EE-125 Project: Continuous Wavelet Transform (CWT)

Brian Tracey (Corrections in RED, **12/14 clarifications in GREEN**)

December 14, 2017

1 Goals and background

After finishing this project, you should:

- Understand how wavelet shapes and the CWT are computed
- Understand how wavelets of different scales can be interpreted as filter banks
- Understand how the CWT compares to the spectrogram
- Get some insight into how the wavelet shape can affect our results

In this project, we'll be focusing on two wavelet types - Morlet and the 'Mexican hat' or Sombrero wavelet. We'll start off with a brief description of those, and then give the equations for the CWT itself.

1.1 Morlet wavelet

The Morlet wavelet can be either complex-valued or real-valued. The complex version is defined as a complex exponential multiplied by a Gaussian window:

$$\Psi_M(t) = e^{j2\pi f_0 t} e^{-t^2/(2\sigma^2)} \tag{1}$$

where f_0 gives the frequency of the exponential, and σ controls the width of the Gaussian. In the EEG literature, where this wavelet is popular, it's common[1] to set $\sigma = nCycl/(2\pi f_0)$, where $nCycl$ is a number of cycles. If we set $nCycl = 3$, for example, the Gaussian will decay in about three cycles of a sinusoid at f_0. You will verify this below.

Eq. 1 defines the 'mother wavelet', which will be scaled and shifted. For a scaling value s and time shift of 0, we have

$$\Psi_M(t/s) = e^{j2\pi f_0 t/s} e^{-t^2/(2s^2\sigma^2)} \tag{2}$$

Looking in the exponential term, you can see this gives an effective frequency of $f = f_0/s$. A major advantage of the Morlet wavelet is that it is easy to map the scale factor s to frequency f in Hz. For other wavelets, it can be harder to interpret the scale factor s.

The real-valued Morlet is found by just taking the real part of Eq. 1 or Eq. 2, so it is a cosine multiplied by a Gaussian window.

[1] for example, see http://mikexcohen.com/lectures.html

1.2 Sombrero (also Mexican hat or Ricker) wavelet

The mother wavelet for the Sombrero or 'Mexican hat' wavelet is the 2nd derivative of a Gaussian shape, and can be defined as

$$\Psi_S(t) = (1 - t^2)e^{-t^2/2} \tag{3}$$

To scale it by a factor s, we just substitute t/s for t above.

It is much harder to define the frequency of this type of wavelet (this is true for most kinds of wavelets, as they are most useful when the signals do not look like sinusoids). However, we can take a Fourier transform of any signal. It turns out that the Fourier transform of the unscaled $\Psi_S(t)$ has its maximum amplitude at roughly $f1 \approx 0.2237$ Hz. Then, if we scale the time by $1/s$, we will scale the maximum frequency to roughly $(0.2237/s)$ Hz.

1.3 Computing the CWT

The CWT is defined as

$$C(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\Psi^*(\frac{t-\tau}{s})dt \tag{4}$$

where Ψ is the mother wavelet, τ is the time shift, s is the scale factor, and $*$ denotes the complex conjugate.

If we digitize the signal and the wavelet shape, sampling at time intervals T, then Eq. 7 can be written using a Reimann sum as

$$C(n, s) = \frac{T}{\sqrt{|s|}} \sum_m x(m)\Psi^*(\frac{m-n}{s}) \tag{5}$$

which looks like a convolution. To make this point clearer, we can define a filter for scale s as

$$h_s(n) \equiv \frac{T}{\sqrt{|s|}}\Psi^*(\frac{n}{s}). \tag{6}$$

which lets us write

$$C(n, s) = x(n) * h_s(n) \tag{7}$$

where '*' is convolution. Assuming we compute Ψ over a finite number time interval, $h_s(n)$ will be an FIR filter. If we use the complex Morlet transform, $h_s(n)$ will be a complex-valued FIR filter - something we haven't used in the class yet, but convolution still works the same as for a real-valued filter.

2 Wavelet shapes, and wavelets as filter banks

In this section, you will code up the wavelet shapes defined above and look at their frequency responses. This project has a lot of plots, so I tried to number them in the discussion to avoid confusion (plot numbers are 2.x, or 3.x for the next section). Sentences starting with 'Discuss' mark discussions that I will look for during grading.

2.1 Complex and real Morlet wavelets

Write a small function called `makeMorlet` that implements Eq 1 and computes σ as $\sigma = nCycl/(2\pi f_0)$. A *suggested* function definition is:

```
function psi = makeMorlet(t,s,f0,ncycles)
```

To test this, set up a time vector t that goes from -2 sec to 2 sec with a sampling rate of 200 Hz. For starting values, set $f0 = 5$ Hz and `ncycles`=8. Compute the 'mother wavelet' this gives you, and plot both the real and imaginary parts of Ψ_M vs time (overlaid on the same plot - plot 2.1).

