Detection of Action Potentials with the Continuous Wavelet Transform

Zoran Nenadic, D.Sc.

May 4, 2015

1 Introduction

The problem of detecting transient signals in a noisy environment has been studied for decades. In the Statistical Signal Detection Theory the presence of a useful signal in a background noise is normally cast as a hypothesis testing, where under the null hypothesis no signal is present. If the statistics of signal to be detected is not perfectly known, usually no uniformly most powerful (UMP) test exists, and the performance of a detector depends on signal representation [1]. In general, a signal representation can be model based and expansion based. When no appropriate model for the signal can be found, one usually resorts to a "canonical set" of basis function where the signal is projected, giving rise to expansion coefficients. We can think of these coefficients as of signal representation in a new coordinate system. Depending on the signal representation the detection problem can be formulated in various domains such as time domain, frequency domain, time-frequency domain, etc. In time-frequency domain, a signal is projected onto a basis of waveforms that are localized (subject to Heisenberg uncertainty principle) in both time and frequency, yielding a two-dimensional signal representation $Tx(w,t)$ of a one-dimensional signal $x(t)$. An example of this representation is a windowed Fourier Transform introduced by Gabor. A breakthrough in the theory of wavelets offered a powerful alternative to windowed Fourier Transform, where a one-dimensional signal $x(t)$ is represented in time-scale domain by virtue of a wavelet transform $Tx(a,b)$.

FAQ:

- What is wrong with the Fourier basis? *Action potentials (APs) are short (transient) pulses, and belong to the family of time-limited signals. The Fourier Transform of a time-limited signal cannot be band-limited. Therefore, the representation of APs in the frequency domain will be spread and possibly highly overlapped with the spectrum of noise. Based on these observations, one can argue that the Fourier basis is suboptimal for representing APs.*

- Why use the continuous wavelet transform (CWT)? *Simply because we have a good idea about the duration of a typical AP. Namely, a large majority of APs last between* 0.5 *and* 2.0 *[ms]. Rather than using dyadic scales (typically associated with the discrete wavelet transform), where one hunts for signals at a wide range of scales, we can zoom in on the scales of interest—relevant scales. In addition, running the wavelet transform continuously, ensures that no AP will be missed due to its location. This property of the CWT is known as a translation-invariance.*

- What types of wavelets? *Naturally, the wavelets of compact support are preferred, because APs are signals of compact support. In particular, wavelets from biorthogonal family have*

bi-phasic shapes that are reminiscent of those of APs. Simply put, this allows a single AP to be represented with a few wavelet basis functions—sparse representation. Therefore, over the set of relevant scales, a bank of approximately matched filters is created, with the hope of getting a large wavelet coefficient once the wavelet function and AP are aligned.

This tutorial is an accompanying document to the computer code for *detection of action potentials with the continuous wavelet transform*. The details of the method can be found in [2], and the code is written in MATLAB^{TM}. The code consists of several functions and to run the main function Wavelet Toolbox needs to be installed. The main function is:

(1) `detect_spikes_wavelet.m`

This is a stand-alone function for the detection of APs in noisy neural data. The auxiliary functions are:

(1) `get_score.m`

(2) `lag_ts.m`

(3) `lead_ts.m`

and are provided for tutorial purposes only. Two synthetic data files are provided as well:

(1) `clean.mat`

(2) `corrupted.mat`

`detect_spikes_wavelet.m` returns the value of the vector `TE` of arrival times of detected spikes. The arrival times are given in samples and can be converted to true times if the sampling rate is known. This function has several input arguments and they will be explained in Section 2. Remark: if you were using the previous version of this function (previously posted at http://robotics.caltech.edu/ zoran/Research/detection.html), there are slight changes. Firstly, the set of relevant scales is now estimated more precisely, by means of a subroutine within `detect_spikes_wavelet.m`. Secondly, the number of relevant scales is now passed to the function as an input argument. Type `help detect_spikes_wavelet` in MATLAB^{TM}command prompt to learn more about this function.

