15 January 2009
10:00 am Ocean View Conference Room

MBARI Project 900907 Meeting
High Resolution Near-Surface Seismic Refraction Tomography

Attending: David Caress, Mark Chaffey, Paul McGill

Source Transducer
The source transducer KT106 has been ordered from Edgetech

- $13,125
- 1-6 kHz
- 6000 m rated

Receivers
Update from Bill Burgess at Acousonde:

- 3000 m design – not yet tested to 3000 m
- Model – B003A Acousonde
- Transducer response 16 Hz – 9 kHz
- Samples at 20 kHz
- Cost $13K
- 8 GB solid state storage
- 16 bit A/D
- Setting time – temperature compensated low power crystal good to 1 minute / year = = > 160 msec / day
- We have to set time and/or measure time offsets
- Time can be set according to preprogrammed acoustic or shock events
- Communication by infrared port
- Battery 3V lithium
- USB port accessible through battery compartment
- Will order 2

Modeling of Chirp Processing
Dave showed Paul and Mark some modeling of chirp processing done using Octave (rather than Matlab). The figures are inserted below, along with the m-file used to generate them. Octave is an open source alternative to Matlab.

Agenda Next Week
We will focus on defining the engineering requirements.

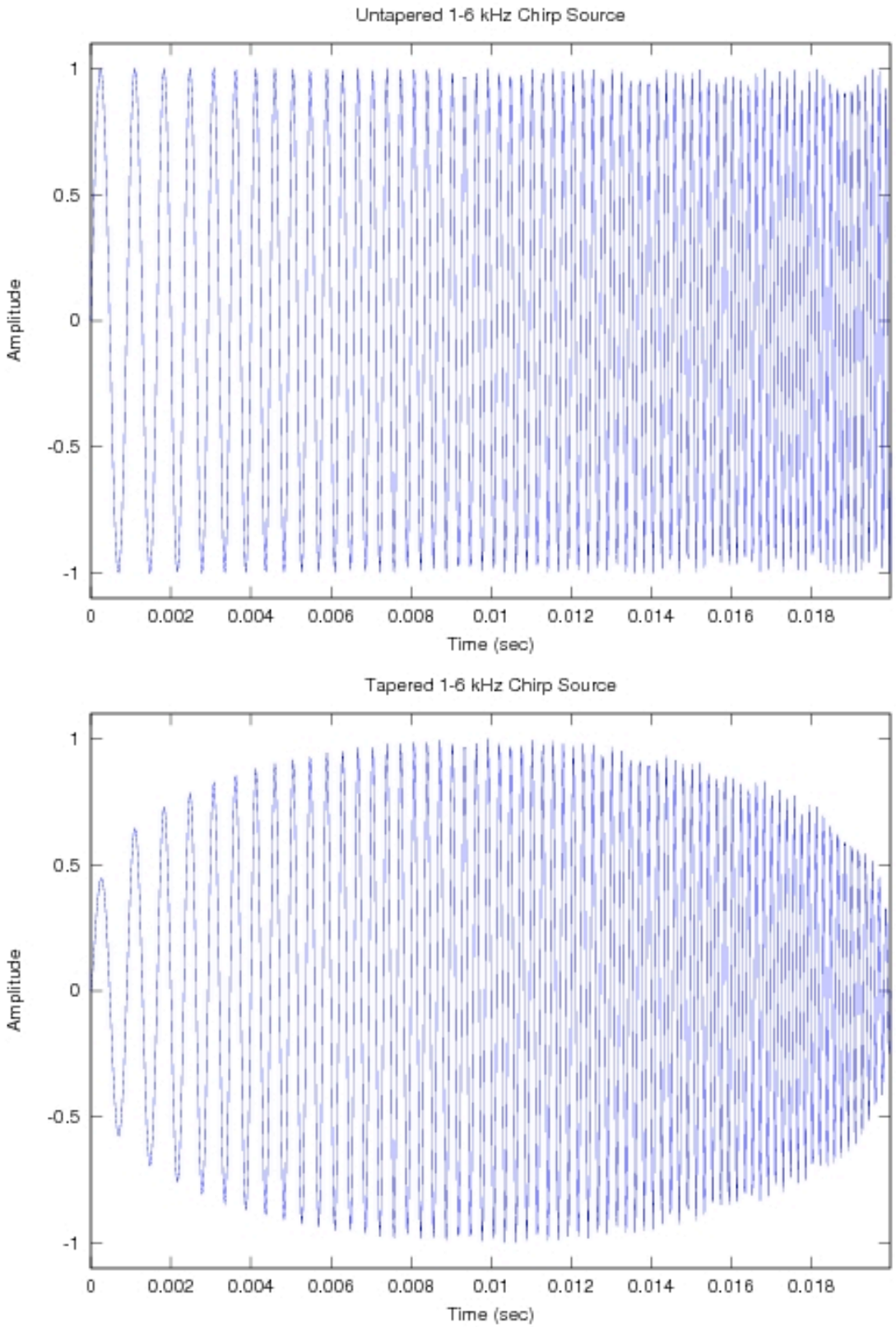

Figure 1. The upper figure shows a simple chirp sweep from 1 to 6 kHz over a 20 msec sweep. The lower figure shows the same chirp with a sqrt(cos) taper applied.