Take the Fourier transform of the mother wavelet, Ψ_M, and plot the magnitude (linear scale) vs frequency in Hz (see note here[2]). Does the spectrum peak around 5 Hz, as expected? Now, do the same thing but take the DFT of the real part of the mother wavelet (i.e., use the real-valued Morlet wavelet). **When you do so, be sure to plot the spectrum for both negative and positive frequencies.** Include both plots in your report (plots 2.2 and 2.3). Discuss why you see a difference (hint: review the frequency-shift property from Table 4.5 of the textbook).

From the discussion above, figure out scale factors s that should create wavelets with center frequencies of 5 Hz, 30 Hz, and 60 Hz. Use your code to compute the scaled wavelets for each s. Make a plot showing the real part of the three scaled wavelets, **vs. time** (either as subplots, or overlaid - plot 2.4. **If you prefer, you can plot both real and imaginary parts; plotting only the real part is just a way to simplify the plot.**). Make another plot that shows the magnitude (linear) of the Fourier transform of each **(complex-valued)** wavelet, plotted vs. frequency in Hz. The three lines should be shown on the same plot (plot 2.5). What you should see is something that looks like three bandpass filters centered around 5, 30 and 60 Hz, but with a passband width that changes with frequency.

The CWT, Eq. 7, includes a normalization factor of $1/\sqrt{|s|}$. Redo plot 2.5, but apply this scaling to the three curves (plot 2.6). Discuss what this scaling accomplishes. **CHANGE: Don't worry much about this discussion - the project would need more detail added to allow you to give a detailed discussion. A very basic description of what you see is fine.**

Create three wavelets **with the same parameters as above** , except with `ncycles`=16. Make a plot of the wavelets in time, similar to plot 2.4 (this is plot 2.7). Make a scaled frequency plot, similar to 2.6 (this is plot 2.8). Discuss how the number of cycles affects the wavelet shapes in time, and how it changes the frequency response of the filters.

When we compute the CWT, we'll want wavelets at many scales. Reset `ncycles` to 8. Set up a vector of 101 evenly-spaced s values that will scale the frequency from 5 Hz up to 100 Hz. Loop over this vector, compute the complex Morlet wavelet at each scale, and store the values in a matrix `psiM`. To avoid saving a bunch of zero values, compute this matrix for a smaller range of times t, from -0.5 sec to +0.5 sec (giving L total samples in time). You can check your results by making 2D plots of the real and imaginary parts (using say, `surf` or `imagesc`), but that is optional - you don't need to include it in your report.

2.2 Sombrero / Mexican hat wavelet

Write a small function called `makeSombrero` which implements Eq. 3.To test this, set up a time vector t that goes from -3 sec to 3 sec with a sampling rate of 200 Hz (note this is a longer time vector than you used before) and plot the resulting mother wavelet (plot 2.9).

As mentioned above, interpreting the scale factor for this wavelet (or most wavelets) in terms of 'frequency' can be tricky. However, let's try anyway. Using the discussion above, calculate s values that should scale the Sombrero's response to 5, 30 and 60 Hz. Plot the three resulting scaled wavelets in time, as you did in plot 2.4 above (this is now plot 2.10). Show their scaled frequency responses, as you did in plot 2.6 above (this is plot 2.11). Discuss the time resolution vs. frequency resolution of the two wavelets, and when you might use each.

As before, when we compute the CWT, we'll want wavelets at many scales. Set up a vector of 101 linearly spaced s values that will scale the wavelet from 5 Hz up to 100 Hz. Compute a matrix `psiS` in the same way you did above for the matrix `psiM` (with time going from -0.5 to 0.5 sec). As a note, picking your scalings to match particular frequencies is a little unusual for wavelets (other than Morlet), but it will be helpful in this project when you compare results between the two wavelet types.

[2] even though you should be very clear by now on how to do this, I'm included the file `ctft.m`, which you are free to use. Read it first to make sure you understand it!

3 Computing and plotting the CWT - toneburst data

Now, we are ready to actually compute the CWT. You are given the shell of a function `myCWT.m`. Fill out the code so it implements the approach discussed in Section 1.3. As a practical note, when you do the convolution at each scale, you'll end up with an output that is much longer the input data. We'll discard the start and end of this output, to give a convolution output which is the same length as the original data (so if the wavelet length L is an even number, we'll discard the first $L/2$ and last $L/2$ samples).

