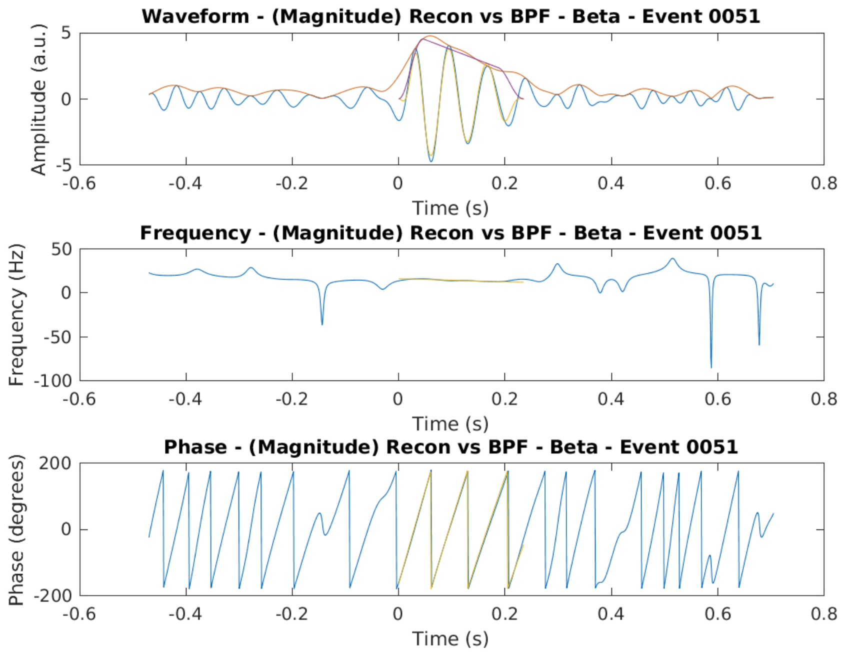
wlBurst Library – Function Reference

Written by Christopher Thomas – May 31, 2024.



Contents

Chapter 1

Overview

The wlBurst library functions are divided into several categories:

- **"Processing"** functions (ch. 3) perform segmentation, feature extraction, and other operations on single data traces. Events detected are returned as "event lists".
- **"Field Trip"** functions (ch. 4) perform event detection and other operations on Field Trip raw data structures. Events detected are returned as "event matrices". Functions that further process these event matrices are also provided in this category.
- **"Statistics"** functions (ch. 5) compile burst rate statistics from "event matrices", perform bootstrapped estimation of confidence intervals on burst rates, and create phase-shuffled surrogate data to estimate background detection rates.
- **"Synthesis"** functions (ch. 6) construct simulated LFP oscillation events based on supplied parameters. These functions are also used when reconstructing nominal detected oscillation waveforms from curve-fit parameters.
- **"Auxiliary"** functions (ch. 7) perform manipulations that don't fall into the previous categories.
- **"Plotting"** functions (ch. 8) produce rough plots of various types of data. These are included as part of the sample code and as an aid to rapid prototyping; the output is generally not publication-ready.

Several types of structure and several types of function handle are used by the library functions. These are described in Chapter 2.

Chapter 2

Special Structures and Function Handles

2.1 BURSTTYPEDEF.txt

```
wlSynth_traceAddBursts() generates oscillatory bursts with randomly varied
parameters. The range of possible values for these parameters is specified
by "burst type definition" structures, with the following fields:

"rate":                    Average number of bursts per second.
"snrrange": [min max]      Signal-to-noise ratio of bursts, in dB.
"noiseband": [min max]     Frequency band in which noise power is measured.
"durrange": [min max]      Burst duration in cycles (at average frequency).
"fctrrange": [min max]     Burst nominal center frequency.
"framprange": [min max]    Ratio of ending/starting frequency.
"aramprange": [min max]    Ratio of ending/starting amplitude.

Burst events of a given type are treated as a Poisson point process with the
specified rate. For each event, parameters are chosen randomly within the
specified ranges.

SNR in decibels is drawn using a uniform distribution. All other parameters
are drawn using a log distribution (the logarithm of the parameter values
has a uniform distribution). This handles diverse scales well.
```

2.2 COMPAREFUNC.txt

```
A "comparison function" is used for determining whether two oscillatory
burst event descriptions refer to the same event or different events, and
to meansure the "distance" between oscillatory burst events in parameter
space.

The implementation of this function is arbitrary; comparison functions are
passed as "lambda functions" (functiton handles) to library functions that
```

```
need them.

An event comparison function has the form:

[ ismatch distance ] = comparefunc(evfirst, evsecond)

The value of "ismatch" should be boolean.
The value of "distance" should be a non-negative scalar.
```

2.3 EVENTFORMAT.txt

```
Events are described by structures with the following fields. Some of these
may be omitted, depending on the function producing the event list.


Event metadata:

"sampstart":  Sample index in the original recording waveform corresponding
              to burst time 0.
"duration":   Time between nominal 50% roll-on and 50% roll-off for the burst.

"s1":         Sample index in "wave" of nominal start (time 0, 50% of roll-on).
"s2":         Sample index in "wave" of nominal stop (50% of roll-off).
"samprate":   Samples per second in the stored waveforms.

"snr":        Nominal signal-to-noise ratio for this burst, in dB.


Curve-fit parameters:

"paramtype":  Type of parameter fit performed. For now, "chirpramp".

"f1":         Burst frequency at nominal start.
"f2":         Burst frequency at nominal stop.
"a1":         Envelope amplitude at nominal start.
"a2":         Envelope amplitude at nominal stop.
"p1":         Phase at nominal start.
"p2":         Phase at nominal stop.

"rollon":     Duration of the envelope's curve-fit cosine roll-on time.
"rolloff":    Duration of the envelope's curve-fit cosine roll-off time.
"ftype":      Frequency ramp type ("linear" or "logarithmic").
"atype":      Amplitude ramp type ("linear" or "logarithmic").


Curve-fit waveform:

"times":      [1xN] array of burst sample times, relative to start time.
```

```
"wave":       [1xN] array of nominal (curve-fit) burst waveform samples.
"mag":        [1xN] array of nominal envelope magnitude samples.
"freq":       [1xN] array of nominal instantaneous frequency samples.
"phase":      [1xN] array of nominal instantaneous phase samples (in radians).


Auxiliary waveforms (optional; typically algorithm-specific):

"auxwaves":   Structure containing fields that each contain a signal.

  Signal labels are arbitrary but typically have the following forms:

  "FOOtimes":    [1xN] array with signal FOO's sample times (relative).
  "FOOwave":     [1xN] array with signal FOO's waveform samples.
  "FOOmag":      [1xN] array with signal FOO's instantaneous magnitude.
  "FOOfreq":     [1xN] array with signal FOO's instantaneous frequency.
  "FOOphase" :   [1xN] array with signal FOO's instantaneous phase.

  Field Trip waves are as follows:


Auxiliary metadata (optional; typically algorithm-specific):

"auxdata":   Structure containing algorithm-specific event metadata.

  Metadata labels are arbitrary but typically have the following form:

  "FOOstat":     Statistic "stat" for algorithm/processing step "FOO".

  Field Trip metadata is as follows:

  ft_trialstart:   Index of the trial's first sample in continuous data
                   (from the first column of ftdata.sampleinfo).
  ft_sampstart:    Index of "sampstart" in the curve-fit wave relative to
                   t=0 in the Field Trip trial.
  ft_sampend:      Index of "sampend" in the curve-fit wave relative to
                   t=0 in the Field Trip trial.
  ft_trialnum:     Field Trip trial index.
  ft_channelnum:   Field Trip channel index.
  ft_bandnum:      Band index from the list of bands supplied to
                   wlFT_doFindEventsInTrials.
  ft_bandmin:      Lower corner frequency of the band.
  ft_bandmax:      Upper corner frequency of the band.
  ft_eventnum:     Event index in the list of detected events in this trial.

  Additional metadata from Field Trip curve fits is also recorded.


(This is the end of the file.)
```

2.4 EVMATRIX.txt

```
The "event matrix" structure holds event information extracted from Field
Trip data. This data is kept in wlBurst library format, to avoid discarding
metadata and diagnostic information. This may be converted to and from
the "event FT" format, with the loss of some metadata.


An "event matrix" structure contains the following fields, which hold data
returned by "wlProc_doSegmentAndParamExtract":

- "events{bidx, tidx, cidx}" holds event lists corresponding to a given
  band, FT trial, and channel within the FT trial.

- "waves{bidx, tidx, cidx}" holds diagnostic waveform information from a
  given band, FT trial, and channel within the FT trial. This includes
  raw FT trial waveforms ("ftwave") and FT timestamps ("fttimes").

- "auxdata{bidx, tidx, cidx}" holds auxiliary diagnostic data from a given
  band, FT trial, and channel within the FT trial.


The following additional fields are also present:

- "bandinfo" is an array of structures with the following fields:
  "band" [ min max ] defines a frequency band of interest.
  "name" is a character array containing a human-readable band name.
  "label" is a character array containing a filename-safe band ID string.

- "samprate" is the sampling rate (samples per second).

- "segconfigbyband{bidx}" records segmentation algorithm information per
  "SEGCONFIG.txt".

- "paramconfigbyband{bidx}" records parameter estimation algorithm
  information per "PARAMCONFIG.txt".


(This is the end of the file.)
```

2.5 FIGCONFIG.txt

```
The "figure configuration information" structure contains the following
fields:

General parameters:
```

```
"fig" - A figure handle to perform rendering on.
"outdir" - The directory to write figure files to.

"fsamp" - Time series sampling rate. Used for spectrum plots.

Spectrum plot tuning parameters [typical values in square brackets]:

"psfres" - Frequency resolution, Hz. [20]
"psolap" - Overlap between time and frequency bins, percent. [99]
"psleak" - "Leakage" between bins (0.5 = Hann window, 1.0 = square).
  [0.75]
"psylim" - Maximum frequency plotted. [50]



Tuning time/frequency plots is a black art. A useful introduction is at:
https://www.mathworks.com/help/signal/examples/practical-introduction-to-time-frequency-analysis.h

Details of the implementation of the "pspectrum" function are at:
https://www.mathworks.com/help/signal/ref/pspectrum.html
```

2.6 PARAMCONFIG.txt

```
Parameter extraction algorithm configuration structures have a mandatory
field "type", with additional fields that depend on the algorithm specified
by the "type" field.

Algorithm-specific arguments are as follows:



== "groundtruth" type:

Parameters are known a priori.



== "default" type:

The wlProc library is to replace this with a configuration of its choice.
FIXME - NYI.



== "custom" type:

This wraps a user-specified parameter extraction function.
```

```
"paramfunc":  A function handle with the form:
  [ newevents, auxdata ] = paramfunc( oldevents, waves, samprate, paramconfig )

The function handle is passed a copy of this structure as its "paramconfig"
argument, allowing additional parameters to be passed.

Function arguments are:
  "oldevents":  The list of events produced by segmentation. Only the
    "sampstart" and "duration" fields are guaranteed to exist.
  "waves":  The list of derived waveforms produced by segmentation. Which
    of these exist depends on the segmentation algorithm. These typically
    include a band-pass-filtered waveform and its Hilbert transform
    magnitude, phase, and frequency series.
  "samprate":  The number of samples per second in the signal data.
  "paramconfig":  A copy of this structure, containing additional parameters.

Function return values are:
  "newevents":  An updated list of events with remaining fields filled in.
  "auxdata":  A structure containing additional statistics or diagnostic
    data. This structure may be empty.



== "grid" type:

Amplitude envelope roll-on and roll-off are fit using a coarse grid search.
For a given roll-on and roll-off, magnitude and frequency are curve-fit.
Endpoints are kept fixed.

"gridsteps":  The number of intermediate time points tested as roll-on and
              roll-off endpoints.

The number of cases tested goes up as the square of "gridsteps".


== "annealamp" type:

Perform "grid" for a coarse fit, then use simulated annealing on the envelope
to fit it to the analytic signal magnitude signal.

"gridsteps":   The number of intermediate time points for the "grid" fit.
"matchfreq":   The maximum allowed frequency ratio between original and
               perturbed events.
"matchamp:     The maximum allowed peak-amplitude ratio between original and
               perturbed events.
"matchlength": The maximum allowed length ratio between original and perturbed
               events.
"matcholap":   The minimum fraction of the shorter event that must be covered
               by the larger event, between original and perturbed events.
"tunnelmax":   The maximum number of consecutive unproductive perturbation
```

```
               attempts that can be made before annealing is assumed to have
               converged.
"totalmax":    The maximum number of perturbation attempts that can be made
               in total while annealing one event record.


== "annealboth" type:

Perform "grid" for a coarse fit, then use simulated annealing on the envelope
to fit it to the analytic signal magnitude signal, then use simulated
annealing on amplitude and frequency to fit the event and signal waveforms.

Arguments are identical to "annealamp".



This is the end of the file.
```

2.7 SEGCONFIG.txt

```
Segmentation algorithm configuration structures have a mandatory field
"type", with additional fields that depend on the algorithm specified by the
"type" field.

Algorithm-specific arguments are as follows:



== "groundtruth" type:

Segments are known a priori.



== "default" type:

The wlProc library is to replace this with a configuration of its choice.
FIXME - NYI.



== "custom" type:

This wraps a user-specified segmentation function.

"segmentfunc":  A function handle with the form:
  [ events, waves ] = segmentfunc( wavedata, samprate, bpfband, segconfig )
```

```
The function handle is passed a copy of this structure as its "segconfig"
argument, allowing additional parameters to be passed.

Function arguments are:
  "wavedata":  The data trace to examine.
  "samprate":  The number of samples per second in the signal data.
  "bpfband" [min max]:  The frequency band of interest. Edges may fade.
  "segconfig":  A copy of this structure, containing additional parameters.

Function return values are:
  "events":  An array of event record structures per EVENTFORMAT.txt.
    Only the "sampstart" and "duration" fields are provided.
  "waves":   A structure containing new waveforms derived from "wavedata".
    These typically include a band-pass-filtered waveform and its Hilbert
    transform magnitude, phase, and frequency series.


== "mag" type:

Detection looking for excursions in analytic signal magnitude.
As "magdual", but omitting "dbend".



== "magdual" type:

Detection looking for excursions in analytic signal magnitude.

"qlong":    Time constant for DC-average filtering, as a multiple of the
            nominal oscillation period.
"qdrop":    Maximum gap in detection to ignore, as a multiple of the nominal
            oscillation period.
"qglitch":  Maximum spurious detection to ignore, as a multiple of the nominal
            oscillation period.
"dbpeak":   Minimum power ratio (in dB) above background for event detection.
"dbend":    Minimum power ratio (in dB) above background at the edges of a
            detected event. This is used to determine event boundaries.

The standard "3 sigma" heuristic for detection corresponds to a "dbpeak"
value of 9.5 dB. A typical "dbend" value is 2 dB.



== "freq" type:

Detection looking for stabilization of analytic signal frequency.
As "freqdual", but omitting "dbend".
```

```
== "freqdual" type:

Detection looking for stabilization of analytic signal frequency.

This involves adding high-frequency noise to the band-pass-filtered signal;
oscillations with amplitude significantly higher than the HF noise floor
stabilize the analytic signal's computed frequency (derivative of phase).

"noiseband":  [min max]  Frequency band for noise injection.
"noisesnr":   Ratio of signal power to injected noise power (in dB).
"qlong":    Time constant for DC-average filtering, as a multiple of the
            nominal oscillation period.
"qshort":   Time constant for smoothing analytic signal frequency, as a
            multiple of the nominal oscillation period.
"qdrop":    Maximum gap in detection to ignore, as a multiple of the nominal
            oscillation period.
"qglitch":  Maximum spurious detection to ignore, as a multiple of the nominal
            oscillation period.
"dbpeak":   Minimum noise power suppression ratio (in dB) for event detection.
"dbend":    Minimum noise power suppression ratio (in dB) at the edges of a
            detected event. This is used to determine event boundaries.

The noise band's low frequency edge is typically at least 10 times the event
band's upper frequency edge.
The standard "3 sigma" heuristic for detection corresponds to a "dbpeak"
value of 9.5 dB. A typical "dbend" value is 2 dB.



This is the end of the file.
```