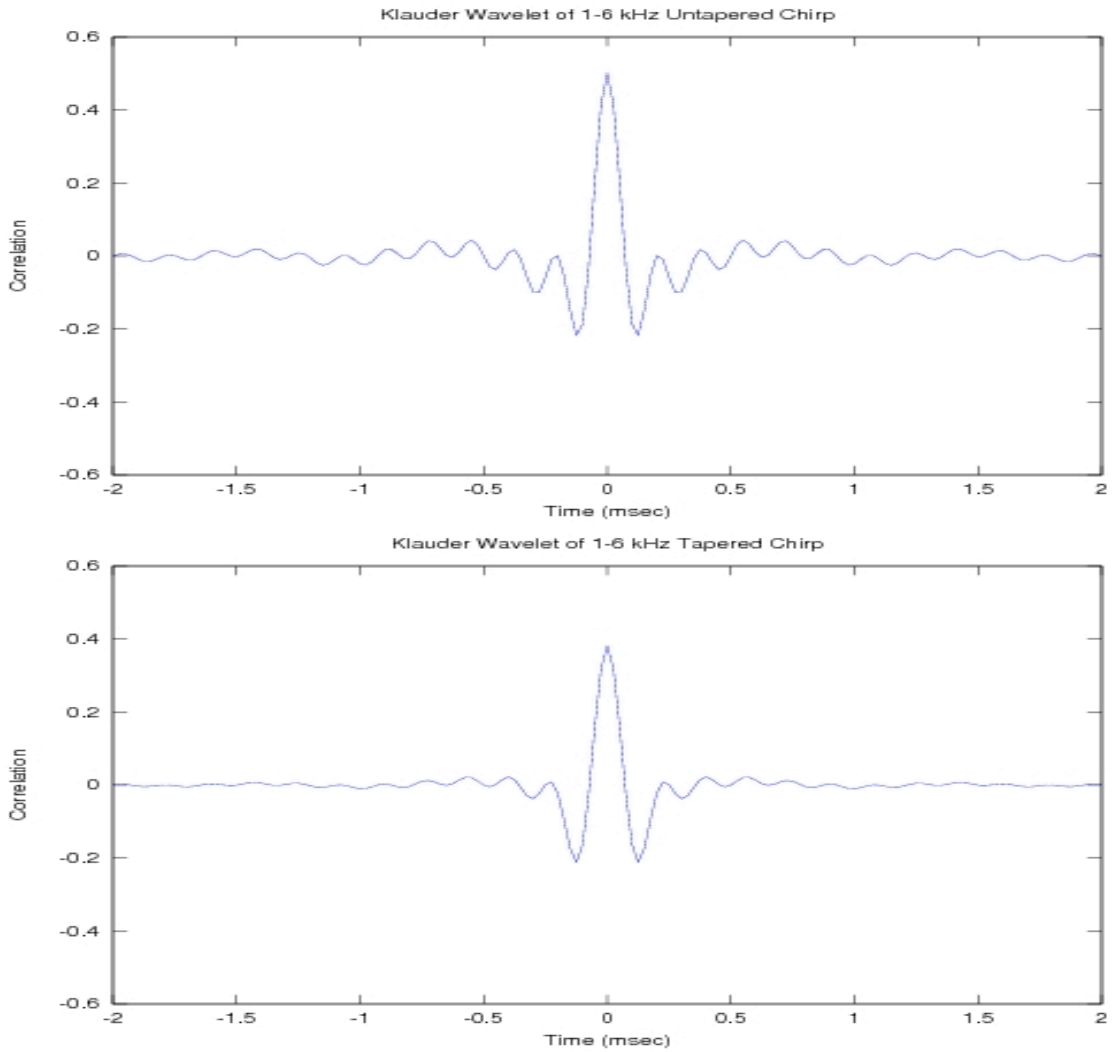

Figure 2. When match filtering (cross correlating) time series data with a source function, the highest possible correlation should correspond to the cross correlation of the source function with itself (autocorrelation), also called the Klauder Wavelet in radar, sonar, and exploration seismology applications. These plots show the Klauder Wavelets of the source functions from the first figure. The advantage of tapering the source is clear – the tapered source has a less "ringy" Klauder Wavelet. The width of the primary three lobes of the Klauder Wavelet is about 0.5 msec.

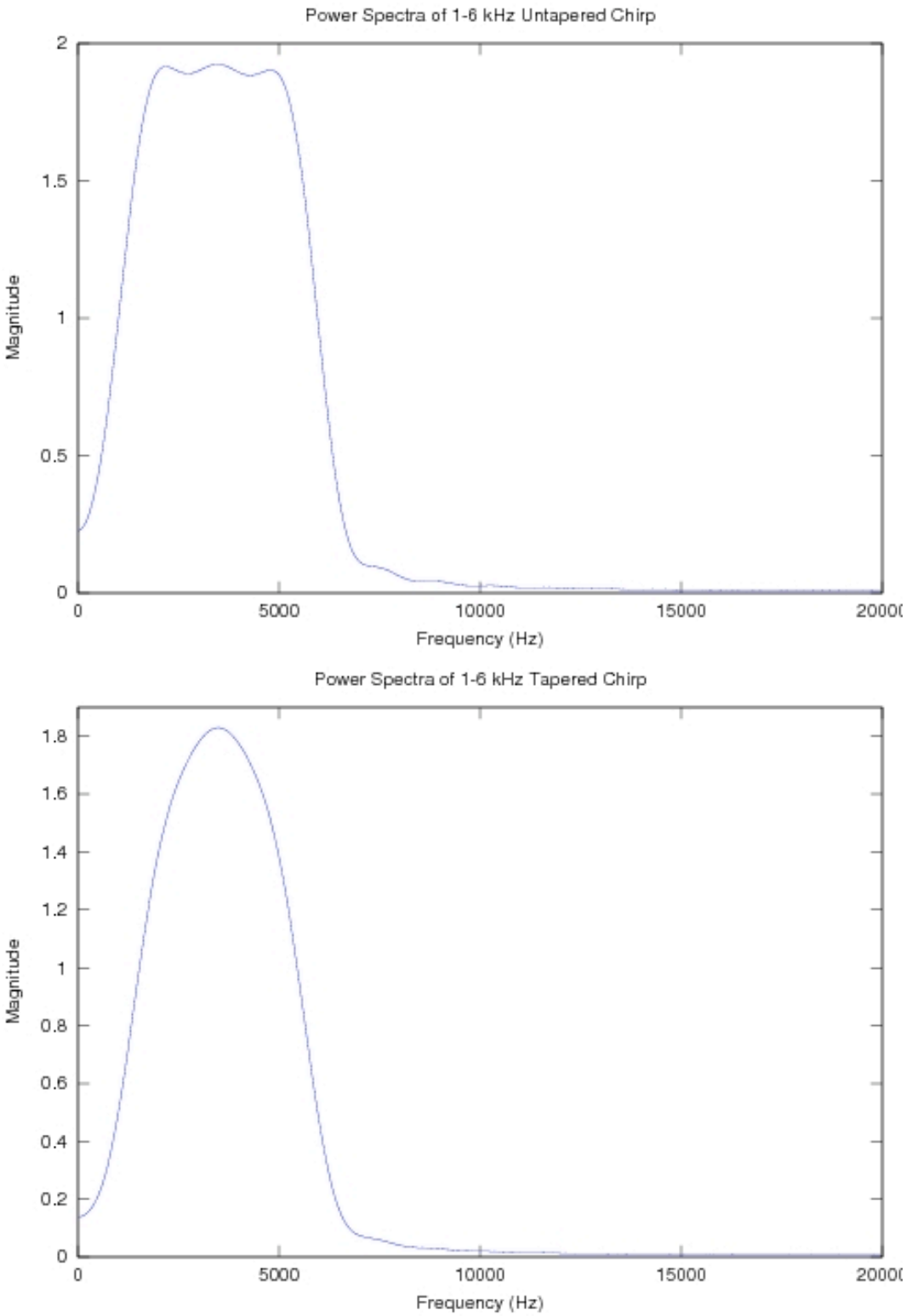

Figure 3. Power Spectra of the tapered (above) and untapered chirp sources. As expected, the tapering removes energy from the lowest and highest frequencies in the chirp.

