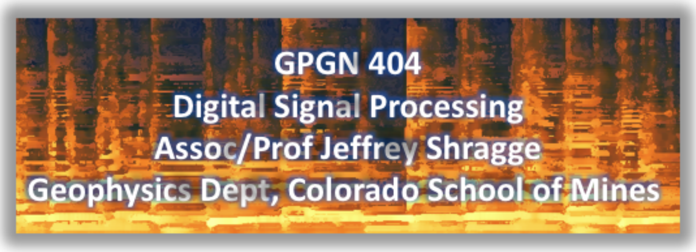
Out[27]: The raw code for this Jupyter notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.


GPGN 404
Digital Signal Processing
Assoc/Prof Jeffrey Shragge
Geophysics Dept, Colorado School of Mines

Module 11: Windows, Short-time Fourier Transforms and Spectrograms

The purpose of this section is for you to get experience in looking at how the spectral content can change in datasets represented by long time series. So far we have looked at how the Fourier Transform can be applied to look at the spectral content. However, the Fourier Transform is a **global** operation that cannot distinguish how the frequency content varies within a signal. Examples of the global nature are evident when one thinks about the following Fourier transform pair:

$$\delta(t-t_0) \leftrightarrow e^{i\omega t_0} \tag{1a}$$

and

$$\cos(\omega_0 t) + i\sin(\omega_0 t) \leftrightarrow \delta(\omega-\omega_0) \tag{1b}$$

In equation 1a, a δ-function spike at one moment in time affects everywhere in the frequency domain. In equation 1b, a δ-function spike at one frequency has an effect everywhere in the time domain. Of course, we have looked at other less extreme examples where the effects are more restricted in the time and frequency domains. For example, the Fourier transform of Gaussian function in the time domain is a Gaussian function in the frequency domain.

One of the underlying assumptions when applying a Fourier Transform is that the signals are **stationary**. Playing a bit fast and loose with nomenclature, this means that if you have a signal $f(t)$ with Fourier Transform $\widehat{F}(\omega)$, then one assumes that the spectral components do not evolve over time.

$$\frac{d\widehat{F}}{dt} = 0 \tag{1c}$$

However, we are all familiar with scenarios around us when transient phenomena is audible (e.g., a passing train) and we know that the frequency structure is changing as a function of time. Does this mean that equation 1c does not hold?

$$\frac{d\widehat{F}}{dt} \neq 0?? \tag{1d}$$

How do we obtain more information about how the signal is evolving? Clearly, we need to think about moving from a **global** transform to one that is more **local**. While this might not make sense with the analytic Fourier Transform defined between $-\infty$ and ∞, one might feel intuitively that this can be done with the discrete Fourier Transform (DFT) because this operation is applied to a sequence of numbers without any requirement of $-\infty$ and ∞!

Short-term thinking

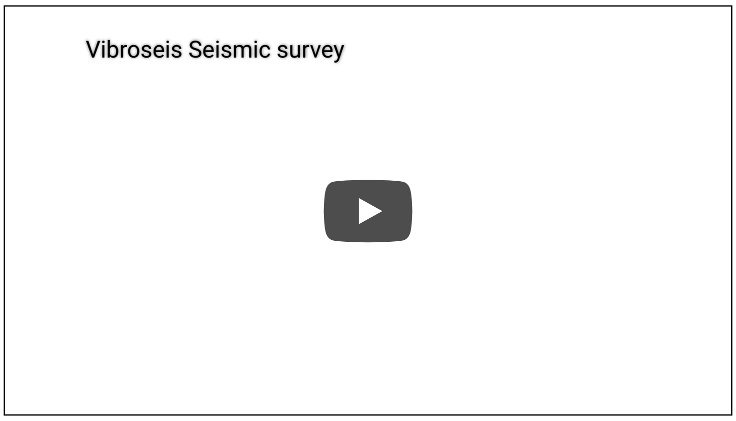
One pathway forward is to think about breaking up a long time series of N_t points into N_w windows of length N_t/N_w and then applying N_w Fourier Transforms (one also hears the term short-time Fourier Transform (https://en.wikipedia.org/wiki/Short-time_Fourier_transform)). You can then concatenate the N_w Fourier Transforms into a 2D representation of the signal: a spectrogram (https://en.wikipedia.org/wiki/Spectrogram).

Completing these operations requires investigating a number of items: Spectrograms, Windowing and Gibbs Phenomena, and the Short-term Fourier Transform (STFT). We'll first look at some examples that complete this operation using the Scipy library *signal.spectrogram* (https://docs.scipy.org/doc/scipy-0.19.1/reference/generated/scipy.signal.spectrogram.html#scipy.signal.spectrogram) code. We'll then dig a bit deeper to investigate the individual components making up this operation.

Example 1 - Chirp Signals (i.e., Vibroseis)

One class of signals that are commonly used in many branches of the physical sciences is a chirp signal (https://en.wikipedia.org/wiki/Chirp). An example in exploration seismology is the Vibroseis machine. Here's an example of one in action:

Out[2]:



Creating a chirp signal

The general form of the chirp signal is given by:

$$x(t) = e^{i\phi(t)}, \tag{1}$$

where $\phi(t)$ is the phase function that is in general time dependent. The instantaneous frequency (https://en.wikipedia.org/wiki/Instantaneous_phase) is given by the time derivative:

$$f(t) = \frac{1}{2\pi}\frac{d\phi(t)}{dt}. \tag{2}$$

The interpretation of the signal in equation 2 is that this tells you at any given moment what the frequency content is of the signal. Let's look at two different types of chirp signals:

Linear:

$$f(t) = f_0 + (f_1 - f_0)\frac{t}{t_1} = f_0 + kt. \tag{3}$$

where $k = (f_1 - f_0)/t_1$ is slope of the linear change in frequency so that at time $t_0 = 0$ is f_0 and at some later reference time t_1 it is f_1.

To get our phase function $\phi(t)$ we want to integrate both sides of the equation 2 for instantaneous frequency. Integrating this yields:

$$\begin{aligned}\phi(t) &= \phi_0 + 2\pi\int_0^t f(\tau)\,d\tau \\ &= \phi_0 + 2\pi\int_0^t (f_0 + k\tau)\,d\tau \\ &= \phi_0 + 2\pi\left(f_0 t + \frac{k}{2}t^2\right),\end{aligned} \tag{4a}$$

where ϕ_0 is the initial phase of the signal at $t = 0$ (which we can set to zero most of the time). We observe a t^2 entering the phase so the frequency content of the signal is going to be time varying. Therefore, the chirp signal that we will be looking at is given by:

$$x(t) = e^{i\left(\phi_0 + 2\pi\left(f_0 t + \frac{k}{2}t^2\right)\right)} = e^{i\phi_0} e^{i2\pi f_0 t} e^{i2\pi(f_1 - f_0)t^2/(2t_1)} \tag{4b}$$

It is clear that if $f_1 = f_0$ then the last term would equal to 1 and we would recover a sinusoidal signal.

