A3b: Filtering

Finn Hittson - fxh157
CSDS 464
Due: March 6th, 2023

1 Filtering

1a. A moving average filter

Part (A):
$y_M[n]$ has the form

$$y_M[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k] = \frac{1}{M}(x[n] + x[n-1] + \cdots + x[n-M+1])$$

$y_{M-1}[n-1]$ has the form

$$y_{M-1}[n-1] = \frac{1}{M-1}\sum_{k=0}^{M-2} x[n-k-1] = \frac{1}{M}(x[n-1] + x[n-2] + \cdots + x[n-M+1]$$

Distribute the $\frac{1}{M}$ in $y_M[n]$.

$$y_M[n] = \frac{x[n]}{M} + \frac{x[n-1]}{M} + \cdots + \frac{x[n-M+1]}{M}$$

Set $1-\lambda = \frac{1}{M}$. This makes $\lambda = \frac{M-1}{M}$. Reformat the expression.

$$y_M[n] = (1-\lambda)x[n] + \frac{1}{M}\sum_{k=0}^{M-1} x[n-k-1] = (1-\lambda)x[n] + y_M[n-1]$$

Note that $\lambda y_{M-1}[n-1] = y_M[n-1]$ so making this substitution gives us the desired result.

$$y_M[n] = \lambda y_{M-1}[n-1] + (1-\lambda)x[n]$$

Part (B):
Write the derived expression in the form of summations.

$$\frac{1}{M}\sum_{k=0}^{M-1} x[n-k] = \lambda\frac{1}{M-1}\sum_{k=0}^{M-2} x[n-k-1] + (1-\lambda)x[n]$$

Approximate $\frac{1}{M-1}\sum_{k=0}^{M-2} x[n-k-1] \approx \frac{1}{M}\sum_{k=0}^{M-1} x[n-k-1]$ and simplify the expression.

$$\frac{1}{M}\sum_{k=0}^{M-1} x[n-k] \approx \lambda\frac{1}{M}\sum_{k=0}^{M-1} x[n-k-1] + (1-\lambda)x[n]$$

Let $y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k]$ to get our desired result.

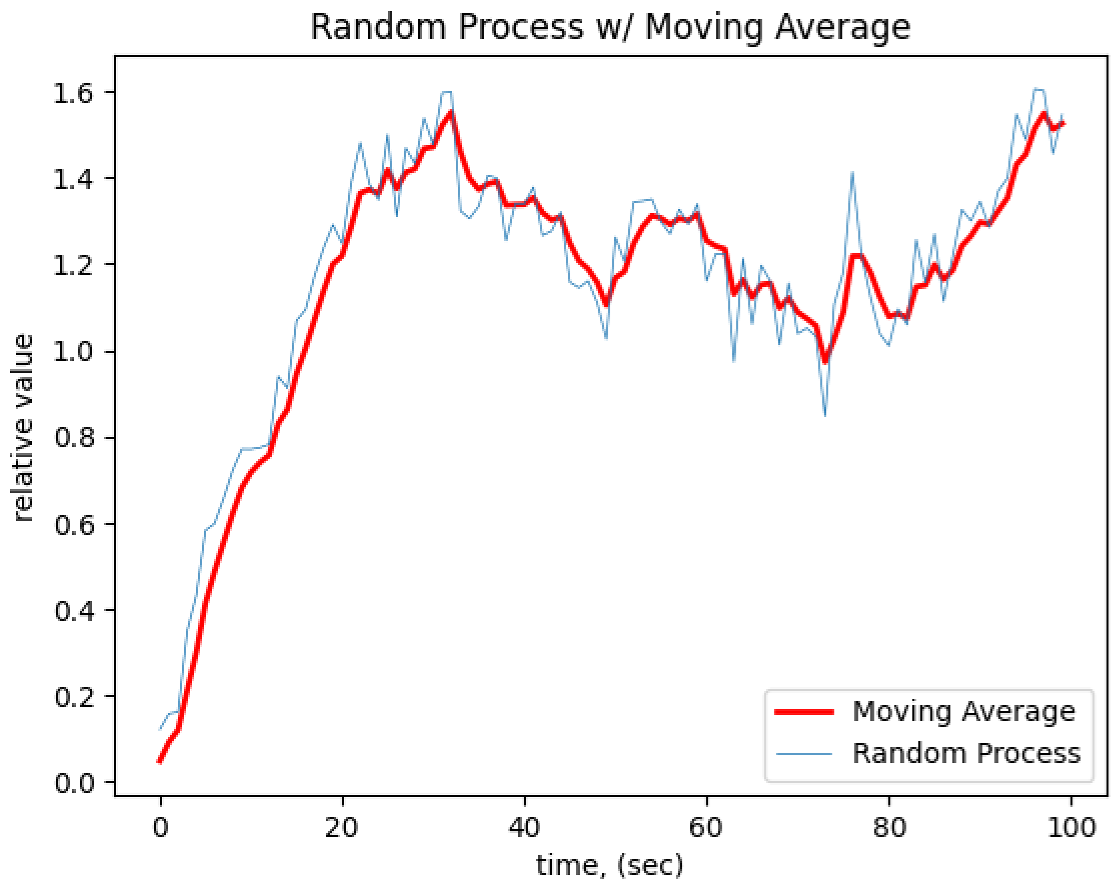
$$y[n] = \lambda y[n-1] + (1-\lambda)x[n]$$

Part (C):
When λ is increased, this causes $y[n-1]$ to become closer in value to $y[n]$ and has the opposite effect when λ is decreased.