2.8 WAVEERRFUNC.txt

```
A "wave error function" is used for evaluating the "error" (distance) between
an event and a reference waveform. Typically this is done via time-domain
comparison between the event waveform and reference waveform, but other
approaches are possible.

The implementation of this function is arbitrary; wave error functions are
passed as "lambda functions" (function handles) to library functions that
need them.


A wave error function has the form:

error = errfunc(event, wavestruct)

The value of "error" should be a non-negative scalar.
```

```
"event" is an event record, per EVENTFORMAT.txt.
"wavestruct" is a structure containing one or more reference waveforms. This
  is typically one cell extracted from the "waves" field of an event matrix,
  per EVMATRIX.txt.


An example implementation is:

waveerrfunc_bpf = @(thisev, thiswave) ...
  wlProc_calcWaveErrorRelative( thiswave.bpfwave, thisev.sampstart, ...
    thisev.wave, thisev.s1 );
```

2.9 WLNOTES.txt

```
In addition to the raw field trip structures, the wlFT wrapper routines
produce a "wlnotes" structure containing additional metadata. Contents
are described below.


"wlnotes.bandinfo" is an array of structures containing band definitions:

- "band" [ min max ] contains the frequency limits of the band.
- "name" is a character array containing a human-readable band name.
- "label" is a unique filename-safe character string identifying the band.


"wlnotes.wavesbybandandtrial{bidx,tidx,cidx}" is a cell array containing
structures that contain diagnostic waveforms. These waveforms are derived
from the original trial data and represent intermediate processing steps.
Different analyses will produce different sets of diagnostic waveforms.

wlBurst event records that are annotated with "context" typically have
corresponding portions of these signals in their "auxwaves" structures.
wlBurst event matrices store these in the "waves" cell array.

- "ftwave" is the original wideband trial waveform.
- "fttimes" contains original timestamps associated with "ftwave" data.

- "bpfwave" is the band-pass-filtered trial waveform.
- "bpf(mag|freq|phase)" are analytic signals derived from "bpfwave", from
  analyses that use the Hilbert transform.

- "magpower(fast|slow)" are low-pass-filtered squared magnitudes computed
  from the analytic magnitude of the signal, from analyses that use
  magnitude-based detection.

- "noisywave" is a copy of "bpfwave" with high-frequency noise added, from
```

  analyses that use frequency-stability detection.
- "noisy(mag|freq|phase)" are analytic signals derived from "noisywave", from analyses that use frequency-stability detection.
- "fvar(fast|slow)" are low-pass-filtered versions of the squared magnitude of the AC component of "noisyfreq". These represent the instantaneous variance of "noisyfreq" averaged over narrow and wide time windows.

"wlnotes.trialinfo_label" is a cell array of character arrays containing labels for columns in the "trialinfo" metadata structure. Metadata present depends on the analyses performed.

wlBurst event records typically store some or all of this metadata in their "auxdata" structures (aside from curve-fit event parameters, which are stored as top-level event record fields). wlBurst event matrices store this in the "auxdata" cell array. In wlBurst event and event matrix structures, the string "ft_" is prepended to the labels associated with these fields (i.e. "ft_sampstart" in auxdata, vs "sampstart" in wlnotes).

Metadata that is always present:

- "sampstart" is a duplicate of "sampleinfo(trial,1)".
- "sampend" is a duplicate of "sampleinfo(trial,2)".
- "trialnum" is the original-data trial in which this event occurred.
- "channelnum" is the original-data channel in which this event occurred.
- "bandnum" is the index of the detection band within "bandinfo".
- "bandmin" is a duplicate of "bandinfo.band(1)".
- "bandmax" is a duplicate of "bandinfo.band(2)".
- "eventnum" is the event number within this trial, channel, and band.

Event detection metadata:

- "detthresh" is the minimum peak power or variance excursion for event detection, in dB.
- "trial_bpf_rms" is the RMS amplitude of the band-pass-filtered version of the original trial ("bpfwave" in "wavesbybandandtrial").
- "event_rms" is the RMS amplitude of the reconstructed event.
- "event_max" is the maximum absolute amplitude of the reconstructed event.

Cropped trial metadata:

- "roistart" is the first copied sample location in the un-cropped trial.
- "roistop" is the last copied sample location in the un-cropped trial.

Parameter fit metadata:

- "f1" is the curve-fit frequency at 50% roll-on.
- "f2" is the curve-fit frequency at 50% roll-off.
- "a1" is the curve-fit envelope magnitude at 50% roll-on.
- "a2" is the curve-fit envelope magnitude at 50% roll-off.

- "p1" is the curve-fit phase at 50% roll-on.
- "p2" is the curve-fit phase at 50% roll-off.
- "rollon" is the duration of roll-on, in seconds.
- "rolloff" is the duration of roll-off, in seconds.
- "duration" is the time between 50% roll-on and 50% roll-off, in seconds.

"wlnotes.segconfigbyband{bidx}" and "wlnotes.paramconfigbyband{bidx}" store the algorithm tuning parameters used for segmentation and parameter extraction of events in the various frequency bands, respectively.

Chapter 3

“wlProc” Functions

3.1 wlProc_calcAnalytic.m

```
% function [ rawmag, rawfreq, rawphase ] = ...
%   wlProc_calcAnalytic(data, samprate)
%
% This function uses the Hilbert transform to generate an analytic signal
% for the specified input signal, and from that extracts magnitude and phase.
% Frequency is calculated using the first difference of phase.
%
% Calculated frequency will be very noisy. The last sample of frequency is
% replicated, for consistent array lengths. The first sample of frequency
% occurs between the first and second samples of the other traces.
%
% "data" is the real-valued signal to analyze.
% "samprate" is the number of samples per second (used to calculate frequency).
%
% "rawmag" contains analytic signal instantaneous magnitudes.
% "rawfreq" contains the estimated instantaneous frequency of the signal.
% This can be anywhere from -nyquist to +nyquist; non-artifact values should
% be strictly positive and in biologically relevant bands.
% "rawphase" contains analytic signal instantaneous phase angles. These are
% in the range 0..2pi (i.e. not unwrapped).
```

3.2 wlProc_calcBPFAnalytic.m

```
% function [ rawwave, hilmag, hilfreq, hilphase ] = ...
%   wlProc_calcBPFAnalytic(data, samprate, bpfband)
%
% This function band-pass filters an input signal, and then returns the
% analytic signal components of the band-limited signal. Frequency is
% calculated using the first difference of phase (this function wraps
% wlProc_calcAnalytic).
%
% "data" is the real-valued signal to analyze.
% "samprate" is the number of samples per second in the signal data.
% "bpfband" [min max] is the frequency band of interest. Edges may fade.
%
% "rawwave" contains the noisy band-limited waveform samples.
% "hilmag" contains analytic signal instantaneous magnitudes.
% "hilfreq" contains the estimated instantaneous frequency of the signal.
%   This can be anywhere from -nyquist to +nyquist for a noisy signal.
% "hilphase" contains analytic signal instantaneous phase angles. These are
%   in the range 0..2pi (i.e. not unwrapped).
```

3.3 wlProc_calcCosineWindow.m

```
% function wave = wlProc_calcCosineWindow(s2, s3, s4)
%
% This function generates a cosine roll-off window with the two roll-off
% lengths independent of each other.
%
% This returns a [1xN] array of samples in the range 0..1.
%
% wave(1) is 0.
% wave(s2) is the first sample with a value of 1.
% wave(s3) is the last sample with a value of 1.
% wave(s4) is 0.
%
% If (s2-1) = (s4-s3), the result is a Tukey window (symmetrical roll-off).
% If s2 = s3 and (s2-1) = (s4-s3), the result is a von Hann window (cosine).
```

3.4 wlProc_calcDeGlitchedVector.m

```
% function newvec = wlProc_calcDeGlitchedVector(oldvec, maxglitch, maxdrop)
%
% This function removes spurious gaps (drop-outs) and brief events (glitches)
% in a one-dimensional logical array. Durations are specified as sample
% counts.
%
% "oldvec" is the vector to process. Elements should be "true" or "false".
% "maxglitch" is the longest event duration to reject as spurious.
% "maxdrop" is the longest gap duration within an event to reject as spurious.
%
% "newvec" is a modified version of "oldvec" with glitches and drop-outs
%   removed.
```

3.5 wlProc_calcEventSNRs.m

```
% function newevents = wlProc_calcEventSNRs(data, samprate, oldevents)
%
% This function calculates the approximate signal-to-noise ratios of a list
% of detected events. The event waveform is reconstructed from its extracted
% paramters, and the power of the middle 80% is compared to the average power
% of the input waveform (which is assumed to be noise-dominated).
%
% "data" is the data trace to examine. This is typically band-pass filtered.
% "samprate" is the number of samples per second in the signal data.
% "oldevents" is the input event list. See EVENTFORMAT.txt for contents.
%
% "newevents" is a copy of the input list with the "snr" field filled in.
```

3.6 wlProc_calcMagPowerFastSlow.m

```
% function [ powerfast, powerslow ] = ...
%   wlProc_calcMagPowerFastSlow(magsignal, samprate, flong, fshort)
%
% This function computes the instantaneous power of the fast-changing and
% slowly-changing portions of a supplied magnitude waveform (a real and
% non-negative signal).
%
% Each of these is computed as the squared value of a low-pass-filtered
% version of the input signal, using different filter corner frequencies.
% If "fshort" is omitted, the "fast-changing" signal is the input signal.
%
% "magsignal" is a real-valued non-negative signal to analyze.
% "samprate" is the number of samples per second in the signal data.
% "flong" is the maximum frequency in the "slowly-changing" signal. Set this
%   to zero to use the DC average.
% "fshort" is the maximum frequency in the "fast-changing" signal.
%
% "powerfast" is the instantaneous power of the fast-changing signal.
% "powerslow" is the instantaneous power of the slowly-changing signal.
```

3.7 wlProc_calcMatchFromParams.m

```
% function [ ismatch distance ] = ...
%   wlProc_calcMatchFromParams(evfirst, evsecond, ...
%     freqratiomax, ampratiomax, lenratiomax, olapfracmin)
%
% This function compares two event records, indicating whether or not they
% "match" (represent the same event), and calculating a "distance" measure
% between them (with smaller values indicating a closer match).
% Distance and match state are computed using extracted parametric burst
% parameters (starting and ending time, amplitude, and frequency).
%
% "evfirst" and "evsecond" are event records to compare. See EVENTFORMAT.txt
%   for a description of record fields.
% "freqratiomax" is the maximum frequency ratio between matching events.
% "ampratiomax" is the maximum amplitude ratio between matching events.
% "lenratiomax" is the maximum duration ratio between matching events.
% "olapfracmin" is the minimum fraction of the shorter event that must be
%   covered by the longer event for a match to be accepted.
%
% "ismatch" is a boolean value that is "true" if the events match.
% "distance" is a non-negative real value representing how far apart the
%   events are. Smaller values indicate a closer match.
% "dvec" is a vector representing the distance between the two events.
%   Components of this vector typically represent axes of parameter space.
% "cnames" is a cell array containing label strings naming the distance
%   vector components.
```

3.8 wlProc_calcNoisyAnalytic.m

```
% function [ rawwave, hilmag, hilfreq, hilphase ] = ...
%   wlProc_calcNoisyAnalytic(data, samprate, sigband, noiseband, noisesnr)
%
% This function adds band-limited noise to a signal and then returns the
% analytic signal components of the noisy signal. Frequency is calculated
% using the first difference of phase (this function wraps
% wlProc_calcAnalytic).
%
% "data" is the real-valued signal to analyze.
% "samprate" is the number of samples per second in the signal data.
% "sigband" [min max] is the band to measure signal power in.
% "noiseband" [min max] is the band to add white noise in.
% "noisesnr" is the ratio of signal power to white noise power, in dB.
%
% "rawwave" contains the noisy waveform samples.
% "hilmag" contains analytic signal instantaneous magnitudes.
% "hilfreq" contains the estimated instantaneous frequency of the signal.
%   This can be anywhere from -nyquist to +nyquist for a noisy signal.
% "hilphase" contains analytic signal instantaneous phase angles. These are
%   in the range 0..2pi (i.e. not unwrapped).
```

3.9 wlProc_calcPaddedBandpass.m

```
% function newdata = wlProc_calcPaddedBandpass(olddata, band, samprate)
```

```
%
% This function wraps Matlab's "bandpass" function.
% Zero-padding is added on both sides of the data to give Matlab's filters
% time to stabilize. This is removed when the trace is returned.
% There will still be artifacts, but amplitude should be much lower than
% calling "bandpass" directly.
%
% "olddata" is the signal to filter.
% "band" [ min max ] is the frequency band to pass.
% "samprate" is the number of samples per second in the signal data.
%
% "newdata" is the filtered signal.
```

3.10 wlProc_calcPaddedLowpass.m

```
% function newdata = wlProc_calcPaddedLowpass(olddata, fcorner, samprate)
%
% This function wraps Matlab's "lowpass" function.
% Zero-padding is added on both sides of the data to give Matlab's filters
% time to stabilize. This is removed when the trace is returned.
% There will still be artifacts, but amplitude should be much lower than
% calling "lowpass" directly.
%
% "olddata" is the signal to filter.
% "fcorner" is the filter's cutoff frequency.
% "samprate" is the number of samples per second in the signal data.
%
% "newdata" is the filtered signal.
```

3.11 wlProc_calcShortLowpass.m

```
% function newdata = wlProc_calcShortLowpass(olddata, fcorner, samprate, fact)
%
% This function implements low-pass filtering using raised cosine windows
% as FIR filters. The farthest any disturbance can propagate is the width
% of the window.
%
% Mathematically we're convolving by an anti-aliasing filter, decimating,
% and then convolving by a reconstruction filter, where the AA and recon
% filters are raised cosine windows. The radius of the window is an
% integer multiple of the decimation sample spacing.
%
% "olddata" is the signal to filter.
% "fcorner" is the filter's cutoff frequency.
% "samprate" is the number of samples per second in the signal data.
% "fact" is a positive integer. The Nyquist frequency of the decimated
%   signal is "fact" times the corner frequency.
%
% Calculations are performed in the time domain, so this will be slow for
% "fact" >> 1. We don’t actually convolve, so for small "fact" it’s efficient.
% Filter slope isn’t very steep so aliasing and leakage are concerns.
%
% "newdata" is the filtered signal.
```