Hyperbolic:

$$f(t) = \frac{f_0 f_1 t_1}{(f_0 - f_1)t + f_1 t_1}. \tag{5}$$

I'll leave the integration of this to you!

Let's first plot the two functions as well as their Fourier Transforms. The first is a **linear** chirp between $[f_0, f_1] = [10, 200]$ over about 16 seconds of recording. The second is a **hyperbolic** chirp with the same frequency range. As expected, the FT panels show us the overall spectral content; however, they do not show us **how the frequencies are distributed throughout the time series**. This is the information we are looking to obtain from a spectrogram.

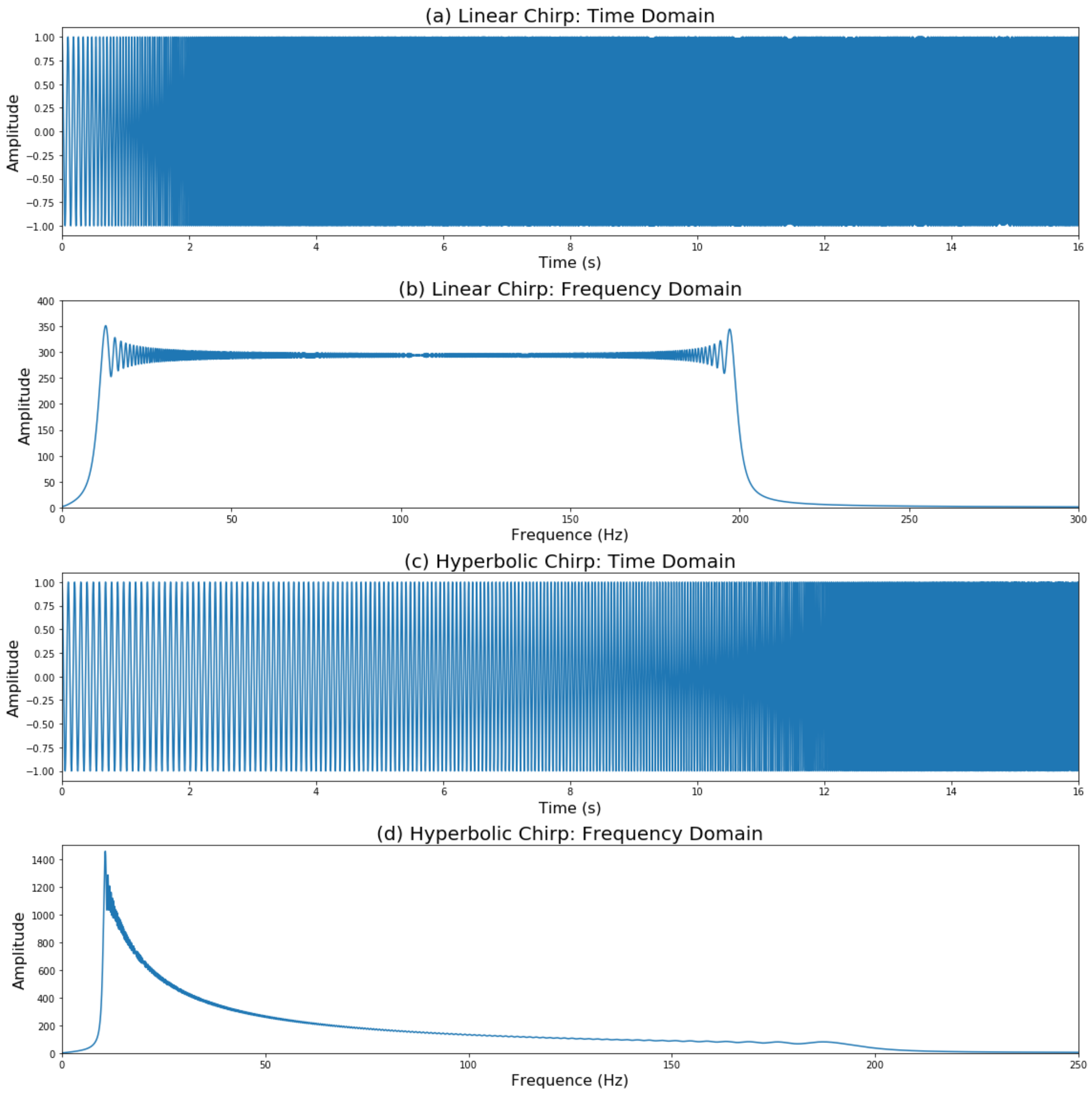


Figure 1. (a) Example of a linear chirp signal along with (b) the associated Fourier magnitude spectra. (c) Example of a hyperbolic chirp signal along with (d) the associated Fourier magnitude spectra.

As expected, the FT panels show us the overall spectral content; however, they do not show us **how the frequencies are distributed throughout the time series**. This is the information we are looking to obtain from a spectrogram.

You can listen to the linear chirp sound here:

```
Out[4]:
```

0:00 / 0:08

You can listen to the hyperbolic chirp sound here:

```
Out[5]:
```

0:00 / 0:08

Now let's plot the spectrograms of these two functions, where the spectrograms on the right

