Tutorial: Wavelet estimation for well ties

Evan Bianco[1]

You can find all the data and code to work through this tutorial for yourself at https://github.com/seg.

Wavelets are filters that keep us from seeing what we set out to see in seismic data: the earth's reflectivity. Through deconvolution, we try to remove the wavelet from the data. In tying wells, we make a wavelet interact with well-log measurements to create a synthetic representation of the seismic according to the borehole.

Such local seismic calibration can be done reliably only if two conditions are met: (1) the borehole path has been converted accurately to the seismic's traveltime domain, and (2) the well's reflectivity has been filtered using an appropriate wavelet. In *Well-tie calculus* (Bianco, 2014), I explored the computational aspects of creating a time-to-depth relationship from sonic logs. At the end of that tutorial, I made a cursory well tie using a zero-phase Ricker wavelet. A Ricker wavelet is a decent wavelet to start with. Compared to other wavelets, it has a relatively simple shape, and it is described by a simple equation where the only variable is the central frequency. The Ricker may serve as an initial filter for checking a well's time-to-depth relationship, but we likely will require more precision in our wavelet if a synthetic is to be useful for seismic interpretation.

There are several ways to estimate seismic wavelets (e.g., Brown et al., 1988). In this tutorial, we'll explore a few of the common approaches. We'll look at two ways you can estimate a wavelet when you only have seismic data: (1) fitting a band-pass filter to the frequency content of the data and (2) doing an autocorrelation. Then we'll look at wavelet estimation options when you have seismic and well-log data. For the sake of illustration, we'll use a portion of the Marmousi2 synthetic (Martin et al., 2006), because the earth model is known exactly, but the data still contains 2D wave propagation effects.

Spectral matching

If we assume the earth's reflectivity spectrum to be flat, then the shape of the spectrum can be attributed to the shape of the wavelet. Then we can analyze the frequency content of the seismic data and create a band-pass filter to match, even when there is no well on the survey.

To look at the frequency content of a single trace, we can take the fast Fourier transform, which yields a vector of complex-valued Fourier coefficients. To get an amplitude spectrum, we take the absolute value of the Fourier coefficients; to get the phase spectrum, we take the angle of the coefficients in the complex plane:

```
>>> import numpy as np
>>> amp_spec = np.abs(np.fft.rfft(s))
>>> phase_spec = np.angle(np.fft.rfft(s))
```

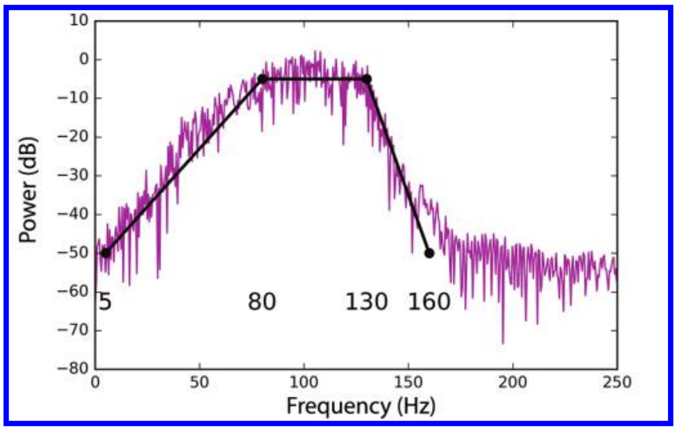

Figure 1: Building an Ormsby wavelet by fitting a band-pass filter (black) to the seismic data (pink) with frequencies 5, 80, 130, and 160 Hz shown.

To make a plot, we can return the frequency values corresponding to the Fourier coefficients,

```
>>> f = np.fft.rfftfreq(len(s), d=dt)
```

where `len(s)` is the length of the signal, `s`, and `dt` is the sample interval of the signal in seconds. To produce the power spectrum shown in Figure 1, we take the log of the spectrum and plot that versus frequency, `f`.

```
>>> power = 20 * np.log10(amp_spec)
>>> plt.plot(f, power)
```

To compare this spectrum with an analytic wavelet, I've selected four corner frequencies, f_1, f_2, f_3, f_4 = 5, 80, 130, 160 Hz, that coincide with the frequency content of the data reasonably well (Figure 1). The four frequencies define the Ormsby wavelet shown in Figure 2.

Autocorrelation

We'll continue with the assumption of a flat reflectivity spectrum. By doing an autocorrelation of the trace — that is, correlating the trace with itself — we get a zero-phase signal from the amplitude spectrum of the trace.

```
>>> np.correlate(s, s, mode='same')
```

The autocorrelation function forces all the cosines to zero phase, so we can think of the autocorrelation as a zero-phase version of our data's frequency content. We know that the data are likely not exactly zero-phase however, so let's move on to methods that preserve the phase content of the data.

