win, t have to be in secs, and Fs has to
  be Hz. If E is in samples, so are win and t, and Fs=1. In case of spike
  times, the units have to be consistent with the units of data as well.
      data        (structure array of spike times with dimension channels/trials;
                  also accepts 1d array of spike times) -- required
      movingwin         (in the form [window winstep] i.e length of moving
                                          window and step size.
                                          Note that units here have
                                          to be consistent with
                                          units of Fs
      phi         (angle for evaluation of derivative) -- required.
                      e.g. phi=[0,pi/2] giving the time and frequency
                      derivatives
      params: structure with fields tapers, pad, Fs, fpass, trialave
      -optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5
                  Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
```

```
                -1 corresponds to no padding, 0 corresponds to padding
                to the next highest power of 2 etc.
                   e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                   to 512 points, if pad=1, we pad to 1024 points etc.
                   Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and
                                Fs/2
          trialave (average over trials when 1, don't average when 0) -
          optional. Default 0
Output:
      dS      (spectral derivative in form phi x time x frequency x channels/trials if trialave=0;
              in form phi x time x frequency if trialave=1)
      t       (times)
      f       (frequencies)
```

This function calls:

extractdatapt (Section 4.38) Extract segements of spike times between t(1) and t(2)

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

mtdspectrumpt (Section 4.54) Multi-taper spectral derivative - point process times

This function is called by:

none No functions call this function

4.52 mtdspectrumc

Purpose:
Multi-taper frequency derivative of the spectrum - continuous process

Synopsis:
`function [dS,f]=mtdspectrumc(data,phi,params)`

Comments:

```
Multi-taper frequency derivative of the spectrum - continuous process

Usage:

[dS,f]=mtdspectrumc(data,phi,params)
Input:
  Note that all times can be in arbitrary units. But the units have to be
  consistent. So, if E is in secs, win, t have to be in secs, and Fs has to
  be Hz. If E is in samples, so are win and t, and Fs=1. In case of spike
  times, the units have to be consistent with the units of data as well.
      data        (in form samples x channels/trials or a single vector) -- required
      phi         (angle for evaluation of derivative) -- required.
                      e.g. phi=[0,pi/2] gives the time and frequency derivatives
      params: structure with fields tapers, pad, Fs, fpass, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
      Fs   (sampling frequency) - optional. Default 1.
      fpass    (frequency band to be used in the calculation in the form
                               [fmin fmax])- optional.
```

```
                                Default all frequencies between 0 and Fs/2
          trialave (average over trials/channels when 1, don't average when 0) - optional. Default 0
Output:
      dS        (spectral derivative in form phi x frequency x channels/trials if trialave=0 or
                in form phi x frequency if trialave=1)
      f         (frequencies)
```

This function calls:

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

This function is called by:

mtdspecgramc (Section 4.49) Multi-taper derivative of the time-frequency spectrum - continuous process

4.53 mtdspectrumpb

Purpose:
Multi-taper spectral derivative - binned point process

Synopsis:
`function [dS,f]=mtdspectrumpb(data,phi,params)`

Comments:

```
Multi-taper spectral derivative - binned point process

Usage:

[dS,f]=mtdspectrumpb(data,phi,params)
Input:
  Note that all times can be in arbitrary units. But the units have to be
  consistent. So, if E is in secs, win, t have to be in secs, and Fs has to
  be Hz. If E is in samples, so are win and t, and Fs=1. In case of spike
  times, the units have to be consistent with the units of data as well.
      data (in form samples x channels/trials or single vector) -- required
      tapers (precalculated tapers from dpss, or in the form [NW K] e.g [3 5]) -- optional.
                                                 If not specified, use [NW K]=[3 5]
      phi         (angle for evaluation of derivative) -- required.
                      e.g. phi=[0,pi/2] giving the time and frequency
                      derivatives
      params: structure with fields tapers, pad, Fs, fpass, trialave
      -optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
```

```
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and
                                Fs/2
          trialave (average over trials when 1, don't average when 0) -
          optional. Default 0
Output:
      dS       (derivative of the spectrum in form phi x frequency x channels/trials if trialave=0;
               in the form phi x frequency if trialave=1)
      f        (frequencies)
```

This function calls:

mtfftpb (Section 4.56) Multi-taper fourier transform - binned point process data

This function is called by:

mtdspecgrampb (Section 4.50) Multi-taper derivatives of time-frequency spectrum - binned point process

4.54 mtdspectrumpt

Purpose:
Multi-taper spectral derivative - point process times

Synopsis:
`function [dS,f]=mtdspectrumpt(data,phi,params,t)`

Comments:

```
Multi-taper spectral derivative - point process times

Usage:

[dS,f]=mtdspectrumpt(data,phi,params,t)
Input:
  Note that all times can be in arbitrary units. But the units have to be
  consistent. So, if E is in secs, win, t have to be in secs, and Fs has to
  be Hz. If E is in samples, so are win and t, and Fs=1. In case of spike
  times, the units have to be consistent with the units of data as well.
      data        (structure array of spike times with dimension channels/trials;
                  also accepts 1d array of spike times) -- required
      phi         (angle for evaluation of derivative) -- required.
                      e.g. phi=[0,pi/2] giving the time and frequency derivatives
      params: structure with fields tapers, pad, Fs, fpass, trialave
      -optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
```

```
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          trialave (average over trials when 1, don't average when 0) -
          optional. Default 0
      t         (time grid over which the tapers are to be calculated:
                      this argument is useful when calling the spectrum
                      calculation routine from a moving window spectrogram
                      calculation routine). If left empty, the spike times
                      are used to define the grid.
Output:
      dS      (spectral derivative in form phi x frequency x channels/trials if trialave=0;
              function of phi x frequency if trialave=1)
      f       (frequencies)
```

This function calls:

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

mtfftpt (Section 4.57) Multi-taper fourier transform for point process given as times

This function is called by:

mtdspecgrampt (Section 4.51) Multi-taper derivative time-frequency spectrum - point process times

4.55 mtfftc

Purpose:
Multi-taper fourier transform - continuous data

Synopsis:
`function J=mtfftc(data,tapers,nfft,Fs)`

Comments:

```
Multi-taper fourier transform - continuous data

Usage:
J=mtfftc(data,tapers,nfft,Fs) - all arguments required
Input:
      data (in form samples x channels/trials or a single vector)
      tapers (precalculated tapers from dpss)
      nfft (length of padded data)
      Fs   (sampling frequency)

Output:
      J (fft in form frequency index x taper index x channels/trials)
```

This function calls:

none This function calls no functions

This function is called by:

CrossSpecMatc (Section 4.1) Multi-taper cross-spectral matrix - another routine, allows for multiple trials and channels

coherencyc (Section 4.8) Multi-taper coherency,cross-spectrum and individual spectra - continuous process

coherencyc_unequal_length_trials (Section 4.9) This routine computes the average multi-taper coherence for a given set of unequal length segments. It is

cohmatrixc (Section 4.26) Multi-taper coherency,cross-spectral matrix - continuous process

ftestc (Section 4.41) computes the F-statistic for sine wave in locally-white noise (continuous data).

mtdspectrumc (Section 4.52) Multi-taper frequency derivative of the spectrum - continuous process

mtspectrumc (Section 4.66) Multi-taper spectrum - continuous process

mtspectrumc_unequal_length_trials (Section 4.67) This routine computes the multi-taper spectrum for a given set of unequal length segments. It is

mtspectrumsegc (Section 4.70) Multi-taper segmented spectrum for a univariate continuous process

nonst_stat (Section 4.76) Nonstationarity test - continuous process

4.56 mtfftpb

Purpose:
Multi-taper fourier transform - binned point process data

Synopsis:
`function [J,Msp,Nsp]=mtfftpb(data,tapers,nfft)`

Comments:

```
Multi-taper fourier transform - binned point process data

Usage:

[J,Msp,Nsp]=mtfftpb(data,tapers,nfft) - all arguments required
Input:
      data   (in form samples x channels/trials or single vector)
      tapers (precalculated tapers from dpss)
      nfft   (length of padded data)
Output:
      J (fft in form frequency index x taper index x channels/trials)
      Msp (number of spikes per sample in each channel)
      Nsp (number of spikes in each channel)
```

This function calls:

none This function calls no functions

This function is called by:

CrossSpecMatpb (Section 4.2)

coherencypb (Section 4.12) Multi-taper coherency,cross-spectrum and individual spectra - binned point process

cohmatrixpb (Section 4.27) Multi-taper coherency matrix - binned point process

mtdspectrumpb (Section 4.53) Multi-taper spectral derivative - binned point process

mtspectrumpb (Section 4.68) Multi-taper spectrum - binned point process

mtspectrumsegpb (Section 4.71) Multi-taper segmented spectrum for a univariate binned point process

4.57 mtfftpt

Purpose:
Multi-taper fourier transform for point process given as times

Synopsis:
`function [J,Msp,Nsp]=mtfftpt(data,tapers,nfft,t,f,findx)`

Comments:

```
Multi-taper fourier transform for point process given as times

Usage:
[J,Msp,Nsp]=mtfftpt (data,tapers,nfft,t,f,findx) - all arguments required
Input:
      data        (struct array of times with dimension channels/trials;
                  also takes in 1d array of spike times as a column vector)
      tapers      (precalculated tapers from dpss)
      nfft        (length of padded data)
      t           (time points at which tapers are calculated)
      f           (frequencies of evaluation)
      findx       (index corresponding to frequencies f)
Output:
      J (fft in form frequency index x taper index x channels/trials)
      Msp (number of spikes per sample in each channel)
      Nsp (number of spikes in each channel)
```

This function calls:

none This function calls no functions

This function is called by:

CrossSpecMatpt (Section 4.3)

coherencypt (Section 4.13) Multi-taper coherency - point process times

cohmatrixpt (Section 4.28) Multi-taper coherency matrix - point process times

mtdspectrumpt (Section 4.54) Multi-taper spectral derivative - point process times

mtspectrumpt (Section 4.69) Multi-taper spectrum - point process times

mtspectrumsegpt (Section 4.72) Multi-taper segmented spectrum for a univariate binned point process

4.58 mtpowerandfstatc

Purpose:
Multi-taper computation of the power and the fstatistic for a particular frequency - continuous process

Synopsis:
`function [P,Fstat,f0]=mtpowerandfstatc(data,params,f0)`

Comments:

```
Multi-taper computation of the power and the fstatistic for a particular frequency - continuous process

Usage:

[P,Fstat,f0]=mtpowerandfstatc(data,params,f0)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (in form samples x channels/trials or a single vector) -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      -optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad             (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
      f0  (frequency of calculation)
Output:
      P       (integrated power within the frequency range of interest (trapezoidal integration))
      Fstat   (F-statistic)
      f0      (frequency)
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.59 mtspecgramc

Purpose:
Multi-taper time-frequency spectrum - continuous process

Synopsis:
`function [S,t,f,Serr]=mtspecgramc(data,movingwin,params)`

Comments:

```
Multi-taper time-frequency spectrum - continuous process

Usage:
[S,t,f,Serr]=mtspecgramc(data,movingwin,params)
Input:
Note units have to be consistent. Thus, if movingwin is in seconds, Fs
has to be in Hz. see chronux.m for more information.
      data        (in form samples x channels/trials) -- required
      movingwin         (in the form [window winstep] i.e length of moving
                                            window and step size)
                                            Note that units here have
                                            to be consistent with
                                            units of Fs - required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5
                    Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
```

```
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials/channels when 1, don't average when 0) - optional. Default 0
Output:
      S       (spectrum in form time x frequency x channels/trials if trialave=0;
              in the form time x frequency if trialave=1)
      t       (times)
      f       (frequencies)
      Serr    (error bars) only for err(1)&gt;=1
```

This function calls:

mtspectrumc (Section 4.66) Multi-taper spectrum - continuous process

This function is called by:

mtspecgramtrigc (Section 4.62) Multi-taper event triggered time-frequency spectrum - continuous process

4.60 mtspecgrampb

Purpose:
Multi-taper time-frequency spectrum - binned point process

Synopsis:
`function [S,t,f,R,Serr]=mtspecgrampb(data,movingwin,params,fscorr)`

Comments:

```
Multi-taper time-frequency spectrum - binned point process

Usage:

[S,t,f,R,Serr]=mtspecgrampb(data,movingwin,params,fscorr)
Input:
      data        (in form samples x channels/trials or single vector) -- required
      movingwin         (in the form [window,winstep] i.e length of moving
                                          window and step size.

      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5
                  Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
```

```
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials/channnels when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      S       (spectrum in form time x frequency x channels/trials for trialave=0;
              or as a function of frequency if trialave=1)
      t       (times)
      f       (frequencies)
      R       (rate)
      Serr    (error bars) - only for err(1)&gt;=1
```

This function calls:

mtspectrumpb (Section 4.68) Multi-taper spectrum - binned point process

This function is called by:

mtspecgramtrigpb (Section 4.63) Multi-taper event triggered time-frequency spectrum - binned point process

4.61 mtspecgrampt

Purpose:
Multi-taper time-frequency spectrum - point process times

Synopsis:
`function [S,t,f,R,Serr]=mtspecgrampt(data,movingwin,params,fscorr)`

Comments:

```
Multi-taper time-frequency spectrum - point process times

Usage:

[S,t,f,R,Serr]=mtspecgrampt(data,movingwin,params,fscorr)
Input:
      data        (structure array of spike times with dimension channels/trials;
                  also accepts 1d array of spike times) -- required
      movingwin         (in the form [window,winstep] i.e length of moving
                                          window and step size.

      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5
                    Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
```

```
                                Default all frequencies between 0 and Fs/2
        err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
        trialave (average over trials/channels when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.

Output:
      S       (spectrogram with dimensions time x frequency x channels/trials if trialave=0;
              dimensions time x frequency if trialave=1)
      t       (times)
      f       (frequencies)

      Serr    (error bars) - only if err(1)&gt;=1
```

This function calls:

extractdatapt (Section 4.38) Extract segements of spike times between t(1) and t(2)

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

mtspectrumpt (Section 4.69) Multi-taper spectrum - point process times

This function is called by:

mtspecgramtrigpt (Section 4.64) Multi-taper event triggered time-frequency spectrum - point process times

4.62 mtspecgramtrigc

Purpose:
Multi-taper event triggered time-frequency spectrum - continuous process

