Deconvolution Project

February 27, 2023

0.0.1 Deconvolution Project

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal
import numpy as np
```

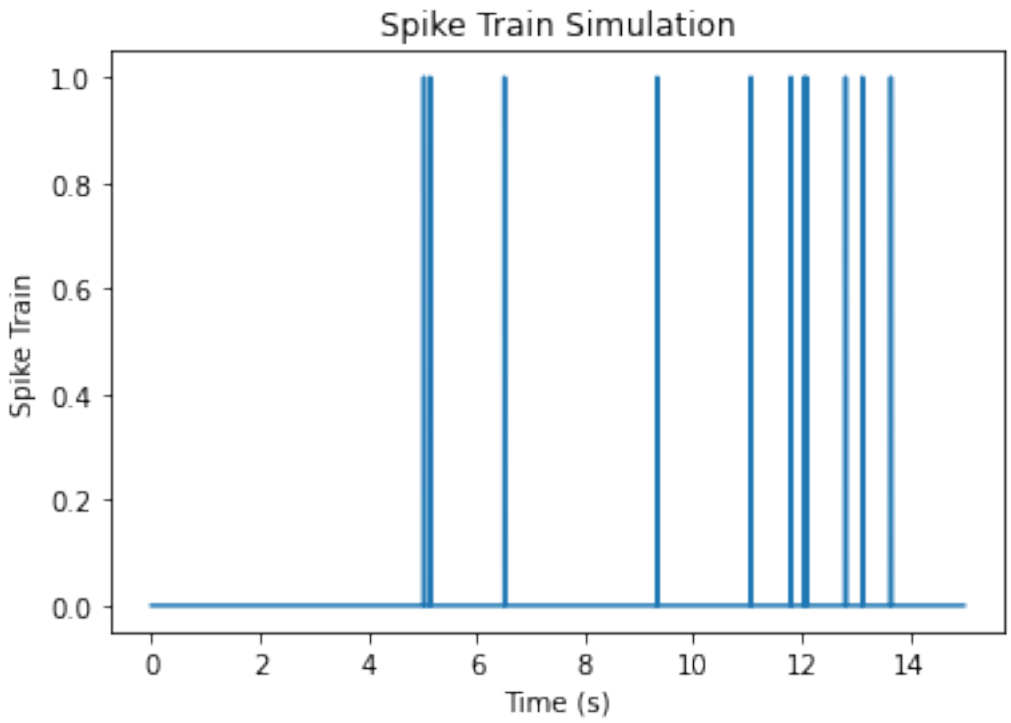
Simulation of a spike train of duration T=15 s, sampled at Fs=2.0 kHz

where the probability of a spike occurring in any time step is 0.03%.

[3]:
```
T = 15
Fs = 2000
p = 0.0003

spike_train = np.array(np.random.rand(T * Fs) < p, dtype=int)
time = np.arange(T * Fs) / Fs