3.12 wlProc_calcVarFastSlow.m

```
% function [ varfast, varslow ] = ...
%   wlProc_calcVarFastSlow(data, samprate, fdc, flong, fshort)
%
% This function calculates the fast-changing and slowly-changing portions
% of the variance of a supplied signal waveform.
%
% The local mean of the signal is obtained by low-pass-filtering with
% cuttoff "fdc". The instantaneous variance is computed by squaring the
% residue after the local mean is subtracted. Rapidly-changing and slowly-
% -changing variances are computed by low-pass-filtering the instantaneous
% variance with cutoffs "fshort" and "flong", respectively; these calculate
% the average of the instantaneous windows within their equivalent windows.
%
% "data" is a real-valued signal to analyze.
% "samprate" is the number of samples per second in the signal data.
% "fdc" is the maximum frequency in the "mean" signal. Set this to zero to
%   use the DC average.
% "flong" is the maximum frequency in the "slowly-changing" variance signal.
%   Set this to zero to use the DC average.
% "fshort" is the maximum frequency in the "fast-changing" variance signal.
%
% "varfast" is the "fast-changing" running variance signal.
% "varslow" is the "slowly-changing" running variance signal.
```

3.13 wlProc_calcWaveErrorAbsolute.m

```
% function error = wlProc_calcWaveErrorAbsolute( ...
%   bigwave, bigstart, subwave, substart)
%
% This function calculates the "error" between an event waveform and a
% reference waveform.
% The "Absolute" version computes this as RMS(error) without normalizing.
%
% "bigwave" contains reference waveform samples.
% "bigstart" is the sample index of the alignment point within "bigwave".
% "subwave" contains the event waveform samples.
% "substart" is the sample index of the alignment point within "subwave".
%
% "error" is a non-negative real value.
```

3.14 wlProc_calcWaveErrorRelative.m

```
% function error = wlProc_calcWaveErrorRelative( ...
%   bigwave, bigstart, subwave, substart)
%
% This function calculates the "error" between an event waveform and a
% reference waveform.
% The "Relative" version computes this as RMS(error) / RMS(reference).
%
% "bigwave" contains reference waveform samples.
% "bigstart" is the sample index of the alignment point within "bigwave".
% "subwave" contains the event waveform samples.
% "substart" is the sample index of the alignment point within "subwave".
%
% "error" is a non-negative real value.
```

3.15 wlProc_doAnneal.m

```
% function [ bestvec longestprobe endtotal ] = ...
%   wlProc_doAnneal(scalevec, errfunc, maxprobes, maxtotal)
%
% This function performs simulated annealing, searching for a perturbation
% vector about a known solution that results in the minimum "error" value.
%
% Search steps are drawn from a scaled Gaussian distribution. Radius is
% scaled by an exponential factor, and components are further scaled by
% the amounts dictated by "scalevec". The purpose of exponential scaling is
% to allow tunnelling between distant minima while also allowing fine-scale
% gradient descent, without foreknowledge of the scales involved. The purpose
% of "scalevec" is to ensure that for a given choice of exponential scale,
% feature sizes in all directions are roughly comparable. If feature sizes
% are not known, all elements can be set to unity and annealing will still
% work (just with slower convergence).
% FIXME - We might want an inverted whitening matrix instead of "scalevec".
%
% Details of the problem being optimized are encapsulated in "errfunc". The
% annealing function provides "errfunc" with a perturbation vector (with the
% starting state being the zero vector). Interpretation of that vector in
% solution space is up to "errfunc".
%
% "scalevec" is a [1xN] vector whose components indicate the "natural scale"
%   of their respective state vector/perturbation vector elements.
% "errfunc" is a function for evaluating proposed perturbed solutions. This
%   has the form: [ isvalid error ] = errfunc(perturbvec)
% "maxprobes" is the maximum number of unproductive perturbation attempts that
%   can be made in one step before annealing terminates.
% "maxtotal" is the maximum number of total perturbation attempts, productive
%   or not, that can be made before annealing terminates.
%
% "bestvec" is the perturbation vector obtained after annealing.
% "longestprobe" is the maximum number of perturbation attempts actually made
%   during any single probe. If this is less than "maxprobes", annealing
%   didn't converge before reaching "maxtotal" attempts.
% "endtotal" is the total number of perturbation attempts actually made. If
%   this is less than "maxtotal", annealing did converge.
```

3.16 wlProc_doEvAnnealAllWave.m

```
% function [ newevents avglongestprobe avgendtotal convergefrac ] = ...
%   wlProc_doEvAnnealAllWave( ...
%     oldevents, samprate, wavedata, ...
%     comparefunc, maxprobes, maxtotal)
%
% This function anneals an event list with approximate parameters, producing
% a list with parameters perturbed to better match the input.
% The "All" implementation anneals all event parameters, rather than fitting.
% The "Wave" implementation fits to the signal waveform, rather than to
% the analytic version.
% This is the single-threaded implementation. Use _MT for multithreaded.
%
% "oldevents" is the input event list, with format per EVENTFORMAT.txt.
% "samprate" is the number of samples per second in the signal data.
% "wavedata" is the signal waveform the event was extracted from.
% "comparefunc" is a function handle for an event comparison function, as
%   described in COMPAREFUNC.txt. Perturbed events must "match" the original
%   event to be valid. This has the form:
%     [ ismatch distance ] = comparefunc(evfirst, evsecond)
% "maxprobes" is the maximum number of unproductive perturbation attempts that
%   can be made in one step before annealing terminates.
% "maxtotal" is the maximum number of total perturbation attempts, productive
%   or not, that can be made before annealing terminates.
%
% "newevents" is the perturbed event list. Annealing statistics are stored in
%   "auxdata" (as "AnnWave_maxtunnel", "AnnWave_total", "AnnWave_errstart",
%   and "AnnWave_errfinal").
% "avglongestprobe" is the average across events of the maximum number of
%   perturbation attempts made during any single probe during annealing.
% "avgendtotal" is the average across events of the total number of
%   perturbation attempts made while annealing an event.
% "convergefrac" is the fraction of events where annealing took fewer than
%   95% of "maxtotal" perturbation attempts.
```

3.17 wlProc_doEvAnnealAllWave_MT.m

```
% function [ newevents avglongestprobe avgendtotal convergefrac ] = ...
%   wlProc_doEvAnnealAllWave_MT( ...
%     oldevents, samprate, wavedata, ...
%     comparefunc, maxprobes, maxtotal)
%
% This function anneals an event list with approximate parameters, producing
% a list with parameters perturbed to better match the input.
% The "All" implementation anneals all event parameters, rather than fitting.
% The "Wave" implementation fits to the signal waveform, rather than to
% the analytic version.
% This is a multithreaded wrapper for the single-threaded version.
%
% "oldevents" is the input event list, with format per EVENTFORMAT.txt.
% "samprate" is the number of samples per second in the signal data.
% "wavedata" is the signal waveform the event was extracted from.
% "comparefunc" is a function handle for an event comparison function, as
%   described in COMPAREFUNC.txt. Perturbed events must "match" the original
%   event to be valid. This has the form:
%     [ ismatch distance ] = comparefunc(evfirst, evsecond)
% "maxprobes" is the maximum number of unproductive perturbation attempts that
%   can be made in one step before annealing terminates.
% "maxtotal" is the maximum number of total perturbation attempts, productive
%   or not, that can be made before annealing terminates.
%
% "newevents" is the perturbed event list. Annealing statistics are stored in
%   "auxdata" (as "AnnWave_maxtunnel", "AnnWave_total", "AnnWave_errstart",
%   and "AnnWave_errfinal").
% "avglongestprobe" is the average across events of the maximum number of
%   perturbation attempts made during any single probe during annealing.
% "avgendtotal" is the average across events of the total number of
%   perturbation attempts made while annealing an event.
% "convergefrac" is the fraction of events where annealing took fewer than
%   95% of "maxtotal" perturbation attempts.
```

3.18 wlProc_doEvAnnealAmplitudeHilbert.m

```
% function [ newevents avglongestprobe avgendtotal convergefrac ] = ...
%   wlProc_doEvAnnealAmplitudeHilbert( ...
%     oldevents, samprate, hilmag, hilfreq, hilphase, ...
%     comparefunc, maxprobes, maxtotal)
%
% This function anneals an event list with approximate parameters, producing
% a list with parameters perturbed to better match the input.
% The "Amplitude" implementation just anneals envelope shape, re-fitting
% frequency and phase afterwards.
% The "Hilbert" implementation fits to analytic parameters.
% This is the single-threaded implementation. Use _MT for multithreaded.
%
% "oldevents" is the input event list, with format per EVENTFORMAT.txt.
% "samprate" is the number of samples per second in the signal data.
% "hilmag" is the instantaneous magnitude of the analytic signal.
% "hilfreq" is the instantaneous frequency of the analytic signal.
% "hilphase" is the instantaneous phase of the analytic signal.
% "comparefunc" is a function handle for an event comparison function.
%   Perturbed events must "match" the original event to be valid. This
%   has the form: [ ismatch distance ] = comparefunc(evfirst, evsecond)
% "maxprobes" is the maximum number of unproductive perturbation attempts that
%   can be made in one step before annealing terminates.
% "maxtotal" is the maximum number of total perturbation attempts, productive
%   or not, that can be made before annealing terminates.
%
% "newevents" is the perturbed event list. Annealing statistics are stored in
%   "auxdata" (as "AnnAmp_maxtunnel", "AnnAmp_total", "AnnAmp_errstart", and
%   "AnnAmp_errfinal").
% "avglongestprobe" is the average across events of the maximum number of
%   perturbation attempts made during any single probe during annealing.
% "avgendtotal" is the average across events of the total number of
%   perturbation attempts made while annealing an event.
% "convergefrac" is the fraction of events where annealing took fewer than
%   95% of "maxtotal" perturbation attempts.
```

3.19 wlProc_doEvAnnealAmplitudeHilbert_MT.m

```
% function [ newevents avglongestprobe avgendtotal convergefrac ] = ...
%   wlProc_doEvAnnealAmplitudeHilbert( ...
%     oldevents, samprate, hilmag, hilfreq, hilphase, ...
%     comparefunc, maxprobes, maxtotal)
%
% This function anneals an event list with approximate parameters, producing
% a list with parameters perturbed to better match the input.
% The "Amplitude" implementation just anneals envelope shape, re-fitting
% frequency and phase afterwards.
% The "Hilbert" implementation fits to analytic parameters.
% This is a multithreaded wrapper for the single-threaded version.
%
% "oldevents" is the input event list, with format per EVENTFORMAT.txt.
% "samprate" is the number of samples per second in the signal data.
% "hilmag" is the instantaneous magnitude of the analytic signal.
% "hilfreq" is the instantaneous frequency of the analytic signal.
% "hilphase" is the instantaneous phase of the analytic signal.
% "comparefunc" is a function handle for an event comparison function.
%   Perturbed events must "match" the original event to be valid. This
%   has the form: [ ismatch distance ] = comparefunc(evfirst, evsecond)
% "maxprobes" is the maximum number of unproductive perturbation attempts that
%   can be made in one step before annealing terminates.
% "maxtotal" is the maximum number of total perturbation attempts, productive
%   or not, that can be made before annealing terminates.
%
% "newevents" is the perturbed event list. Annealing statistics are stored in
%   "auxdata" (as "AnnAmp_maxtunnel", "AnnAmp_total", "AnnAmp_errstart", and
%   "AnnAmp_errfinal").
% "avglongestprobe" is the average across events of the maximum number of
%   perturbation attempts made during any single probe during annealing.
% "avgendtotal" is the average across events of the total number of
%   perturbation attempts made while annealing an event.
% "convergefrac" is the fraction of events where annealing took fewer than
%   95% of "maxtotal" perturbation attempts.
```

3.20 wlProc_doSegmentAndParamExtract.m

```
% function [ events auxwaves auxdata ] = wlProc_doSegmentAndParamExtract( ...
%   wavedata, samprate, bpfband, segconfig, paramconfig, wantmultithread)
%
% This function performs event segmentation and parameter extraction on the
% specified waveform. Specific algorithms and their tuning parameters are
% selected/specified via configuration structures.
% This version defaults to single-threaded; "wantmultithreaded" selects MT.
%
% "wavedata" is the data trace to examine.
% "samprate" is the number of samples per second in the signal data.
% "bpfband" [min max] is the frequency band of interest. Edges may fade.
% "segconfig" is a structure containing configuration information for
%   event segmentation, per SEGCONFIG.txt.
% "paramconfig" is a structure containing configuration information for
%   event parameter estimation, per PARAMCONFIG.txt.
% "wantmultithread" is "true" if _MT variants of functions are to be called.
%   This parameter is optional; the default is "false" (single-threaded).
%
% "events" is an array of structures describing detected events. Format is
%   per EVENTFORMAT.txt.
% "auxwaves" is a structure containing derived waveforms. These are
%   algorithm-specific, but typically include a band-pass-filtered version
%   of the original waveform and an analytic signal derived from this.
% "auxdata" is a structure containing other algorithm-specific metadata about
%   the analysis.
```

3.21 wlProc_doSegmentAndParamExtract_MT.m

```
% function [ events auxwaves auxdata ] = ...
%   wlProc_doSegmentAndParamExtract_MT( ...
%     wavedata, samprate, bpfband, segconfig, paramconfig)
%
% This function performs event segmentation and parameter extraction on the
% specified waveform. Specific algorithms and their tuning parameters are
% selected/specified via configuration structures.
% This is a wrapper that forces multithreaded operation.
%
% "wavedata" is the data trace to examine.
% "samprate" is the number of samples per second in the signal data.
% "bpfband" [min max] is the frequency band of interest. Edges may fade.
% "segconfig" is a structure containing configuration information for
%   event segmentation, per SEGCONFIG.txt.
% "paramconfig" is a structure containing configuration information for
%   event parameter estimation, per PARAMCONFIG.txt.
% "wantmultithread" is "true" if _MT variants of functions are to be called.
%   This parameter is optional; the default is "false" (single-threaded).
%
% "events" is an array of structures describing detected events. Format is
%   per EVENTFORMAT.txt.
% "auxwaves" is a structure containing derived waveforms. These are
%   algorithm-specific, but typically include a band-pass-filtered version
%   of the original waveform and an analytic signal derived from this.
% "auxdata" is a structure containing other algorithm-specific metadata about
%   the analysis.
```