Synopsis:
`function [S,t,f,Serr]=mtspecgramtrigc(data,E,win,movingwin,params)`

Comments:

```
Multi-taper event triggered time-frequency spectrum - continuous process

Usage:

[S,t,f,Serr]=mtspecgramtrigc(data,E,win,movingwin,params)
Input:
Note units have to be consistent. Thus, if movingwin is in seconds, Fs
has to be in Hz. see chronux.m for more information.
      data        (single channel data) -- required
      E           (event times) -- required
      win         (in the form [winl winr] i.e window around each event)
                                          required
      movingwin         (in the form [window winstep] i.e length of moving
                                          window and step size) -
                                          required
                                          Note that units for the windows have
                                          to be consistent with
                                          units of Fs
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5
                   Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
```

```
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over events when 1, don't average when 0) - optional. Default 0
Output:
      S       (triggered spectrum in form time x frequency x events for trialave=0;
              or in the form time x frequency trialave=1)
      t       (times)
      f       (frequencies)
      Serr    (error bars) only for err(1)&gt;=1
```

This function calls:

createdatamatc (Section 4.30) Helper function to create an event triggered matrix from univariate

mtspecgramc (Section 4.59) Multi-taper time-frequency spectrum - continuous process

This function is called by:

none No functions call this function

4.63 mtspecgramtrigpb

Purpose:
Multi-taper event triggered time-frequency spectrum - binned point process

Synopsis:
`function [S,t,f,R,Serr]=mtspecgramtrigpb(data,E,win,movingwin,params,fscorr)`

Comments:

```
Multi-taper event triggered time-frequency spectrum - binned point process

Usage:

[S,t,f,R,Serr]=mtspecgramtrigpb(data,E,win,movingwin,params,fscorr)
Input:
      data        (single channel data vector) -- required
      E           (event times) - required
      win         (in the form [winl,winr] i.e window around each event
                                          required
      movingwin         (in the form [window,winstep] i.e length of moving
                                          window and step size) -
                                          required
                                          Note that units for the windows have
                                          to be consistent with
                                          units of Fs
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5
                  Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
```

```
                    to 512 points, if pad=1, we pad to 1024 points etc.
                    Defaults to 0.
        Fs   (sampling frequency) - optional. Default 1.
        fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
        err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
        trialave (average over events when 1, don't average when 0) - optional. Default 0
    fscorr   (finite size corrections, 0 (don't use finite size corrections) or
             1 (use finite size corrections) - optional
             (available only for spikes). Defaults 0.
Output:
    S       (triggered time-frequency spectrum in form time x frequency x events if segave=0;
            or in the form time x frequency segave=1)
    t       (times)
    f       (frequencies)
    R       (spike rate)
    Serr    (error bars) - only for err(1)&gt;=1
```

This function calls:

createdatamatpb (Section 4.31)

mtspecgrampb (Section 4.60) Multi-taper time-frequency spectrum - binned point process

This function is called by:

none No functions call this function

4.64 mtspecgramtrigpt

Purpose:
Multi-taper event triggered time-frequency spectrum - point process times

Synopsis:
`function [S,t,f,R,Serr]=mtspecgramtrigpt(data,E,win,movingwin,params,fscorr)`

Comments:

```
Multi-taper event triggered time-frequency spectrum - point process times

Usage:

[S,t,f,R,Serr]=mtspecgramtrigpt(data,E,win,movingwin,params,fscorr)
Input:
      data        (univariate spike data -1d structure array of spike times;
                  also accepts 1d array of spike times) -- required
      E           (event times) - required
      win         (in the form [winl,winr] i.e window around each event
                                          required
      movingwin         (in the form [window winstep] i.e length of moving
                                          window and step size) -
                                          required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5
                    Note that T has to be equal to movingwin(1).

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
```

```
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over events when 1, don't average when 0) - optional. Default 0
       fscorr   (finite size corrections, 0 (don't use finite size corrections) or
                1 (use finite size corrections) - optional
                (available only for spikes). Defaults 0.

Output:
       S       (Spectrogram with dimensions time x frequency x events if trialave=0;
               dimensions time x frequency if trialave=1)
       t       (times)
       f       (frequencies)
       R       (spike rate)
       Serr    (error bars)-only if err(1)&gt;=1
```

This function calls:

createdatamatpt (Section 4.32) Helper function to create an event triggered matrix from a single

mtspecgrampt (Section 4.61) Multi-taper time-frequency spectrum - point process times

This function is called by:

none No functions call this function

4.65 mtspectrum_of_spectrumc

Purpose:
Multi-taper segmented, second spectrum (spectrum of the log spectrum) for a continuous process

Synopsis:
```
function [SS,tau]=mtspectrum_of_spectrumc(data,win,tapers2spec,params)
```

Comments:

```
Multi-taper segmented, second spectrum (spectrum of the log spectrum) for a continuous process
This routine computes the second spectrum by explicitly evaluating the
Fourier transform (since the spectrum is symmetric in frequency, it uses
a cosine transform)

Usage:

[SS,tau]=mtspectrum_of_spectrumc(data,win,tapers2spec,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (single channel) -- required
      win  (duration of the segments) - required.
      tapers2spec (tapers used for the spectrum of spectrum computation) -
      required in the form [use TW K] - Note that spectrum of the
      spectrum involves computing two Fourier transforms. While the first
      transform (of the original data) is always computed using the
      multi-taper method, the current routine allows the user to specify
      whether or not to use this method for the second transform. use=1
      means use tapers, use=anything other than 1 means do not use the
      multitaper method. If use=1, then tapers2spec controls the
      smoothing for the second Fourier transform. Otherwise, a direct
      Fourier transform is computed.
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
```

```
                    The default is to use form 1 with TW=3 and K=5

         pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                -1 corresponds to no padding, 0 corresponds to padding
                to the next highest power of 2 etc.
                   e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                   to 512 points, if pad=1, we pad to 1024 points etc.
                   Defaults to 0.
        Fs   (sampling frequency) - optional. Default 1.
        fpass    (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and
                                Fs/2
Output:
      SS       (second spectrum in form frequency x segments x trials x channels
               if segave=0; in the form frequency x trials x channels if segave=1)
      tau      (frequencies)
```

This function calls:

createdatamatc (Section 4.30) Helper function to create an event triggered matrix from univariate

mtspectrumc (Section 4.66) Multi-taper spectrum - continuous process

mtspectrumsegc (Section 4.70) Multi-taper segmented spectrum for a univariate continuous process

This function is called by:

none No functions call this function

4.66 mtspectrumc

Purpose:
Multi-taper spectrum - continuous process

Synopsis:
`function [S,f,Serr]=mtspectrumc(data,params)`

Comments:

```
Multi-taper spectrum - continuous process

Usage:

[S,f,Serr]=mtspectrumc(data,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (in form samples x channels/trials) -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      -optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over trials/channels when 1, don't average when 0) - optional. Default 0
Output:
```

```
S       (spectrum in form frequency x channels/trials if trialave=0;
        in the form frequency if trialave=1)
f       (frequencies)
Serr    (error bars) only for err(1)&gt;=1
```

This function calls:

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

This function is called by:

ftestc (Section 4.41) computes the F-statistic for sine wave in locally-white noise (continuous data).

mtspecgramc (Section 4.59) Multi-taper time-frequency spectrum - continuous process

mtspectrum_of_spectrumc (Section 4.65) Multi-taper segmented, second spectrum (spectrum of the log spectrum) for a continuous process

mtspectrumtrigc (Section 4.73) Multi-taper event triggered time-frequency spectrum - continuous process

rmlinesc (Section 4.85) removes significant sine waves from data (continuous data).

