BrainWavelet Toolbox v2.0 Docmentation

– *fMRI Edition* –

Ameera X Patel

November 26, 2017

If you use these methods or any code from this toolbox in a publication, please cite the following two articles:

Patel, AX et al. (2014). ***A wavelet method for modeling and despiking motion artifacts from resting-state fMRI time series.*** **NeuroImage. 95: 287–304.**

Patel, AX and Bullmore, ET (2016). ***A wavelet-based estimator of the degrees of freedom in denoised fMRI time series for probabilistic testing of functional connectivity and brain graphs.*** **NeuroImage. 142: 14–26.**

Contents

1 Introduction

Welcome to the BrainWavelet Toolbox (BWT). This toolbox was designed for the removal of motion artefacts from functional Magnetic Resonance Imaging (fMRI) data. Motion artefacts, caused by abrupt subject movement, can be notoriously difficult to model and remove from fMRI time series. They can cause a number of problems, ranging from voxel-shift intensity changes, due to differences in magnetic susceptibility between adjacent brain regions, to proton density changes, and geometric deformation of the brain.

In the time series themselves, motion artefacts appear as non-stationary events, usually in the form of negative spikes, due to spin de-phasing and subsequent signal intensity loss. The wavelet despiking algorithm was designed to remove these non-stationary events, and to accommodate two additional observations. First that the manifestation of motion artefacts may be spatially variable, for example, the frontal and occipital lobes are particularly vulnerable to damage from rotational movements. Secondly, their effects may persist for many frames after the movement has occurred as lower frequency artefacts (spin history effects).

Below you will find details on how to implement the wavelet despikig algorithm on fMRI datasets. The algorithm is written in MATLAB and despikes time series on a voxel-wise basis. For additional details on this algorithm, and for more details on the method, please see section 2, and the following paper:

Toolbox Citation 1: Patel, AX et al. (2014). *A wavelet method for modeling and despiking motion artifacts from resting-state fMRI time series.* NeuroImage. 95:287–304.

This release of the BWT also incorporates the new probabilistic inference methods described in Patel and Bullmore [2016]. These enable estimation of effective degrees of freedom (*df* or EDOF) in the wavelet domain after denoising. Normalising connectivity by *df* enables control over the type I error and significantly reduces the false positive rate. This method can be applied at a single subject and group level. For more information on this method, see section 2, and the following paper:

Toolbox Citation 2: Patel, AX and Bullmore, ET (2016). *A wavelet-based estimator of the degrees of freedom in denoised fMRI time series for probabilistic testing of functional connectivity and brain graphs.* NeuroImage. 142:14–26.

2 Description of Methods

2.1 Wavelet despiking

Wavelet transforms provide multi-resolution (multi-frequency) information about signals, and are known to be effective at detecting transient phenomena, such as spikes, for which Fourier methods are relatively ineffective. They also enable denoising in multiple frequency bands

due to this property. The wavelet despiking algorithm identifies non-stationary events, many of which will be related to subject movement, in the wavelet domain. The algorithm relies on the fundamental premise in fMRI that the underlying noise is more non-stationary than the underlying signal of interest, i.e. that signal properties such as variance are larger in the underlying noise signal.

2.2 Effecive *df* (EDOF) estimation

One additional property of discrete wavelet transforms that is particularly useful is their whitening ability (or ability to decorrelate signals across time and frequency). fMRI time series are coloured, i.e. there are certain properties of the signals such as low frequency or positive autocorrelations that means that the time points are not independent. This means that the effective *df* $\neq$ the number of time points (N), but are <N. Estimating the effective *df* is thus non-trivial, especially after denoising for motion artefacts, and requires a whitening transform. As shown in [Patel and Bullmore, 2016], use of nominal *df* where one assumes *df* = N significantly inflates the false positive rate (type I error). The wavelet despiking algorithm, by virtue of the fact that it denoises in the wavelet domain, can be used to estimate the effective *df* (EDOF) after denoising using a simple extrapolation of the existing mathematical methods described in [Percival and Walden, 2006]. These extended mathematical equations are described in [Patel and Bullmore, 2016].