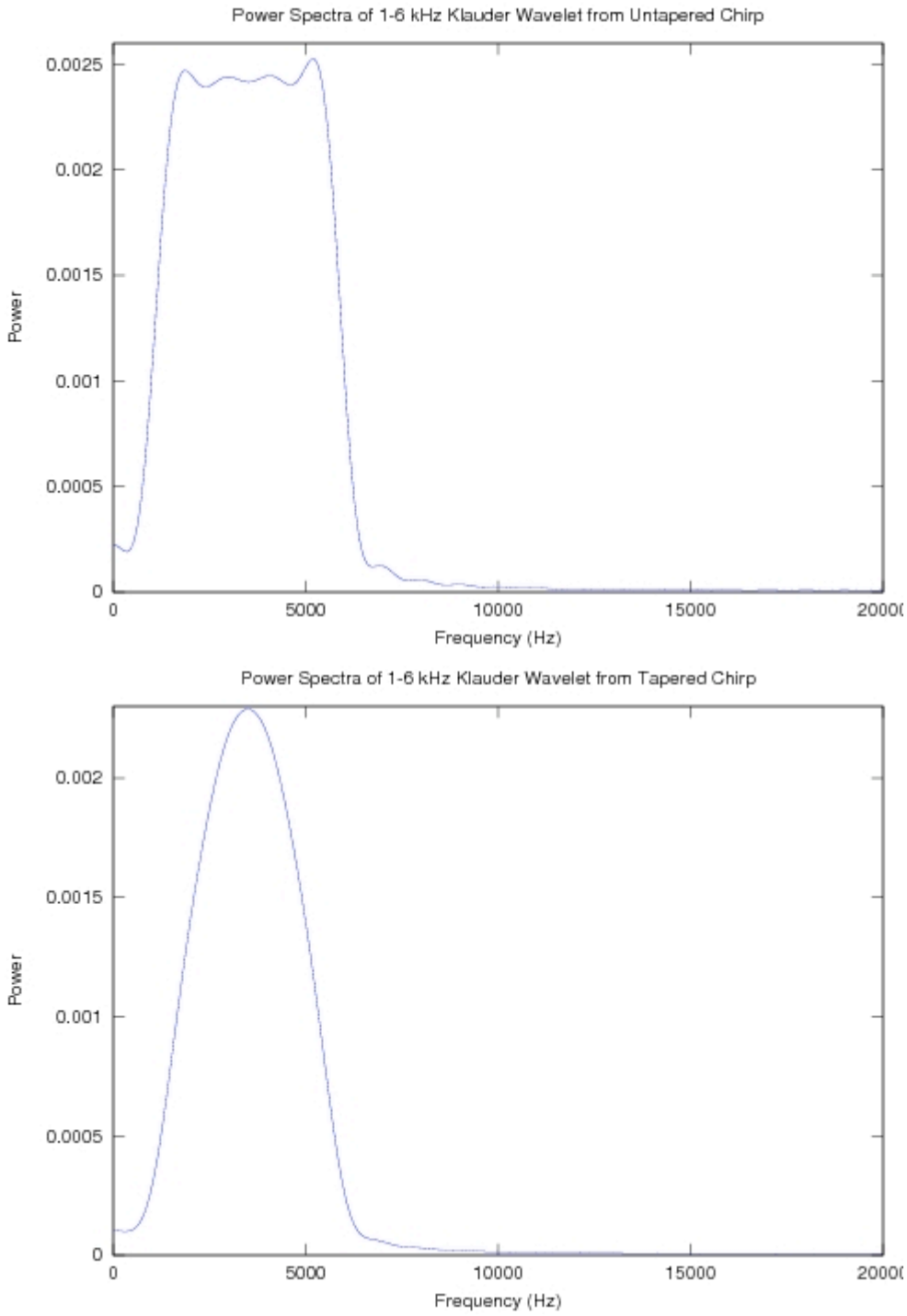

Figure 4. Power Spectra of the Klauder Wavelets of the tapered (above) and untapered (below) chirp sources. The frequency characteristics of the Klauder wavelets are very similar to those of the respective chirp sources.