4.67 mtspectrumc_unequal_length_trials

Purpose:
This routine computes the multi-taper spectrum for a given set of unequal length segments. It is

Synopsis:
`function [ S, f, Serr ]= mtspectrumc_unequal_length_trials( data, movingwin, params, sMarkers )`

Comments:

```
This routine computes the multi-taper spectrum for a given set of unequal length segments. It is
based on modifications to the Chronux routines. The segments are continuously structured in the
data matrix, with the segment boundaries given by markers. Below,
movingwin is used in a non-overlaping way to partition each segment into
various windows. Th spectrum is evaluated for each window, and then the
window spectrum estimates averaged. Further averaging is conducted by
repeating the process for each segment.

Inputs:

  data = data( samples, channels )- here segments must be stacked
  as explained in the email
  movingwin = [window winstep] i.e length of moving
              window and step size. Note that units here have
              to be consistent with units of Fs. If Fs=1 (ie normalized)
              then [window winstep]should be in samples, or else if Fs is
              unnormalized then they should be in time (secs).
  sMarkers = N x 2 array of segment start &amp; stop marks. sMarkers(n, 1) = start
          sample index; sMarkers(n,2) = stop sample index for the nth segment
  params = see Chronux help on mtspecgramc

Output:

      S       frequency x channels
      f       frequencies x 1
      Serr    (error bars) only for err(1)&gt;=1
```

This function calls:

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

This function is called by:

none No functions call this function

4.68 mtspectrumpb

Purpose:
Multi-taper spectrum - binned point process

Synopsis:
`function [S,f,R,Serr]=mtspectrumpb(data,params,fscorr)`

Comments:

```
Multi-taper spectrum - binned point process

Usage:

[S,f,R,Serr]=mtspectrumpb(data,params,fscorr)
Input:
      data (in form samples x channels/trials or a single vector) -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over channels/trials when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
```

```
               (available only for spikes). Defaults 0.
Output:
      S       (spectrum in form frequency x channels/trials if trialave=0;
              as a function of frequency if trialave=1)
      f       (frequencies)
      R       (spike rate)
      Serr    (error bars) - only for err(1)&gt;=1
```

This function calls:

mtfftpb (Section 4.56) Multi-taper fourier transform - binned point process data

This function is called by:

mtspecgrampb (Section 4.60) Multi-taper time-frequency spectrum - binned point process

mtspectrumtrigpb (Section 4.74) Multi-taper event triggered time-frequency spectrum - binned point process

4.69 mtspectrumpt

Purpose:
Multi-taper spectrum - point process times

Synopsis:
`function [S,f,R,Serr]=mtspectrumpt(data,params,fscorr,t)`

Comments:

```
Multi-taper spectrum - point process times

Usage:

[S,f,R,Serr]=mtspectrumpt(data,params,fscorr,t)
Input:
      data        (structure array of spike times with dimension channels/trials;
                  also accepts 1d array of spike times) -- required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over channels/trials when 1, don't average when 0) - optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
```

```
                1 (use finite size corrections) - optional
                (available only for spikes). Defaults 0.
       t        (time grid over which the tapers are to be calculated:
                      this argument is useful when calling the spectrum
                      calculation routine from a moving window spectrogram
                      calculation routine). If left empty, the spike times
                      are used to define the grid.
Output:
       S       (spectrum with dimensions frequency x channels/trials if trialave=0;
               dimension frequency if trialave=1)
       f       (frequencies)
       R       (rate)
       Serr    (error bars) - only if err(1)&gt;=1
```

This function calls:

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

mtfftpt (Section 4.57) Multi-taper fourier transform for point process given as times

This function is called by:

mtspecgrampt (Section 4.61) Multi-taper time-frequency spectrum - point process times

mtspectrumtrigpt (Section 4.75) Multi-taper time-frequency spectrum - point process times

4.70 mtspectrumsegc

Purpose:
Multi-taper segmented spectrum for a univariate continuous process

Synopsis:
`function [S,f,varS,C,Serr]=mtspectrumsegc(data,win,params,segave)`

Comments:

```
Multi-taper segmented spectrum for a univariate continuous process

Usage:

[S,f,varS,C,Serr]=mtspectrumsegc(data,win,params,segave)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (single channel) -- required
      win  (duration of the segments) - required.
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad             (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
```

```
          trialave - not used
      segave - optional 0 for don't average over segments, 1 for average - default
      1
Output:
      S       (spectrum in form frequency x segments if segave=0; in the form frequency if segave=1)
      f       (frequencies)
      varS    (variance of the log spectrum)
      C       (covariance matrix of the log spectrum - frequency x
      frequency matrix)
      Serr    (error bars) only for err(1)&gt;=1
```

This function calls:

createdatamatc (Section 4.30) Helper function to create an event triggered matrix from univariate

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

This function is called by:

mtspectrum_of_spectrumc (Section 4.65) Multi-taper segmented, second spectrum (spectrum of the log spectrum) for a continuous process

rmlinesmovingwinc (Section 4.86) fits significant sine waves to data (continuous data) using overlapping windows.

4.71 mtspectrumsegpb

Purpose:
Multi-taper segmented spectrum for a univariate binned point process

Synopsis:
`function [S,f,R,varS,zerosp,C,Serr]=mtspectrumsegpb(data,win,params,segave,fscorr)`

Comments:

```
Multi-taper segmented spectrum for a univariate binned point process

Usage:

[S,f,R,varS,zerosp,C,Serr]=mtspectrumsegpb(data,win,params,segave,fscorr)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (single vector) -- required
      win  (duration of the segments) - required.
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
           tapers : precalculated tapers from dpss or in the one of the following
                    forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

            pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
           Fs   (sampling frequency) - optional. Default 1.
           fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
           err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                  [0 p] or 0 - no error bars) - optional. Default 0.
```

```
      segave (1 for averaging across segments, 0 otherwise; default 1)
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      S       (spectrum in form frequency x segments if segave=0; as a function of frequency if segave=1)
      f       (frequencies)
      R       (spike rate)
      varS    (variance of the log spectrum)
      zerosp  (0 for segments in which spikes were found, 1 for segments
      in which there are no spikes)
      C       (covariance matrix of the log spectrum - frequency x
      frequency matrix)
      Serr    (error bars) - only for err(1)&gt;=1
```

This function calls:

createdatamatpb (Section 4.31)

mtfftpb (Section 4.56) Multi-taper fourier transform - binned point process data

This function is called by:

none No functions call this function

4.72 mtspectrumsegpt

Purpose:
Multi-taper segmented spectrum for a univariate binned point process

Synopsis:
`function [S,f,R,varS,zerosp,C,Serr]=mtspectrumsegpt(data,win,params,segave,fscorr)`

