WAVELET INFORMATION GUI v1.0
TUTORIAL by Vítor Lopes-dos-Santos and Rodrigo Quian Quiroga

INTRODUCTION

This is a tutorial on our MATLAB GUI implementation of the Wavelet-information method, originally described in the publication *Extracting information in spike time patterns with wavelets and information theory* by Vítor Lopes-dos-Santos, Stefano Panzeri, Christoph Kayser, Mathew E. Diamond and Rodrigo Quian Quiroga, Journal of Neurophysiology, 2014 (10.1152/jn.00380.2014).

Reading this manuscript before going through this tutorial will certainly help if you are not familiar with the theoretical background involved.

The toolbox is available online at http://www.le.ac.uk/csn/WI/ or upon request from vtlsantos@gmail.com (Vítor). Please send any suggestions, comments or criticisms regarding this tutorial or our paper. We really appreciate your feedback to help improving our work.

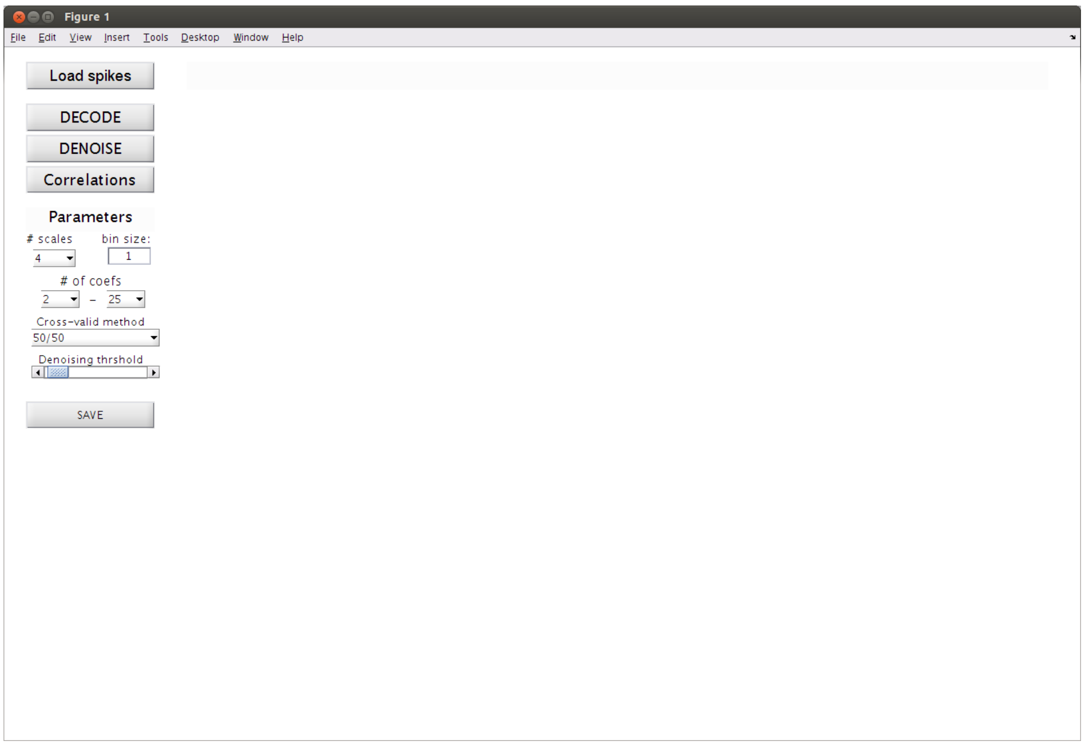
GETTING STARTED

First, you need to extract the provided zip file and add the corresponding folder to MATLAB's search path. Then, you should be able to open the GUI by typing 'WIgui' in the command window.

Let's say for example that you extracted the toolbox in the folder '/home/user/WItoolbox/', then, in MATLAB's command window, you can simply enter:

```
addpath('/home/user/WItoolbox')
WIgui
```

A figure like the one below should pop up.



Now you are ready to start.

LOADING DATA

In order to run decoding or denoising you need load a mat file with two variables: ‘spiketimes’ and `class_id'.