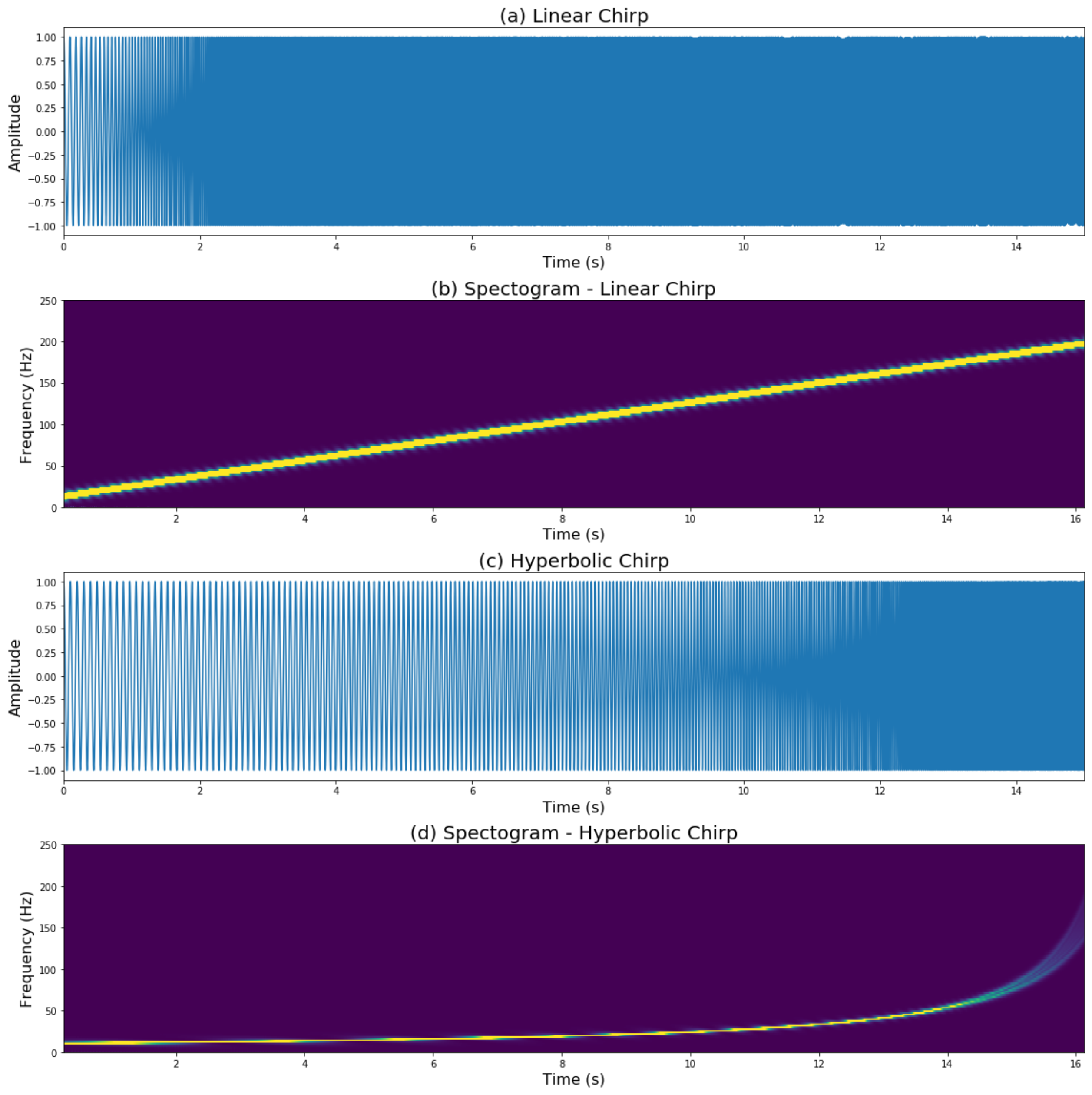


Figure 2. Demonstration of the utility of spectrograms. (a) Example of a linear chirp signal along with (b) the associated spectrogram. (c) Example of a hyperbolic chirp signal along with (d) the associated spectrogram.

Let's now compute a nonlinear chirp where we have a modulation that changes as a function of $\sin(t)$:

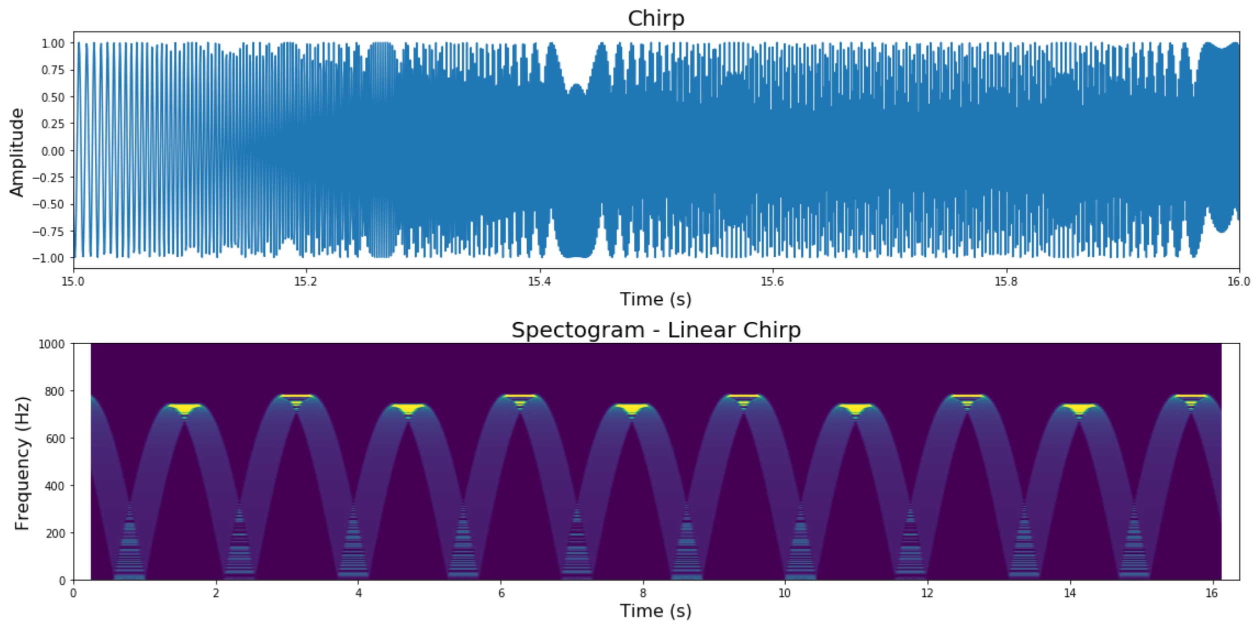


Figure 3. (a) Example of a non-linear chirp signal along with (b) the associated spectrogram.

Let's listen to this signal:

```
Out[8]:
```

0:00 / 0:08

Example 2 - Passing Train with Doppler Effect and Clickity Clacks

Let's next examine a the sound that a passing train makes. We learned in freshman physics that, due to the Doppler effect, a train coming toward an observer will have a frequency spectrum that is shifted from the one heard from a receding train. Listen to the following example to hear what happens at between 6-7s.

```
Out[10]:
```

0:00 / 0:26

Let's now look at the plot of the time series as well as the spectrogram. The latter was computed in blocks of 2048 time samples with consecutive windows overlapping by 1024 samples. (Note: powers of 2 were chosen to facilitate FFT computations.)