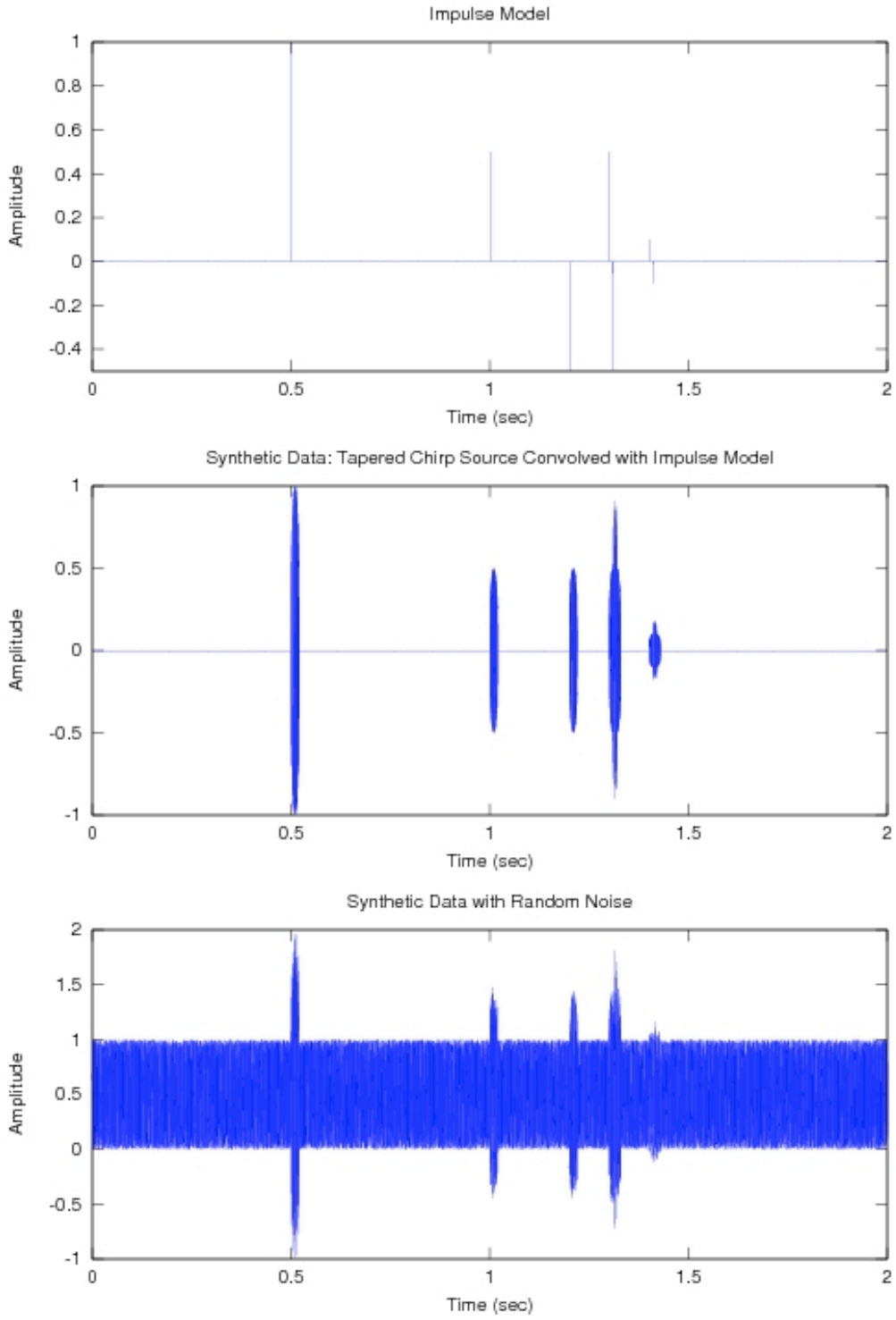

Figure 5. Constructing a synthetic chirp time series. (Top) Begin with a 2 second long time series punctuated with several impulses of both positive and negative polarity. (Middle) Convolve tapered chirp source of Figure 1 with the impulse time series. (Lower) Add random (white) noise.

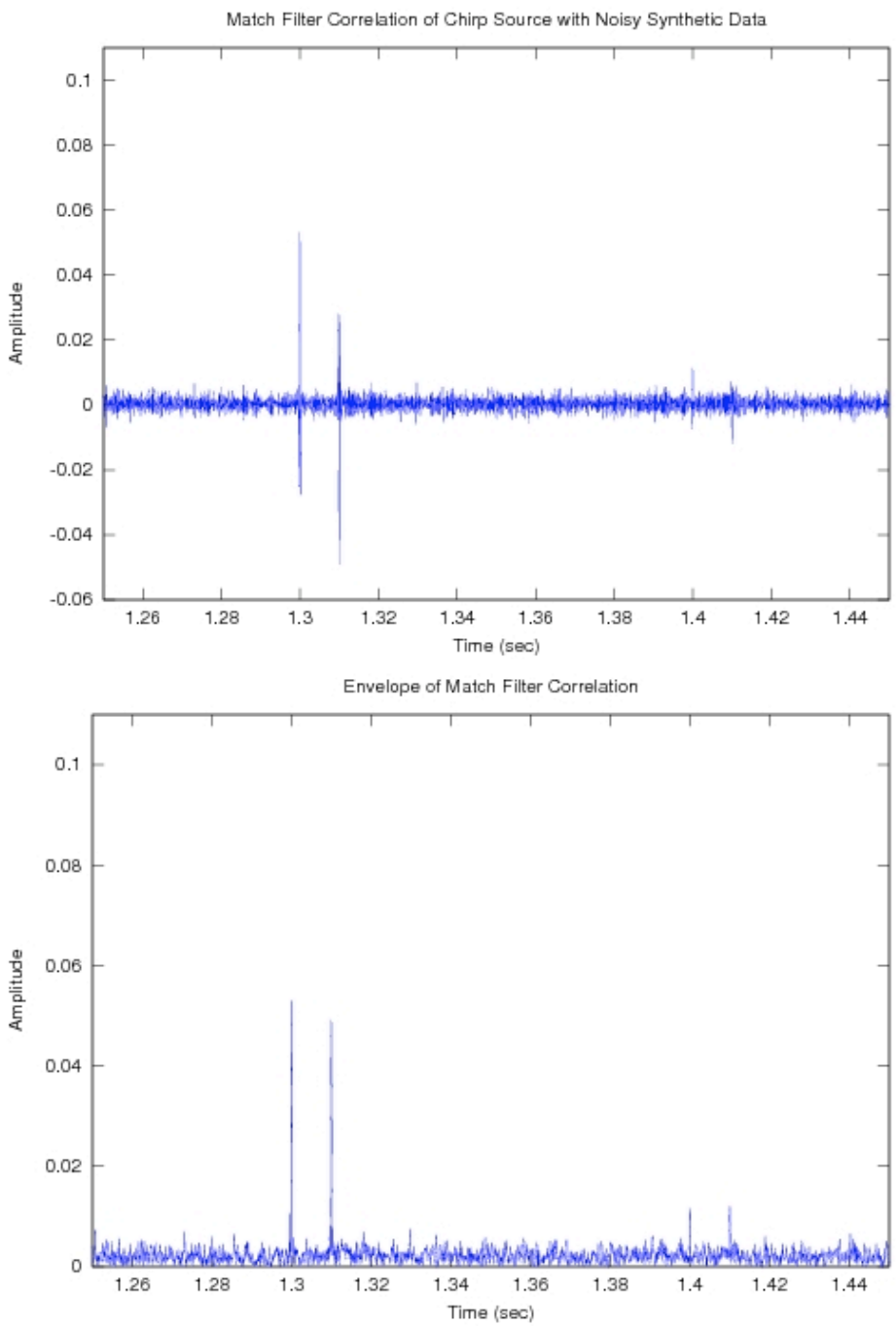

Figure 6. Chirp processing. (Top) This figure shows the results of match filtering the synthetic data for the tapered chirp source; the match filtering is simply the cross correlation of the source with the data. This resulting time series is also called the "correlate". (Lower) Envelope function commonly used as the result of chirp processing. This function is calculated by first constructing an analytic (complex) times series in which the real part is the correlate and the imaginary part the Hilbert transform of the correlate, and then taking the magnitude of the analytic series at every time slice. This is a positive-only function that contains no phase or polarity information.