2.3 Step-by-step method

The methods for denoising and *df* estimation can be summarised in 6 key steps:

1. **Wavelet Transform** Each voxel time series is decomposed into a set of wavelet scales for analysis in the wavelet domain, using the Maximal Overlap Discrete Wavelet Transform (MODWT). The MODWT is implemented using the pyramid algorithm [Percival and Walden, 2006, Cornish, 2006]. The MODWT has a number of key advantages over the Discrete Wavelet Transform (DWT) that makes it particularly useful for denoising transients in fMRI time series. These include: i) it is naturally defined for all sample sizes, and ii), it can eliminate alignment artifacts [Percival and Walden, 2006], making it easier to locate transients in coarser scales (lower frequencies) to a specific point in time.

2. **Identifying maximal and minimal wavelet coefficients** After decomposition, wavelet coefficients are temporally aligned according to the phase delay properties of the filter used (see section 3.1). Maximal and minimal wavelet coefficients are then identified (within each scale), within a 2×2 temporal window. The method used is an adaptation of the more commonly used "modulus maxima method"; the adaptation allows for detection of transients in consecutive time points (or data frames), which is particularly important in high movement datasets.

3. **Chain detection** Non-stationary events caused by movement are represented as chains of maximal and minimal wavelet coefficients, present at the same point in time, but in multiple scales. These chains capture the energy related to non-stationary events. Transients

of all sizes will be represented in this way. Thus, in order to target denoising to larger transients, such as those associated with movement, only those above a specified threshold for denoising will be flagged. Next, to prove mathematically which coefficients are part of maxima and minima chains, one would need to generate a dense set of scales [Percival and Walden, 2006], which is computationally expensive. Instead, the algorithm implements a commonly used mathematical approximation via a sliding window function, which looks at the position of maximal and minimal coefficients at any given scale, relative to other maxima and minima in adjacent scales [Percival and Walden, 2006].

4. **Chain removal** All coefficients that are part of a maxima or minima chains ('noise' coefficients, ϕ) are set to zero in the wavelet domain, a process known as hard thresholding. The coefficients are then temporally de-aligned (the inverse to the shift applied in step 2) according to the phase delay properties of the filter.

5. **Effective *df* (EDOF) estimation** The effective *df* are estimated from the set of coefficients that are not part of maxima or minima chains ('signal' coefficients, α), at each wavelet scale. This yields an estimate of effective *df* for each voxel time series at each wavelet scale (or frequency band). The effective *df* are additive across scales. So, for example if the effective *df* for a single voxel time series at scale 1 = 50, and at scale 2 = 30, then the total effective *df* for scales 1 and 2 = 80.

6. **Time series recomposition** Each voxel time series is then recomposed from the 'signal' coefficients (α), using the inverse MODWT (iMODWT), to generate the "wavelet despiked" signal. The iMODWT is implemented using the inverse pyramid algorithm [Percival and Walden, 2006, Cornish, 2006]. A "noise" signal is also recomposed from the non-stationary (or 'noise') coefficients (ϕ) removed in step 4 using the same process in case further analysis on this signal is required.

There are a number of options included in the MATLAB code to tailor the above steps to different types of data, including data with different levels of artefact and different artefact characteristics. The defaults are set to the values used in Patel et al. [2014]. Please see below for more details.