3.22 wlProc_evalEventsVsTruth.m

```
% function [ fp fn tp matchlist fplist fnlist ] = ...
%   wlProc_evalEventsVsTruth(evtruth, evtest, bandlim, snrlim, ...
%     comparefunc)
%
% This function compares an event list to a "ground truth" event list.
% Only events within a specified frequency band are considered. Match rate
% statistics are computed and a list of matching event pairs is returned.
% Event record format is per EVENTFORMAT.txt.
%
% "evtruth" is a list of ground truth event records.
% "evtest" is a list of event records to test.
% "bandlim" [min max] is the frequency band for events of interest.
% "snrlim" [min max] is the burst SNR range for events of interest.
% "comparefunc" is a function handle for an event comparison function, as
%   described in COMPAREFUNC.txt. This has the form:
%     [ ismatch distance ] = comparefunc(evfirst, evsecond)
```

```
%
% "fp" is the false positive count (evtest events that aren't in evtruth).
% "fn" is the false negative count (evtruth events that aren't in evtest).
% "tp" is the true positive count (events in both evtest and evtruth).
% "matchlist" is a list of structures containing matching event pairs:
%   "truth":  The matching event from evtruth.
%   "test":   The matching event from evtest.
% "fplist" is a list of in-band "evtest" event structures that didn't match.
% "fnlist" is a list of in-band "evtruth" event structures that didn't match.
```

3.23 wlProc_evalEventsVsTruthBinned.m

```
% function [ fp fn tp matchlist fplist fnlist ] = ...
%   wlProc_evalEventsVsTruthBinned(evtruth, evtest, bandlim, snrlim, ...
%     comparefunc, binsecs, huntbins)
%
% This function compares an event list to a "ground truth" event list.
% Only events within a specified frequency band are considered. Match rate
% statistics are computed and a list of matching event pairs is returned.
% Event record format is per EVENTFORMAT.txt.
% The "Binned" version of the function sorts events into time bins and only
% compares events that are sufficiently close to each other. Each event is
% mapped to only one bin.
%
% "evtruth" is a list of ground truth event records.
% "evtest" is a list of event records to test.
% "bandlim" [min max] is the frequency band for events of interest.
% "snrlim" [min max] is the burst SNR range for events of interest.
% "comparefunc" is a function handle for an event comparison function, as
%   described in COMPAREFUNC.txt. This has the form:
%     [ ismatch distance ] = comparefunc(evfirst, evsecond)
% "binsecs" is the size of the time bins, in seconds.
% "huntbins" is the number of bins on either side of an event to check for
%   matches.
%
% "fp" is the false positive count (evtest events that aren't in evtruth).
% "fn" is the false negative count (evtruth events that aren't in evtest).
% "tp" is the true positive count (events in both evtest and evtruth).
% "matchlist" is a list of structures containing matching event pairs:
%   "truth":  The matching event from evtruth.
%   "test":   The matching event from evtest.
% "fplist" is a list of in-band "evtest" event structures that didn't match.
% "fnlist" is a list of in-band "evtruth" event structures that didn't match.
```

3.24 wlProc_fitAmpBasic.m

```
% function [ newevent, error ] = wlProc_fitAmpBasic(hilmag, oldevent, atype)
%
% This function estimates oscillatory burst amplitude by curve fitting a
% logarithmic or linear ramp in the region of interest. Roll-on and roll-off
% are used if supplied, or fixed at 20% of the duration if not supplied (this
% function does not curve-fit the roll-on and roll-off).
%
% "hilmag" contains analytic signal instantaneous magnitude samples.
% "oldevent" is a record structure per EVENTFORMAT.txt. Required fields are:
%   "sampstart":  Sample index in the original recording waveform
%                 corresponding to event time 0.
%   "duration":   Time between the start and end of the event. This is where
%                 the curve fit is performed.
%   "samprate":   Number of samples per second in the input waveform.
%   Optional fields "rollon" and "rolloff" are used if present.
%
% "atype" is 'linear' or 'logarithmic', specifying the curve to fit. If it's
% omitted or set to 'auto', the option with least error is used.
%
% "newevent" is a record structure per EVENTFORMAT.txt. It contains the
% fields of "oldevent", as well as the following:
%   "a1":       Burst amplitude at nominal start.
%   "a2":       Burst amplitude at nominal stop.
%   "atype":    Amplitude ramp type.
%   "rollon":   Cosine roll-on time (copied or set to 0.2*duration).
%   "rolloff":  Cosine roll-off time (copied or set to 0.2*duration).
%
% "error" is the RMS error between the fitted amplitude and the magnitude
%   of the analytic signal.
```

3.25 wlProc_fitAmpGrid.m

```
% function [ newevent, error ] = wlProc_fitAmpGrid(hilmag, oldevent, gsteps)
%
% This function estimates oscillatory burst amplitude by curve fitting a
% logarithmic or linear ramp in the region of interest. Roll-on and roll-off
% are chosen via a brute force grid search with the indicated number of steps.
%
% "hilmag" contains analytic signal instantaneous magnitude samples.
% "oldevent" is a record structure per EVENTFORMAT.txt. Required fields are:
%   "sampstart":  Sample index in the original recording waveform
%                 corresponding to event time 0.
%   "duration":   Time between the start and end of the event. This is where
%                 the curve fit is performed.
%   "samprate":   Number of samples per second in the input waveform.
```

```
%
% "gsteps" is the number of steps to use when sweeping each of the roll times.
%
% "newevent" is a record structure per EVENTFORMAT.txt. It contains the
% fields of "oldevent", as well as the following:
%   "a1":       Burst amplitude at nominal start.
%   "a2":       Burst amplitude at nominal stop.
%   "atype":    Amplitude ramp type.
%   "rollon":   Cosine roll-on time.
%   "rolloff":  Cosine roll-off time.
%
% "error" is the RMS error between the fitted amplitude and the magnitude
%   of the analytic signal.
```

3.26 wlProc_fitFreqPhase.m

```
% function [ newevent, error ] = wlProc_fitFreqPhase( ...
%   hilfreq, hilphase, oldevent)
%
% This function estimates oscillatory burst frequency and phase by curve
% fitting. Linear and logarithmic fits are performed, and the version with
% the least RMS phase error is returned.
%
% "hilfreq" contains input signal instantaneous frequency samples.
% "hilphase" contains input signal instantaneous phase samples.
% "oldevent" is a record structure per EVENTFORMAT.txt. Required fields are:
%   "sampstart":  Sample index in the original recording waveform
%                 corresponding to event time 0.
%   "duration":   Time between the start and end of the event. This is where
%                 the curve fit is performed.
%   "samprate":   Number of samples per second in the input waveform.
%
% "newevent" is a record structure per EVENTFORMAT.txt. It contains the
% fields of "oldevent", as well as the following:
%   "f1":     Burst frequency at nominal start.
%   "f2":     Burst frequency at nominal stop.
%   "p1":     Burst phase at nominal start.
%   "p2":     Burst phase at nominal stop.
%   "ftype":  Frequency ramp type (set to "linear").
%
% "error" is the RMS phase error (with phase error being the difference
%   between original and reconstructed phases wrapped to -pi..pi).
```

3.27 wlProc_fitFreqPhaseLinear.m

```
% function [ newevent, error ] = wlProc_fitFreqPhaseLinear( ...
%   hilfreq, hilphase, oldevent)
%
% This function estimates oscillatory burst frequency and phase by curve
% fitting a linear ramp in the region of interest.
%
% "hilfreq" contains input signal instantaneous frequency samples.
% "hilphase" contains input signal instantaneous phase samples.
% "oldevent" is a record structure per EVENTFORMAT.txt. Required fields are:
%   "sampstart":  Sample index in the original recording waveform
%                 corresponding to event time 0.
%   "duration":   Time between the start and end of the event. This is where
%                 the curve fit is performed.
%   "samprate":   Number of samples per second in the input waveform.
%
% "newevent" is a record structure per EVENTFORMAT.txt. It contains the
% fields of "oldevent", as well as the following:
%   "f1":     Burst frequency at nominal start.
%   "f2":     Burst frequency at nominal stop.
%   "p1":     Burst phase at nominal start.
%   "p2":     Burst phase at nominal stop.
%   "ftype":  Frequency ramp type (set to "linear").
%
% "error" is the RMS phase error (with phase error being the difference
%   between original and reconstructed phases wrapped to -pi..pi).
```

3.28 wlProc_fitFreqPhaseLogarithmic.m

```
% function [ newevent, error ] = wlProc_fitFreqPhaseLogarithmic( ...
%   hilfreq, hilphase, oldevent)
%
% This function estimates oscillatory burst frequency and phase by curve
% fitting a logarithmic ramp in the region of interest.
%
% "hilfreq" contains input signal instantaneous frequency samples.
% "hilphase" contains input signal instantaneous phase samples.
% "oldevent" is a record structure per EVENTFORMAT.txt. Required fields are:
%   "sampstart":  Sample index in the original recording waveform
%                 corresponding to event time 0.
%   "duration":   Time between the start and end of the event. This is where
%                 the curve fit is performed.
%   "samprate":   Number of samples per second in the input waveform.
%
% "newevent" is a record structure per EVENTFORMAT.txt. It contains the
% fields of "oldevent", as well as the following:
%   "f1":     Burst frequency at nominal start.
%   "f2":     Burst frequency at nominal stop.
%   "p1":     Burst phase at nominal start.
%   "p2":     Burst phase at nominal stop.
%   "ftype":  Frequency ramp type (set to "logarithmic").
%
% "error" is the RMS phase error (with phase error being the difference
%   between original and reconstructed phases wrapped to -pi..pi).
```

3.29 wlProc_getEvParamsUsingHilbert.m

```
% function newevents = wlProc_getEvParamsUsingHilbert(data, samprate, ...
%   hilmag, hilfreq, hilphase, oldevents, gridsteps)
%
% This function estimates oscillatory burst parameters given a list of
% putative event locations and a signal waveform. An updated list of events
% is generated.
% NOTE - The signal waveform should already be band-pass filtered.
% The "UsingHilbert" implementation takes the analytic signal components as
% input and curve-fits to them directly.
%
% "data" is the data trace to examine. This is typically band-pass filtered.
% "samprate" is the number of samples per second in the signal data.
% "hilmag" is the instantaneous magnitude of the analytic signal.
% "hilfreq" is the instantaneous frequency of the analytic signal.
% "hilphase" is the instantaneous phase of the analytic signal.
% "oldevents" is the input event list. Only the "sampstart", "duration",
%   and "samprate" fields must be present.
% "gridsteps" is an optional argument specifying the number of steps to use
%   when sweeping roll-on and roll-off times during curve fitting.
%
% "newevents" is an array of event record structures following the
%   conventions given in EVENTFORMAT.txt. Curve fit parameters are generated,
%   but "times", "wave", "mag", "freq", and "phase" are left unset. The
%   "auxwaves" and "auxdata" fields are initialized as empty if not present.
```

3.30 wlProc_getEvSegmentsByComparing.m

```
% function events = wlProc_getEvSegmentsByComparing( ...
%   databigger, datasmaller, samprate, threshfactor, maxglitch, maxdrop);
%
% This function identifies the location of events in a data stream, returning
% a list of skeletal putative event records. Events are found by comparing
% two input signals, flagging events when the ratio of the larger to the
% smaller exceeds some threshold.
%
% Gaps between events that are shorter than "maxdrop" are removed, after which
% isolated events that are shorter than "maxglitch" are also removed.
%
% This is intended to be used as a helper function, called by a variety of
% detection routines with several different derived signals as input.
%
% "databigger" is the larger-valued data trace to examine.
% "datasmaller" is the smaller-valued data trace to examine.
% "samprate" is the number of samples per second in the signal data.
% "threshfactor" is the amount by which the larger signal must exceed the
%   smaller signal for an event to occur.
% "maxglitch" is the longest event duration to reject as spurious.
% "maxdrop" is the longest gap duration within an event to reject as spurious.
%
% "events" is an array of event record structures following the conventions
%   given in EVENTFORMAT.txt. Only the following fields are provided:
%
%   "sampstart":  Sample index in "data" corresponding to event nominal start.
%   "duration":   Time between burst nominal start and burst nominal stop.
%   "samprate":   Samples per second in the signal data.
```

3.31 wlProc_getEvSegmentsByComparingDual.m

```
% function events = wlProc_getEvSegmentsByComparingDual( ...
%   databigger, datasmaller, samprate, ...
%   threshfactpeak, threshfactend, maxglitch, maxdrop);
%
% This function identifies the location of events in a data stream, returning
% a list of skeletal putative event records. Events are found by comparing
% two input signals, flagging event regions where the ratio of the larger to
% the smaller exceeds some threshold. Event endpoints are the nearest points
% where the ratio of the larger to the smaller exceeds a different (lower)
% threshold.
%
% Gaps between events that are shorter than "maxdrop" are removed, after which
% isolated events that are shorter than "maxglitch" are also removed.
%
% This is intended to be used as a helper function, called by a variety of
% detection routines with several different derived signals as input.
%
% "databigger" is the larger-valued data trace to examine.
% "datasmaller" is the smaller-valued data trace to examine.
% "samprate" is the number of samples per second in the signal data.
% "threshfactpeak" is the amount by which the larger signal must exceed the
%   smaller signal for an event to occur.
% "threshfactend" is the amount by which the larger signal must exceed the
%   smaller signal at event endpoints
% "maxglitch" is the longest event duration to reject as spurious.
% "maxdrop" is the longest gap duration within an event to reject as spurious.
%
% "events" is an array of event record structures following the conventions
%   given in EVENTFORMAT.txt. Only the following fields are provided:
```

```
%
%   "sampstart":  Sample index in "data" corresponding to event nominal start.
%   "duration":   Time between burst nominal start and burst nominal stop.
%   "samprate":   Samples per second in the signal data.
```

3.32 wlProc_getEvSegmentsUsingFreq.m

```
% function events = wlProc_getEvSegmentsUsingFreq(data, samprate, ...
%   bpfband, noiseband, noisesnr, maxglitch, maxdrop, taudc, tauac, threshdb)
%
% This function identifies the locations of oscillatory bursts in a data
% stream, returning a list of skeletal putative burst event records.
% The "UsingFreq" implementation band-pass-filters the input signal, computes
% the analytic signal, and looks for regions with stable instantaneous
% frequency. High-frequency white noise is added to reduce false positives.
%
% "data" is the data trace to examine.
% "samprate" is the number of samples per second in the signal data.
% "bpfband" [min max] is the frequency band of interest. Edges may fade.
% "noiseband" [min max] is the band to add white noise in.
% "noisesnr" is the ratio of signal power to white noise power, in dB.
% "maxglitch" is the longest duration event to reject as spurious.
% "maxdrop" is the longest duration gap in an event to reject as spurious.
% "taudc" is the time constant for smoothing "average" frequency variance.
%   Set this to infinity to use the DC average.
% "tauac" is the time constant for smoothing short-term frequency variance.
% "threshdb" is the factor by which short-term frequency variance must be
%   suppressed with respect to average frequency variance for an event
%   candidate to be generated.
%
% "events" is an array of event record structures following the conventions
%   of wlSynth_traceAddBursts(). Only the following fields are provided:
%
%   "sampstart":  Sample index in "data" corresponding to burst nominal start.
%   "duration":   Time between burst nominal start and burst nominal stop.
%
% "waves" is a structure containing waveforms derived from "data":
%   "bpfwave" is the band-pass-filtered waveform.
%   "bpfmag" is the analytic magnitude of the bpf waveform.
%   "bpffreq" is the analytic frequency of the bpf waveform.
%   "bpfphase" is the analytic phase of the bpf waveform.
%   "noisywave" is the noisy version of the bpf waveform.
%   "noisymag" is the analytic magnitude of the noisy waveform.
%   "noisyfreq" is the analytic frequency of the noisy waveform.
%   "noisyphase" is the analytic phase of the noisy waveform.
%   "fvarfast" is the rapidly-changing variance of the instantaneous frequency.
%   "fvarslow" is the slowly-changing variance of the instantaneous frequency.
```