plt.plot(time, spike_train)
plt.xlabel('Time (s)')
plt.ylabel('Spike Train')
plt.title('Spike Train Simulation')
plt.show()
```

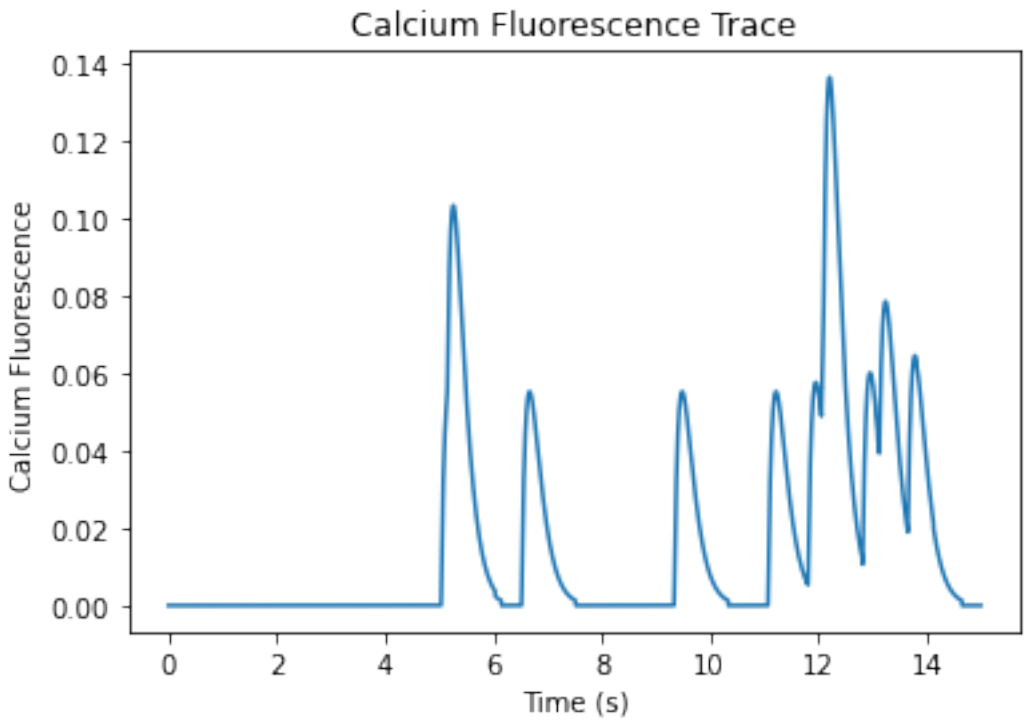


Impulse response function convolved with my spike train.

Resulting simulated calcium fluorescence trace plotted.

[4]:
```
T = 15
Fs = 2000
p = 0.0003
t = np.arange(0, 1, 1/Fs)
h_t = t * (1 - np.exp(-t/.005)) * np.exp(-t/.15)
trace = np.convolve(spike_train, h_t)[:T * Fs]
time = np.arange(T * Fs) / Fs

plt.plot(time, trace)
plt.xlabel('Time (s)')
plt.ylabel('Calcium Fluorescence')
plt.title('Calcium Fluorescence Trace')
plt.show()
```

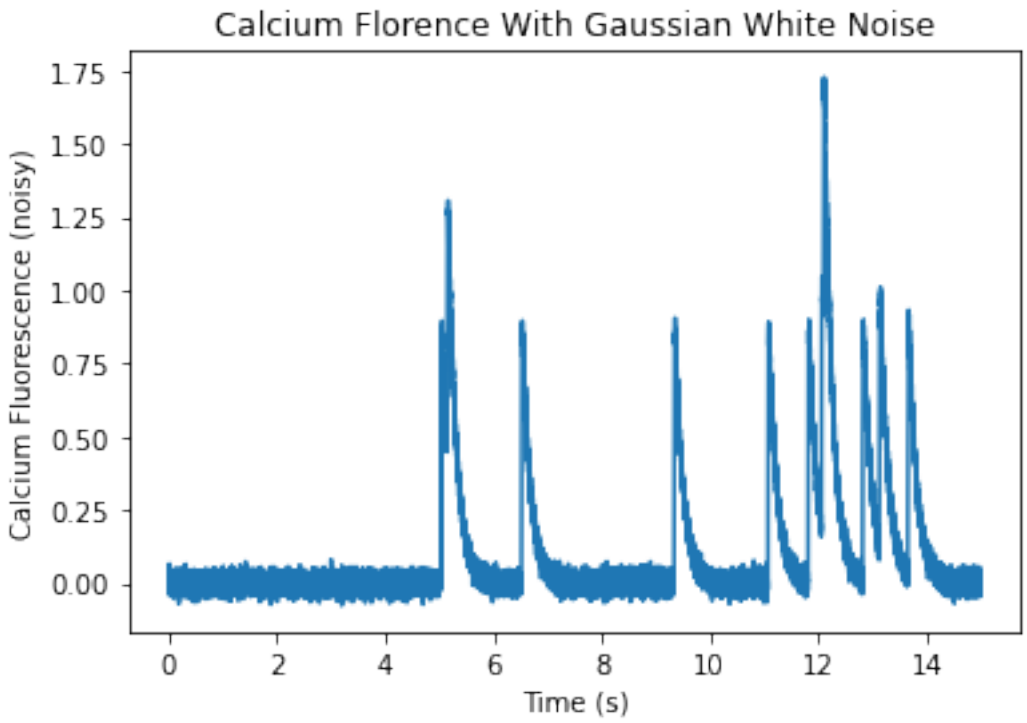


Gaussian white noise added to my signal

[5]:
```
T = 15
Fs = 2000
p = 0.0003