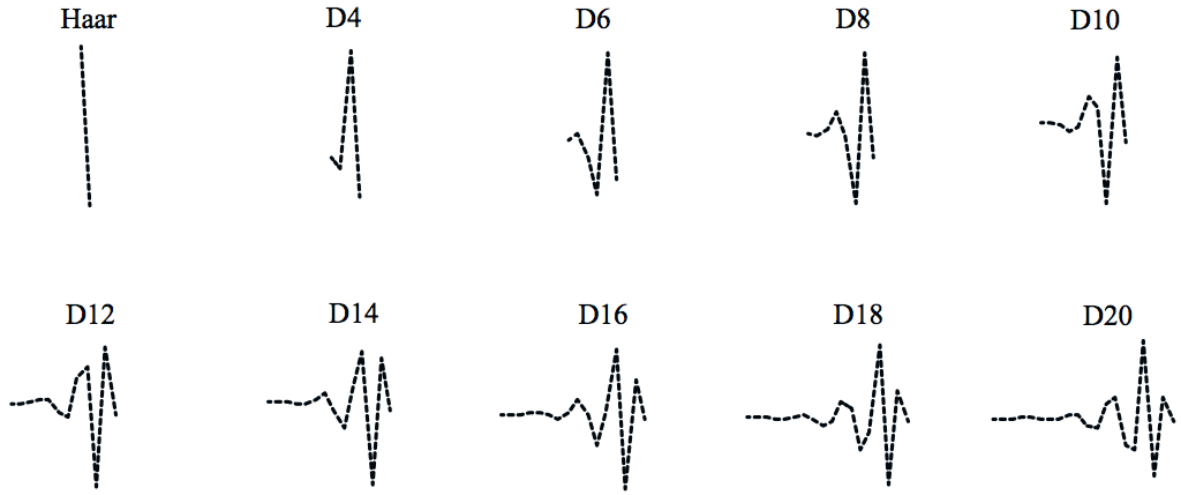
3 Algorithm Options

3.1 Wavelets

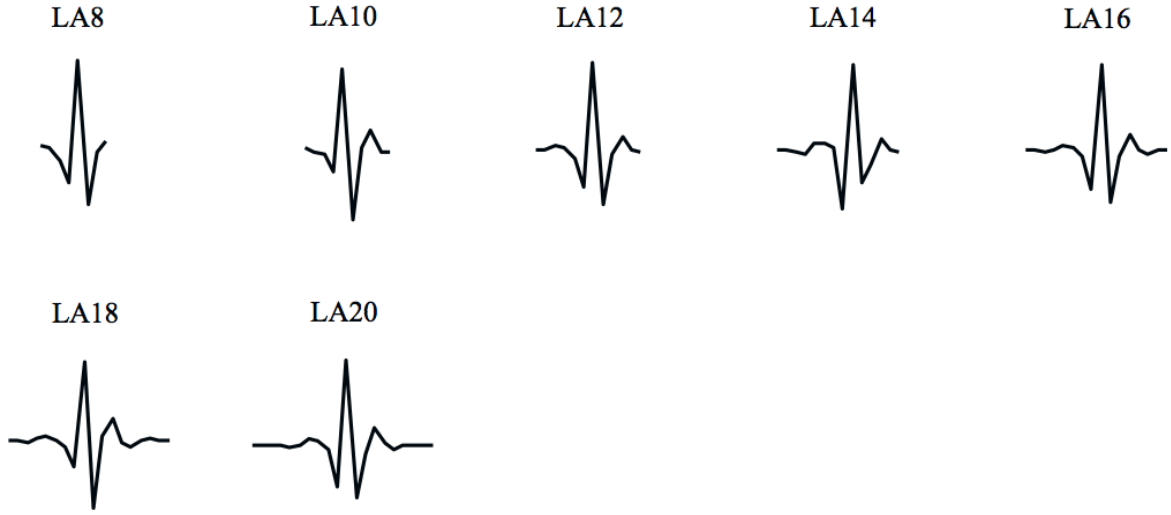
The following wavelet options are available for computing the wavelet transform to be used in wavelet despiking (as in section 2 step 1) . Please note that some of the wavelets with longer filter lengths are not ideal for fMRI time series. Different wavelets will be effective at identifying different types of artefact, but a good all-round choice for denoising motion artefacts in fMRI data is the extremal phase (Daubechies) wavelet filter with length 4 ('D4' or 'db2') or the filter with length 8 ('D8' or 'db4'). D4 is the default option, as in [Patel et al., 2014]. Please note that in theory longer filter lengths are preferred for better decorrelation (i.e. if estimating effective

df), but shorter filter lengths are preferred for denoising artefacts that occur in adjacent frames (as discussed in [Patel and Bullmore, 2016]). However, in practice both D4 and D8 filters produce similar results.