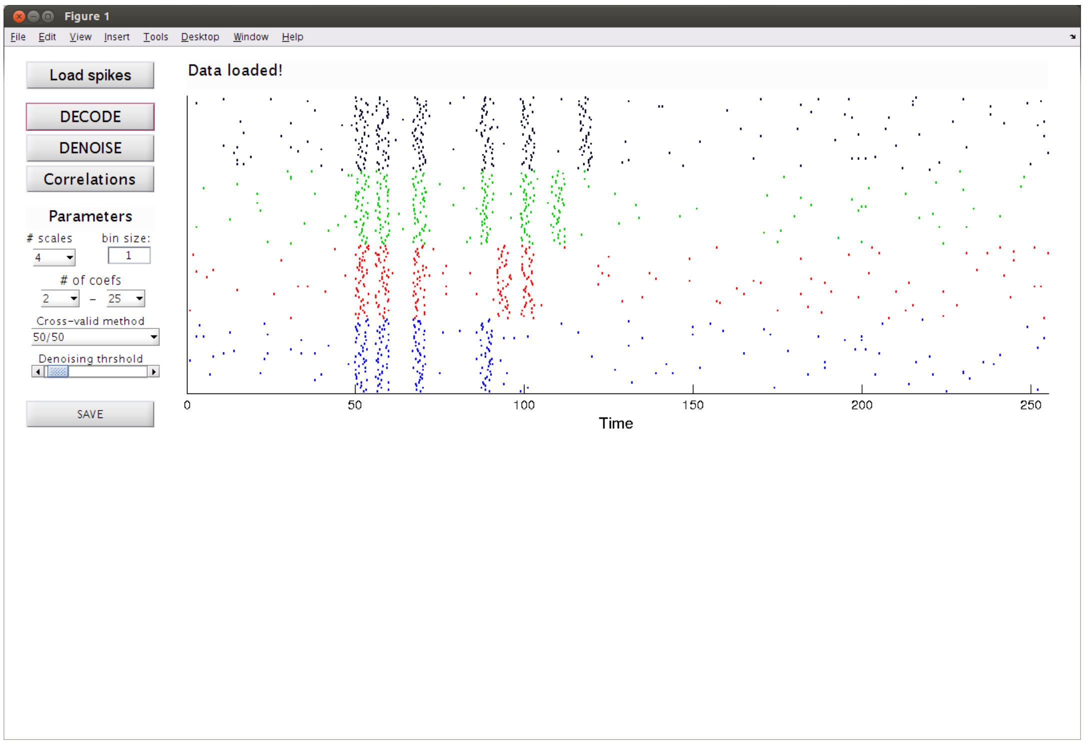
‘spiketimes’ is a cell and each of its element carries the spike times of a given trial.
‘class_id’ is a vector that assigns class to each trial.

We provide some examples with synthetic data with this toolbox (check the folder ‘data’ in the folder you extracted the zip file).

In this tutorial we will use the file ‘simdata_ex1.mat’.

In order to load it hit the button ‘Load spikes’ in the GUI and select it (it is in the same directory you extracted the toolbox).

The figure should be updated to something similar to the one below.



Data loaded!

scales: 4
bin size: 1
# of coefs: 2 – 25
Cross-valid method: 50/50

A rastergram of the loaded data is displayed.

Trials of the same class are shown in the same color.

PARAMETERS

Before you run the methods take a look in the parameters used.
The default values here are the same used on our original publication.
Below we provide a brief description of parameters you can set in the GUI.

# scales: (default = 4) Here select the number of scales (decomposition levels) you want to use.

bin size: (default = 1) Here select the bin size.

\# **of coefs**: (default = 2, 25) Here select the minimum and the maximum number of wavelet coefficients you want to use for decoding/denoising.

cross-valid method: (default = 50/50) Here select the cross validation method for decoding. If you choose ‘50/50’, half of the trials (randomly chosen) of each class will be used for training and the remaining will be used for testing the classifier. If you choose 'Leave-one out' instead, each trial will be classified by a decoder trained with all other trials.

Denoising thrshold: (default = 1) Here select the value (in terms of standard deviations) you want to use to threshold the wavelet reconstructed raster plot when denoising.