3.33 wlProc_getEvSegmentsUsingFreqDual.m

```
% function [ events, waves ] = wlProc_getEvSegmentsUsingFreqDual( ...
%   data, samprate, bpfband, noiseband, noisesnr, ...
%   maxglitch, maxdrop, taudc, tauac, peakdb, enddb)
%
% This function identifies the locations of oscillatory bursts in a data
% stream, returning a list of skeletal putative burst event records.
% The "UsingFreqDual" implementation band-pass-filters the input signal,
% computes the analytic signal, and looks for regions with stable
% instantaneous frequency. High-frequency white noise is added to reduce
% false positives.
% Frequency variance must be reduced by "peakdb" to be considered "stable",
% but endpoints only have to be reduced by "enddb".
%
% "data" is the data trace to examine.
% "samprate" is the number of samples per second in the signal data.
% "bpfband" [min max] is the frequency band of interest. Edges may fade.
% "noiseband" [min max] is the band to add white noise in.
% "noisesnr" is the ratio of signal power to white noise power, in dB.
% "maxglitch" is the longest duration event to reject as spurious.
% "maxdrop" is the longest duration gap in an event to reject as spurious.
% "taudc" is the time constant for smoothing "average" frequency variance.
%   Set this to infinity to use the DC average.
% "tauac" is the time constant for smoothing short-term frequency variance.
% "peakdb" is the factor by which short-term frequency variance must be
%   suppressed with respect to average frequency variance for an event
%   candidate to be generated.
% "enddb" is the factor by which short-term frequency variance must be
%   suppressed with respect to average frequency variance at event endpoints.
%
% "events" is an array of event record structures following the conventions
%   of wlSynth_traceAddBursts(). Only the following fields are provided:
%
%   "sampstart":  Sample index in "data" corresponding to burst nominal start.
%   "duration":   Time between burst nominal start and burst nominal stop.
%
% "waves" is a structure containing waveforms derived from "data":
%   "bpfwave" is the band-pass-filtered waveform.
%   "bpfmag" is the analytic magnitude of the bpf waveform.
%   "bpffreq" is the analytic frequency of the bpf waveform.
%   "bpfphase" is the analytic phase of the bpf waveform.
%   "noisywave" is the noisy version of the bpf waveform.
%   "noisymag" is the analytic magnitude of the noisy waveform.
%   "noisyfreq" is the analytic frequency of the noisy waveform.
%   "noisyphase" is the analytic phase of the noisy waveform.
%   "fvarfast" is the rapidly-changing variance of the instantaneous frequency.
%   "fvarslow" is the slowly-changing variance of the instantaneous frequency.
```

3.34 wlProc_getEvSegmentsUsingMag.m

```
% function [ events, waves ] = wlProc_getEvSegmentsUsingMag( ...
%   data, samprate, bpfband, maxglitch, maxdrop, taudc, threshdb)
%
% This function identifies the locations of oscillatory bursts in a data
% stream, returning a list of skeletal putative burst event records.
% The "UsingMag" implementation band-pass-filters the input signal, computes
% the analytic signal, and looks for excursions in analytic magnitude.
%
% "data" is the data trace to examine.
% "samprate" is the number of samples per second in the signal data.
% "bpfband" [min max] is the frequency band of interest. Edges may fade.
% "maxglitch" is the longest duration event to reject as spurious.
% "maxdrop" is the longest duration gap in an event to reject as spurious.
% "taudc" is the time constant for smoothing the "average" magnitude. Set this
%   to infinity to use the DC average.
% "threshdb" is the factor by which instantaneous power must exceed average
%   power for an event candidate to be generated.
%
% "events" is an array of event record structures following the conventions
%   of wlSynth_traceAddBursts(). Only the following fields are provided:
%
%   "sampstart":  Sample index in "data" corresponding to burst nominal start.
%   "duration":   Time between burst nominal start and burst nominal stop.
%
% "waves" is a structure containing waveforms derived from "data":
%   "bpfwave" is the band-pass-filtered waveform.
%   "bpfmag" is the analytic magnitude of the bpf waveform.
%   "bpffreq" is the analytic frequency of the bpf waveform.
%   "bpfphase" is the analytic phase of the bpf waveform.
%   "magpowerfast" is the rapidly-changing instantaneous power.
%   "magpowerslow" is the slowly-changing instantaneous power.
```

3.35 wlProc_getEvSegmentsUsingMagDual.m

```
% function [ events, waves ] = wlProc_getEvSegmentsUsingMagDual( ...
%   data, samprate, bpfband, maxglitch, maxdrop, taudc, peakdb, enddb)
%
% This function identifies the locations of oscillatory bursts in a data
% stream, returning a list of skeletal putative burst event records.
% The "UsingMagDual" implementation band-pass-filters the input signal,
% computes the analytic signal, and looks for excursions in analytic
% magnitude. Excursion maxima must exceed "peakdb", but endpoints only have
% to exceed "enddb".
%
% "data" is the data trace to examine.
% "samprate" is the number of samples per second in the signal data.
% "bpfband" [min max] is the frequency band of interest. Edges may fade.
% "maxglitch" is the longest duration event to reject as spurious.
% "maxdrop" is the longest duration gap in an event to reject as spurious.
% "taudc" is the time constant for smoothing the "average" magnitude. Set
%   this to infinity to use the DC average.
% "peakdb" is the factor by which instantaneous power must exceed average
%   power for an event candidate to be generated.
% "enddb" is the factor by which instantaneous power must exceed average
%   power at the event endpoints. This should be less than "peakdb".
%
% "events" is an array of event record structures following the conventions
%   of wlSynth_traceAddBursts(). Only the following fields are provided:
%
%   "sampstart":  Sample index in "data" corresponding to burst nominal start.
%   "duration":   Time between burst nominal start and burst nominal stop.
%
% "waves" is a structure containing waveforms derived from "data":
%   "bpfwave" is the band-pass-filtered waveform.
%   "bpfmag" is the analytic magnitude of the bpf waveform.
%   "bpffreq" is the analytic frequency of the bpf waveform.
%   "bpfphase" is the analytic phase of the bpf waveform.
%   "magpowerfast" is the rapidly-changing instantaneous power.
%   "magpowerslow" is the slowly-changing instantaneous power.
```

Chapter 4

"wlFT" Functions

4.1 wlFT_calcEventErrors.m

```
% function newmatrix = wlFT_calcEventErrors(oldmatrix, errfunc, errlabel)
%
% This function calculates reconstruction error for each detected event in
% the specified event matrix. Error is computed between the event and one or
% more context waves using the specified event function, and the resulting
% error value is stored in the specified field in each event record's
% "auxdata" structure.
%
% "oldmatrix" is the event matrix to process.
% "errfunc" is a function handle for a wave error function, as described
%   in WAVEERRFUNC.txt. This has the form:
%     error = errfunc(event, wavestruct)
% "errlabel" is the name of the field within (event).auxdata to store event
%   reconstruction error values in.
%
% "newmatrix" is a copy of "oldmatrix" with error values added.
```

4.2 wlFT_compareMatrixEventsVsTruth.m

```
% function [ fp fn tp matchtruth matchtest missingtruth missingtest ] = ...
%   wlFT_compareMatrixEventsVsTruth(truthmatrix, testmatrix, comparefunc, ...
%   bandlim, snrlim)
%
% This function compares a detected event matrix to a "ground truth" event
% matrix. Only events within a specified frequency band and SNR range are
% considered. Match rate statistics are computed and matrices containing
% matching and non-matching event pairs are returned. Event matrix format
% is per EVMATRIX.txt.
%
```

```
% "truthmatrix" is an event matrix containing ground-truth events.
% "testmatrix" is an event matrix containing putative detected events.
% "comparefunc" is a function handle for an event comparison function, as
%   described in COMPAREFUNC.txt. This has the form:
%     [ ismatch distance ] = comparefunc(evfirst, evsecond)
% "bandlim" [min max] is the frequency band for events of interest. This is
%   optional; omitting it accepts all frequencies.
% "snrlim" [min max] is the burst SNR range (in dB) for events of interest.
%   This is optional; omitting it accepts all SNRs.
%
% "fp(bidx, tidx, cidx)" is the false positive count (bad putative events).
% "fn(bidx, tidx, cidx)" is the false negative count (unmatched truth events).
% "tp(bidx, tidx, cidx)" is the true positive count (matched events).
% "matchtruth" is an event matrix containing copies of ground truth events
%   that matched.
% "matchtest" is an event matrix containing copies of putative events that
%   matched.
% "missingtruth" is an event matrix containing copies of ground truth events
%   that did not match (false negatives).
% "missingtest" is an event matrix containing copies of putative events that
%   did not match (false positives).
```

4.3 wlFT_doFindEventsInTrials.m

```
% function eventmatrixdata = wlFT_doFindEventsInTrials( trialftdata, ...
%   bandlist, segconfig, paramconfig, bandoverrides, tattleprogress )
%
% This function calls wlProc_doSegmentAndParamExtract() for each trial in
% "trialftdata", and for each band in "bandlist", storing the resulting
% events and auxiliary data in an "event matrix" structure per EVMATRIX.txt.
%
% This is the single-threaded implementation. Use the "_MT" version to
% parallelize across bands/trials/channels.
%
% "trialftdata" is a Field Trip data structure containing trials to process.
% "bandlist" is an array of structures with the following fields:
%   "band" [ min max ] defines a frequency band of interest.
%   "name" is a character array containing a human-readable band name.
%   "label" is a character array containing a filename-safe band ID string.
% "segconfig" specifies the segmentation algorithm to use, per SEGCONFIG.txt.
% "paramconfig" specifies the parameter extraction algorithm to use, per
%   PARAMCONFIG.txt.
% "bandoverrides" is an array of structures containing band-specific
%   override values for "segconfig" and "paramconfig". There are as many
%   array elements as there are bands. Each element is a structure with
%   "seg" and "param" fields, each of which contains a structure that
%   holds field values that should override those specified in "segconfig"
%   and "paramconfig".
% "tattleprogress" is an optional argument. If present and set to "true",
%   progress messages are displayed.
%
% "eventmatrixdata" is a structure describing detected events and auxiliary
%   data, per EVMATRIX.txt.
```

4.4 wlFT_doFindEventsInTrials_MT.m

```
% function eventmatrixdata = wlFT_doFindEventsInTrials_MT( trialftdata, ...
%   bandlist, segconfig, paramconfig, bandoverrides, tattleprogress )
%
% This function calls wlProc_doSegmentAndParamExtract() for each trial in
% "trialftdata", and for each band in "bandlist", storing the resulting
% events and auxiliary data in an "event matrix" structure per EVMATRIX.txt.
%
% The "_MT" implementation is multithreaded, using "parfor" to process trials
% from "trialftdata" in parallel. Per Matlab's documentation, this falls back
% to single-threaded operation if called from within another parfor loop.
%
% Bands, trials, and channels are all parallelized.
%
% "trialftdata" is a Field Trip data structure containing trials to process.
% "bandlist" is an array of structures with the following fields:
%   "band" [ min max ] defines a frequency band of interest.
%   "name" is a character array containing a human-readable band name.
%   "label" is a character array containing a filename-safe band ID string.
% "segconfig" specifies the segmentation algorithm to use, per SEGCONFIG.txt.
% "paramconfig" specifies the parameter extraction algorithm to use, per
%   PARAMCONFIG.txt.
% "bandoverrides" is an array of structures containing band-specific
%   override values for "segconfig" and "paramconfig". There are as many
%   array elements as there are bands. Each element is a structure with
%   "seg" and "param" fields, each of which contains a structure that
%   holds field values that should override those specified in "segconfig"
%   and "paramconfig".
% "tattleprogress" is an optional argument. If present and set to "true",
%   progress messages are displayed.
%
% "eventmatrixdata" is a structure describing detected events and auxiliary
%   data, per EVMATRIX.txt.
```

4.5 wlFT_getChanSubset.m

```
% function [ newftdata chandefs ] = wlFT_getChanSubset(oldftdata, chanlist)
%
% This function extracts a subset of channels from within a Field Trip data
% structure. Channels are selected by name (matching names in the "label"
% cell array).
%
% "oldftdata" points to a Field Trip data structure to process.
% "chanlist" is a cell array containing desired channel name strings.
%
% "newftdata" is a modified Field Trip data structure containing only the
%   desired channels.
% "chandefs" is an array containing the old channel index corresponding to
%   each new channel.
%
% NOTE - While the indended situation is for "chandefs" to have the same
% number of entries as "chanlist", that will not be the case if some requested
% channels couldn't be found in the trial data.
```

4.6 wlFT_getEventTrialsFromMatrix.m

```
% function eventftdata = wlFT_getEventTrialsFromMatrix( eventmatrixdata );
%
% This function converts an "event matrix" structure as described by
% EVMATRIX.txt into a Field Trip data structure.
%
% "eventftdata" uses Field Trip's format, storing each detected event as a
% trial. Event metadata (including original trial/channel) goes in
% "trialinfo". An additional "wlnotes" metadata structure provides field
% information for "trialinfo", and holds metadata that isn't stored per-trial.
% This is described in "WLNOTES.txt".
%
% "eventmatrixdata" is a structure describing detected events and auxiliary
%   data, per EVMATRIX.txt.
%
% "eventftdata" is a Field Trip data structure describing detected events.
```

4.7 wlFT_getEventsAsArtifacts.m

```
% function artifactstruct = wlFT_getEventsAsArtifacts( ...
%   eventmatrixdata, ftlabels, bandwanted )
%
% This function converts an "event matrix" structure as described by
% EVMATRIX.txt into an artifact definition structure per ft_artifact_XXX.
% Nx2 matrix of the type returned by ft_artifact_XXX.
%
% This is intended to be used with ft_databrowser to visualize bursts in
% trials as if they were artifacts.
%
% "eventmatrixdata" is a structure describing detected events and auxiliary
%   data, per EVMATRIX.txt.
% "ftlabels" is a copy of the "label" field of the Field Trip dataset.
%   This maps event channel indices to FT labels.
% "bandwanted" specifies the band to use (first index into events{}). If this
%   is omitted or NaN, all bands are used.
%
% "artifactstruct" is a structure with the following fields, intended to be
%   stored in cfg.artfctdef.wlburst:
%   "artifact" is a Nx2 matrix containing the locations of detected
%     events in the original data, in a format similar to "trl" from
%     ft_definetrial (per ft_artifact_XXX).
%   "channel" is a Nx1 cell array with channel labels for each detected event.
```