Comments:

```
Multi-taper segmented spectrum for a univariate binned point process

Usage:

[S,f,R,varS,zerosp,C,Serr]=mtspectrumsegpt(data,win,params,segave,fscorr)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (structure array of one channel of spike times;
           also accepts 1d vector of spike times) -- required
      win  (duration of the segments) - required.
      params: structure with fields tapers, pad, Fs, fpass, err
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
```

```
                                [0 p] or 0 - no error bars) - optional. Default 0.
      segave - (0 for don't average over segments, 1 for average) - optional - default  1
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      S       (spectrum in form frequency x segments if segave=0; function of frequency if segave=1)
      f       (frequencies)
      R       (spike rate)
      varS    (variance of the spectrum as a function of frequency)
      zerosp  (0 for segments in which spikes were found, 1 for segments
      C       (covariance matrix of the log spectrum - frequency x
      frequency matrix)
      Serr    (error bars) - only if err(1)&gt;=1
```

This function calls:

createdatamatpt (Section 4.32) Helper function to create an event triggered matrix from a single

minmaxsptimes (Section 4.48) Find the minimum and maximum of the spike times in each channel

mtfftpt (Section 4.57) Multi-taper fourier transform for point process given as times

This function is called by:

none No functions call this function

4.73 mtspectrumtrigc

Purpose:
Multi-taper event triggered time-frequency spectrum - continuous process

Synopsis:
`function [S,f,Serr]=mtspectrumtrigc(data,E,win,params)`

Comments:

```
Multi-taper event triggered time-frequency spectrum - continuous process

Usage:

[S,f,Serr]=mtspectrumtrigc(data,E,win,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data        (single channel) -- required
      E           (event times) -- required
      win         (in the form [winl winr] i.e window around each event
                                          required)
                                          Note that units here have
                                          to be consistent with
                                          units of Fs
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      -optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
```

```
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over events when 1, don't average when 0) - optional. Default 0
Output:
      S       (triggered spectrum in form frequency x events for trialave=0 -
              or as a function of frequency for trialave=1)
      f       (frequencies)
      Serr    (error bars) only for err(1)&gt;=1
```

This function calls:

createdatamatc (Section 4.30) Helper function to create an event triggered matrix from univariate

mtspectrumc (Section 4.66) Multi-taper spectrum - continuous process

This function is called by:

none No functions call this function

4.74 mtspectrumtrigpb

Purpose:
Multi-taper event triggered time-frequency spectrum - binned point process

Synopsis:
`function [S,f,R,Serr]=mtspectrumtrigpb(data,E,win,params,fscorr)`

Comments:

```
Multi-taper event triggered time-frequency spectrum - binned point process

Usage:

[S,f,R,Serr]=mtspectrumtrigpb(data,E,win,params,fscorr)
Input:
      data        (single channel data) -- required
      E           (event times) - required
      win         (in the form [winl winr] i.e window around each event
                                          required
                                          Note that units here have
                                          to be consistent with
                                          units of Fs
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
```

```
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over events when 1, don't average when 0) -
          optional. Default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.
Output:
      S       (triggered spectrum in form frequency x events for trialave=0,
              or as a function of frequency for trialave=1)
      f       (frequencies)
      R       (spike rate)
      Serr    (error bars) - only for err(1)&gt;=1
```

This function calls:

createdatamatpb (Section 4.31)

mtspectrumpb (Section 4.68) Multi-taper spectrum - binned point process

This function is called by:

none No functions call this function

4.75 mtspectrumtrigpt

Purpose:
Multi-taper time-frequency spectrum - point process times

Synopsis:
`function [S,f,R,Serr]=mtspectrumtrigpt(data,E,win,params,fscorr)`

Comments:

```
Multi-taper time-frequency spectrum - point process times

Usage:

[S,f,R,Serr]=mtspectrumtrigpt(data,E,win,params,fscorr)
Input:
      data        (structure array of one channel of spike times;
                  also accepts 1d column vector of spike times) -- required
      E           (event times) - required
      win         (in the form [winl winr] i.e window around each event)--
                                              required
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      - optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
          fpass    (frequency band to be used in the calculation in the form
                                  [fmin fmax])- optional.
                                  Default all frequencies between 0 and Fs/2
```

```
          err  (error calculation [1 p] - Theoretical error bars; [2 p] - Jackknife error bars
                                 [0 p] or 0 - no error bars) - optional. Default 0.
          trialave (average over events when 1, don't average when 0) -
          default 0
      fscorr   (finite size corrections, 0 (don't use finite size corrections) or
               1 (use finite size corrections) - optional
               (available only for spikes). Defaults 0.

Output:
      S       (triggered spectrum in form frequency x events if trialave=0;
              function of frequency if trialave=1)
      f       (frequencies)
      R       (spike rate)
      Serr    (error bars) - only for err(1)&gt;=1
```

This function calls:

createdatamatpt (Section 4.32) Helper function to create an event triggered matrix from a single

mtspectrumpt (Section 4.69) Multi-taper spectrum - point process times

This function is called by:

none No functions call this function

4.76 nonst_stat

Purpose:
Nonstationarity test - continuous process

Synopsis:
`function sigma = nonst_stat(data,A,sumV,params)`

Comments:

```
Nonstationarity test - continuous process

Usage:

sigma=nonst_test(data,A,sumV,params)
Input:
Note units have to be consistent. See chronux.m for more information.
      data (1d array in samples) -- required
      A   quadratic coefficient matrix - (Compute this separately since
      the computation is time consuming - [A,sumV]=quadcof(N,NW,order). order
      has to &lt; 4NW.)
      sumV   sum of the quadratic inverse basis vectors
      params: structure with fields tapers, pad, Fs, fpass, err, trialave
      -optional
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                   (1) A numeric vector [TW K] where TW is the
                       time-bandwidth product and K is the number of
                       tapers to be used (less than or equal to
                       2TW-1).
                   (2) A numeric vector [W T p] where W is the
                       bandwidth, T is the duration of the data and p
                       is an integer such that 2TW-p tapers are used. In
                       this form there is no default i.e. to specify
                       the bandwidth, you have to specify T and p as
                       well. Note that the units of W and T have to be
                       consistent: if W is in Hz, T must be in seconds
                       and vice versa. Note that these units must also
                       be consistent with the units of params.Fs: W can
                       be in Hz if and only if params.Fs is in Hz.
                       The default is to use form 1 with TW=3 and K=5

           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
          Fs   (sampling frequency) - optional. Default 1.
Output:
      sigma   (nonstationarity index Thomson, 2000)
```

This function calls:

mtfftc (Section 4.55) Multi-taper fourier transform - continuous data

This function is called by:

none No functions call this function

4.77 padNaN

Purpose:
Creates a padded data matrix from input structural array of spike times

Synopsis:
`function data=padNaN(data)`

Comments:

```
Creates a padded data matrix from input structural array of spike times
pads with NaN
Usage: data=padNaN(data)
Input:
data : structural array of spike times
Output:
data : data matrix (zero padded)
```

This function calls:

none This function calls no functions

This function is called by:

countsig (Section 4.29) Give the program two spike data sets and one

isi (Section 4.44) Calculate the inter-spike-interval histogram

psth (Section 4.82) function to plot trial averaged rate smoothed by

4.78 plot_matrix

Purpose:
Function to plot a time-frequency matrix X. Time and frequency axes are in t and f.