`get_score.m` is an auxiliary function that is useful for comparing the true arrival times (if they are known) and the estimated arrival times given by `TE`. This function is useful since it returns the number of correctly detected APs, as well as the number of omissions and false alarms.

`lag_ts.m` and `lead_ts.m` are the auxiliary functions that are used by `get_score.m`.

2 Example

The use of the above functions will be demonstrated with synthetic data provided above. After downloading the data files `clean.mat` and `corrupted.mat` in appropriate directory, these data can be loaded in MATLAB^{TM}workspace:

```
>> load clean.mat; who

Your variables are:

Signal
```

Your workspace now contains a variable `Signal`, which is a row vector of "neural data" to be analyzed. This data contains 10 equally spaced APs (transients). The attribute "clean" indicates that the data does not contain any noise. You can visualize the signal by:

```
>> plot(Signal)
>> set(gca,'XLim',[1 length(Signal)],'YLim',[-3 3])
```

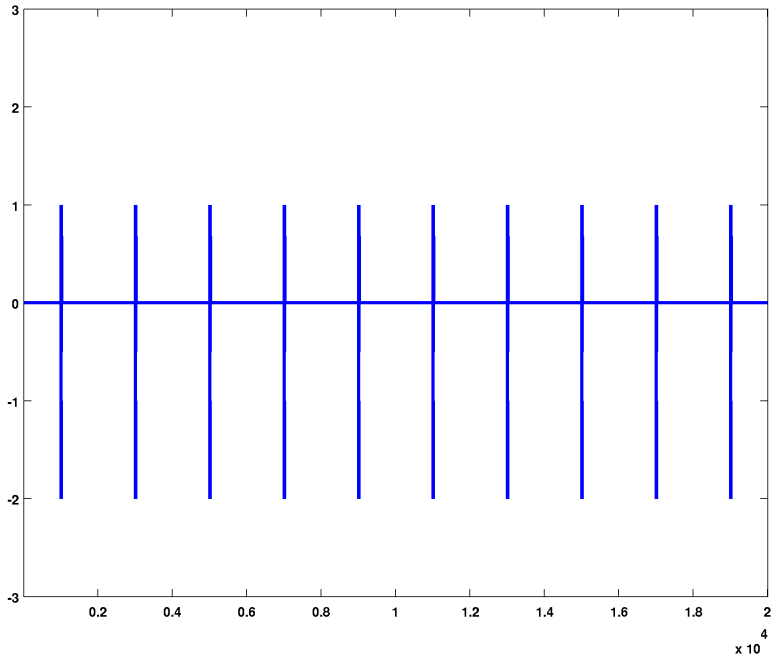

Figure 1: Signal given in `clean.mat` plotted against time [samples].

The result is shown in Fig. 1. The sampling rate of the synthetic signal is 20 [kHz], and so Fig. 1 corresponds to 1 [s] of data. A close inspection of individual APs shows that they consist of two sinusoid semi-cycles of different amplitudes. The duration of the APs is set to 0.6 [msec]. The detection of APs in `Signal` is not very challenging, but reveals the true arrival times of APs:

```
>> TT = detect_spikes_wavelet(Signal,20,[0.5 1.0],6,'l',0.1,'bior1.5',1,1)
wavelet family: bior1.5
10 spikes found
elapsed time: 0.26422

TT =

  Columns 1 through 6

       1004        3004        5004        7004        9004       11004

  Columns 7 through 10

      13004       15004       17004       19004
```

The last two arguments in the function are plot and comment flags. To suppress comments and plots, you should pass a number different from 1, for example:

```
>> TT = detect_spikes_wavelet(Signal,20,[0.5 1.0],6,'l',0.1,'bior1.5',0,1);
```

produces comments but not plots, and

```
>> TT = detect_spikes_wavelet(Signal,20,[0.5 1.0],6,'l',0.1,'bior1.5',1,0);
```

produces plots but not comments. If you opt for plots, the script generates two figures: Fig. 2 and Fig. 3 (shown below).