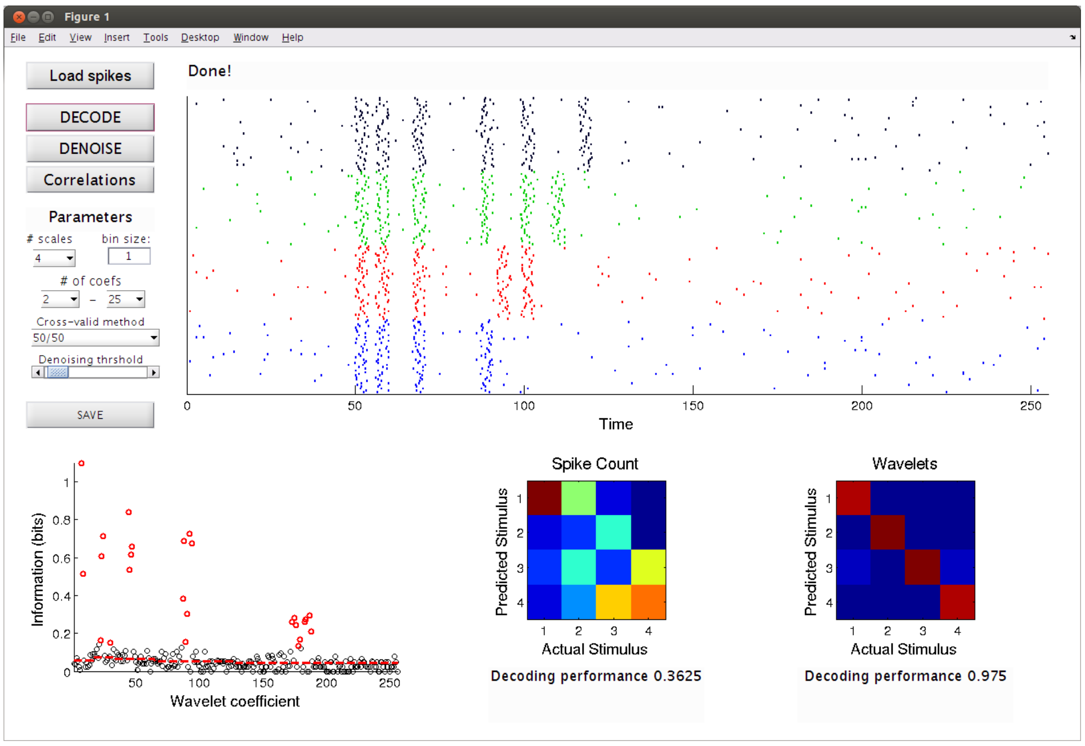
DECODING

When you hit 'DECODE', decoding will be initiated with the parameters selected.

If you are using '50/50' to cross-validate, the algorithm will select which wavelet coefficients will be used for the classification based on the the trials randomly assigned to the training set.

When selection is done, the naïve Bayesian decoder is trained and the remaining trails (the testing set) will be classified. A similar decoder will also be used but using spike counts only.

Then, the GUI will be updated to something like this:



Done!

# scales: 4
bin size: 1
# of coefs: 2 – 25
Cross-valid method: 50/50
Denoising thrshold

Decoding performance 0.3625

Decoding performance 0.975

The bottom left panel shows the Shannon information (vertical axis) of each wavelet coefficient (horizontal axis). Red dashed lines indicate the statistical thresholds for information significance computed by shuffling trial labels. The information of the selected coefficients are displayed in red while the non-selected ones in black.

The bottom middle and right panels show the confusion matrices obtained by classifying spike counts and wavelet coefficients, respectively. The decoding performances are shown below their respective confusion matrices.

Note that for this synthetic example, time patterns provide more information than spike counts alone.

Note that when using '50-50', the trials will be randomly assigned to training and testing sets and therefore the outcome of decoding will not be deterministic; i.e., running DECODE repeatedly may provide different outcomes. If the outcome is very different, you should use 'Leave-one out'.

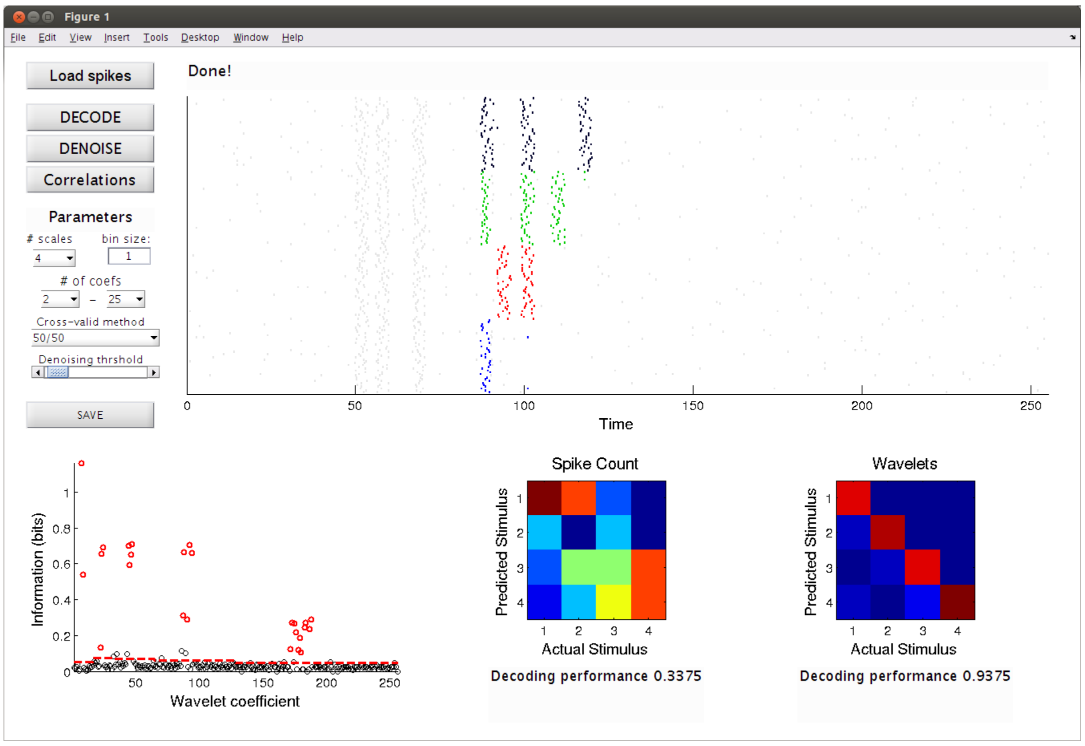
If you select 'Leave-one out' as cross-validation method, you will be using all the information you can use in order to classify each trial; however this means you need to train a new classifier for each trial, which will take much longer. When you have few trials in your dataset, this is the adequate alternative.