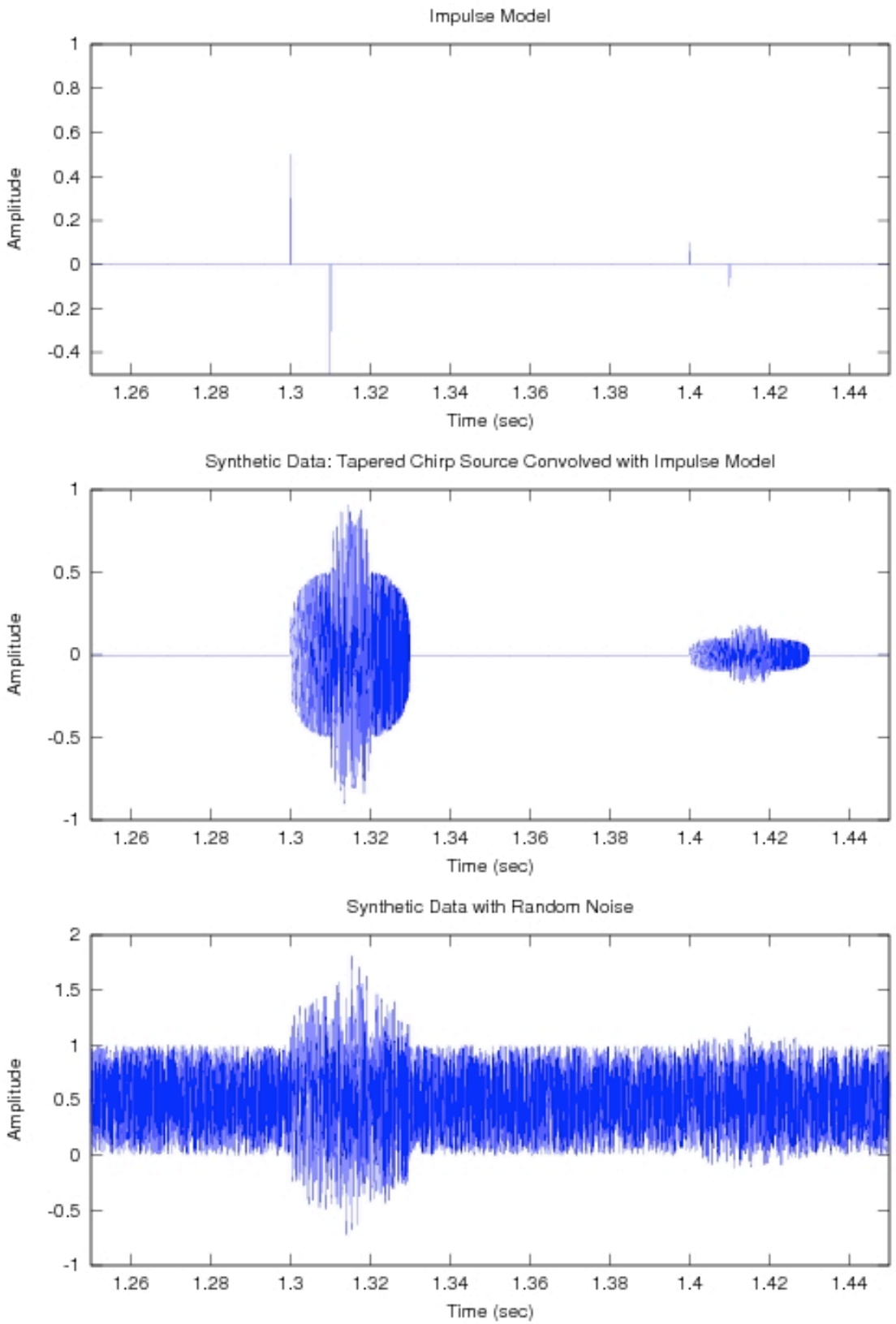

Figure 7. View of a 0.20 second section of the synthetic data.

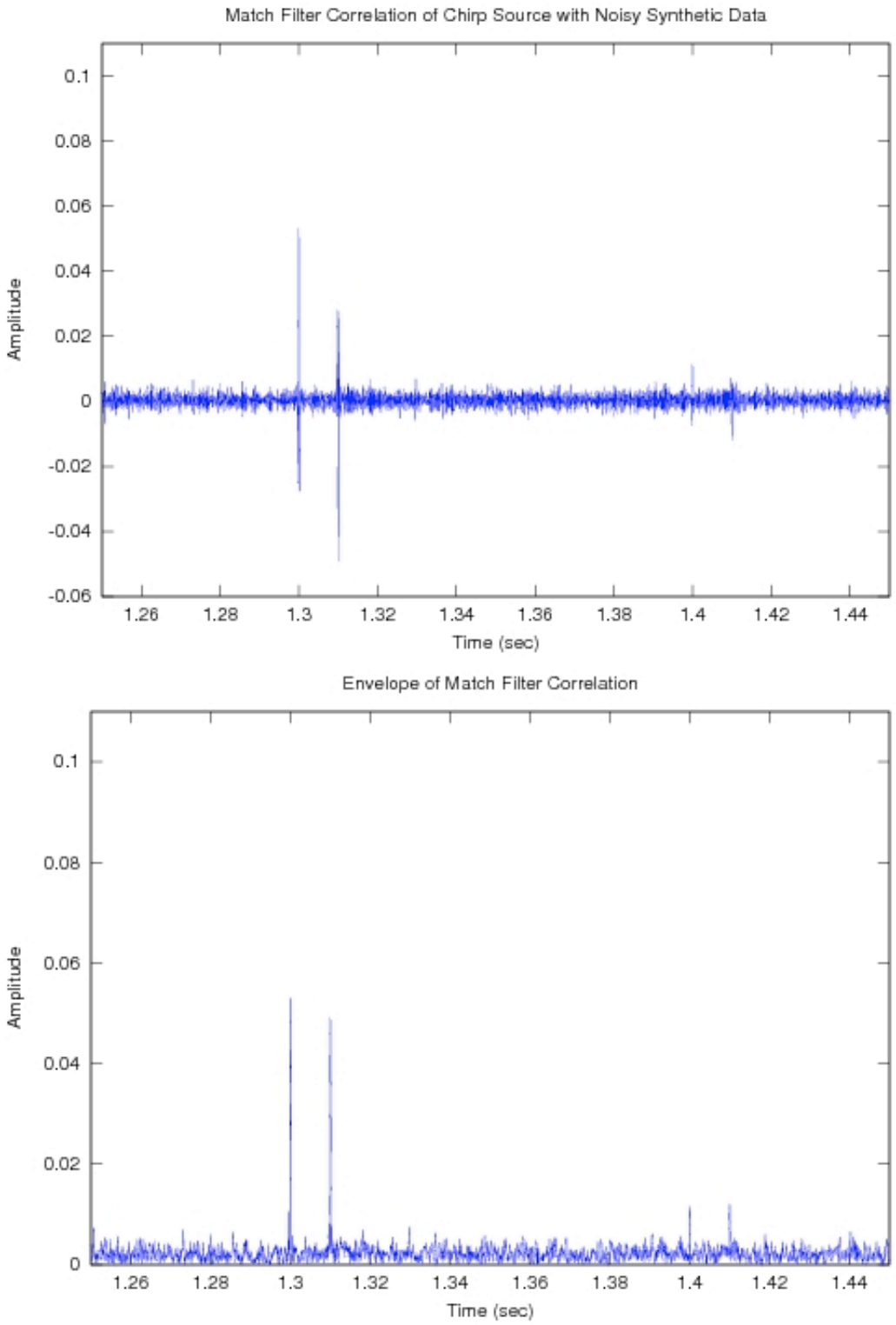

Figure 8. Chirp processing of a 0.20 second section of the synthetic data. The envelope function demonstrates that arrivals with amplitudes lower than the noise can be identified by the chirp processing.