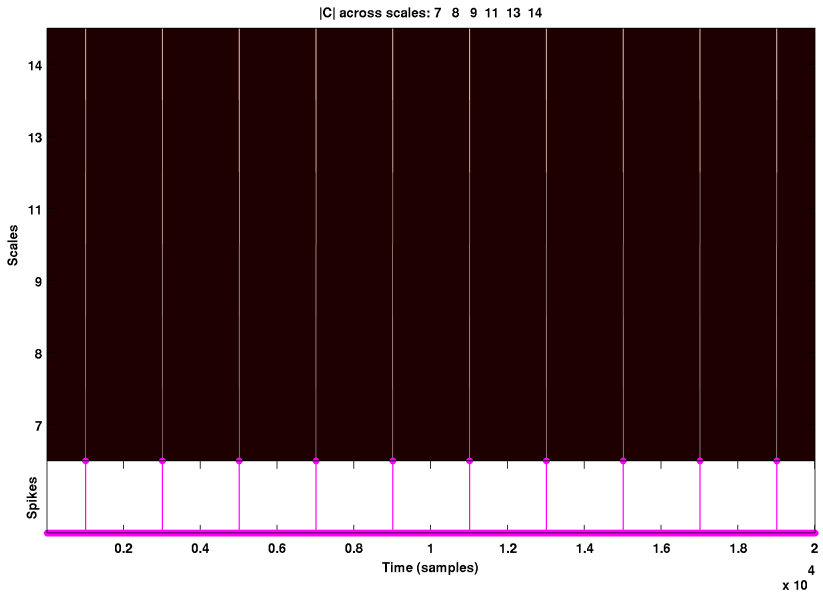

Figure 2: Time-scale representation of the signal in `clean.mat`. The presence of strong coefficients at relevant scales coincides with the arrival times of APs.

Fig. 2 shows the time-scale representation of the signal, where the absolute values of wavelet coefficients are color coded. At the bottom of the same plot are the locations (arrival times) of the APs. By zooming in on this figure, it is obvious that the APs correspond to "large" coefficients across various scales.

Fig. 3 shows the wavelet coefficients at different scales that cross the detection threshold. The coefficients at various scales are color-coded, e.g red, blue, green etc. The coefficients that do not cross the detection threshold are shrunk to 0.

The 1st argument of `detect_spikes_wavelet.m` is the signal itself. The 2nd argument is the sampling rate of `Signal` in [kHz]. The 3rd argument is a vector that specifies the minimal and maximal duration in [ms] of APs to be detected. The 4th argument is the number of scales used in detection. In this case, 6 scales were used and they are roughly matched to APs lasting