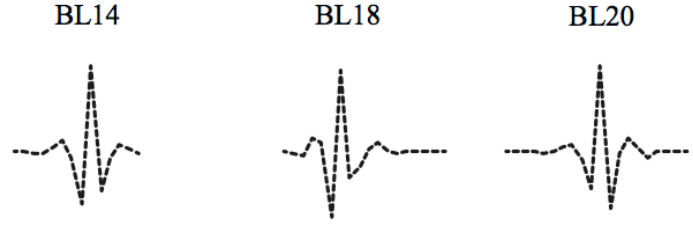
Extremal phase filters:



Least assymetric filters:



Best localised filters:



Coiflet filters:

3.2 Boundary conditions

There are two options for dealing with time series boundaries: 'circular' (or 'periodic') and 'reflection'. The periodic boundary attempts to model the end of the time series using the start. This is a good approximation if there are no non-stationary behaviours such as drifts present in the time series. If the baseline changes through time, for example as a result of scanner drift or a step change in intensity (from voxel shifts), a bias will be incurred in estimation of some wavelet coefficients. The number of biased coefficients will vary depending on the scale and the wavelet used. These are known as boundary coefficients. The reflection boundary, as the name suggests, models the end of the time series by reflecting it in the time dimension. If the time series had N time points originally, the result of applying this boundary condition is a time series of $2N$ time points. This has a computational cost (increased memory usage and computation time), but boundary coefficients will not be biased by a circularity assumption. The reflection boundary condition is set as the default option.

3.3 Number of scales

The number of scales computed will depend on the number of time points over which the data has been collected. There are three methods for calculating the number of scales: 'conservative', 'liberal' and 'extreme'. All options depend on estimating the number of scales from the nearest power of 2.

Conservative The number of scales is defined as the highest power of 2 that yields a number less than the number of non-boundary coefficients. As described above in section 3.2, the number of non-boundary coefficients will vary from scale to scale. This consideration is not so important if bias from boundary conditions is minimised, e.g. by use of the reflection boundary.

Liberal This method computes the number of scales as the highest power of 2 that yields a number less than or equal to the number of time points. There is no consideration of the number of boundary coefficients, and therefore, if a circular boundary condition is applied, the highest scale(s) may contain coefficients that are all biased by the circularity assumption.

Extreme Unlike the Discrete Wavelet Transform (DWT), the MODWT allows for computation of more scales than would be allowed for by an equivalent time series transformed using the DWT [Percival and Walden, 2006]. This is slightly counter-intuitive, as it would involve deriving information in frequencies lower than those allowed for by the length of the time series. In these non-intuitive scales, the level of additional detail $\rightarrow$ zero in frequencies lower than those that would normally require $1.5\times$ the number of time points to calculate. For this option, the number of scales is therefore defined as the highest power of 2 that yields a number less than or equal to the number of time points $\times 1.5$. This option may be useful for denoising in certain (rare) circumstances, e.g. in the presence of extreme amplitude artefacts.

3.4 Threshold

This is the coefficient threshold value used during maxima and minima coefficient definition. For fMRI data, the default has been set to retain all local maxima and minima corresponding to large non-stationary events (as in Patel et al. [2014]). This step is included to reduce computational cost, to improve the successful identification of maxima and minima chains related to large non-stationarities, and conversely, to prevent removal of stationary coefficients (i.e. to reduce false positive artefact detection). This is set by default to 10, however, this number will need to be tuned for datasets collected using different sequences and processed using different pre-processing pipelines to those used in [Patel et al., 2014]. More detail on this is provided in section 7.2.

3.5 Chain search options

There are three options, each implementing a different set of rules for defining maxima and minima chains. These rulesets are termed: 'conservative', 'moderate' and 'harsh'.

Conservative This set of rules stipulates that maxima or minima chains, flagged for removal, have to start in scale 1. This means that the artefacts removed will all have a high frequency component starting in scale 1. Of course, if lower frequency components (such as prolonged spin-history-type effects) are associated with these higher frequency components, they will also be removed.

Moderate The 'moderate' ruleset allows maxima and minima chains to start in either scale 1 or 2. Both scales 1 and 2 are relatively high frequency bands, but given the observation that on some (rarer) occasions, artefacts caused by abrupt movements are represented as maxima or minima chains starting in scale 2, this set of rules allows these latter artefacts to be flagged for removal as well. This is the default option (as in Patel et al. [2014]).

Harsh This final set of rules allows maxima and minima chains to start in any scale. This is a useful denoising option if there are low frequency as well as high frequency artefacts present, independently of each other. Note, that use of these criteria may result in harsh reduction of

the effective *df* (EDOF).