4.8 wlFT_getSamplingRate.m

```
% function ftrate = wlFT_getSamplingRate(trialdata)
%
% This function returns the sampling rate used within a Field Trip trial data
% structure. Priority is given to "rawsample", then to "hdr.Fs", and then to
% "1.0 / (time{1}(2) - time{1}(1))". NaN is returned on error.
```

4.9 wlFT_getTimeBinList.m

```
% function binlist = wlFT_getTimeBinList(ftdata, time_bin_ms, offset)
%
% This generates a list of time bin spans that covers a set of Field Trip
% trials.
%
% "ftdata" is a ft_datatype_raw data structure produced by Field Trip.
% "time_bin_ms" is the time bin width in milliseconds.
% "offset" is 'center', 'edge', or a time in milliseconds. If it's 'center',
%   one time bin's midpoint is at t=0. If it's 'edge', one time bin starts
%   at t=0. If it's a time in milliseconds, one time bin's midpoint is at
%   that time.
%
% "binlist" is a cell array. Each cell contains a [ min max ] time pair
%   specifying the time bin extents in seconds (not milliseconds).
```

4.10 wlFT_getTrialStartSamples.m

```
% function trialstarts = wlFT_getTrialStartSamples( ftdata )
%
% This extracts the continuous-data sample indices of the first samples of
% each trial in a Field Trip dataset.
%
% This is the first column in ftdata.sampleinfo or ftdata.trialdef or
% cfg.trl.
%
% "ftdata" is a ft_datatype_raw structure to process.
%
% "trialstarts" is a 1xNtrials vector with trial start sample indices.
```

4.11 wlFT_pruneEventsByTime.m

```
% function newevmatrix = wlFT_pruneEventsByTime(oldevmatrix, ...
%   padbegin, padend)
%
% This function processes an event matrix structure and removes events with
% nominal starting or stopping times that are too close to the ends of the
% trials from which they were extracted.
%
% "oldevmatrix" is a structure describing detected events and auxiliary
%   data, per "EVMATRIX.txt".
% "padbegin" is the duration of the exclusion region at the start of the trial.
% "padend" is the duration of the exclusion region at the end of the trial.
%
% "newevmatrix" is an event matrix struture containing edited event lists.
```

4.12 wlFT_trimTrials.m

```
% function [ newftdata cropdefs ] = wlFT_trimTrials(oldftdata, limitsfunc)
%
% This function truncates trial records to limits computed by "limitsfunc",
% adjusting "sampleinfo" appropriately.
%
% "oldftdata" points to a Field Trip data structure to process.
% "limitsfunc" is a function handle for computing the region of interest for
%   each trial (with a sample index of "1" being the first sample in the
%   trial). This has the form:  [ firstsample lastsample ] = ...
%     limitsfunc(thistrial, sampleinfo_row, trialinfo_row)
%
% The intention is that "limitsfunc" may do anything from echoing the original
% trial size to reporting bounds recorded in "trialinfo" to performing
% artifact detection in the trial waveform directly.
%
% "newftdata" is a modified Field Trip data structure with cropped trials.
% "cropdefs" is an Nx2 array containing each original trial's cropping limits.
```

4.13 wlFT_unMapChannels.m

```
% function newftevdata = wlFT_unMapChannels(subsetftevdata, chandefs)
%
% This function alters "ft_channelnum" fields in "trialinfo" metadata for
% events extracted from trial data with channel subsets (processed with
% wlFT_getChanSubset()) to use channel indices from the original trial data.
%
% "subsetftevdata" points to a Field Trip data structure to process. This
%   structure should contain event data and "wlnotes" metadata.
% "chandefs" is an array containing the original channel index corresponding
%   to each remapped channel index.
%
% "newftevdata" is a modified Field Trip data structure containing event
%   data and metadata with modified "ft_channelnum" indices.
```

4.14 wlFT_unTrimMetadata.m

```
% function newftevdata = wlFT_unTrimMetadata(trimmedftevdata, cropdefs)
%
% This function alters "sampleinfo" and "trialinfo" metadata for events
% extracted from trimmed trial data to use sample indices from the original
% un-trimmed trial data.
%
% "trimmedftevdata" points to a Field Trip data structure to process. This
%   structure should contain event data and "wlnotes" metadata.
% "cropdefs" is an Nx2 array containing each original trial's cropping limits.
%
% "newftevdata" is a modified Field Trip data structure containing event
%   data and metadata with modified sample indices.
```

Chapter 5

“wlStats” Functions

5.1 wlStats_getBootstrappedStats.m

```
% function [ stat_mean stat_dev stat_sem ] = ...
%   wlStats_getBootstrappedStats( datavals, bootcount )
%
% This computes the mean, standard deviation, and standard error of the
% mean for a set of samples. SEM is estimated via bootstrapping or by
% assuming a normal distribution.
%
% "datavals" is a vector containing data samples.
% "bootcount" is the number of distributions to generate when estimating
%   the SEM. As a special case, this may be 'normal' to estimate it by
%   dividing the deviation by the square root of the number of samples.
%
% "stat_mean" is the average (mean) sample value.
% "stat_dev" is the standard deviation of the sample values.
% "stat_sem" is the standard error of the mean (the estimated standard
%   deviation of stat_mean).
```

5.2 wlStats_getMatrixBurstRates.m

```
% function [ rate_avg rate_dev rate_sem ] = wlStats_getMatrixBurstRates( ...
%   detectmatrix, time_bins_sec, bootcount )
%
% This computes the average burst rate across trials within a set of
% time bins. Statistics are estimated via bootstrapping.
%
% "Burst rate" is the average number of bursts overlapping a given time
% window. In  most cases, this will be very much less than 1, and can be
% interpreted as "burst probability" within that window.
%
```

```
% "detectmatrix" is an event matrix structure per EVMATRIX.txt.
% "time_bins_sec" is a cell array. Each cell contains a [ min max ] time pair
%   specifying time bin extents in seconds.
% "bootcount" is the number of distributions to generate when estimating the
%   SEM for the average burst rates. As a special case, this may be 'normal'
%   to estimate it by dividing the deviation by the square root of the number
%   of trials.
%
% "rate_avg" is a matrix indexed by (bidx, cidx, widx) that holds the burst
%   rate (expected number of bursts) observed for each band, channel, and
%   time window, averaged across trials.
% "rate_dev" is a matrix per "rate_avg" holding the standard deviation of the
%   burst rate across trials.
% "rate_sem" is a matrix per "rate_avg" holding the standard error of the
%   mean of the burst rate (the estimated standard deviation of the value
%   of "rate_avg").
```

5.3 wlStats_makePhaseShuffledWave.m

```
% function newwave = wlStats_makePhaseShuffledWave( oldwave )
%
% This scrambles the phases of all components of a wave, smearing out
% position-related information while keeping the power spectrum the same.
%
% This is intended to be used for surrogate testing of burst detection. The
% rate at which spurious bursts appear by chance will be the same as with
% the original data.
%
% NOTE - Applying a rolloff window to the waveform data is recommended, as
% otherwise endpoint discontinuity will show up as sawtooth harmonics.
%
% "oldwave" is a vector containing waveform data.
%
% "newwave" is a vector containing phase-shuffled waveform data.
```

5.4 wlStats_makePhaseSurrogateFT.m

```
% function [ newdata trialmap ] = ...
%   wlStats_makePhaseSurrogateFT( olddata, replicount )
%
% This generates a phase-scrambled surrogate of a Field Trip dataset.
%
% "olddata" is a Field Trip dataset to scramble.
% "replicount" is the number of new trials to make from each old trial.
%
% "newdata" is a scrambled Field Trip dataset.
% "trialmap" is a vector mapping new trial indices to old trial indices,
%   such that oldtrial = trialmap(newtrial).
```

Chapter 6

"wlSynth" Functions

6.1 wlSynth_genFieldTrip.m

```
% function [ ftdata, evmatrix ] = wlSynth_genFieldTrip( ...
%   samprate, chancount, trialcount, trialdur, burstdefs, ...
%   channelratevar, channelnoisevar )
%
% This function generates a minimal Field Trip dataset with synthetic
% oscillatory burst events. An "event matrix" containing ground truth
% information for oscillations is also returned.
%
% "samprate" is the number of samples per second.
% "chancount" is the number of channels to generate.
% "trialcount" is the number of trials to generate.
% "trialdur" [ min max ] specifies minimum and maximum trial durations, in
%   seconds.
% "burstdefs" is an array of burst type definition structures, with fields
%   as defined in BURSTTYPEDEF.txt.
% "channelratevar" [ min max ] specifies the minimum and maximum factor by
%   which burst rates in a channel are to be scaled. Each channel gets its
%   own scale factor (picked uniformly in the log domain).
% "channelnoisevar" [ min max ] specifies the minimum and maximum values to
%   add to signal to noise ratios in a channel, in dB. Each channel gets its
%   own SNR offset (picked uniformly).
%
% "ftdata" is a Field Trip data structure containing trial records with
%   synthetic oscillatory bursts. Trial timestamps abut each other (the
%   trials can be concatenated to get a valid signal waveform with no
%   discontinuities).
% "evmatrix" is a data structure containing ground-truth information about
%   oscillatory burst events in the trials, using the format described in
%   EVMATRIX.txt. A single frequency band is defined.
%   NOTE - Events may be recorded multiple times if they span multiple trials.
```

6.2 wlSynth_makeOneBurst.m

```
% function [errcode, times, wave, mag, freq, phase] = ...
%  wlSynth_makeOneBurst(duration, rollon, rolloff, samprate, ...
%    f1, f2, ftype, a1, a2, atype, pstart)
%
% This function generates an oscillatory burst aligned to t=0.
%
% "duration" is the time between the midpoints of the roll-on and roll-off
% (the FWHM of the roll-off window; this is not the FHWM of the burst itself).
% "rollon" is the roll-on time of the cosine roll-off window.
% "rolloff" is the roll-off time of the cosine roll-off window.
% "samprate" is the number of samples per second in the generated data.
% "f1" and "f2" are the frequencies at time 0 and time "duration".
% "a1" and "a2" are the amplitudes at time 0 and time "duration" BEFORE the
% application of the cosine roll-off window.
% "ftype" and "atype" are "linear" or "logarithmic", defining how frequency
% and amplitude vary. These use the same equations as "chirp".
% "pstart" is the phase at time zero, in radians.
%
% "errcode" is "ok" on success or a descriptive error message on failure.
% "times" is a [1xN] array of sample timestamps.
% "wave" is a [1xN] array containing the oscillatory burst waveform.
% "mag" is a [1xN] array containing the ground-truth envelope magnitude.
% "freq" is a [1xN] array containing the ground-truth instantaneous frequency.
% "phase" is a [1xN] array containing the ground-truth instantaneous phase in radians.
```

6.3 wlSynth_makeOneBurst_Gabor.m

```
% function [errcode, times, wave, mag, freq, phase] = ...
%   wlSynth_makeOneBurst_Gabor(samprate, f1, a1, pmid, periods)
%
% This function generates approximations of Gabor wavelets using
% wlSynth_makeOneBurst. These are cosine waves modulated by a von Hann
% window (raised cosine window).
% The center of the wavelet is aligned to t=0 (differing from the convention
% used by wlSynth_makeOneBurst).
%
% "samprate" is the number of samples per second in the generated data.
% "f1" is the nominal frequency of the wavelet.
% "a1" is the maximum amplitude of the modulation envelope.
% "pmid" is the cosine wave's phase at the midpoint of the wavelet.
%   A value of 0 gives a cosine wavelet; -pi/2 gives a sine wavelet.
% "periods" is the duration of the wavelet, in cycles. This is the full
%   width of the von Hann window, not the FHWM.
%
% "errcode" is "ok" on success or a descriptive error message on failure.
% "times" is a [1xN] array of sample timestamps.
% "wave" is a [1xN] array containing the oscillatory burst waveform.
% "mag" is a [1xN] array containing the ground-truth envelope magnitude.
% "freq" is a [1xN] array containing the ground-truth instantaneous frequency.
% "phase" is a [1xN] array containing the ground-truth instantaneous phase in radians.
```

6.4 wlSynth_makeOneBurst_Simple.m

```
% function [errcode, times, wave, mag, freq, phase] = ...
%   wlSynth_makeOneBurst_Simple(samprate, f1, f2, a1, a2, pstart, periods)
%
% This function generates an oscillatory burst aligned to t=0.
% This is a wrapper for wlSynth_makeOneBurst with simplified arguments.
%
% "samprate" is the number of samples per second in the generated data.
% "f1" and "f2" are the frequencies at the start and end of the burst.
% "a1" and "a2" are the amplitudes at the start and end of the burst.
% "pstart" is the phase at the start of the burst.
% "periods" is the time between midpoints of the envelope's rising/falling
%   edges, in cycles at the average frequency ( (f1 + f2) / 2 ).
%
% The burst uses a Tukey window (with cosine roll-off). The midpoint of the
% rising edge of the window is at t=0; the distance between midpoints of
% the rising and falling edges is the duration. Rise time and fall time are
% one (average) period.
%
% "errcode" is "ok" on success or a descriptive error message on failure.
% "times" is a [1xN] array of sample timestamps.
% "wave" is a [1xN] array containing the oscillatory burst waveform.
% "mag" is a [1xN] array containing the ground-truth envelope magnitude.
% "freq" is a [1xN] array containing the ground-truth instantaneous frequency.
% "phase" is a [1xN] array containing the ground-truth instantaneous phase in radians.
```

6.5 wlSynth_traceAddBursts.m

```
% function [newdata, events] = ...
%   wlSynth_traceAddBursts(olddata, samprate, deflist)
%
% This function adds a series of oscillatory bursts to a data stream.
%
% "olddata" is the signal to add bursts to.
% "samprate" is the number of samples per second in the signal data.
% "deflist" is an array of burst type definition structures, with the
%   following fields (per BURSTTYPEDEF.txt):
%
%   "rate":                 Average number of bursts per second.
%   "snrrange": [min max]   Signal-to-noise ratio of bursts, in dB.
%   "noiseband": [min max]  Frequency band in which noise power is measured.
%   "durrange": [min max]   Burst duration in cycles (at average frequency).
%   "fctrrange": [min max]  Burst nominal center frequency.
%   "framprange": [min max] Ratio of ending/starting frequency.
%   "aramprange": [min max] Ratio of ending/starting amplitude.
%
% Burst parameters are defined using the conventions for
% wlSynth_makeOneBurst().
% Burst events are treated as a Poisson point process. For each event,
% parameters are chosen randomly within the specified ranges.
%
% "newdata" is a copy of "olddata" with burst events added.
% "events" is an array of structures with fields defined per EVENTFORMAT.txt.
% The following fields are filled in:
%
%   "sampstart":  Sample index in "newdata" corresponding to burst time 0.
%   "duration":   Time between 50% roll-on and 50% roll-off for the burst.
%   "s1":         Sample index in "wave" of nominal start (50% of roll-on).
%   "s2":         Sample index in "wave" of nominal stop (50% of roll-off).
%   "samprate":   Samples per second in the stored waveforms.
%   "snr":        Signal-to-noise ratio for this burst, in dB.
%
%   "paramtype"   Parameter fit type. For this function, it's "chirpramp".
%
%   "f1":         Burst frequency at nominal start.
%   "f2":         Burst frequency at nominal stop.
%   "a1":         Envelope amplitude at nominal start.
%   "a2":         Envelope amplitude at nominal stop.
%   "p1":         Phase at nominal start.
%   "p2":         Phase at nominal stop.
%
%   "rollon":     Duration of the envelope's cosine roll-on time.
%   "rolloff":    Duration of the envelope's cosine roll-off time.
%   "ftype":      Frequency ramp type.
%   "atype":      Amplitude ramp type.
%
%   In "auxwaves", the following waveforms:
%   "truthtimes": [1xN] array of burst sample times, relative to 50% roll-on.
%   "truthwave":  [1xN] array of ground-truth burst samples.
%   "truthmag":   [1xN] array with ground-truth burst envelope magnitude.
%   "truthfreq":  [1xN] array with ground-truth instantaneous frequency.
%   "truthphase": [1xN] array with ground-truth instantaneous phase (radians).
```