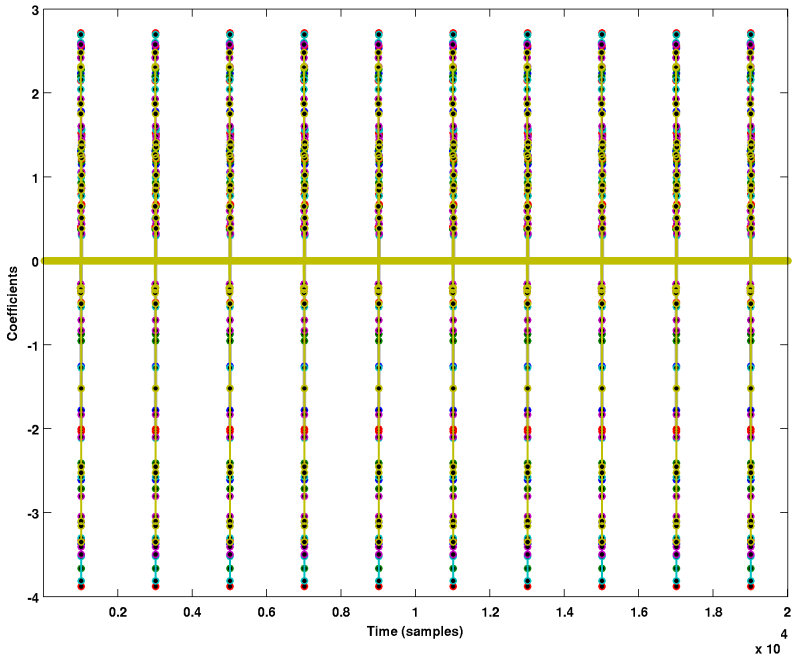

Figure 3: The coefficients of the signal in `clean.mat` at relevant scales.

$\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ [ms]. The 5th argument is an option `l` or `c`, referring to "liberal" and "conservative" mode of operation (see [2]). These two modes will produce similar results in most cases. Differences will arise in detection under extremely noisy conditions, where in the conservative mode the function will typically not return any detected APs, while in the liberal mode the function will most likely find APs of very small amplitude. The 6th argument is a sensitivity parameter, whose value typically ranges between -0.2 and 0.2. This parameter trades-off the cost of false alarm and omission. The higher the value of the parameter, the less likely the false alarms are and the more likely omissions are (see below). Finally, the 7th argument is the family of wavelet to use for detection. The wavelet from the family `bior1.5` is used in this example.

We now take the time series from above and corrupt it by adding a white noise, whose standard deviation determined from desired signal-to-noise ratio (SNR).

```
>> load corrupted.mat
>> plot(Signal)
>> set(gca,'XLim',[1 length(Signal)],'YLim',[-3 3])
```

By executing the code above, we can visualize the noisy neural data (see Fig. 4). Black arrows (plotted separately) indicate the arrival times of APs. Clearly, a simple detection based on amplitude thresholding could not detect low-amplitude APs (e.g. APs 1, 5 and 7) without accepting a number of false positives.

Then, we detect APs in the noisy data by

```
>> TE = detect_spikes_wavelet(Signal,20,[0.5 1.0],6,'l',0.1,'bior1.5',1,0);
```

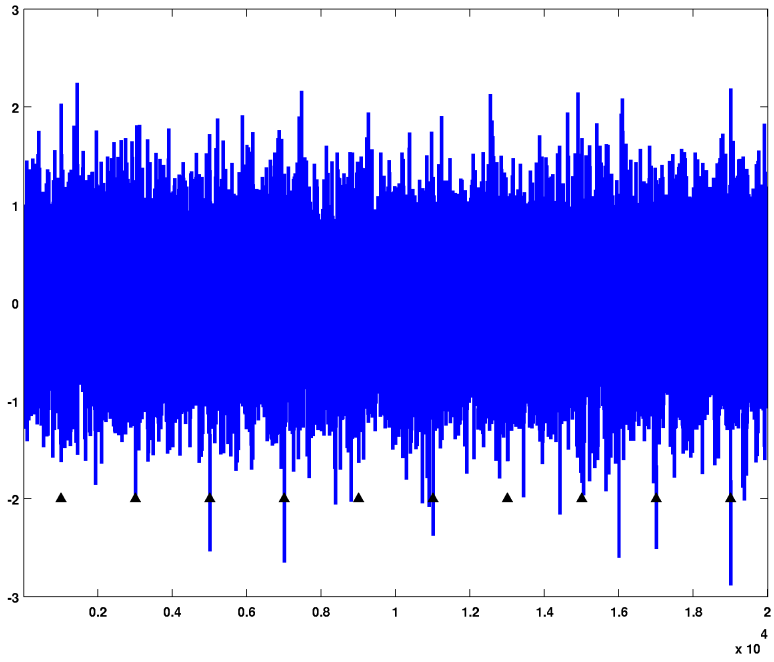

Figure 4: Signal given in `corrupted.mat` plotted against time [samples].