3.6 Spike Percentage

The Spike Percentage is a vector of length N, where N is the number of time points in the time series. For each time point, it quantifies the percentage of voxels containing spikes. A voxel time series is considered to contain a spike at any point in time if there is a maxima or minima chain at that point in time starting in scale 1. It therefore shows how badly the voxel time series are affected by artefacts (most commonly movement related), without any prior knowledge of the physical movement parameters. The Spike Percentage is an optional output from the wavelet despike function, and offers a useful way to diagnose how badly datasets are affected by movement artefacts, in a similar way to the movement parameters. This artefact diagnostic is output by default, but can be turned off with the input options.

3.7 Effective *df*

Use of this flag when running WaveletDespike.m results in a NIfTI output of the effective *df* (EDOF) at each voxel and at each wavelet scale (j) of the denoised (wavelet despiked) data. The number of scales (or frequency bands) is set by the nscale option (see section 3.3). So if the original input NIfTI had dimensions 30×35×40×300 (i.e. 30×35×40 voxels with a time series length of 300), the effective *df* NIfTI will have dimensions of 30×35×40×nscales. Use of this option does not affect how denoising is conducted, it simply provides an additional output of the effective *df*. So, if you have denoised data with an older version of this toolbox, you could simply re-run the wavelet despiking step with this version of the toolbox, adding the EDOF flag. This will generate voxel-wise effective *df* maps without changing how the denoising is done. To work out what frequency bands the wavelet scales correspond to requires the TR. So, if TR = 2s, then $\frac{1}{4}$ to $\frac{1}{2}$ Hz will be Nyquist frequencies. Scale (j) = 1 will represent $\frac{1}{8}$ to $\frac{1}{4}$ Hz; scale (j) = 2 will represent $\frac{1}{16}$ to $\frac{1}{8}$ Hz; scale (j) = 3 will represent $\frac{1}{32}$ to $\frac{1}{16}$ Hz and so on.

4 Memory considerations

4.1 Batch processing

For large datasets, the memory required for wavelet despiking may be quite high. For example, an adult brain imaged with voxel dimensions 3×3×3 mm over 250 time points, would comprise approximately 100,000 voxels and would require approximately 19 Gb of memory to despike. The MemorySolver function in this toolbox will automatically detect the amount of memory available, and if there is not enough to process the brain as one large 3D+time matrix, it will initiate a 'Batch Mode'. This will divide up the brain and batch process groups of voxels. The denoised output will not be affected by whether the Batch Mode is initiated or not, but the runtime might be slightly longer.

To prevent greedy usage of all available memory, which may slow other running applications, there is an option to cap the amount of memory (in Gb) this algorithm is allowed to use. Please see section 6.2 for more information on this.

4.2 Image compression

To reduce the memory footprint, there is an option to compress the input NIfTI file before wavelet despiking commences. As NIfTI files contain a large number of non-brain voxels, which are represented as time series of zeros, compressing out these voxels before wavelet despiking can reduce the RAM footprint by around half in most cases. The NIfTI parsing function in this toolbox will store the location of these non-brain voxels in a structure with the dataset header, and will add them back to the dataset when wavelet despiking has finished, prior to recompiling the NIfTI file. There is therefore no adverse consequence of including image compression to despiking, and as a result, this option is highly recommended and is set by default. For more information, see section 6.2.

5 Setup

Setting up this toolbox should be straightforward. The first step is to move the folder titled "BrainWavelet" inside the downloaded folder to the desired location on your hard drive. Next, within the MATLAB environment, switch to the toolbox directory using:

>> cd /path/to/BrainWavelet/

and run the setup.m file included in the download folder by entering the following into the MATLAB command prompt.