t = np.arange(0, 1, 1/Fs)
h_t = np.heaviside(t, 1) * (1 - np.exp(-t/.005)) * np.exp(-t/.15)
trace = np.convolve(spike_train, h_t)[:T * Fs]
ctn = trace + np.random.normal(0, 0.02, (T*Fs))
time = np.arange(T * Fs) / Fs

plt.plot(time, ctn)
plt.xlabel('Time (s)')
plt.ylabel('Calcium Fluorescence (noisy)')
plt.title('Calcium Florence With Gaussian White Noise')
plt.show()
```

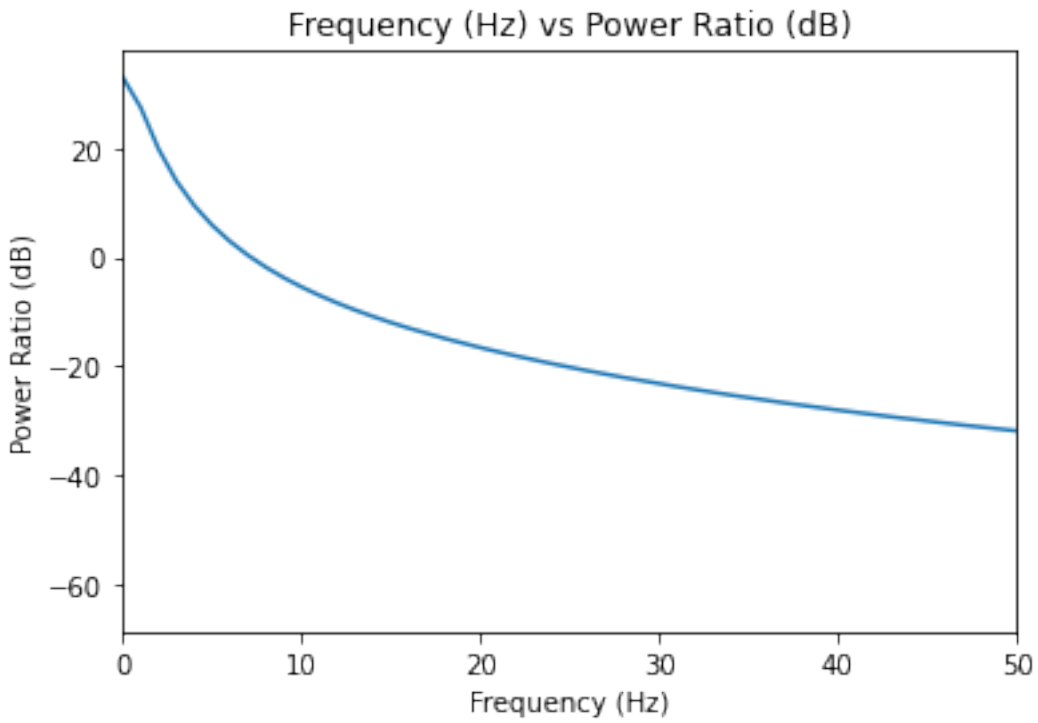


Transfer function calculated using fft

[6]:
```
Fs = 2000

t = np.arange(0, 1, 1/Fs)
h_t = t * (1 - np.exp(-t/.005)) * np.exp(-t/.15)
H = np.fft.fft(h_t)
pr = np.abs(H)**2
prdb = 10 * np.log10(pr)
l = len(h_t)//2
freqs = np.fft.fftfreq(len(h_t), 1/Fs)[:l]