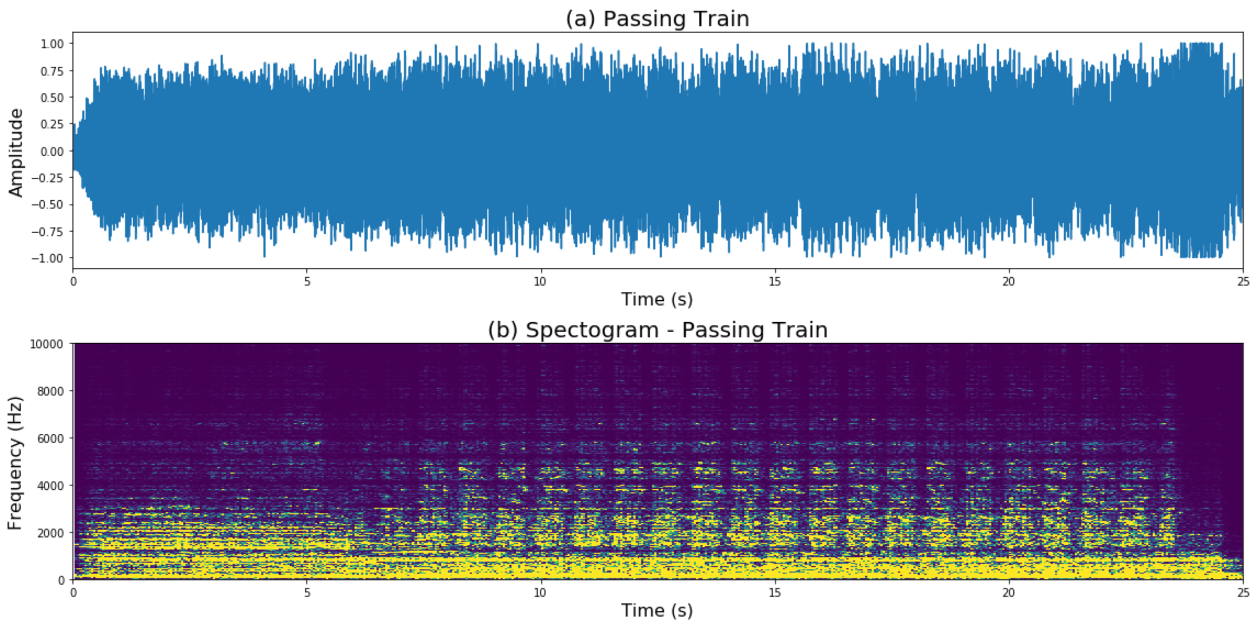


Figure 4. (a) Example of a recorded passing train along with (b) the associated spectrogram.

Looking at the spectrogram, you can probably see that there is a bit of a down shift in frequency content from between 1-5s to 5-7s. You'll then notice the periodic clickity-clack of the train rolling over a gap in the tracks. This definitely causes a higher-frequency noise. Again, this type of analysis would not be possible using a single global Fourier Transform.

Example 3 - Tohoku Earthquake recorded in Germany

Let's now listen to the disasterous Tohoku earthquake as it was recorded at a station in Germany. This signal represents the vertical motion at this station location.

```
Out[12]:
```

0:00 / 0:10

You likely would have heard the first arrival (P-waves), as well as the second and third louder arrivals (S- and surface waves) coming in soon afterward. If you listen again, can you detect some of the later arrivals as well?

Let's now look at the time series data as well as the spectrogram. You'll see that the peaks of the spectrogram are well aligned with the visible arrivals in the plot of the time-series data. You'll also notice that there are spikes in the spectrogram where you do not really see arriving waveforms: these are likely the more exotic P-wave phases that are weaker in amplitude ... but have higher frequency content!

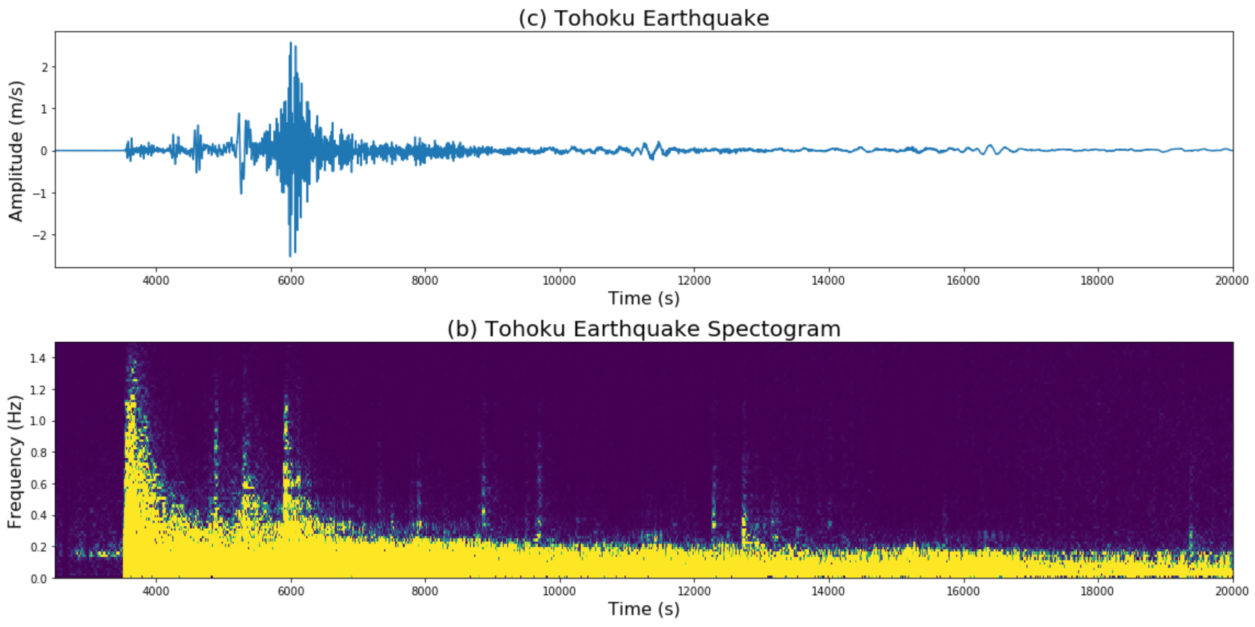


Figure 5. (a) Example from the Tohoku earthquake g with (b) the associated spectrogram.

Windowing

The first major step in deciding how to construct a spectrogram is to decide on the windowing procedure that will be applied to diving the long time series into N_w windows.

The first question to address is: does how you apply the windowing matter? The answer in general is **yes**. To illustrate why, look at the example below of a Gaussian waveform that is in the middle of a time series. The frequency spectrum is located to the immediate right. Note that the maximum frequency is about 35 Hz.

The second pair of panels illustrates what happens when you half the time-series into two different windows and then apply the Fourier Transform. We see that the overall magnitude is decreased and their appears to be energy at higher frequencies.