>> setup

This will then ensue to add the relevant paths to your computer, and compile the mex files for the MODWT and inverse MODWT. For this, you will need a mex compiler. Pre-compiled binaries for a UNIX 64-bit system have been included, so you will not need to compile the mex files if you run this system. Please note that if you do not have write access for setting paths globally, e.g. on a shared server, you will need to add these locally to your startup.m file. You will know that this is the case, if the matlab error 'Undefined function or variable 'WaveletDespike' for input arguments of type 'char'' appears when you try to run wavelet despiking. The startup.m is often located within a subfolder called 'matlab' in the home directory. If this the case, the following paths should be appended to the startup file manually:

path('/path/to/BrainWavelet/BWT',path)
path('/path/to/BrainWavelet/third_party/cprintf',path)
path('/path/to/BrainWavelet/third_party/NIfTI',path)

path('/path/to/BrainWavelet/third_party/wmtsa/dwt',path)
path('/path/to/BrainWavelet/third_party/wmtsa/utils',path)
path('/path/to/BrainWavelet/BWT/ProbConn',path)

Please note, that if you already have functions with the same names as those included in this toolbox, MATLAB will throw errors. There are no know clashes with MATLAB built-in functions, but some may exist with other open-source add-ons. The quickest solution is to temporarily disable the duplicate functions that are not included in this toolbox. Please report any such clashes back to be addressed in future releases.

This toolbox relies on some open-source third-party functions, all of which are included in the toolbox with their respective copyright licenses (in accordance with the BSD license). For more information on these functions, please see the 'third-party' subfolder in this toolbox.

6 Usage

6.1 Example of WaveletDespike.m

Wavelet despiking is run from the MATLAB command prompt using the function WaveletDespike.m. This will call other functions from the toolbox. All options have default presets in order to make the function easy to use. The only required inputs are the name of the file to denoise, and an output prefix. The function will write files to the current working directory, so please run the code from the directory in which you would like to output the denoised data. Effective *df* (EDOF) and Spike Percentage (SP) are output by default. To turn these outputs off, set the respective flags to 0 (see below).

For example, in order to wavelet despike the file 'rest.nii.gz', and output files with the prefix 'rest_dn', you would change into the directory where you want the files written (using the cd command), and then run the following command:

>> WaveletDespike('/path/to/rest.nii.gz','rest_dn');

This will write the following 4 files to your working directory:

rest_dn_wds.nii.gz – This is the NIfTI file of wavelet despiked time series. The output file dimensions are identical to the input file (rest.nii.gz) dimensions.

rest_dn_noise.nii.gz – This is the NIfTI file of noise timecourses removed during wavelet despiking. The output file dimensions are identical to the input file (rest.nii.gz) dimensions.

rest_dn_SP.txt – This is Spike Percentage timecourse for the dataset. This is a vector of length N, where N is the number of time points in the input NIfTI file.

rest_dn_EDOF.nii.gz – This is the NIfTI file of voxel-wise effective *df*. The first 3 spatial dimensions of this output file (i.e. $x \times y \times z$) are the same as the input NIfTI. The 4th

dimension (originally time), now contains wavelet scales from 1 to J (J being the maximum number of scales with meaningful information - this is defined and preset by the nscales option).

The algorithm will process all non-zero voxels in the input NIfTI file, so please ensure that you mask out non-brain regions prior to despiking, otherwise the estimate of Spike Percentage will be biased.

Additional input options must be specified as MATLAB string-value pairs, for example, if you want to use a circular boundary and the 'la8' wavelet with a RAM usage limit of 5 Gb, you would type:

>> WaveletDespike('rest.nii.gz','rest_dn','boundary','circular','wavelet','la8','LimitRAM',5);

6.2 Built-in help files

All functions in this toolbox have built-in help files. For example, entering the following command:

>> WaveletDespike

in the MATLAB command prompt will display the following built-in help file, which outlines the various input options for the function.

FUNCTION: WaveletDespike – wavelet despikes time series, and outputs effective df as described in:

Patel, AX. et al (2014). A wavelet method for modeling and despiking motion artifacts from resting-state fMRI time series. NeuroImage. 95: 287–304.

Patel, AX. and Bullmore, ET. (2016). A wavelet-based estimator of the effective degrees of freedom in denoised fMRI time series for probabilistic testing of functional connectivity and brain graphs. NeuroImage. 142: 14–26.

USAGE: WaveletDespike(Inii,Opref,varargin)

Inputs:
- Inii – NIfTI file (3D+t dataset) containing time series to despike.
- Opref – Output prefix for despiked NIfTI files and Spike Percentage / EDOF (if specified).