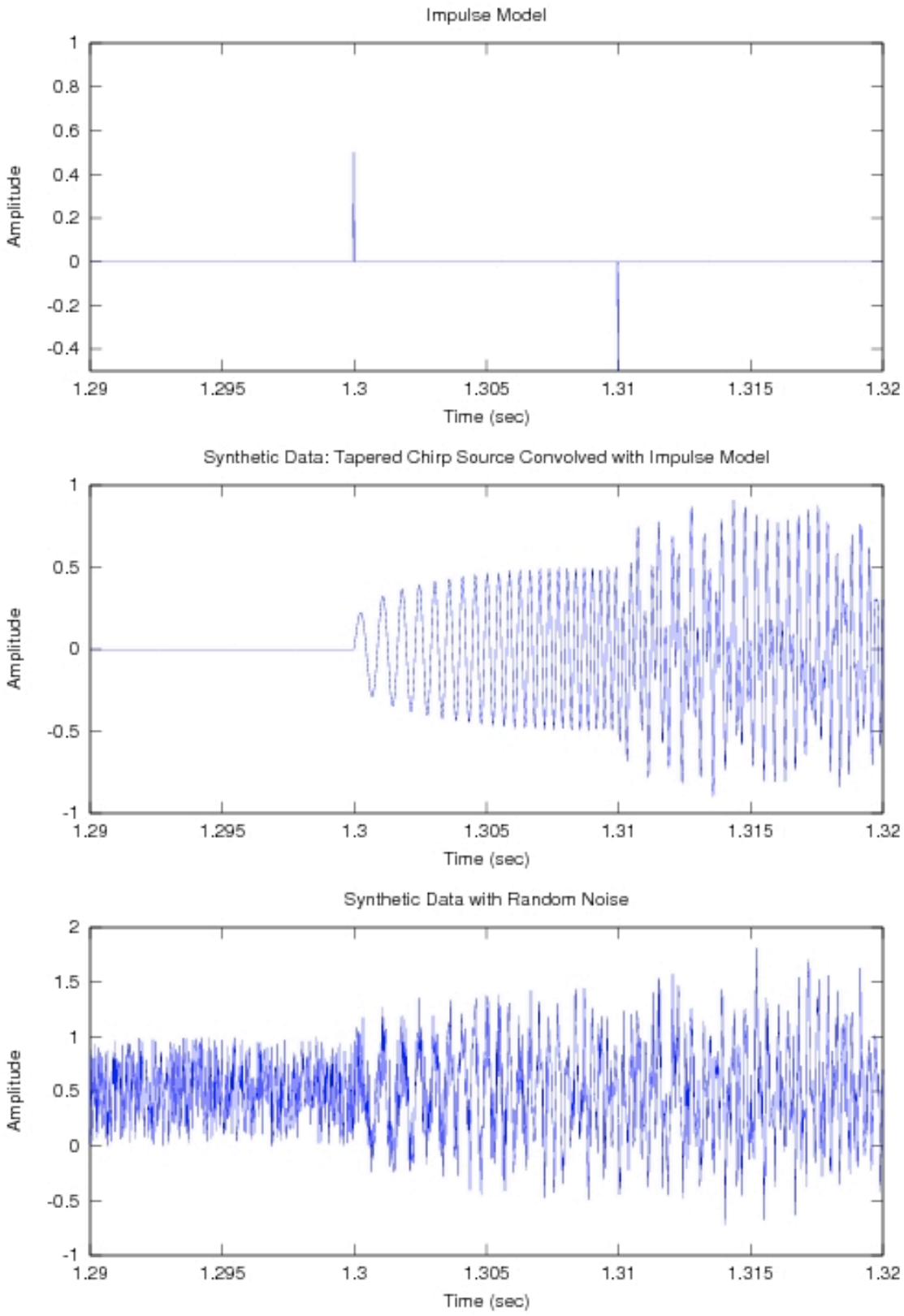

Figure 9. View of a 0.03 second section of the synthetic data.