Synopsis:
`function plot_matrix(X,t,f,plt,Xerr)`

Comments:

```
Function to plot a time-frequency matrix X. Time and frequency axes are in t and f.
If error bars are specified in Xerr,
it also plots them. Xerr contains upper and lower confidence intervals
on X.
Usage: plot_matrix(X,t,f,plt,Xerr)
Inputs:
X: input vector as a function of time and frequency (t x f)
t: t axis grid for plot. Default [1:size(X,1)]
f: f axis grid for plot. Default. [1:size(X,2)]
plt: 'l' for log, 'n' for no log.
Xerr: lower and upper confidence intervals for X1: lower/upper x t x f.
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.79 plot_vector

Purpose:
Function to plot a frequency dependent vector X. If error bars are specified in Xerr,

Synopsis:
`function plot_vector(X,f,plt,Xerr,c,w)`

Comments:

```
Function to plot a frequency dependent vector X. If error bars are specified in Xerr,
it also plots them. Xerr can either contain upper and lower confidence intervals
on X, or simply a theoretical confidence level (for the coherence). Used
to plot the spectrum and coherency.
Usage: plot_vector(X,f,plt,Xerr,c)
Inputs:
X: input vector as a function of frequency (f), see third argument
f: f axis grid for plot. Default. [1:length(X)]
plt: 'l' for log, 'n' for no log.
Xerr: lower and upper confidence intervals for X1: lower/upper x f. Or
      simply a single number specifying an f-independent confidence
      level.
c: controls the color of the plot - input 'b','g','r' etc. Default 'b'
w: controls the width of the lines - input 1, 1.5, 2 etc
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.80 plotsig

Purpose:
Function to plot C where it is higher than a threshold sig

Synopsis:
`function plotsig(C,sig,t,f,c)`

Comments:

```
Function to plot C where it is higher than a threshold sig
useful for plotting coherence
Usage: plotsig(C,sig,t,f)
Inputs:
C: input array t x f - also works for a single vector
sig: significance level
t: t axis grid for plot
f: f axis grid for plot.
c: color to use (default blue)-only meaningful for a line plot
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.81 plotsigdiff

Purpose:
Function to plot significant differences between two time-frequency arrays X1 and X2

Synopsis:
`function [mask,Xdiff]=plotsigdiff(X1,X1err,X2,X2err,plt,t,f)`

Comments:

```
Function to plot significant differences between two time-frequency arrays X1 and X2
given errors X1err, X2err.
Usage: mask=plotsigdiff(X1,X1err,X2,X2err,plt,t,f)

X1 err and X2err contain upper and lower confidence intervals for X1 and X2
The plot generated is shows X1-X2 where the difference is significant
either in dB or on a linear scale.

Inputs:
X1: input array t x f. Can also be a function of just the frequency.
X1err: lower and upper confidence intervals for X1: lower/upper x t x f
X2: input array t x f. if vector then as row vector
X2err: lower and upper condidence intervals for X2: lower/upper x t x f
plt: 'l' for log, 'nl' for no log,'n' for no plot at all.
t: t axis grid for plot. If X1,X2 are vectors, then specify t=1.
f: f axis grid for plot.

Outputs:
mask: +1 for all t-f (or f) indices for which the X1 significantly greater than
X2, -1 for all t-f (or f) indices for which X1 is significantly less than X2,
and zero otherwise

Xdiff: X1-X2
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.82 psth

Purpose:
function to plot trial averaged rate smoothed by

Synopsis:
```
function [R,t,E] = psth(data,sig,plt,T,err,t)
```

Comments:

```
 function to plot trial averaged rate smoothed by
 a Gaussian kernel - visual check on stationarity
 Usage: [R,t,E] = psth(data,sig,plt,T,err,t)

 Inputs:
Note that all times have to be consistent. If data
is in seconds, so must be sig and t. If data is in
samples, so must sig and t. The default is seconds.
data            structural array of spike times
sig             std dev of Gaussian (default 50ms)
                (minus indicates adaptive with
                 approx width equal to mod sig)
plt = 'n'|'r' etc      (default 'r')
T is the time interval (default all)
err - 0 = none
      1 = Poisson
      2 = Bootstrap over trials (default)
(both are based on 2* std err rather than 95%)
t   = times to evaluate psth at

The adaptive estimate works by first estimating
the psth with a fixed kernel width (-sig) it
then alters the kernel width so that the number
of spikes falling under the kernel is the same on
average but is time dependent.  Reagions rich
in data therefore have their kernel width reduced

Outputs:

R = rate
t = times
E = errors (standard error)
```

This function calls:

padNaN (Section 4.77) Creates a padded data matrix from input structural array of spike times

This function is called by:

none No functions call this function

4.83 quadcof

Purpose:
Helper function to calculate the nonstationary quadratic inverse matrix

Synopsis:
```
function [A,sumV] = quadcof(N,NW,order)
```

Comments:

```
Helper function to calculate the nonstationary quadratic inverse matrix
Usage: [A,sumV] = quadcof(N,NW,order)
N              (number of samples)
NW: Time bandwidth product
order: order (number of coefficients, upto 4NW)

Outputs:

A: quadratic inverse coefficient matrix
sumV: sum of the quadratic inverse eigenvectors
```

This function calls:

quadinv (Section 4.84) calculates the quadratic inverse eigenvectors

This function is called by:

none No functions call this function

4.84 quadinv

Purpose:
calculates the quadratic inverse eigenvectors

Synopsis:
```
function [V,E] = quadinv(N,NW)
```

Comments:

```
calculates the quadratic inverse eigenvectors

Input
N: number of samples
NW: time-bandwidth product

Output
V: The quadratic inverse eigenvectors
E: the quadratic inverse eigenvalues
```

This function calls:

none This function calls no functions

This function is called by:

quadcof (Section 4.83) Helper function to calculate the nonstationary quadratic inverse matrix

4.85 rmlinesc

Purpose:
removes significant sine waves from data (continuous data).

Synopsis:
`function data=rmlinesc(data,params,p,plt,f0)`

Comments:

```
removes significant sine waves from data (continuous data).

Usage: data=rmlinesc(data,params,p,plt,f0)

 Inputs:
Note that units of Fs, fpass have to be consistent.
      data        (data in [N,C] i.e. time x channels/trials or a single vector) - required.
      params      structure containing parameters - params has the
      following fields: tapers, Fs, fpass, pad
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

          Fs             (sampling frequency) -- optional. Defaults to 1.
             fpass       (frequency band to be used in the calculation in the form
                                 [fmin fmax])- optional.
                                 Default all frequencies between 0 and Fs/2
          pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
       p            (P-value for F-test) - optional. Defaults to 0.05/N
       where N is data length. This corresponds to a false detect
       probability of approximately 0.05

      plt         (y/n for plot and no plot respectively)
```

```
      f0          frequencies at which you want to remove the
                  lines - if unspecified the program uses the f statistic
                  to determine appropriate lines.

Outputs:
      data        (data with significant lines removed)
```

This function calls:

fitlinesc (Section 4.40) fits significant sine waves to data (continuous data).

mtspectrumc (Section 4.66) Multi-taper spectrum - continuous process

This function is called by:

none No functions call this function

4.86 rmlinesmovingwinc

Purpose:
fits significant sine waves to data (continuous data) using overlapping windows.