The third pair of panels shows what happens when the window is translated to be centered over the Gaussian. We see that the spectrum returns to that of the first panel.

The final pair of panels shows the complement of the second panels with the altered frequency content.

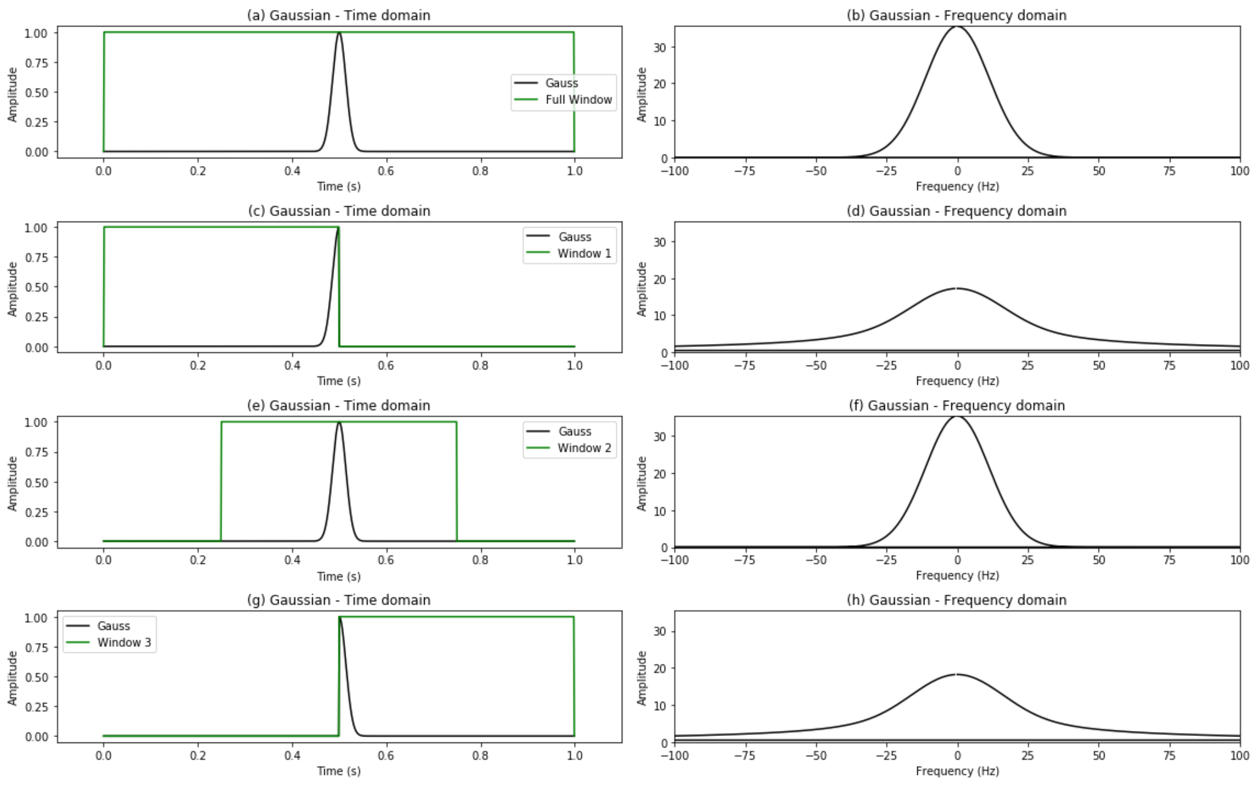


Figure 6. Illustrating how the location of the slide window can affect the resulting frequency calculation forr a Gaussian signal. (a) Gaussian signal and full selection window that results in the Fourier magnitude spectrum in (b). (c) Gaussian signal trucated by left-half window length that results in the Fourier magnitude spectrum in (d). (e) Gaussian signal with centered window that results in the Fourier magnitude spectrum in (f). (g) Gaussian signal trucated by right-half window length that results in the Fourier magnitude spectrum in (h).

Let's now look at a spectrogram of the 1000-point time Gaussian that is computed on windows of length 400, and then translated by 1 sample. Thus, we expect a total of 600 different frequency distributions. The plot below shows the results of this where the time coordinate is the center of the sliding window. We again see that the frequency distribution appears to change.

Let's recompute this using smallering windows of 200 samples (and then shifting the output by one sample). Again, we see the frequency content varying with lag - but in a different way than the example to the left. We also see that by halving the number of samples, the frequency resolution also has dropped by a factor of two.

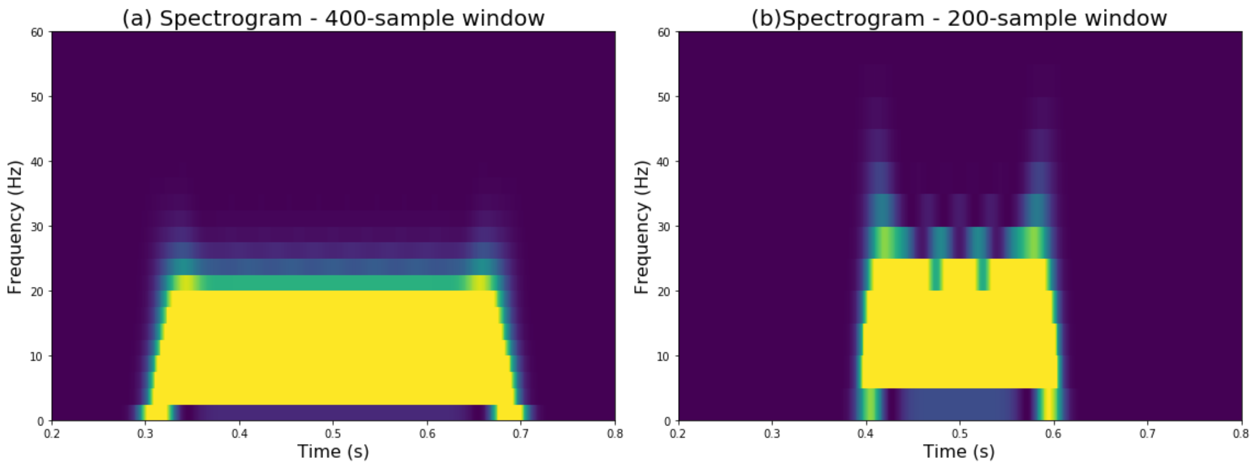


Figure 7. Example of how spectrogram can change depending on window length. (a) $N = 400$ window length. (b) $N = 200$ window length. Note that the longer window length gives finer resolution in frequency.