[1]Agile Geoscience.

http://dx.doi.org/10.1190/tle35060541.1.

Spectral division

A simplistic representation of the seismic reflection experiment is that the observed seismic trace, $s(t)$, is approximated by convolving the wavelet, $w(t)$, with the earth's reflectivity, $r(t)$.

$$s(t) = r(t) * w(t). \quad (1)$$

One nice property of the Fourier transform is that convolution of two signals in the time domain is achieved by multiplying them in the frequency domain. Conversely, deconvolution in the time domain can be done by division in the frequency domain. Thus, we would expect that if the earth's reflectivity spectrum can be estimated, say from well logs, then we should be able to obtain our wavelet by spectral division (Brown et al., 1988):

$$W(f) = S(f) \,/\, R(f), \quad (2)$$

where $W(f)$ denotes the Fourier transform of the signal $w(t)$, and so on. Then we can take the inverse Fourier transform of the wavelet's spectrum to get the wavelet in the time domain. Unfortunately, this method relies on the basic assumptions of convolution and will fail in the presence of internal multiples, migration effects, system noise, cosmetic processing steps, and so on. In other words, it is unlikely to work well in real data.

Least-squares fit

An alternative is to solve for the wavelet by doing least-squares regression. We can write the convolutional model above in vector form as

$$s = \boldsymbol{R}w, \quad (3)$$

where s and w are vectors, and $\boldsymbol{R}$ is a Toeplitz matrix constructed from the reflectivity log. Toeplitz matrices have the property that each column is a shifted version of the one before it, so they provide another way to perform convolution. SciPy's `linalg` module has a function we can use for creating $\boldsymbol{R}$ from our reflectivity series:

```
>>> R = scipy.linalg.toeplitz(r)
```

Since $\boldsymbol{R}$ isn't necessarily invertible, we instead solve the system using least squares:

$$\min|\boldsymbol{R}w - s|^2. \quad (4)$$

```
>>> w = np.linalg.lstsq(R, s)[0]
```

This solution is often called the naïve solution, in that it just tries to fit the data. It happens to do this very well, because we haven't imposed any constraints. We've essentially found a wavelet that fits the data too well — both the signal and the noise. It doesn't have a short time duration, or "compact support," like a wavelet is supposed to have.

To fix this, we can bias the least-squares solution toward an answer we are looking for using regularization techniques. For instance, since we know wavelets have a short time duration, we can penalize solutions with many nonzero terms. Using Python's `scikit-learn` module, we can apply a Tikhonov regularization, also known as ridge regression or constrained linear inversion, which is just a least-squares regression with an L_2 penalty:

```
>>> clf = linear_model.Ridge(alpha=0.5, fit_intercept=False)
>>> clf.fit(R, s)
>>> w = clf.coef_
```

This wavelet, shown in Figure 3, has a compact time duration, and its spectrum matches the spectrum of the data, without all the noise. Is this wavelet any good? Well, the degree to which the synthetic matches the data can be calculated by using a correlation coefficient.

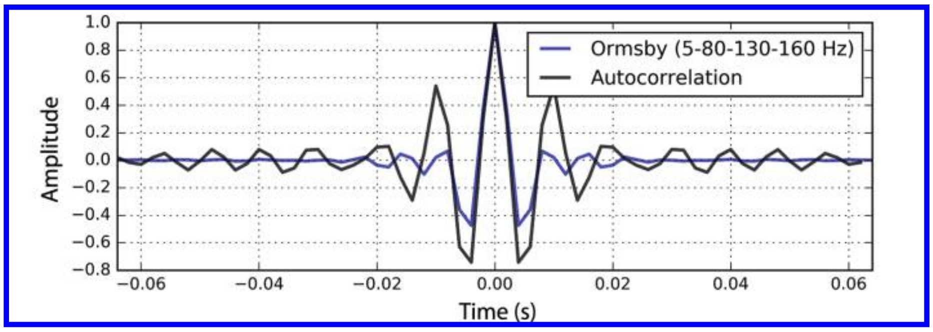

Figure 2: Wavelet estimation by autocorrelation, compared to Ormsby band-pass filter matching the spectrum of the data.