Synopsis:
```
function [datac,datafit,Amps,freqs]=rmlinesmovingwinc(data,movingwin,tau,params,p,plt,f0)
```

Comments:

```
 fits significant sine waves to data (continuous data) using overlapping windows.

 Usage: [datac,datafit]=rmlinesmovingwinc(data,movingwin,tau,params,p,plt)

  Inputs:
 Note that units of Fs, fpass have to be consistent.
       data        (data in [N,C] i.e. time x channels/trials or as a single vector) - required.
       movingwin         (in the form [window winstep] i.e length of moving
                                              window and step size)
                                              Note that units here have
                                              to be consistent with
                                              units of Fs - required
       tau      parameter controlling degree of smoothing for the amplitudes - we use the
       function 1-1/(1+exp(-tau*(x-Noverlap/2)/Noverlap) in the region of overlap to smooth
       the sinewaves across the overlap region. Noverlap is the number of points
       in the overlap region. Increasing tau leads to greater overlap smoothing,
       typically specifying tau~10 or higher is reasonable. tau=1 gives an almost
       linear smoothing function. tau=100 gives a very steep sigmoidal. The default is tau=10.
       params      structure containing parameters - params has the
       following fields: tapers, Fs, fpass, pad
           tapers : precalculated tapers from dpss or in the one of the following
                    forms:
                    (1) A numeric vector [TW K] where TW is the
                        time-bandwidth product and K is the number of
                        tapers to be used (less than or equal to
                        2TW-1).
                    (2) A numeric vector [W T p] where W is the
                        bandwidth, T is the duration of the data and p
                        is an integer such that 2TW-p tapers are used. In
                        this form there is no default i.e. to specify
                        the bandwidth, you have to specify T and p as
                        well. Note that the units of W and T have to be
                        consistent: if W is in Hz, T must be in seconds
                        and vice versa. Note that these units must also
                        be consistent with the units of params.Fs: W can
                        be in Hz if and only if params.Fs is in Hz.
                        The default is to use form 1 with TW=3 and K=5
                    Note that T has to be equal to movingwin(1).

            Fs             (sampling frequency) -- optional. Defaults to 1.
```

```
            fpass       (frequency band to be used in the calculation in the form
                                [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
          pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                  -1 corresponds to no padding, 0 corresponds to padding
                  to the next highest power of 2 etc.
                     e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                     to 512 points, if pad=1, we pad to 1024 points etc.
                     Defaults to 0.
       p             (P-value to calculate error bars for) - optional.
       Defaults to 0.05/Nwin where Nwin is length of window which
       corresponds to a false detect probability of approximately 0.05.
      plt          (y/n for plot and no plot respectively) - default no
                   plot.
      f0           frequencies at which you want to remove the
                   lines - if unspecified the program uses the f statistic
                   to determine appropriate lines.

Outputs:
     datafit        (fitted sine waves)
     datac          (cleaned up data)
```

This function calls:

fitlinesc (Section 4.40) fits significant sine waves to data (continuous data).

mtspectrumsegc (Section 4.70) Multi-taper segmented spectrum for a univariate continuous process

This function is called by:

none No functions call this function

4.87 runline

Purpose:
Running line fit (local linear regression)

Synopsis:
`function y_line=runline(y,n,dn)`

Comments:

```
Running line fit (local linear regression)

Usage: y_line=runline(y,n,dn);

Inputs:
y: input 1-d time series (real)
n: length of running window in samples
dn: stepsize of window in samples

Outputs:
y_line: local line fit to data
```

This function calls:

none This function calls no functions

This function is called by:

locdetrend (Section 4.46) Remove running line fit (using local linear regression)- continuous

locsmooth (Section 4.47) Running line fit (using local linear regression) - 1d only, continuous

4.88 specerr

Purpose:
Function to compute lower and upper confidence intervals on the spectrum

Synopsis:
`function Serr=specerr(S,J,err,trialave,numsp)`

Comments:

```
Function to compute lower and upper confidence intervals on the spectrum
Usage: Serr=specerr(S,J,err,trialave,numsp)
Outputs: Serr (Serr(1,...) - lower confidence level, Serr(2,...) upper confidence level)

Inputs:
S - spectrum
J - tapered fourier transforms
err - [errtype p] (errtype=1 - asymptotic estimates; errchk=2 - Jackknife estimates;
                   p - p value for error estimates)
trialave - 0: no averaging over trials/channels
           1 : perform trial averaging
numsp    - number of spikes in each channel. specify only when finite
           size correction required (and of course, only for point
           process data)

Outputs:
Serr - error estimates. Only for err(1)&gt;=1. If err=[1 p] or [2 p] Serr(...,1) and Serr(...,2)
contain the lower and upper error bars with the specified method.
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.89 spsvd

Purpose:
Space frequency SVD of input data - continuous processes

Synopsis:
`function [sv,sp,fm] = spsvd(data,params,mdkp)`

Comments:

```
Space frequency SVD of input data - continuous processes
Usage: [sv,sp,fm] = spsvd(data,params,mdkp)
Inputs:
data       (data matrix in timexchannels form)-required
      params      structure containing parameters - params has the
      following fields: tapers, Fs, fpass, pad
          tapers : precalculated tapers from dpss or in the one of the following
                   forms:
                  (1) A numeric vector [TW K] where TW is the
                      time-bandwidth product and K is the number of
                      tapers to be used (less than or equal to
                      2TW-1).
                  (2) A numeric vector [W T p] where W is the
                      bandwidth, T is the duration of the data and p
                      is an integer such that 2TW-p tapers are used. In
                      this form there is no default i.e. to specify
                      the bandwidth, you have to specify T and p as
                      well. Note that the units of W and T have to be
                      consistent: if W is in Hz, T must be in seconds
                      and vice versa. Note that these units must also
                      be consistent with the units of params.Fs: W can
                      be in Hz if and only if params.Fs is in Hz.
                      The default is to use form 1 with TW=3 and K=5

           Fs             (sampling frequency) -- optional. Defaults to 1.
          fpass       (frequency band to be used in the calculation in the form
                                 [fmin fmax])- optional.
                                Default all frequencies between 0 and Fs/2
           pad            (padding factor for the FFT) - optional (can take values -1,0,1,2...).
                   -1 corresponds to no padding, 0 corresponds to padding
                   to the next highest power of 2 etc.
                      e.g. For N = 500, if PAD = -1, we do not pad; if PAD = 0, we pad the FFT
                      to 512 points, if pad=1, we pad to 1024 points etc.
                      Defaults to 0.
mdkp       (number of dimensions to be kept)-optional. Default is the
              maximum possible modes determined by taper parameters

Outputs:
sv sp fm  : singular values, space modes, frequency modes
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.90 sta