6.6 wlSynth_traceAddNoise.m

```
% function newdata = ...
%   wlSynth_traceAddNoise(olddata, samprate, colortype, ...
%     noisef1, noisef2, noisepower, powerf1, powerf2)
%
% This function adds noise to a signal.
%
% "olddata" is the signal to add noise to.
% "samprate" is the number of samples per second in the signal data.
% "colortype" is a noise color using the "dsp.ColoredNoise" conventions.
%   Typical values are "pink" and "white".
% "noisef1" is the lowest frequency present in the noise.
% "noisef2" is the highest frequency present in the noise.
% "noisepower" is the average noise power (measured with "bandpower").
% "powerf1" is the lowest frequency of the band-limited power measurement.
% "powerf2" is the highest frequency of the band-limited power measurement.
%
% "newdata" is a copy of "olddata" with appropriately scaled noise added.
```

Chapter 7

"wlAux" Functions

7.1 wlAux_checkParallelToolbox.m

```
% function haveparallel = wlAux_checkParallelToolbox
%
% This function tests for the presence of the Parallel Computing Toolbox.
% That toolbox is needed for running the _MT versions of the processing
% functions.
%
% No inputs.
%
% "haveparallel" is true if the toolbox is present, false otherwise.
```

7.2 wlAux_getContextTrace.m

```
% function [ wavecontext, timescontext ] = ...
%   wlAux_getContextTrace(wave, samprate, ...
%     minlengths, mincycles, ...
%     evfreq, evlength, evsampstart)
%
% This function extracts a section of the input waveform containing "context"
% for an event. Timestamps are generated aligned with the event start time.
%
% "wave" is a [1xN] vector containing signal waveform data.
% "samprate" is the number of samples per second.
% "minlengths" is the minimum size of the surrounding context time as a
%   fraction of event duration.
% "mincycles" is the minimum size of the surrounding context time as a
%   fraction of the nominal cycle period of the event signal.
% "evfreq" is the nominal frequency of the event signal.
% "evlength" is the duration of the event in seconds.
% "evsampstart" is the sample index in "wave" corresponding to the nominal
%   starting time of the event.
%
% "wavecontext" is a [1xM] vector containing context waveform data.
% "timescontext" is a [1xM] vector containing context timestamp data.
```

7.3 wlAux_getMatrixEventCount.m

```
% function evcount = wlAux_getMatrixEventCount(evmatrix)
%
% This function returns the number of events in a matrix, across all bands,
% trials, and channels.
%
% "evmatrix" is an event matrix, per EVMATRIX.txt.
%
% "evcount" is the total number of events recorded in the event matrix.
```

7.4 wlAux_getReconFromParams.m

```
% function newevlist = wlAux_getReconFromParams(oldevlist)
%
% This function fills in "times", "wave", "mag", "freq", and "phase" in a
% seires of event records by calling wlSynth_makeOneBurst() using the
% extracted event parameters. s1 and s2 are adjusted accordingly.
% The new wave is also stored in "auxwaves" with the prefix "param".
%
% "oldevlist" is an array of event records per EVENTFORMAT.txt.
%
% "newevent" is an array of updated event records with "curve-fit waveform"
% fields filled in.
```

7.5 wlAux_getReconFromWave.m

```
% function newevlist = wlAux_getReconFromWave(oldevlist, label)
%
% This function fills in "times", "wave", "mag", "freq", and "phase" in a
% seires of event records by copying a set of signal traces stored in
% "auxwaves" that begin with the specified prefix. "s1" and "s2" are set to
% point to time 0 and time "duration", respectively.
%
% "oldevlist" is an array of event records per EVENTFORMAT.txt.
% "label" is the prefix to use when constructing "auxwaves" signal names.
% These names have "times", "wave", "mag", "freq", and "phase" appended to
% the label.
```

```
%
% "newevent" is an array of updated event records with "curve-fit waveform"
% fields filled in.
```

7.6 wlAux_getStringFromParams.m

```
% function paramstr = wlAux_getStringFromParams(thisev)
%
% This function builds a human-readable text string describing the fit
% parameters in the specified event record.
%
% "thisev" is the event structure to describe, per EVENTFORMAT.txt.
%
% "paramstr" is the text representation of the event parameters.
```

7.7 wlAux_makeSafeString.m

```
% function [ newlabel newtitle ] = wlAux_makeSafeString( oldstring )
%
% This function makes label- and title-safe versions of an input string.
% The label-safe version strips anything that's not alphanumeric.
% The title-safe version replaces stripped characters with spaces.
%
% This is more aggressive than filename- or fieldname-safe strings; in
% particular, underscores are interpreted as typesetting metacharacters
% in plot labels and titles.
%
% This accepts character vectors (single strings) or cell arrays (lists of
% strings), returning the same type.
%
% "oldstring" is a character vector to convert, or a cell array of character
%   vectors to convert.
%
% "newlabel" is a character vector with only alphanumeric characters, or
%   a cell array of such character vectors.
% "newtitle" is a character vector with non-alphanumeric characters replaced
%   with spaces, or a cell array of such character vectors.
```

7.8 wlAux_mergeEventLists.m

```
% function newlist = wlAux_mergeEventLists(oldlistarray, ...
%   comparefunc, binsecs, huntbins)
%
```

```
% This function merges multiple event lists, removing duplicates.
% Duplicates are those that match according to "comparefunc". Events are
% sorted in ascending order of "sampstart".
%
% FIXME - For now, an arbitrarily selected duplicate is kept, and the rest
% are removed. This should be upgraded to find a proper cluster average.
%
% "oldlistarray" is a cell array of event lists to merge.
% "comparefunc" is a function handle for an event comparison function, as
%   described in COMPAREFUNC.txt. This has the form:
%     [ ismatch distance ] = comparefunc(evfirst, evsecond)
% "binsecs" is the size of time bins, in seconds. Only events in nearby bins
%   are compared for duplication.
% "huntbins" is the number of bins on either side of an event to check for
%   duplicates.
%
% "newlist" is an event list containing all entries from "oldlistarray",
%   with duplicates removed, sorted in ascending order of "sampstart".
```

7.9 wlAux_mergeTrialsChannels.m

```
% function newmatrix = wlAux_mergeTrialsChannels(oldmatrix, comparefunc)
%
% This function merges an event matrix's events across trials and channels
% (so that each band has a single trial with a single channel). Duplicate
% events (as detected by "comparefunc") are merged.
% Event matrix format is described in EVMATRIX.txt.
%
% NOTE - Old "auxdata" values are discarded. An arbitrarily chosen "waves"
% entry within each band is kept as a placeholder, but should be considered
% meaningless.
%
% The only reason to merge trials this way is to consolidate the event
% list. Almost all context information is lost.
%
% NOTE - This can take O(n^2) time. Use it with caution.
%
% "oldmatrix" is an event matrix containing the events to process.
% "comparefunc" is a function handle for an event comparison function, as
%   described in COMPAREFUNC.txt. This has the form:
%     [ ismatch distance ] = comparefunc(evfirst, evsecond)
%
% "newmatrix" is an event matrix containing re-binned events.
```

7.10 wlAux_pruneMatrix.m

```
% function newmatrix = wlAux_pruneMatrix(oldmatrix, passfunc)
%
% This function removes a subset of the event records from an event matrix.
% Events for which "passfunc" returns "true" are kept; events for which
% "passfunc" returns "false" are removed.
% Event matrix format is described in EVMATRIX.txt.
%
% "oldmatrix" is an event matrix containing events to process.
% "passfunc" is a function handle for an event evaluation function, with
%   the form:  result = passfunc(event)
%
% "newmatrix" is an event matrix containing only events for which "passfunc"
%   returned "true".
```

7.11 wlAux_selectEventsByTime.m

```
% function newlist = wlAux_selectEventsByTime( oldlist, timerange, timestamps )
%
% This selects events from an event list that overlap a specified time
% range.
%
% "oldlist" is an array of event structures, per EVENTFORMAT.txt.
% "timerange" [ min max ] is the range of timestamps to accept.
% "timestamps" is a vector containing sample timestamps.
%
% "newlist" is a copy of "oldlist" containing only those events that passed
%   the time range check.
```

7.12 wlAux_splitEvMatrixByBand.m

```
% function newmatrix = wlAux_splitEvMatrixByBand(oldmatrix, bandlist)
%
% This function sorts an event matrix's events into a new set of frequency
% bands. Old per-band lists are merged and repartitioned. Event matrix format
% is described in EVMATRIX.txt.
%
% Old "auxdata" values are discarded. Old "waves" waveforms are discarded
% except for "ftwave" and "fttimes". Approximate band-pass-filtered analytic
% waveforms are then generated.
%
% "oldmatrix" is an event matrix containing the events to process.
% "bandlist" is an array containing band definition structures:
%   "band" [ min max ] defines a frequency band of interest.
%   "name" is a character array containing a human-readable band name.
%   "label" is a character array containing a filename-safe band ID string.
%
% "newmatrix" is an event matrix containing re-binned events.
```

7.13 wlAux_splitEventsByBand.m

```
% function bandevlist = wlAux_splitEventsByBand(oldlist, bandlist)
%
% This function splits an event list into smaller lists of events that match
% the supplied frequency bands.
%
% "oldlist" is an array of event records per EVENTFORMAT.txt.
% "bandlist" is a cell array with entries containing [ low high ] band defs.
%
% "bandevlist" is a cell array containing arrays of event indices for
%   in-band event records. Indices index events in "oldlist". Bands are
%   presented in the same order as the bands in "bandlist".
%
% Getting band events themselves is straightforward: oldlist(bandevlist{band}).
```

Chapter 8

"wlPlot" Functions

8.1 wlPlot_plotAllMatrixEvents.m

```
% function wlPlot_plotAllMatrixEvents(cfg, evmatrix, figtitle, figlabel, ...
%   plotsperband, trialstride, channelstride)
%
% This function plots all or some of the events in an "event matrix", such
% as that returned by wlFT_doFindEventsInTrial_MT(). Reconstructed events are
% plotted against the corresponding band-pass-filtered and wideband trial
% waveforms. The spectrogram of the wideband signal is also provided.
%
% "cfg" contains figure configuration information (per "FIGCONFIG.txt").
% Certain plot parameters may be overridden.
% "evmatrix" is a structure describing detected events and auxiliary data,
% (per "EVMATRIX.txt").
% "figtitle" is the title to apply to the figure series. Subfigures have a
% prefix prepended to this title.
% "figlabel" is used within figure filenames to identify this figure series.
% "plotsperband" suppresses plots; a maximum of "plotsperband" figures is
% generated per frequency band. This argument is optional.
% "trialstride" suppresses plots; one out of every "trialstride" trials has
% events plotted. This argument is optional.
% "channelstride" suppresses plots; one out of every "channelstride" channels
% has events plotted. This argument is optional.
```

8.2 wlPlot_plotAllMatrixEventsDebug.m

```
% function wlPlot_plotAllMatrixEventsDebug(cfg, evmatrix, ...
%   figtitle, figlabel, plotsperband, trialstride, channelstride, eventstride)
%
% This function plots all or some of the events in an "event matrix", such
% as that returned by wlFT_doFindEventsInTrial_MT(). Reconstructed events are
% plotted individually against band-pass-filtered and wideband waveforms,
% with detection diagnostic waveforms added.
%
% "cfg" contains figure configuration information (per "FIGCONFIG.txt").
% Certain plot parameters may be overridden.
% "evmatrix" is a structure describing detected events and auxiliary data,
% (per "EVMATRIX.txt").
% "figtitle" is the title to apply to the figure series. Subfigures have a
% prefix prepended to this title.
% "figlabel" is used within figure filenames to identify this figure series.
% "plotsperband" suppresses plots; a maximum of "plotsperband" figures is
% generated per frequency band. This argument is optional.
% "trialstride" suppresses plots; one out of every "trialstride" trials has
% events plotted. This argument is optional.
% "channelstride" suppresses plots; one out of every "channelstride" channels
% has events plotted. This argument is optional.
% "eventstride" suppresses plots; one out of every "eventstride" events is
% plotted. This argument is optional.
```

8.3 wlPlot_plotDetectCurves.m

```
% function wlPlot_plotDetectCurves(cfg, tuneseries, tunelabel, ...
%   dataseries, figtitle, filelabel)
%
% This function plots one or more accuracy-vs-tuning-parameter curves.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "tuneseries" is an array with the independent parameter values (x axis).
% "tunelabel" is the axis label for the independent parameter.
% "dataseries" is an array of structures defining dependent data series.
% "figtitle" is the title to apply to the figure.
% "filelabel" is used within the figure filename to identify this figure.
%
% Data series definition structures contain the following fields:
% "fp" is a [1xN] array containing false-positive values.
% "fn" is a [1xN] array containing false-negative values.
% "tp" is a [1xN] array containing true-positive values.
% "color" [ r g b ] is the colour to use when plotting this series.
% "label" is a string to identify this data series with.
```

8.4 wlPlot_plotEventFreqQAmp.m

```
% function wlPlot_plotEventFreqQAmp(cfg, events, f1, f2, figtitle, filelabel)
%
% This function makes a scatter-plot of frequency and duration (in periods)
% for detected bursts, and a scatter-plot of frequency and detected
% amplitudes.
%
% Plotted frequency is the arithmetic mean of instantaneous frequency.
% Error bars for frequency (and corresponding uncertainty in duration in
% periods) are shown. Plotted amplitude is the arithmetic mean of
% instantaneous envelope magnitude.
%
% "cfg" contains figure configuration information (see FIGCONFIG.txt").
% "events" is an array of event records (per wlSynth_traceAddBursts).
% "f1" and "f2" define the minimum and maximum frequencies to be plotted.
% "figtitle" is the title to apply to the figure.
% "filelabel" is used within figure filenames to identify this figure.
```