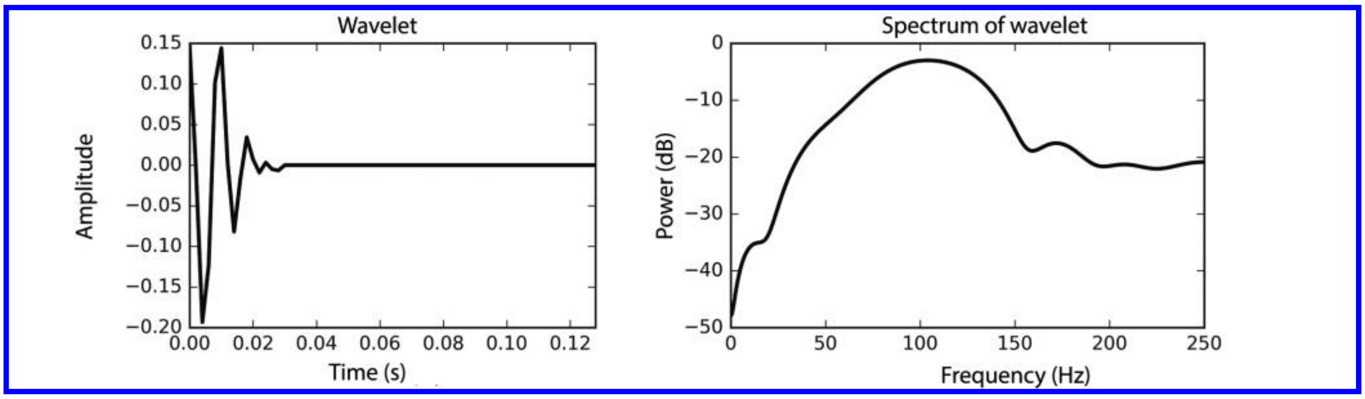

Figure 3: Time (left) and frequency (right) representation of wavelet.

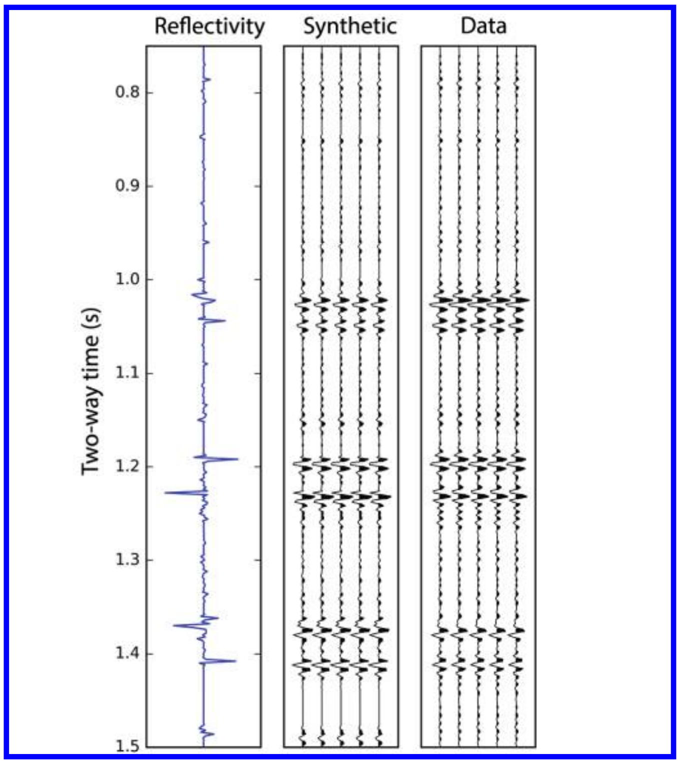

Figure 4: Comparison between synthetic seismogram and seismic data using the wavelet shown in Figure 3. Correlation coefficient = 0.82.

```
>>> numpy.corrcoef(s, synth)[0][1]
0.81788648323700142
```

A correlation coefficient of 0.82 is one way of making a quantitative statement about the quality of this tie (Figure 4). If we use the symmetric, zero-phased version of this wavelet, the correlation coefficient falls to 0.72.

We have reviewed several wavelet estimation methods. There are many others, and some are listed in the accompanying Jupyter notebook at http://github.com/seg. TLE

Acknowledgments

Thanks to my colleagues Matt Hall and Ben Bougher who provided many useful insights and clarifications on mathy bits.

Corresponding author: evan@agilegeoscience.com

References

Bianco, E., 2014, Geophysical tutorial: Well-tie calculus: The Leading Edge, **33**, no. 6, 674–677, http://dx.doi.org/10.1190/tle33060674.1.

Brown, R. L., W. McElhattan, and D. J. Santiago, 1988, Wavelet estimation: An interpretive approach: The Leading Edge, **7**, no. 12, 16–19, http://dx.doi.org/10.1190/1.1439470.

Martin, G., R. Wiley, and K. Marfurt, 2006, Marmousi2: An elastic upgrade for Marmousi: The Leading Edge, **25**, no. 2, 156–166, http://dx.doi.org/10.1190/1.2172306.