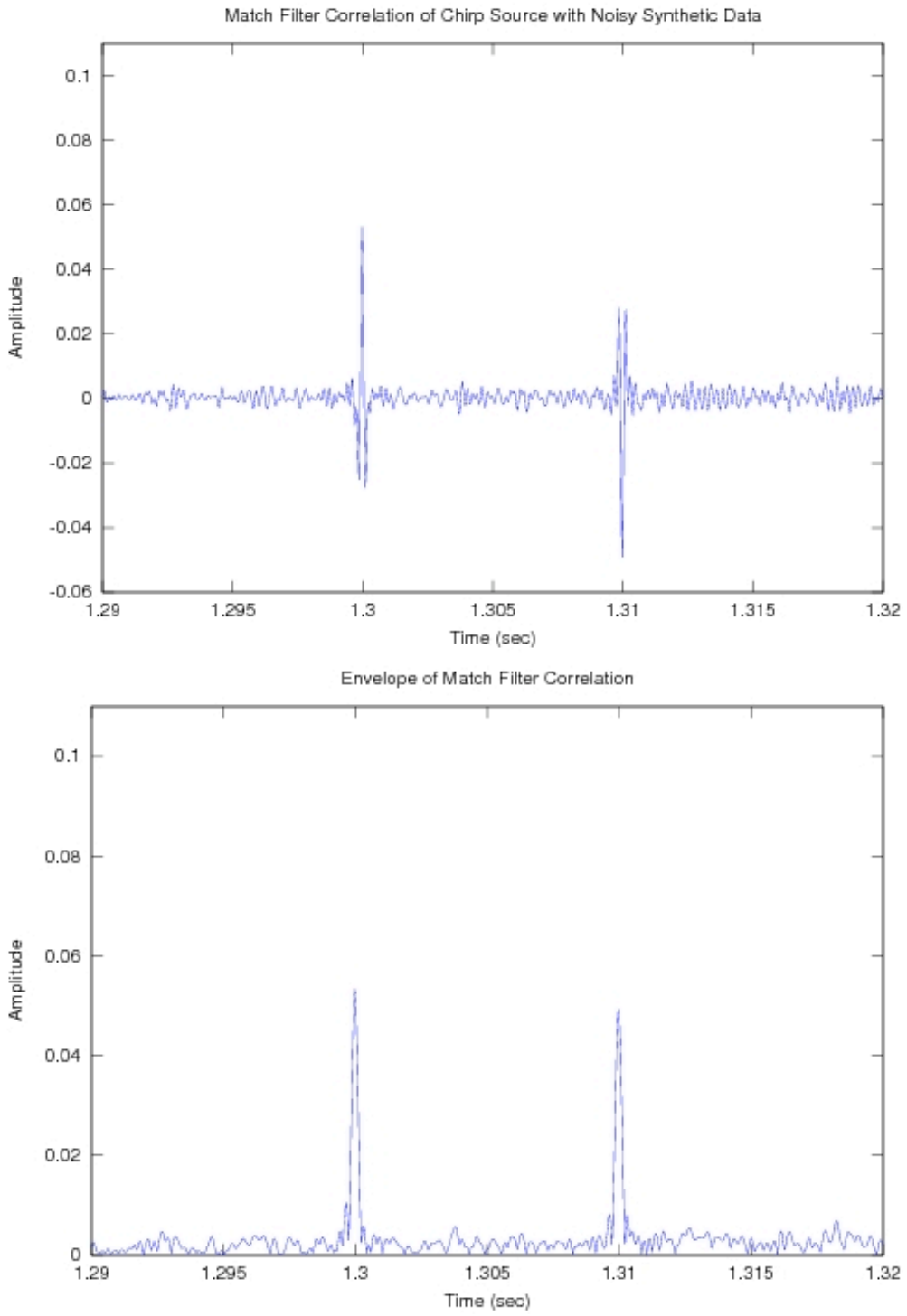

Figure 10. Chirp processing of a 0.20 second section of the synthetic data. The arrivals in the correlate (top) show the Klauder wavelet structure, and also demonstrate how with real data it is difficult to detect the polarity of arrivals. The envelope function arrivals (bottom) has the same width as the Klauder wavelet, and thus this width (~0.5 msec) represents the theoretical best possible time resolution for the chirp source function.