Additional Input Options:
(These must be specified as MATLAB string-value pairs.)

wavelet	–	Wavelet to use for wavelet transform. Input must be a string containing one of the following: ‘d4’,‘d6’,‘d8’,‘d10’,‘d12’,‘d14’,‘d16’,‘d18’,‘d20’,‘la8’, ‘la10’,‘la12’,‘la14’,‘la16’,‘la18’,‘la20’,‘bl14’,‘bl18’,‘bl20’, ‘c6’,‘c12’,‘c18’,‘c24’,‘haar’. [Default=‘d4’].
threshold	–	Threshold for maximal and minimal wavelet coefficients. [Default=10].
boundary	–	Boundary condition to apply. Input must be a string containing one of the following: ‘circular’,‘reflection’. [Default=‘reflection’].
chsearch	–	Rules for identifying maxima and minima chains. Input must be a string containing one of the following: ‘conservative’, ‘moderate’, ‘harsh’. [Default=‘moderate’].
nscale	–	Method for computing number of scales. Input must be a string containing one of the following: ‘conservative’, ‘liberal’, ‘extreme’ [Default: ‘liberal’], or a number between 0 and 1. If a number is specified the number of scales used will be a fraction of the maximum allowed by the liberal method.
compress	–	Binary flag indicating whether to compress out non-brain regions, 1, or not, 0, from input NIfTI file before wavelet despiking. This saves RAM and reduces runtime [Default=1].
SP	–	Binary flag indicating whether to output the Spike Percentage for the dataset [Default=1].
EDOF	–	Binary flag indicating whether to output the effective degrees of freedom maps for the dataset at each wavelet scale. [Default = 1].
EDOF_method	–	Method for calculating effective degrees of freedom – ‘biased’ or ‘unbiased’. NB biased method does not apply for reflection boundary condition. [Default: ‘unbiased’].
LimitRAM	–	Specify an upper bound of RAM usage (in Gigabytes). Default is to use all available RAM.
verbose	–	Binary flag indicating whether to display incremental output from the algorithm, 1, or not, 0. [Default = 1].

Outputs: This function will write the following files to the current directory:

Opref_wds.nii.gz	–	Wavelet despiked time series.
Opref_noise.nii.gz	–	Noise time series removed in wavelet despiking.
Opref_SP.txt	–	The Spike Percentage (if specified at input).

Opref_EDOF.nii.gz – Degrees of freedom map for each wavelet scale.

EXAMPLE: WaveletDespike('rest.nii.gz','rest','wavelet','la8','LimitRAM',5)

AUTHOR: Ameera X Patel
CREATED: 26-12-2013

CREATED IN: MATLAB 7.13

REVISION: 24 (19-11-2017)

TOOLBOX: BrainWavelet Toolbox v2.0

---

7 Other useful tips

7.1 ParseInNii.m

This toolbox function can be used to read NIfTI files into matlab. It converts a 3D+t (NIfTI) file into a 2D matrix of voxels (columns) by time (rows). In other words it compresses out the spatial information. The spatial information (needed for recompiling the NIfTI) is contained within an output variable called 'Info'. So for example, the NIfTI file rest.nii.gz could be read into MATLAB using the following command:

>> [ts, Info, error] = ParseInNii('/path/to/rest.nii.gz');.

The function by default compresses out non-brain voxels to reduce the load on memory and stores the coordinates in the 'Info' file. This option can be turned off by using the MATLAB string-value pair ['compress',0]. So to stop removal of non-brain voxels from the matrix of time series (ts), you would input:

>> [ts, Info, error] = ParseInNii('/path/to/rest.nii.gz','compress',0).

This is useful for comparing the effects of wavelet despiking on the voxel time series. E.g. you could use ParseInNii.m to read in the raw (un-despiked) NIfTI file 'rest.nii.gz', the wavelet despiked output from WaveletDespike.m 'rest_dn_wds.nii.gz' (using the example above), and the noise output from wavelet despiking 'rest_dn_noise.nii.gz'. You could then perform a voxel-by-voxel comparison of the effects of wavelet despiking on the original time series.