which results in Figs. 5 and 6:

As can be seen from the figure above, one of the spikes (spike 7) was not detected. This can be verified by running the function `get_score.m` which returns the number of correctly detected spikes, the number of omissions as well as the number of false positives. Indeed, running

```
>> get_score(TT,TE,20)

ans =

     9     1     0
```

indicates that there are 9 correctly detected spikes, 1 omission and 0 false positives. You can trade-off omissions and false positives by varying the parameter `L` (type `help detect_spikes_wavelet` to learn more about this parameter), i.e.

```
>> TE = detect_spikes_wavelet(Signal,20,[0.5 1.0],6,'l',0.05,'bior1.5',0,0);
>> get_score(TT,TE,20)

ans =

    10     0     3
```

indicating that all APs have been detected (no omissions). The paid price, however, is in the 3 false positives. You can now play with different choice of wavelet functions, such as `'bior1.3' 'db2'`

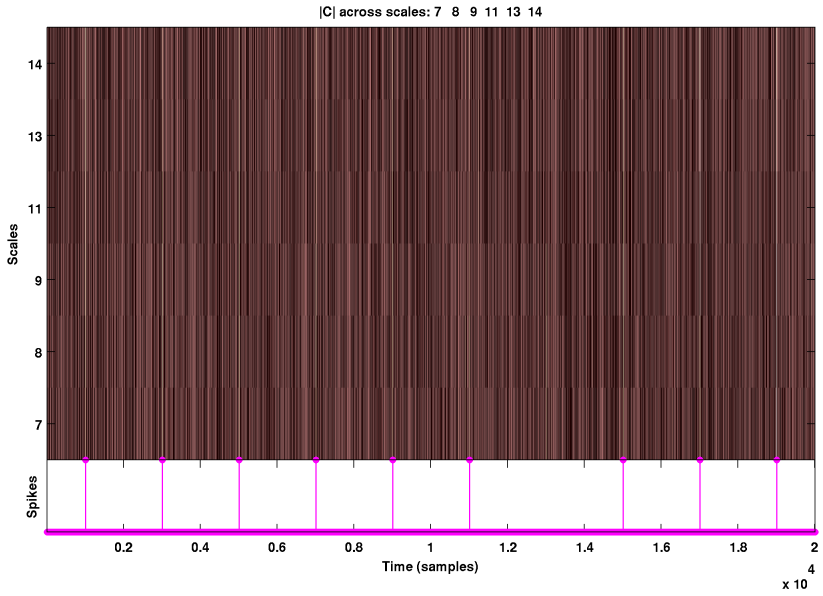

Figure 5: Time-scale representation of the signal in `corrupted.mat`. The presence of strong coefficients at relevant scales coincides with the arrival times of APs.