Here is the m-file used for generating the above plots
:

```
# test script for Octave

# Generate a chirp signal with cosine taper
#octave

        # define sampling frequency in Hz
        fsamp = 40000;
        dt = 1/fsamp;

        # define start and end frequencies in Hz
        fstartkHz = 1;
        fendkHz = 6;
        fstart = 1000 * fstartkHz;
        fend = 1000 * fendkHz;

        # start cosine taper with phase of pi/2
        phasestart = -.25;

        # chirp duration in seconds
        dur = 0.02;
        Ntt = dur / dt;
        Ntt = 2 * floor(Ntt / 2);
        dur = dt * (Ntt - 1);

        # make time vector for time series
        tt = 0 : dt : dur;
        Ntt = length(tt);

        # Calculate time dependent instantaneous frequency
        psi = 2*pi*(phasestart + fstart*tt + 0.5 *(fend - fstart)/dur*tt.*tt);

        # Calculate time dependent raw chirp signal
        chirp = real( 1.0*exp(j*psi) );

        # Calculate cosine taper and apply it to the raw chirp
        # to get the source signal
        taper = sin(pi * tt / dur);
        source = chirp .* sqrt(sqrt(taper));

        # Plot the chirp source signal
        figure(1);
        subplot(2,1,1);
        plot( tt, chirp);
        titlestring = sprintf("Untapered %d-%d kHz Chirp Source", fstartkHz, fendkHz);
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        axis([0, dur, -1.1, 1.1]);
        subplot(2,1,2);
        plot(tt, source);
        titlestring = sprintf("Tapered %d-%d kHz Chirp Source", fstartkHz, fendkHz);
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
```

```
        axis([0, dur, -1.1, 1.1]);
        print ("Fig01_ChirpSource.ps", "-dpsc2", "-F:10");

        # Calculate autocorrelation of raw chirp by
        # calculating crosscorrelation with itself
        cklauder = xcorr(chirp, "biased");
        lt = -dur + dt : dt : dur;

        # Calculate autocorrelation of tapered chirp by
        # calculating crosscorrelation with itself
        sklauder = xcorr(source, "biased");
        lt = -dur : dt : dur;
        ltms = 1000 .* lt;

        # Plot the klauder wavelet
        figure(2);
        subplot(2,1,1);
        plot( ltms, cklauder);
        titlestring = sprintf("Klauder Wavelet of %d-%d kHz Untapered Chirp", fstartkHz,
fendkHz);
        title(titlestring);
        xlabel ("Time (msec)");
        ylabel ("Correlation");
        axis([-2,2,-0.6, 0.6]);
        subplot(2,1,2);
        plot(ltms, sklauder);
        titlestring = sprintf("Klauder Wavelet of %d-%d kHz Tapered Chirp", fstartkHz,
fendkHz);
        title(titlestring);
        xlabel ("Time (msec)");
        ylabel ("Correlation");
        axis([-2,2,-0.6, 0.6]);
        print ("Fig02_KlauderWavelet.ps", "-dpsc2", "-F:10");

        # Fourier transform the various signals
        chirp_fft = fft(chirp);
        source_fft = fft(source);
        fchirp=[0:Ntt/2-1]/(Ntt*dt);
        Nlt = length(cklauder);
        cklauder_fft = fft(cklauder);
        sklauder_fft = fft(sklauder);
        fklauder=[0:Nlt/2]/(Nlt*dt);

        # Plot the chirp ffts
        figure(3);
        subplot(2,1,1);
        plot( fchirp, abs(chirp_fft(1:Ntt/2)));
        titlestring = sprintf("FFT Magnitude of %d-%d kHz Untapered Chirp", fstartkHz,
fendkHz);
        title(titlestring);
        xlabel ("Frequency (Hz)");
        ylabel ("Magnitude");
        subplot(2,1,2);
        plot( fchirp, abs(source_fft(1:Ntt/2)));
        titlestring = sprintf("FFT Magnitude of %d-%d kHz Tapered Chirp", fstartkHz,
fendkHz);
        title(titlestring);
        xlabel ("Frequency (Hz)");
        ylabel ("Magnitude");
        print ("Fig03_ChirpFFT.ps", "-dpsc2", "-F:10");
```

```
        # Plot the Klauder wavelet ffts
        figure(4);
        subplot(2,1,1);
        plot( fklauder, abs(cklauder_fft(1:Nlt/2+1)));
        titlestring = sprintf("FFT Magnitude of %d-%d kHz Klauder Wavelet from Untapered
Chirp", fstartkHz, fendkHz);
        title(titlestring);
        xlabel ("Frequency (Hz)");
        ylabel ("Power");
        subplot(2,1,2);
        plot( fklauder, abs(sklauder_fft(1:Nlt/2+1)));
        titlestring = sprintf("FFT Magnitude of %d-%d kHz Klauder Wavelet from Tapered
Chirp", fstartkHz, fendkHz);
        title(titlestring);
        xlabel ("Frequency (Hz)");
        ylabel ("Power");
        print ("Fig04_KlauderFFT.ps", "-dpsc2", "-F:10");

        # Get spectral density function estimates from the various signals
        chirp_sdf = spectral_xdf(chirp);
        source_sdf = spectral_xdf(source);
        Nlt = length(cklauder);
        cklauder_sdf = spectral_xdf(cklauder);
        sklauder_sdf = spectral_xdf(sklauder);

        # Plot the chirp sdfs
        figure(5);
        subplot(2,1,1);
        plot( fsamp*chirp_sdf(1:Ntt/2,1), chirp_sdf(1:Ntt/2,2));
        titlestring = sprintf("Power Spectra of %d-%d kHz Untapered Chirp", fstartkHz,
fendkHz);
        title(titlestring);
        xlabel ("Frequency (Hz)");
        ylabel ("Magnitude");
        subplot(2,1,2);
        plot( fsamp*source_sdf(1:Ntt/2,1), source_sdf(1:Ntt/2,2));
        titlestring = sprintf("Power Spectra of %d-%d kHz Tapered Chirp", fstartkHz,
fendkHz);
        title(titlestring);
        xlabel ("Frequency (Hz)");
        ylabel ("Magnitude");
        print ("Fig05_ChirpPSD.ps", "-dpsc2", "-F:10");

        # Plot the Klauder wavelet sdfs
        figure(6);
        subplot(2,1,1);
        plot( fsamp*cklauder_sdf(1:Nlt/2+1,1), cklauder_sdf(1:Nlt/2+1,2));
        titlestring = sprintf("Power Spectra of %d-%d kHz Klauder Wavelet from Untapered
Chirp", fstartkHz, fendkHz);
        title(titlestring);
        xlabel ("Frequency (Hz)");
        ylabel ("Power");
        subplot(2,1,2);
        plot( fsamp*sklauder_sdf(1:Nlt/2+1,1), sklauder_sdf(1:Nlt/2+1,2));
        titlestring = sprintf("Power Spectra of %d-%d kHz Klauder Wavelet from Tapered
Chirp", fstartkHz, fendkHz);
        title(titlestring);
        xlabel ("Frequency (Hz)");
        ylabel ("Power");
```

```
        print ("Fig06_KlauderPSD.ps", "-dpsc2", "-F:10");

        # generate a test data time series
        # start with impulses in a long time series
        sweep = 2;
        data_impulse = zeros(sweep*fsamp+1, 1);
        data_tt = 0 : dt : sweep;

        data_impulse(floor(0.5*fsamp)) += 1.0;
        data_impulse(floor(1.0*fsamp)) += 0.5;
        data_impulse(floor(1.2*fsamp)) += -0.5;
        data_impulse(floor(1.30*fsamp)) += 0.5;
        data_impulse(floor(1.31*fsamp)) += -0.5;
        data_impulse(floor(1.40*fsamp)) += 0.1;
        data_impulse(floor(1.41*fsamp)) += -0.1;

        # convolve the impulse data with the source
        data_conv = conv(source,data_impulse);
        data_ttconv = 0 : dt : sweep+dur;

        # add noise to the synthetic data
        data_noise = (data_conv.') .+ (rand(length(data_conv), 1));

        # do the match filtering relative to the source
        normalize = sqrt(sumsq(data_noise) * sumsq(source));
        data_match = xcorr(data_noise, flipud(source)) ./ normalize;
        data_ttmatch = -sweep-dur : dt : sweep+dur;

        # calculate analytic signal and envelope
        data_analytic = hilbert(data_match);
        data_envelope = abs(data_analytic);

        # Plot the synthetic data
        figure(7);
        subplot(3,1,1);
        plot( data_tt, data_impulse);
        axis([0, sweep]);
        titlestring = sprintf("Impulse Model");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        subplot(3,1,2);
        plot( data_ttconv, data_conv);
        axis([0, sweep]);
        titlestring = sprintf("Synthetic Data: Tapered Chirp Source Convolved with Impulse
Model");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        subplot(3,1,3);
        plot( data_ttconv, data_noise);
        axis([0, sweep]);
        titlestring = sprintf("Synthetic Data with Random Noise");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        print ("Fig07_SyntheticDataAll.ps", "-dpsc2", "-F:10");

        # Plot the match-filtering result
        figure(8);
```

```
        subplot(2,1,1);
        plot( data_ttmatch, data_match);
        axis([0, sweep]);
        titlestring = sprintf("Match Filter Correlation of Chirp Source with Noisy
Synthetic Data");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        subplot(2,1,2);
        plot( data_ttmatch, data_envelope);
        axis([0, sweep]);
        titlestring = sprintf("Envelope of Match Filter Correlation");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        print ("Fig08_MatchFilterAll.ps", "-dpsc2", "-F:10");

        # Plot a small bit of the synthetic data
        figure(9);
        subplot(3,1,1);
        plot( data_tt, data_impulse);
        axis([1.25, 1.45]);
        titlestring = sprintf("Impulse Model");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        subplot(3,1,2);
        plot( data_ttconv, data_conv);
        axis([1.25, 1.45]);
        titlestring = sprintf("Synthetic Data: Tapered Chirp Source Convolved with Impulse
Model");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        subplot(3,1,3);
        plot( data_ttconv, data_noise);
        axis([1.25, 1.45]);
        titlestring = sprintf("Synthetic Data with Random Noise");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        print ("Fig09_SyntheticDataSmall.ps", "-dpsc2", "-F:10");

        # Plot a small bit of the match-filtering result
        figure(10);
        subplot(2,1,1);
        plot( data_ttmatch, data_match);
        axis([1.25, 1.45]);
        titlestring = sprintf("Match Filter Correlation of Chirp Source with Noisy
Synthetic Data");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        subplot(2,1,2);
        plot( data_ttmatch, data_envelope);
        axis([1.25, 1.45]);
        titlestring = sprintf("Envelope of Match Filter Correlation");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
```

```
        print ("Fig10_MatchFilterSmall.ps", "-dpsc2", "-F:10");

        # Plot a smaller bit of the synthetic data
        figure(11);
        subplot(3,1,1);
        plot( data_tt, data_impulse);
        axis([1.29, 1.32]);
        titlestring = sprintf("Impulse Model");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        subplot(3,1,2);
        plot( data_ttconv, data_conv);
        axis([1.29, 1.32]);
        titlestring = sprintf("Synthetic Data: Tapered Chirp Source Convolved with Impulse
Model");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        subplot(3,1,3);
        plot( data_ttconv, data_noise);
        axis([1.29, 1.32]);
        titlestring = sprintf("Synthetic Data with Random Noise");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        print ("Fig11_SyntheticDataSmaller.ps", "-dpsc2", "-F:10");

        # Plot a smaller bit of the match-filtering result
        figure(12);
        subplot(2,1,1);
        plot( data_ttmatch, data_match);
        axis([1.29, 1.32]);
        titlestring = sprintf("Match Filter Correlation of Chirp Source with Noisy
Synthetic Data");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        subplot(2,1,2);
        plot( data_ttmatch, data_envelope);
        axis([1.29, 1.32]);
        titlestring = sprintf("Envelope of Match Filter Correlation");
        title(titlestring);
        xlabel ("Time (sec)");
        ylabel ("Amplitude");
        print ("Fig12_MatchFilterSmaller.ps", "-dpsc2", "-F:10");



#exit
```