7.2 Tuning the algorithm

The default wavelet despiking options are set for the specific pre-processing pipeline used in Patel et al. [2014]. Use with other pipelines may require some additional tuning of the harshness

of denoising using the 'threshold' string-value pair. For example, if more spatial smoothing is applied during pre-processing, then the raw time series amplitude will be smaller and the 'threshold' parameter will need to be made smaller. Alternatively, if time series normalisation is done, this alters the variance of the raw time series so the threshold parameter may need to be tuned upwards or downwards. Adjusting this parameter will alter the harshness of denoising. You should not ordinarily need to make any other adjustments to the other default options when applying wavelet despiking.

If you obtain very large Spike Percentage values after running wavelet despiking, it is likely you will need to reduce the 'threshold' parameter.

The threshold chosen should be the highest value which gives adequate denoising of artefacts. Some plots / methods for looking at efficacy of denoising can be found in Patel et al. [2014].

Once a threshold has been identified, this can then be fixed for that pre-processing pipeline / scanner sequence combination and will not require further adjustment in subsequent datasets collected and processed using that sequence and pre-processing pipeline.

8 Probabilistic connectivity workflow

This release of the BWT includes scripts and code for probabilistic connectivity or graph analysis as described in Patel and Bullmore [2016]. After running wavelet despiking, the following steps may be implemented to generate P value matrices as described in Patel and Bullmore [2016].

1. **Parcellation.** The outputs from WaveletDespike.m should first be parcellated. You will need to parcellate the pre-processed and wavelet despiked NIfTI file (rest_pp.nii.gz) and the EDOF NIfTI output (rest_dn_EDOF.nii.gz) using the same parcellation template.

2. **Edge EDOF.** Edge *df* estimates can be calculated as the minimum of the *df* of the connecting nodes / regions / voxels. This will need to be done for each wavelet scale. Note that the time dimension in the rest_dn_EDOF.nii.gz file output by WaveletDespike.m is wavelet scales. This will be in the order of 8 scales for a dataset of 256 time points. This step can be performed using the toolbox function **MinPairEDOF.m**.

3. **Wavelet correlation.** Next, wavelet correlation matrices need to be generated for each wavelet scale using the parcellated time series. This can be done using the toolbox function **wavcorr.m**

4. **P value matrices.** P value matrices can then be generated using the edge *df* values from step 2, and the wavelet correlation matrices from step 3. The first step is to do a Fisher r-to-Z transform using the toolbox function **RtoZ.m** normalising for *df*, at each wavelet scale. Z scores can then be converted to 1- or 2-tailed P values using the toolbox function **ZtoP.m**. P values then need to be adjusted for multiple comparisons, e.g. using

a FDR adjustment. An algorithm for adjusting for FDR (called **FDR.m**) is provided in the toolbox .

Please note, for ease of implementation, a script has been included in this release of BWT called **MakeP.m** which automates steps 2 to 4 above.

For seed correlation analysis, the same principles apply (as described in [Patel and Bullmore, 2016]. First the minimum EDOF of the ROI to be used as a seed needs to be identified. The ROI is then correlated to all other time series to generate a correlation (R) map. MinPairEDOF.m can be used to work out the minimum EDOF between each pair of voxel and ROI. P values can then be generated for each voxel using the Fisher r-to-Z transform and compared to the standard normal distribution to generate P values (using RtoZ.m and ZtoP.m). These P values can then be adjusted for multiple comparisons.

References

Charlie Cornish. Wavelet Methods for Time Series Analysis - implementation in MATLAB, version 0.2.6, 2006. URL `http://www.atmos.washington.edu/wmtsa/`.

Ameera X Patel and Edward T Bullmore. A wavelet-based estimator of the degrees of freedom in denoised fMRI time series for probabilistic testing of functional connectivity and brain graphs. *NeuroImage*, 142:14–26, 2016.

Ameera X Patel, Prantik Kundu, Mikail Rubinov, P Simon Jones, Petra E Vertes, Karen D Ersche, John Suckling, and Edward T Bullmore. A wavelet method for modeling and despiking motion artifacts from resting-state fMRI time series. *NeuroImage*, 95:287–304, 2014.

Donald B Percival and Andrew T Walden. *Wavelet Methods for Time Series Analysis (Cambridge Series in Statistical and Probabilistic Mathematics)*. Cambridge University Press, 2006. ISBN 978-0521685085.