A second consideration when applying of a windowing procedure is that if one applies a windowing function that has sharp corners (e.g., a boxcar Π_n function) is that this can impart a significant ringing of the resulting spectrum (which is an example of Gibbs phenomena (https://en.wikipedia.org/wiki/Gibbs_phenomenon)). To illustrate this, let's look at an example where we have fairly broad waveform (shown in blue in upper left) that has a fairly low frequency distribution (upper right).

Let's now apply a box-car windowing function (shown in black in the lower right panel) that has a *sinc-function* frequency-domain representation. If we look at the resulting waveform in the time domain (shown in red in lower right panel), we see that the truncation has caused the waveform to become very ringy. This is because **the multiplication of the two waveforms in the time domain (e.g., the red curve) is equivalent the convolution of the blue and black curves in the frequency domain (red curve)**.

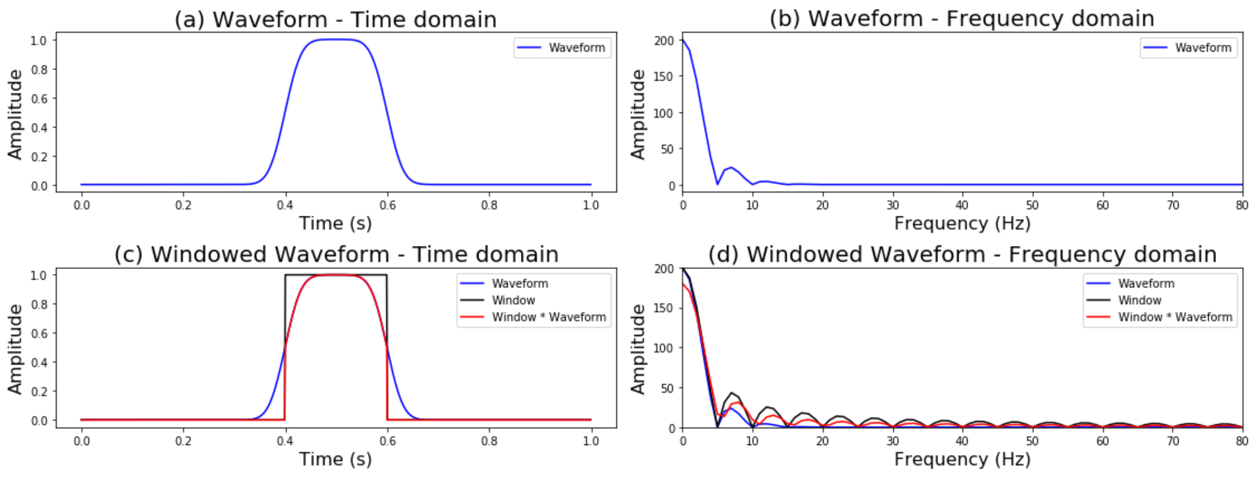


Figure 8. Example showing the Fourier-domain consequences of windowing. (a) Original waveform along with (b) the associated Fourier magnitude spectrum. (c) Windowed signal from (a) that results in (d) Gibb's effects in the Fourier magnitude spectrum.

Playing with Smart Windowing Functions

There are many different windows types available for use in window functions. In the plot below I investigate a number of them that are available in the *scipy.signal* toolbox (https://docs.scipy.org/doc/scipy-0.19.1/reference/signal.html): (1) Blackman-Harris; (2) Chebeshev; (3) Hamming; (4) Gaussian; (5) Hann; (6) Kaiser; (7) Slepian; and (8) Tukey. Let's look at what the windows look like and what the corresponding frequency spectra are!

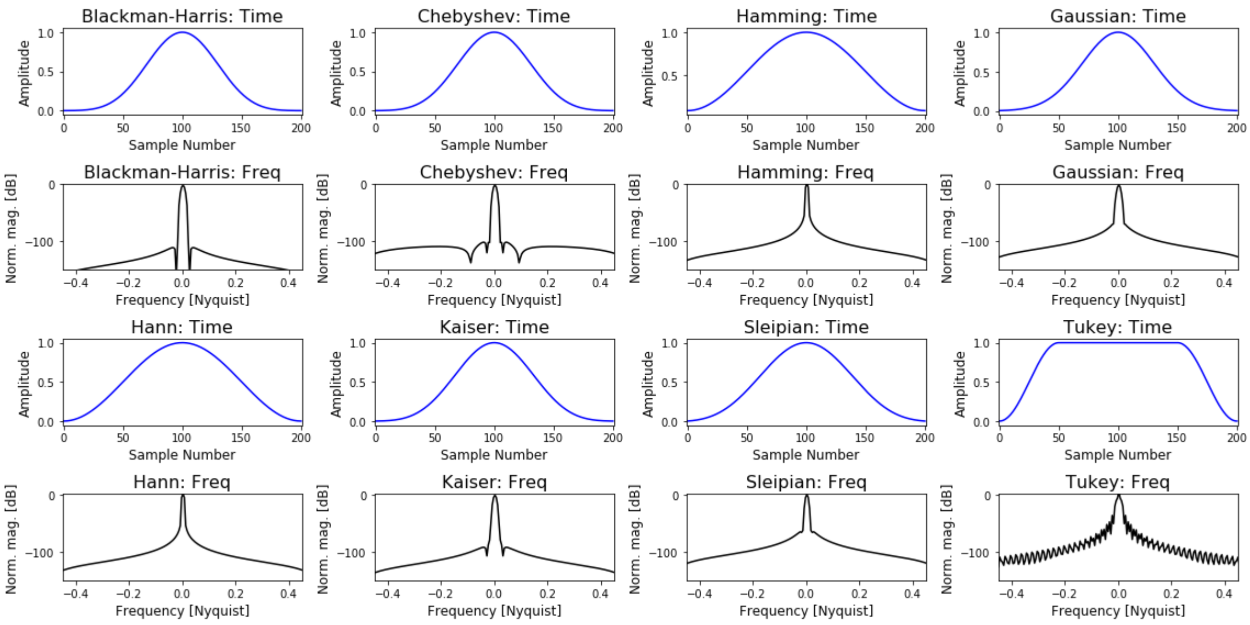


Figure 9. Illustration of eight differernt windowing functions along with their associated Fourier magnitude spectra (in dB down).

In each of the frequency-domain plots the horizontal variables run from $[-0.45f_{max}, 0.45f_{max}]$, and the vertical axis is in normalized magnitude (dB), which is a logarithmic scale. Thus, each window is doing very well in reducing any of the Gibbs ringing effect. Even by slightly varying the shape of the window operator, one can cause a signifant change in the resulting spectrum. A lot of electical engineering time has been spent designing these operators!