8.5 wlPlot_plotEventScatterMulti.m

```
% function wlPlot_plotEventScatterMulti(cfg, eventseries, ...
%   freqrange, durrange, amprange, figtitle, filelabel)
%
% This function makes a scatter-plot of burst amplitudes and burst durations
% (in periods), each as a function of burst frequency.
%
% Plotted frequency and amplitude are the arithmetic means of the
% instantaneous frequency and amplitude curves in the event records.
% Error bars for these values are shown.
%
% "cfg" contains figure configuration information (see FIGCONFIG.txt").
% "eventseries" is an array of structures with the following fields:
%   "eventlist":  A [1xN] array of event records (per wlSynth_traceAddBursts).
%   "color" [r g b]:  The color to use for this series of records.
%   "legend":  The legend title to use for this series of records.
% "freqrange" [min max] defines the frequency range to be plotted (logscale).
% "durrange" [min max] defines the duration range to be plotted (linear).
% "amprange" [min max] defines the amplitude range to be plotted (logscale).
% "figtitle" is the title to apply to the figure.
% "filelabel" is used within figure filenames to identify this figure.
```

8.6 wlPlot_plotEventXYMulti.m

```
% function wlPlot_plotEventXYMulti(cfg, matchseries, ...
%   freqrange, durrange, amprange, figtitle, filelabel)
%
% This function makes a scatter-plot of detected parameter values vs
% ground-truth parameter values for frequency, duration, and amplitude.
%
% Plotted frequency and amplitude are the arithmetic means of the
% instantaneous frequency and amplitude curves in the event records.
% Error bars for these values are shown.
%
% "cfg" contains figure configuration information (see FIGCONFIG.txt").
% "matchseries" is an array of structures with the following fields:
%   "color" [r g b]:  The color to use for this series of records.
%   "legend":  The legend title to use for this series of records.
%   "matchlist":  A [1xN] array of structures with the following fields:
%     "truth":  Ground-truth event record (per wlSynth_traceAddBursts).
%     "test":   Detected event record (per wlSynth_traceAddBursts).
% "freqrange" [min max] defines the frequency range to be plotted (logscale).
% "durrange" [min max] defines the duration range to be plotted (linear).
% "amprange" [min max] defines the amplitude range to be plotted (logscale).
% "figtitle" is the title to apply to the figure.
% "filelabel" is used within figure filenames to identify this figure.
```

8.7 wlPlot_plotEventXYMultiDual.m

```
% function wlPlot_plotEventXYMultiDual(cfg, matchseries, ...
%   freqrange, durrange, amprange, figtitle, filelabel)
%
% This function makes a scatter-plot of detected parameter values vs
% ground-truth parameter values for frequency, duration, and amplitude.
%
% This function differs from wlPlot_plotEventXYMulti in that two event
% lists are provided, rather that one list containing event pairs.
%
% Plotted frequency and amplitude are the arithmetic means of the
% instantaneous frequency and amplitude curves in the event records.
% Error bars for these values are shown.
%
% "cfg" contains figure configuration information (see FIGCONFIG.txt").
% "matchseries" is an array of structures with the following fields:
%   "color" [r g b]:  The color to use for this series of records.
%   "legend":  The legend title to use for this series of records.
%   "truthlist":  A [1xN] array of ground truth event records.
%   "testlist":  A [1xN] array of detected event records.
% "freqrange" [min max] defines the frequency range to be plotted (logscale).
% "durrange" [min max] defines the duration range to be plotted (linear).
% "amprange" [min max] defines the amplitude range to be plotted (logscale).
% "figtitle" is the title to apply to the figure.
% "filelabel" is used within figure filenames to identify this figure.
```

8.8 wlPlot_plotEventsAgainstCommon.m

```
% function wlPlot_plotEventsAgainstCommon(cfg, evlist, evstride, ...
%   signal, samprate, figtitle, filelabel)
```

```
%
% This function plots a series of events, showing the reconstructed signal's
% analytic components against the original signal's components.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "evlist" is a list of event records, in the form used by traceAddBursts().
% "evstride" suppresses plots; one out of every "evstride" events is plotted.
% "signal" is a structure containing the following fields:
%   "wave":   The signal amplitude values.
%   "mag":    The instantaneous magnitude of the signal.
%   "freq":   The instantaneous frequency of the signal.
%   "phase":  The instantaneous phase of the signal.
% "samprate" is the number of samples per second in signal and event data.
% "figtitle" is the title to apply to the figure. Subfigures have a prefix
%   prepended to this title.
% "filelabel" is used within the figure filename to identify this figure.
```

8.9 wlPlot_plotEventsAgainstTrialDebug.m

```
% function wlPlot_plotEventsAgainstTrialDebug(cfg, evlist, samprate, ...
%   wavelist, figtitle, filelabel)
%
% This function plots a series of events, showing the reconstructed signal's
% waveform against the band-pass-filtered and wideband original signals.
% Detection threshold curves are also provided where present.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "evlist" is a list of event records, in the form used by traceAddBursts().
% "samprate" is the number of samples per second in signal and event data.
% "wavelist" is a copy of the event matrix structure's "wave" structure.
%   We look for "bpfwave", "bpfmag", "ftwave", and "fttimes" for signals,
%   "magpowerfast" and "magpowerslow" for magnitude-based detection, and
%   "fvarfast" and "fvarslow" for frequency-based detection.
% "figtitle" is the title to apply to the figure. Subfigures have a prefix
%   prepended to this title.
% "filelabel" is used within the figure filename to identify this figure.
```

8.10 wlPlot_plotEventsAgainstWideband.m

```
% function wlPlot_plotEventsAgainstWideband(cfg, evlist, evstride, ...
%   samprate, timestamps, bpfwave, widewave, figtitle, filelabel)
%
% This function plots a series of events, showing the reconstructed signal's
% waveform against the band-pass-filtered and wideband original signals. A
% spectrogram of the wideband signal is also provided.
%
```

```
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "evlist" is a list of event records, in the form used by traceAddBursts().
% "evstride" suppresses plots; one out of every "evstride" events is plotted.
% "samprate" is the number of samples per second in signal and event data.
% "timestamps" is the time series for signal samples.
% "bpfwave" contains band-pass-filtered signal samples.
% "widewave" contains wideband signal samples.
% "figtitle" is the title to apply to the figure. Subfigures have a prefix
%   prepended to this title.
% "filelabel" is used within the figure filename to identify this figure.
```

8.11 wlPlot_plotEventsWithAux.m

```
% function wlPlot_plotEventsWithAux(cfg, evlist, evstride, ...
%   topcurves, midcurves, botcurves, figtitle, filelabel)
%
% This function plots a series of events, making three-pane plots with
% selected "auxwaves" curves in each pane.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "evlist" is a list of event records, in the form used by traceAddBursts().
% "evstride" suppresses plots; one out of every "evstride" events is plotted.
% "topcurves", "midcurves", and "botcurves" are cell arrays describing pane
%   contents. The first element is the pane's subtitle prefix; remaining
%   elements are groups of three cells containing "auxwaves" curve labels
%   for time series, curve labels for data series, and plotting colours (RGB
%   triplets), for each curve to be plotted.
% "figtitle" is the title to apply to the figure. Subfigures have a prefix
%   prepended to this title. The event number is appended.
% "filelabel" is used within the figure filename to identify this figure.
```

8.12 wlPlot_plotEventsWithContext.m

```
% function wlPlot_plotEventsWithContext(cfg, evlist, evstride, ...
%   signal, samprate, fnom, contextlengths, contextcycles, ...
%   figtitle, filelabel)
%
% This function plots a series of events, showing the reconstructed signal's
% analytic components against the original signal's components.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "evlist" is a list of event records, in the form used by traceAddBursts().
% "evstride" suppresses plots; one out of every "evstride" events is plotted.
% "signal" is a structure containing the following fields:
%   "wave":   The signal amplitude values.
%   "mag":    The instantaneous magnitude of the signal.
%   "freq":   The instantaneous frequency of the signal.
%   "phase":  The instantaneous phase of the signal.
% "samprate" is the number of samples per second in signal and event data.
% "fnom" is the nominal frequency for purposes of context window size.
% "contextlengths" is the minimum size of the surrounding context as a
%   fraction of event duration.
% "contextcycles" is the minimum size of the surrounding context as a
%   fraction of the nominal cycle period of the event signal.
% "figtitle" is the title to apply to the figure. Subfigures have a prefix
%   prepended to this title. The event number is appended.
% "filelabel" is used within the figure filename to identify this figure.
```

8.13 wlPlot_plotExtractedParams.m

```
% function wlPlot_plotExtractedParams(cfg, times, wave, env, freq, phase, ...
%   figtitle, filelabel)
%
% This function plots a time-series waveform, its envelope, its instantaneous
% frequency, and its instantaneous phase as three subfigures (envelope is
% superimposed on the waveform plot).
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "times" is a [1xN] array of time values.
% "wave" is a [1xN] array of signal waveform values.
% "env" is a [1xN] array of envelope waveform values.
% "freq" is a [1xN] array of instantaneous frequency values.
% "phase" is a [1xN] array of instantanous phase values, in radians. This is
%   converted to degrees when plotted.
% "figtitle" is the title to apply to the figure. Subfigures have a prefix
%   prepended to this title.
% "filelabel" is used within the figure filename to identify this figure.
```

8.14 wlPlot_plotInphaseQuadrature.m

```
% function wlPlot_plotInphaseQuadrature(cfg, times, wave_i, wave_q, ...
%   figtitle, filelabel)
%
% This function plots two time-series waveforms, labeled as the in-phase and
% quadrature portions of a signal.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "times" is a [1xN] array of time values.
% "wave_i" is a [1xN] array of in-phase signal waveform values.
% "wave_q" is a [1xN] array of quadrature signal waveform values.
% "figtitle" is the title to apply to the figure.
% "filelabel" is used within the figure filename to identify this figure.
```

8.15 wlPlot_plotMatrixBurstRates.m

```
% function wlPlot_plotMatrixBurstRates(cfg, ...
%   rate_avg, rate_dev, rate_sem, bg_avg, bg_dev, bg_sem, ...
%   timebins, bandtitles, bandlabels, channelnames, ...
%   figtitle, filelabel, timetitle)
%
% This function plots burst rate data returned by wlStats_getMatrixBurstRates.
%
% If only one time window is present, one plot per channel is generated, with
% frequency band as the independent variable.
%
% If multiple time windows are present, multiple plots per channel are
% generated (one per band), with time as the independent variable.
%
% The "timetitle" argument is optional. If given, it overrides the time
% axis label (which defaults to "Time (s)"). This is intended to be used
% to plot detection against threshold or other independent parameters,
% in addition to window time.
%
% "cfg" contains figure configuration information (per "FIGCONFIG.txt").
% "rate_avg" contains burst rates, indexed by (bidx, cidx, widx).
% "rate_dev" contains the standard deviation of burst rates across trials.
% "rate_sem" contains the standard error of the mean of the burst rate
%   (the estimated standard deviation of the value of "rate_avg").
% "bg_avg" contains the background burst rate, indexed by (bidx, cidx, widx).
% "bg_dev" contains the standard deviation of the background rates.
% "bg_sem" contains the standard error of the mean of the background rates.
% "timebins" is a vector of length Nbins+1 containing time window edges, or
%   a vector of length Nbins containing time window midpoints, or a cell
%   array of length Nbins containing [ min max ] time ranges.
% "bandtitles" is a cell array containing human-readable plot-safe band names.
% "bandlabels" is a cell array containing filename-safe band labes.
% "channelnames" is a cell array containing raw channel names (such as
%   from Field Trip's "label" field).
% "figtitle" is the title to apply to the figure series.
% "filelabel" is used within figure filenames to identify this figure series.
% "timetitle" (optional) is a character vector indicating the time axis label
%   in plots. If this argument is omitted, it defaults to 'Time (s)'.
%
% No return value.
```

8.16 wlPlot_plotMatrixErrorStats.m

```
% function wlPlot_plotMatrixErrorStats(cfg, serieslist, figtitle, filelabel)
%
% This function creates bar plots and scatter-plots of specified error
% metrics for a specified series of event matrices.
%
% Bar plots show a histogram of event counts in each error value bin.
% Scatter plots show error values vs event frequency, amplitude, duration,
% and signal-to-noise ratio.
%
% "cfg" contains figure configuration information (per FIGCONFIG.txt).
% "serieslist" is an array of structures with the following fields:
%   "evmatrix":  The event matrix to plot.
%   "errfield":  The field name within (event).auxdata with the error value.
%   "legend":  The label to apply to this data series in the plot legend.
%   "color" [r g b]:  The color to use for this data series in the plot.
% "figtitle" is the title to apply to the figure. Additional text is appended.
% "figlabel" is used within figure filenames to identify this figure.
```

8.17 wlPlot_plotMultipleExtracted.m

```
% function wlPlot_plotMultipleExtracted(cfg, ...
%   wavelist, freqlist, phaselist, ...
%   figtitle, filelabel);
%
% This function plots multiple time-series waveforms, multiple instantaneous
% frequency series, and multiple instantaneous phase series as three
% subfigures.
%
% No legends are specified; there usually isn't enough room with subfigures.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "wavelist" is an array of structures defining waveform traces.
% "freqlist" is an array of structures defining frequency plots.
% "phaselist" is an array of structures defining phase plots.
%   Phase is in radians; it's converted to degrees for plotting.
% "figtitle" is the title to apply to the figure. Subfigures have a prefix
%   prepended to this title.
% "filelabel" is used within the figure filename to identify this figure.
%
% Waveform definition structures contain three fields:
% "waveform.times" is a [1xN] array of time values.
% "waveform.data" is a [1xN] array of data samples.
% "waveform.color" is a [1x3] array of color component values (0..1 RGB).
```

8.18 wlPlot_plotPowerAndPersist.m

```
% function wlPlot_plotPowerAndPersist(cfg, wave, fmin, fmax, ...
%   figtitle, filelabel)
%
```

```
% This function plots power spectral density and a persistence spectrum
% for the specified waveform.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "wave" is a [1xN] array of signal waveform values.
% "fmin" and "fmax" define the frequency range to be plotted.
% "figtitle" is the title to apply to the figure.
% "filelabel" is used within figure filenames to identify this figure set.
```

8.19 wlPlot_plotReconParams.m

```
% function wlPlot_plotReconParams(cfg, realtimes, realwave, ...
%   exttimes, extmag, extphase, figtitle, filelabel)
%
% This function plots a reconstructed time-series burst waveform against
% the real data from which it was extracted.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "realtimes" is a [1xN] array of time values from the real waveform.
% "realwave" is a [1xN] array of real signal waveform values.
% "exttimes" is a [1xN] array of time values for extracted magnitude/phase.
% "extmag" is a [1xN] array of extracted instantaneous magnitude values.
% "extphase" is a [1xN] array of extracted instantaneous phase angles (in rad).
% "figtitle" is the title to apply to the figure.
% "filelabel" is used within the figure filename to identify this figure.
```

8.20 wlPlot_plotWaveAndSpec.m

```
% function wlPlot_plotWaveAndSpec(cfg, times, wave, figtitle, filelabel)
%
% This function plots a time-series waveform and a corresponding spectrogram.
%
% "cfg" contains figure configuration information (see "FIGCONFIG.txt").
% "times" is a [1xN] array of time values.
% "wave" is a [1xN] array of signal waveform values.
% "figtitle" is the title to apply to the figure.
% "filelabel" is used within figure filenames to identify this figure set.
```