plt.plot(freqs, prdb[:l])
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power Ratio (dB)')
plt.title('Frequency (Hz) vs Power Ratio (dB)')
plt.xlim(0,50)
plt.show()
```

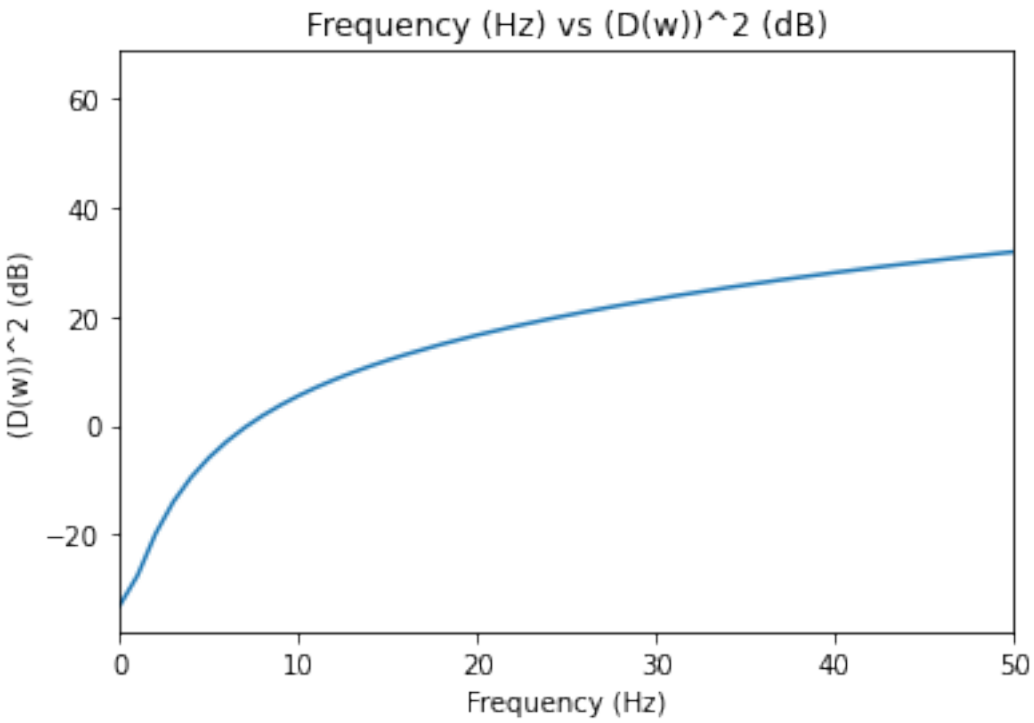


Plotting the magnitude of the deconvolution transfer function for the calcium impulse response

[123]:
```
Fs = 2000

t = np.arange(0, 1, 1/Fs)
h_t = t * (1 - np.exp(-t/.005)) * np.exp(-t/.15)
H = np.fft.fft(h_t)
D = 1 / H
pr = np.abs(D)**2
prdb = 10 * np.log10(pr)
l = len(h_t)//2
freq = np.fft.fftfreq(len(h_t), 1/Fs)[:l]

plt.plot(freq, prdb[:l])
plt.xlabel('Frequency (Hz)')
plt.ylabel('(D(w))^2 (dB)')
plt.title('Frequency (Hz) vs (D(w))^2 (dB)')
plt.xlim(0,50)
plt.show()
```



Noisy calcium signal from above deconvolved

[124]:
```
T = 15
Fs = 2000
p = 0.0003

t = np.arange(0, 1, 1/Fs)
h_t = np.heaviside(t, 1) * (1 - np.exp(-t/.005)) * np.exp(-t/.15)
trace = np.convolve(spike_train, h_t)[:T * Fs]
ctn = trace + np.random.normal(0, 0.02, (T*Fs))
time = np.arange(T * Fs) / Fs


H = np.fft.fft(h_t, n=T * Fs)
D = 1/H


ctd = np.fft.ifft(np.fft.fft(ctn) * D)[:T * Fs]


plt.plot(time, ctd)
plt.xlabel('Time (s)')
plt.ylabel('Calcium fluorescence (Deconvolved)')
plt.title('Time (s) vs Calcium Fluorescence (Deconvolved)')
```

```
plt.show()
```

```
/opt/conda/lib/python3.9/site-packages/matplotlib/cbook/__init__.py:1333:
ComplexWarning: Casting complex values to real discards the imaginary part
  return np.asarray(x, float)
```