Purpose:
Spike Triggered Average

Synopsis:
`function[s,t,E] = sta(data_spk,data_lfp,smp,plt,w,T,D,err)`

Comments:

```
Spike Triggered Average
    Usage: [s,t,E] = sta(data_spk,data_lfp,smp,plt,w,T,D,err)

Inputs

Note that all times have to be consistent. If data_spk
is in seconds, so must be sig and t. If data_spk is in
samples, so must sig and t. The default is seconds.

data_spk    - strucuture array of spike times data
              or NaN padded matrix
data_lfp    - array of lfp data(samples x trials)

Optional...
plt 'n'|'r' etc
width kernel smoothing in s
T = [-0.1 0.1] - extract this range about each spk
D = plot spike triggered average out to [D1 D2]
err = calcluate error bars (bootstrap)

Outputs:

s  spike triggered average
t  times
E  bootstrap standard err
```

This function calls:

none This function calls no functions

This function is called by:

staogram (Section 4.91)

4.91 staogram

Synopsis:
`function[S,tau,tc] = staogram(data_spk,data_lfp,smp,plt,Tc,Tinc,Tw,w,D)`

Comments:

```
staogram : calculates a moving window spike triggered ave %
  Usage:[S,tau,tc] = staogram(data_spk,data_lfp,smp,plt,Tc,Tinc,Tw,w,D)

               ******** INPUT *********
Note that all times have to be consistent. If data_spk
is in seconds, so must be sig and t. If data_spk is in
samples, so must sig and t. The default is seconds.

data_spk    - strucuture array of spike times data
data_lfp    - array of lfp data(samples x trials)
smp         - lfp times of samples

Optional...

Parameter

plt     'y'|'n'

'y' standard staogram
'n' no plot

Tc = start and end times (centres)           whole trial
Tinc = time increment between windows             0.1
Tw = time window width                          0.3
w = smoothing width in seconds
D = plot sta out to on axis [D(1) D(2)] s

               ******** OUTPUT ********
 S spike triggered average
 tau - lag
 tc  - bin centers
```

This function calls:

sta (Section 4.90) Spike Triggered Average

This function is called by:

none No functions call this function

4.92 two_group_test_coherence

Purpose:
function [dz,vdz,Adz]=two_group_test_coherence(J1c1,J2c1,J1c2,J2c2,p)

Synopsis:
`function [dz,vdz,Adz]=two_group_test_coherence(J1c1,J2c1,J1c2,J2c2,p,plt,f)`

Comments:

```
function [dz,vdz,Adz]=two_group_test_coherence(J1c1,J2c1,J1c2,J2c2,p)
Test the null hypothesis (H0) that data sets J1c1,J2c1,J1c2,J2c2 in
two conditions c1,c2 have equal population coherence

Usage:
[dz,vdz,Adz]=two_sample_test_coherence(J1c1,J2c1,J1c2,J2c2,p)

Inputs:
J1c1   tapered fourier transform of dataset 1 in condition 1
J2c1   tapered fourier transform of dataset 1 in condition 1
J1c2   tapered fourier transform of dataset 1 in condition 2
J2c2   tapered fourier transform of dataset 1 in condition 2
p      p value for test (default: 0.05)
plt    'y' for plot and 'n' for no plot
f      frequencies (useful for plotting)


Dimensions: J1c1,J2c2: frequencies x number of samples in condition 1
             J1c2,J2c2: frequencies x number of samples in condition 2
             number of samples = number of trials x number of tapers
Outputs:
dz    test statistic (will be distributed as N(0,1) under H0
vdz   Arvesen estimate of the variance of dz
Adz   1/0 for accept/reject null hypothesis of equal population
      coherences based dz ~ N(0,1)

Note: all outputs are functions of frequency

References: Arvesen, Jackkknifing U-statistics, Annals of Mathematical
Statisitics, vol 40, no. 6, pg 2076-2100 (1969)
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

4.93 two_group_test_spectrum

Purpose:
function [dz,vdz,Adz]=two_group_test_spectrum(J1,J2,p)

Synopsis:
`function [dz,vdz,Adz]=two_group_test_spectrum(J1,J2,p,plt,f)`

Comments:

```
function [dz,vdz,Adz]=two_group_test_spectrum(J1,J2,p)
Test the null hypothesis (H0) that data sets J1, J2 in
two conditions c1,c2 have equal population spectrum

Usage:
[dz,vdz,Adz]=two_sample_test_spectrum(J1,J2,p)

Inputs:
J1   tapered fourier transform in condition 1
J2   tapered fourier transform in condition 2
p      p value for test (default: 0.05)
plt    'y' for plot and 'n' for no plot
f      frequencies (useful for plotting)


Dimensions: J1: frequencies x number of samples in condition 1
            J2: frequencies x number of samples in condition 2
             number of samples = number of trials x number of tapers
Outputs:
dz    test statistic (will be distributed as N(0,1) under H0
vdz   Arvesen estimate of the variance of dz
Adz   1/0 for accept/reject null hypothesis of equal population
      coherences based dz ~ N(0,1)


Note: all outputs are functions of frequency

References: Arvesen, Jackkknifing U-statistics, Annals of Mathematical
Statisitics, vol 40, no. 6, pg 2076-2100 (1969)
```

This function calls:

none This function calls no functions

This function is called by:

none No functions call this function

Bibliography

[1] Hemant Bokil, Bijan Pesaran, Richard A. Andersen, and Partha P. Mitra. A method for detection and classification of events in neural activity. *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, 53:1678–1687, 2006.

[2] Hemant Bokil, Keith Purpura, Jan-Mathijs Schofflen, David Thompson, Bijan Pesaran, and Partha P. Mitra. Comparing spectra and coherences for groups of unequal size. *J Neurosci Methods*, 159:337–345, 2006.

[3] Hemant Bokil, Ofer Tchernichovski, and Partha P. Mitra. Dynamic phenotypes: Time series analysis techniques for characterising neuronal and behavioral dynamics. *Neuroinformatics Special Issue on Genotype-Phenotype Imaging in Neuroscience*, 4:119–128, 2006.

[4] Michael S. Fee, Partha P. Mitra, and David Kleinfeld. Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-gaussian variability. *JOURNAL OF NEUROSCIENCE METHODS*, 69:175–188, 1996.

[5] Rodolfo R. Llinas, Urs Ribary, Daniel Jeanmonod, Eugene Kronberg, and Partha P. Mitra. Thalamocortical dysrhythmia: A neurological and neuropsychiatric syndrome characterized by magnetoencephalography. *PROCEEDINGS OF THE NATIONAL ACADEMY OF SCIENCES OF THE UNITED STATES OF AMERICA*, 96:15222–15227, 1999.

[6] Clive Loader. *Local Regression and Likelihood*. Springer, 1999.

[7] Partha P. Mitra and Bijan Pesaran. Analysis of dynamic brain imaging data. *BIOPHYSICAL JOURNAL*, 76:691–708, 1999.

[8] B Pesaran, JS Pezaris, M Sahani, PP Mitra, and RA Andersen. Temporal structure in neuronal activity during working memory in macaque parietal cortex. *NATURE NEUROSCIENCE*, 5:805–811, 2002.

[9] O Tchernichovski, F Nottebohm, CE Ho, Bijan Pesaran, and PP Mitra. A procedure for an automated measurement of song similarity. *ANIMAL BEHAVIOUR*, 59:1167–1176, 2000.

[10] Thilo Womelsdorf, Pascal Fries, Partha P. Mitra, and Robert Desimone. Gamma-band synchronization in visual cortex predicts speed of change detection. *NATURE*, 439:733–736, 2006.