Short-time Fourier Transform (SFTF)

Evidently, spectrograms are a powerful tool for investigating the **frequency structure** of signals. However, it would also be nice to be able to perform signal processing tasks on these types of functions in the frequency domain, and then transform the results back to the time domain for use in other operations. To do this one can use the short-time Fourier Transform.

Example of a Noisy FM broadcast

Your FM radio broadcasts signals using the concept the frequency modulation (https://en.wikipedia.org/wiki/Frequency_modulation) as a way of carrying information, which is achieved by encoding of information in a carrier wave by varying the instantaneous frequency of the wave.

$$y(t) = A_c \cos\left(2\pi f_c t + \frac{A_m}{f_m}\sin(2\pi f_m t)\right).$$

Here, A_c is the overall amplitude, f_c is the carrier frequency, A_m is the amplitude of the modulated signal, and f_m is the modulation frequency.

Let's create a FM signal with $A_c = 2\sqrt{2}$, $f_c = 1000$ Hz, $A_m = 500$ Hz, and $f_m = 1$~Hz. Let's first listen to the clean signal.

```
Out[18]:
```

0:00 / 0:10

Now let's corrupt this signal with Gaussian noise such that it is strongly audible when listening to the FM signal.

```
Out[19]:
```

0:00 / 0:10

Now let's visualize the signal and the noisy signal. Here's a plot of the two signals overtop of each other. You'll notice that I have applied a Tukey window to the first and last 10% to order to have a smooth tapering into the ends (which is a requirement for stability of the STFT).

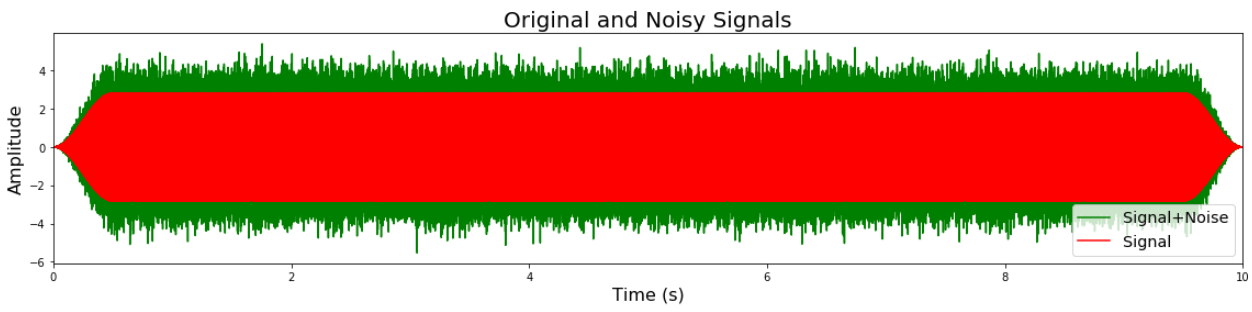


Figure 11. Illustration of the carrier signal (orange) along with additive noise (green).

Let's now apply the STFT to the noisy signal using sliding window of 512 samples with an overlap of 256 samples. The top panel shows the output of the SFTF with the signal clearly visible as the yellow modulating line. However, one can still see the contribution of the random Gaussian noise in the background. It's our goal to get rid of this!

To help eliminate the "salt-and-pepper" background, we're going to apply a **thresholding** operation. Essentially, this scans the entire file and sets to zero any absolute value (recall this is a complex-valued function) that is lower than X% of the maximum value. For the example here, I decided that the threshold value will be 10%.

The middle panel shows the result of thresholdeding the STFT noisy signal. We see that most of the background Gaussian noise has disappeared, leaving on the desired modulating signal. The lower panel shows the difference between the top and middle panels - effectively showing the **residual estimated noise**.