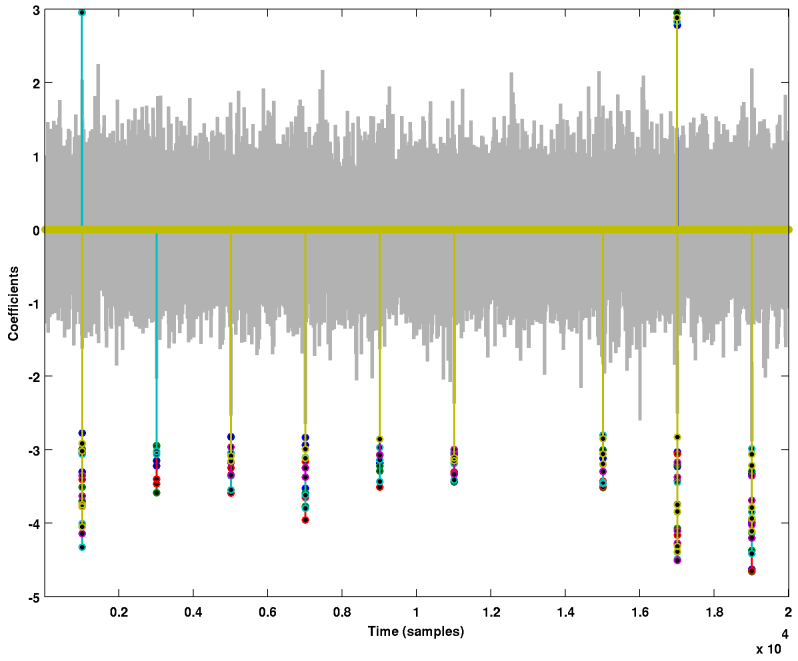

Figure 6: The coefficients of the signal in `corrupted.mat` at relevant scales.

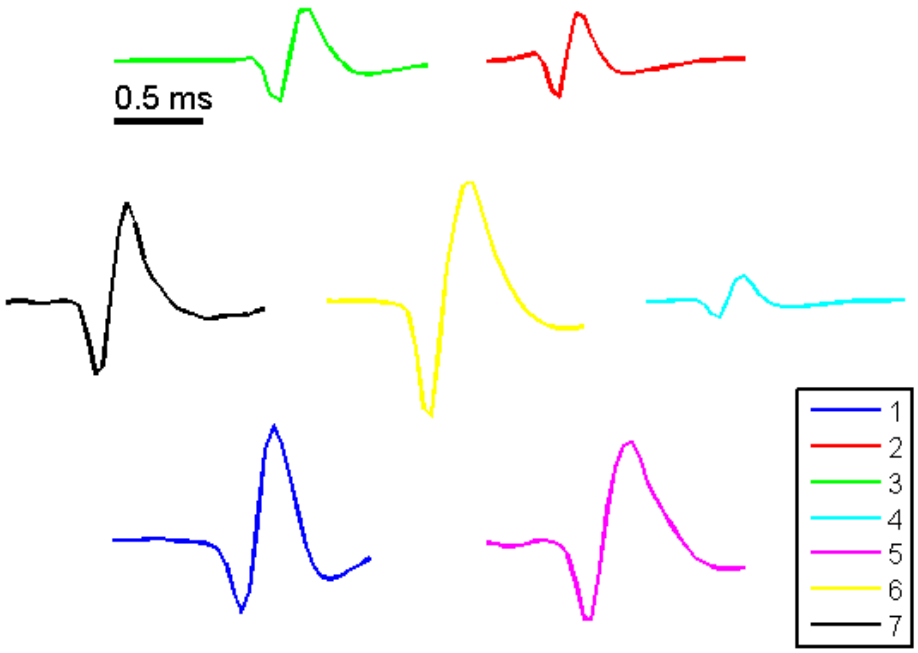

Figure 7: AP templates [3] constructed from experimentally recorded data.

`'haar'` etc., and compare their performances.

3 Downloadable Spike Templates (added May 1, 2015)

In case you are interested in generating your own data for testing, this section provides a data file with 7 AP templates (see Fig. 7). These templates are constructed from experimentally recorded data. They were previously used in [3], which also provides the description of the recording procedure. The file

(1) `templates.mat`

contains the scalar variable `SR` that represents the sampling rate (Hz) and the matrix `Templates`. Should you end up using these data, please make sure you refer to [3].

References

[1] S. M. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1998.

[2] **Z. Nenadic** and J.W. Burdick, Spike detection using the continuous wavelet transform, *IEEE T. Bio-med. Eng.*, vol. 52, pp. 74-87, 2005.

[3] R. Benitez and **Z. Nenadic**, Robust unsupervised detection of action potentials with probabilistic models, *IEEE T. Bio-med. Eng.*, vol. 55(4), pp. 1344-1354, 2008.