3.1 CWT using Morlet transform

Now we'll use the code you just wrote. First, run the file `demoData.m`, which will generate a test signal sampled at $Fs = 200$ Hz (conveniently matching the wavelets you generated). Take a look at the code, and plot the signal in time - you'll see that there are two lower-frequency tones, and two later high-frequency bursts (you don't need to include this plot in your report). Note the duration of the two tonebursts.

Before using CWT, try forming spectrograms of this signal. A suggestion is to start with a Hanning window of, say 64 samples, and with a large amount of overlap. Make a spectrogram with settings that lets you cleanly resolve the two low-freqeuncy tones. (plot 3.1). Make another spectrogram with settings that let you do a better job of estimating duration of the tonebursts (plot 3.2). In both cases, adjust the colorbar range so it covers from roughly the maximum down to 30 dB below the maximum (the `caxis` command should help). Discuss the tradeoffs involved in these settings, and any shortcomings of the spectrogram that you see.

Now, process this signal with your CWT function `myCWT`, for the complex-valued Morlet transform. Because the output is complex, you'll want to take the absolute value to get magnitude. Normalize the output by dividing it by the maximum magnitude, and plot the results in dB on a range of -30 dB to 0 dB (plot 3.3) Label the axes as time vs. scale (of course, you should label all axes in all plots!).

Congratulations - you just made a 'scalogram'! This is the wavelet version of a spectrogram. Be warned - interpreting signals in terms of scale is not always easy. Discuss whether or not you can see evidence of the tones and tonebursts in your scalogram.

In the scalogram above, you had evenly-spaced values in the scale factor, which is typical for wavelet analysis. However, a main advantage of the Morlet wavelet is that we can easily relate scale to frequency. For your vector of s values defined above, compute the corresponding frequencies, and plot scale vs. frequency (plot 3.4). **This will be a line plot.** Clearly, an even spacing in scale gives a non-linear scaling in frequency.

For plotting the results, it would be much better to have an even spacing in frequency. Set up a vector of frequencies at 101 points from 5 Hz up to 100 Hz, and use those to compute a new s vector and a new matrix of wavelet shapes - call this `psiM2`. Use `psiM2` to get CWT outputs, and plot the result (normalized and in dB, 30 dB range, as before), vs time and frequency (plot 3.5). Discuss how this result compares to the spectrogram and scalogram results you got before.

Now, repeat the last step, but just for the real-valued Morlet; this is easily done as you can say `psiMReal = real(psiM2)`. Make the new plot (plot 3.6). Discuss the differences between the real-valued and complex results. You might be interested to plot the results from `imag(psiM2)`, but that's optional.

3.2 CWT using Mexican hat transform

Run your code and make a scalogram using `psiS`, and make a plot similar to plot 3.3 - this one will be plot 3.7. Discuss the result - can you see the tonebursts here? can you explain why the low-frequency tones look the way they do?

4 Computing and plotting the CWT - ECG data

Now, we'll compare the two wavelets on a very different kind of data. Load the (very short) ECG signal provided with the project, and take a look at it. ECG signals clearly have a lot of important time-domain information (the very large spike, called the 'R' wave, is the electrical activity generated when the larger half of your heart contracts to push blood out into the body). It's less obvious that there is useful frequency-domain information, but it's still interesting to see how our various types of time-frequency analysis work on this data.

Make three plots (3.8-3.10) for this data, which show a spectrogram (using parameters you pick, but a 32 sample window is a good starting point), a complex Morlet CWT result (using `psiM2`, so you can interpret the result in terms of frequency), and a Mexican hat result. As above, all should be in dB on a 30 dB range. Discuss what you see - specficially, a) discuss how the time resolution changes vs. frequency in the spectrogram vs. the Morlet result - why is this? b) discuss which result gives you the best time-domain resolution, and why.

One of the ideas of wavelet processing is that it can be useful to pick wavelets that 'look like' the signals we are interested in. Let's briefly explore that. For the two wavelet results for the ECG, try to estimate which scale gives you the highest output on the R-wave peak (this will be easier if you change to plot axis to, say -5 dB up to 0 dB). Plot the two wavelets for those scales (these will be columns taken from `psiM2` and `psiS`), and also plot the original ECG signal. **Thus, these should be three line plots - one lineplot for the Morlet wavelet at the scale which gave the highest response, one lineplot of the Sombrero at the scale which gave its highest response, and the ECG signal.** These can all one on plot (plot 3.11). For easy comparison, **limit your ECG signal so you plot only a small portion with the same number of time samples as the two wavelets.** This should let you see at least one heartbeat in the data. Discuss any benefits of matching the wavelet shape to the expected data, based on this plot, and also on your earlier plots 3.5 and 3.7.

5 Happy holidays!

Congrats - you just finished your last EE-125 assignment. Have a good break!