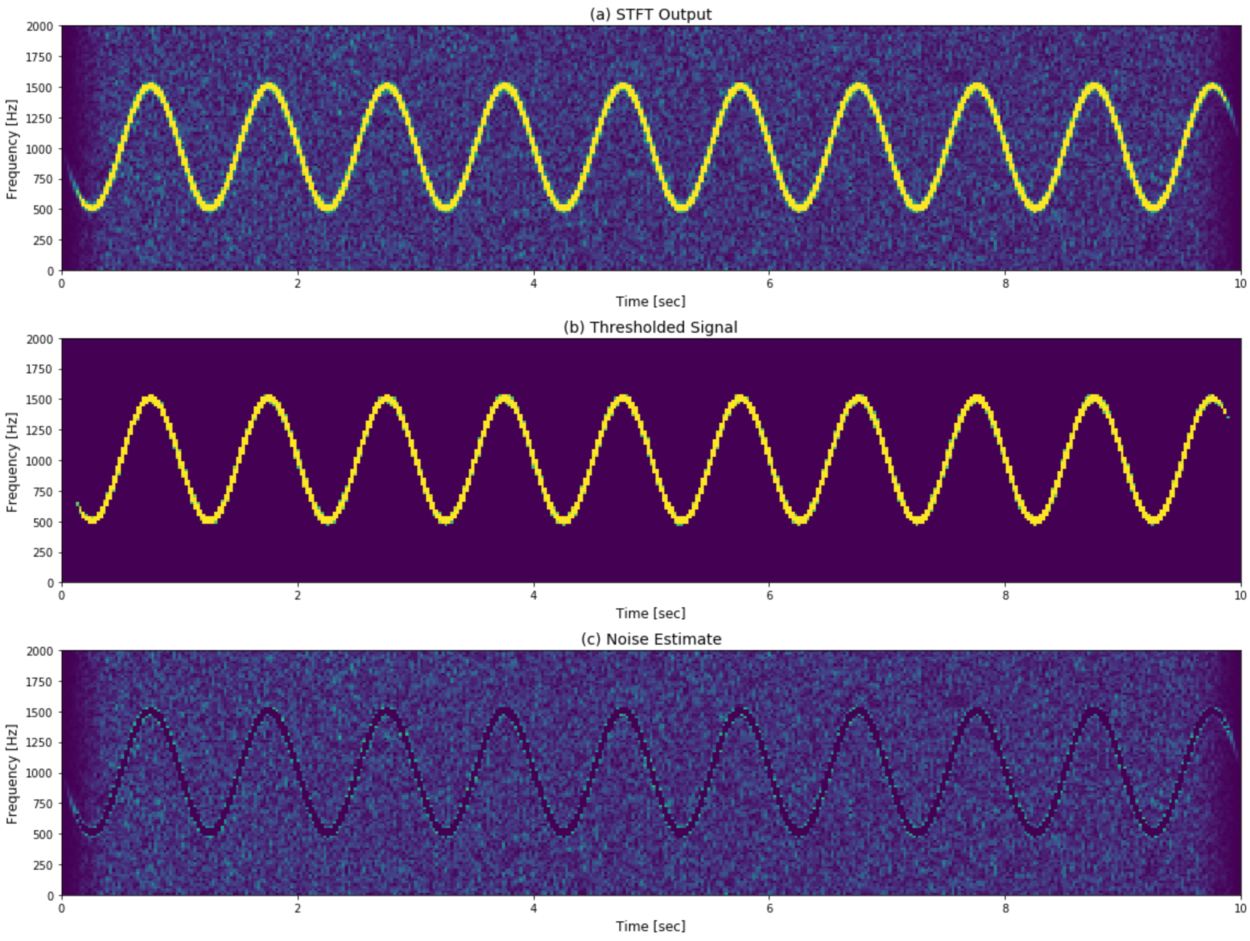


Figure 12. Illustration of thresholding as a denoising technique. (a) STFT output of the signal. (b) Result from (a) after thresholding. (c) Estimate from the subtracted thresholded noise.

Let's now reconstruct the signal using the **inverse short-time Fourier Transform (ISTFT)**. This can be achieved with the *signal.istft* function (that also requires passing the window length in samples and the overlap).

The figure shows a zoom in on a section of the signals between $t = 4.3$s and $t = 4.7$s. The top panel shows the **original signal**, which demonstrably is showing frequency modulation behaviour. The second panel shows the **noise-corrupted signal** as described above. The third panel shows the **reconstructed signal** after applying thresholding and the ISTFT. We see that the general shape is much closer to the expected result; however, there are longer wavelength variations (i.e., with a 20-40 Hz period) that are indicating that we haven't done a perfect job.

The final panel shows the **residual estimated noise** that is the difference between the middle two panels. The noise appears as quite high frequency, which is consistent with the frequency spread of noise that we saw in the ISTFT panels.

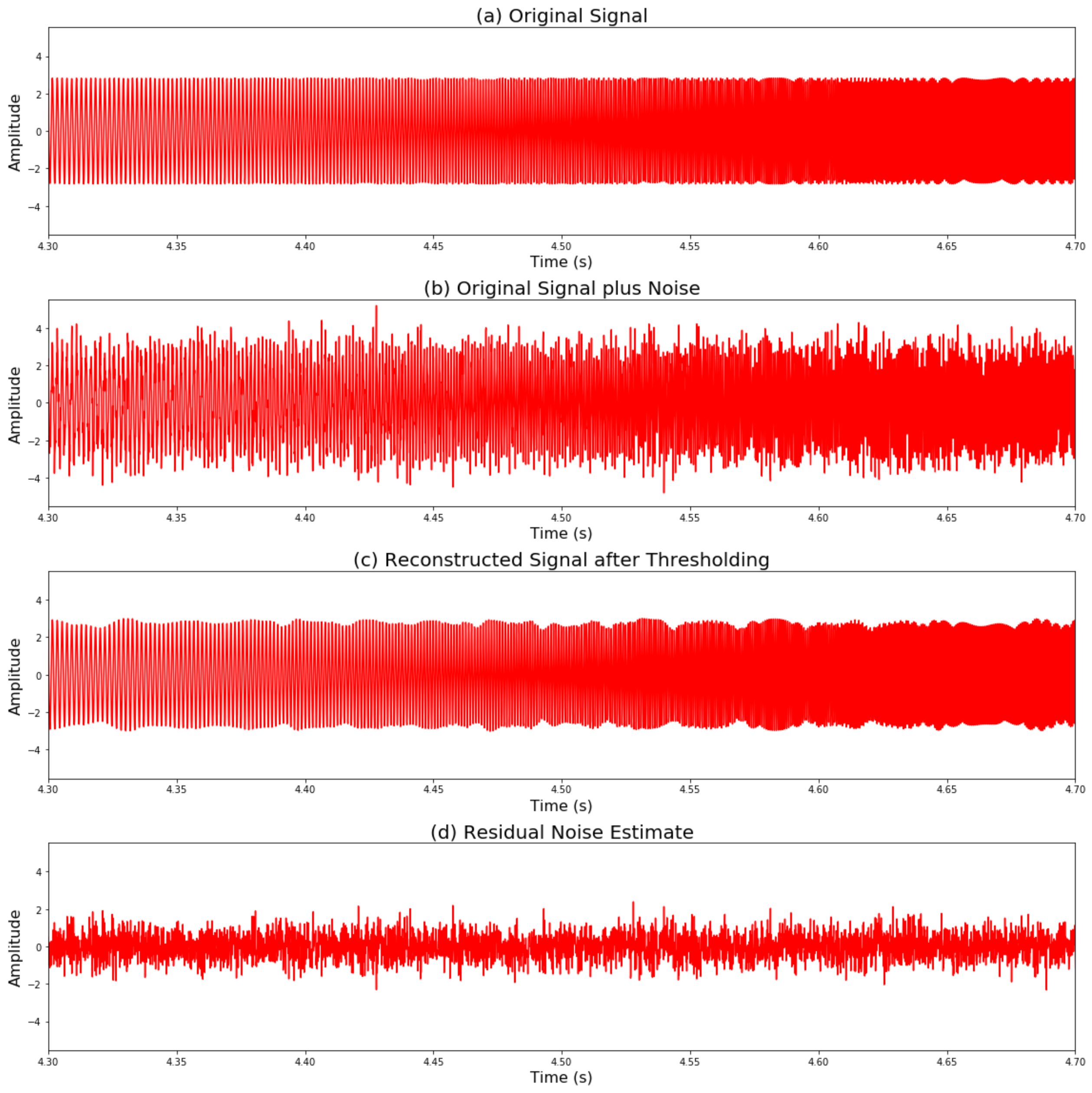


Figure 13. Looking at the denoising process in the time domain. (a) Original signal. (b) Original signal plus noise. (c) Estimated reconstructed signal after thresholding and reconstrution. (d) Noise estimate.

Let's now listen again to the original signal:

```
Out[23]:
```

0:00 / 0:09

as well as the noisy signal:

```
Out[24]:
```

0:00 / 0:09

Now's let's listen to the reconstructed signal recovered through the STFT-thresholding-ISTFT process:

```
Out[25]:
```

0:00 / 0:09

This now seems to be pretty noise free (even if the low frequencies aren't perfectly reconstructed!) Finally, let's listed to the residual noise estimate to hear if any of our signal is present.

```
Out[26]:
```

0:00 / 0:09

Overall, the thresholding operation seems to have done a pretty good job ... and there isn't much signal in the residual (which is what we want)!