DENOISING

When you hit ‘DENOISE’, denoising will be performed based on the parameters selected.

Differently from decoding, there is no cross-validation for denoising: all trials are used for selecting wavelet coefficients.

When denoising is finished, the GUI will update to something similar to the next figure below.



Done!

Parameters
# scales: 4
bin size: 1
# of coefs: 2 – 25
Cross-valid method: 50/50
Denoising thrshold

Decoding performance 0.3375

Decoding performance 0.9375

The spikes that are not found to provide information will be displayed in gray.
A similar plot for the information values of each wavelet coefficient is also displayed in the bottom left panel.

You can also change ‘Denoising threshold’ in the scrollbar. Increasing this threshold will make the denoising more rigorous.

SAVING RESULTS

Hit the SAVE button to save your results.

A struct variable called ‘WIdata’ will be constructed with the results obtained with the last data you loaded.

The fields of the struct are shown below for the example shown above.

```
WIdata =
        binsize: 1
        nscales: 4
        crossvalidation method: '50/50'
        SPKCNTconfusionmatrix: [4x4 double]
        SPKCNTperf: 0.3375
        WVconfusionmatrix: [4x4 double]
        WVperf: 0.9375
        Denoised_spiketimes: {1x160 cell}
```

The field `denoised_spiketimes' is a cell variable similar to ‘spiketimes’ but only with the spikes that remained after denoising (gray spikes are deleted).

Note that if the user has changed Denoising threshold, the last value set will be used (i.e., the one applied in the current displayed rastergram when results are saved).

DECODING WITHOUT THE GUI

Here we show an example of how you can perform decoding without the GUI.

This approach can be useful if you have too many neurons and want to program a for loop, for example.

Below an example of how it works.
You can also find this same code in the file ‘WIscript_caller.m’.

```
% make handles global so all function will 'see' all parameters they need
global handles

% below set the required parameters as you wish
handles.binsize = 1; % window length used to bin data
handles.nscales = 4; % number of scales for wavelet decomposition
handles.nsurr = 50; % number of surrogates for computing shuffling distribution
handles.percentile = 95; % percentile of surrogate distribution for significance
handles.maxwvcoefs = 25; % maximum number of coefs to use
handles.minwvcoefs = 2; % mininum number of coefs to use

% below a mat file and add the data to the global variable
load('simdata_ex2.mat')
handles.spiketimes = spiketimes;
handles.class_id = class id;
WIfunc_binmatrix() % constructs binned matrix
WIfunc_wavedec() % computes wavelet decomposition
WIfunc_decode_5050() % classifies with 50/50 cross-validation
% WIfunc_decode_leaveoneout() % classifies with leave-one out cross-validation
```

When the script below is run, you can find all results in the variable ‘handles’.

The subfield ‘handles.decode’ provides:

```
handles.decode.trainingtrials: % which trials were used for training
handles.decode.testingtrials: % which trials were used for testing
handles.decode.SPKCNTconfusionmatrix: % confusion matrix from spike count decoding
handles.decode.SPKCNTperf: % performance from spike count decoding
handles.decode.WVconfusionmatrix: % confusion matrix from wavelet-information decoding
handles.decode.WVperf: % performance from wavelet-information decoding
```

For the example above:

```
>> handles.decode

ans =

         trainingtrials: [40x1 double]

         testingtrials: [40x1 double]

         SPKCNTconfusionmatrix: [4x4 double]

         SPKCNTperf: 0.6750

         WVconfusionmatrix: [4x4 double]

         WVperf: 1
```

Additionally, more information can be accessed in `handles.matrices'.

```
handles.matrices.actmatrix: % binned activity
handles.matrices.wvmatrix: % wavelet decomposition
handles.matrices.selected_wvcoefs: % wavelet coefficients selected for decoding
```

FINAL REMARKS

This is the tutorial of the first version of the GUI implementation of the Wavelet-information framework. We will be further developing the method to improve its performance and we will be happy to hear any feedback.

We hope that you found this tutorial useful.

Any doubts, comments or problems, please report to vtlsantos@gmail.com (Vítor).

Cheers.