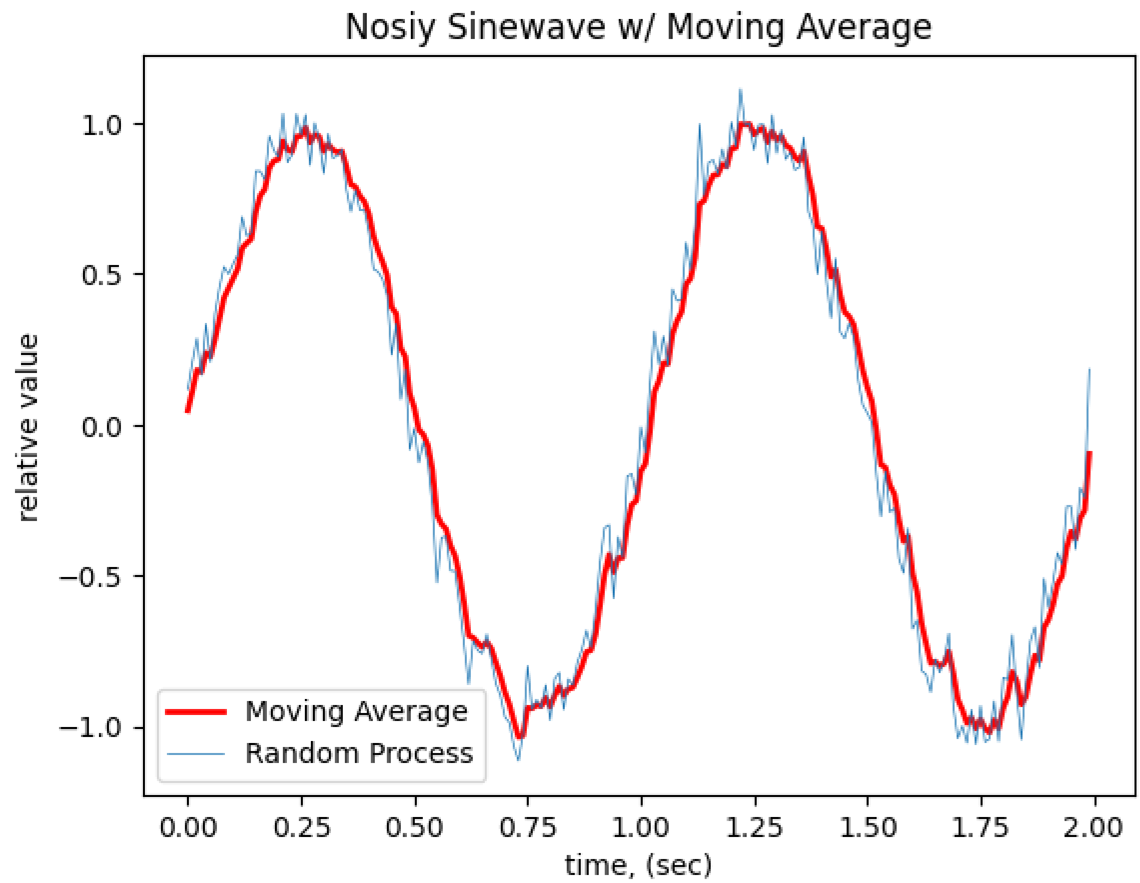
1b. Implementation

In [ ]:
```
import A3b_fxh157 as a3b
import numpy as np

import sys
sys.path.append('../464-A3a_fxh157_files/')
import A3a_fxh157 as a3a

sys.path.append('../464-A1b_fxh157_files/')
import A1b_fxh157 as a1b
```

In [ ]:
```
rand = a3b.randprocess(N=100, s=0.1)
avg = a3b.movingavg(x=rand, l=0.6)
a3b.plot_movingavg(rand=rand, avg=avg)
```



In [ ]:
```
t, x, n = a3a.noisysignal(t=0, g=a1b.sinewave, fs=100, tau=0, T=2, s=0.1, tscale=1,
avg = a3b.movingavg(x+n, l=0.6)
a3b.plot_movingavg(rand=x+n, avg=avg, t=t, title="Nosiy Sinewave w/ Moving Average"
```

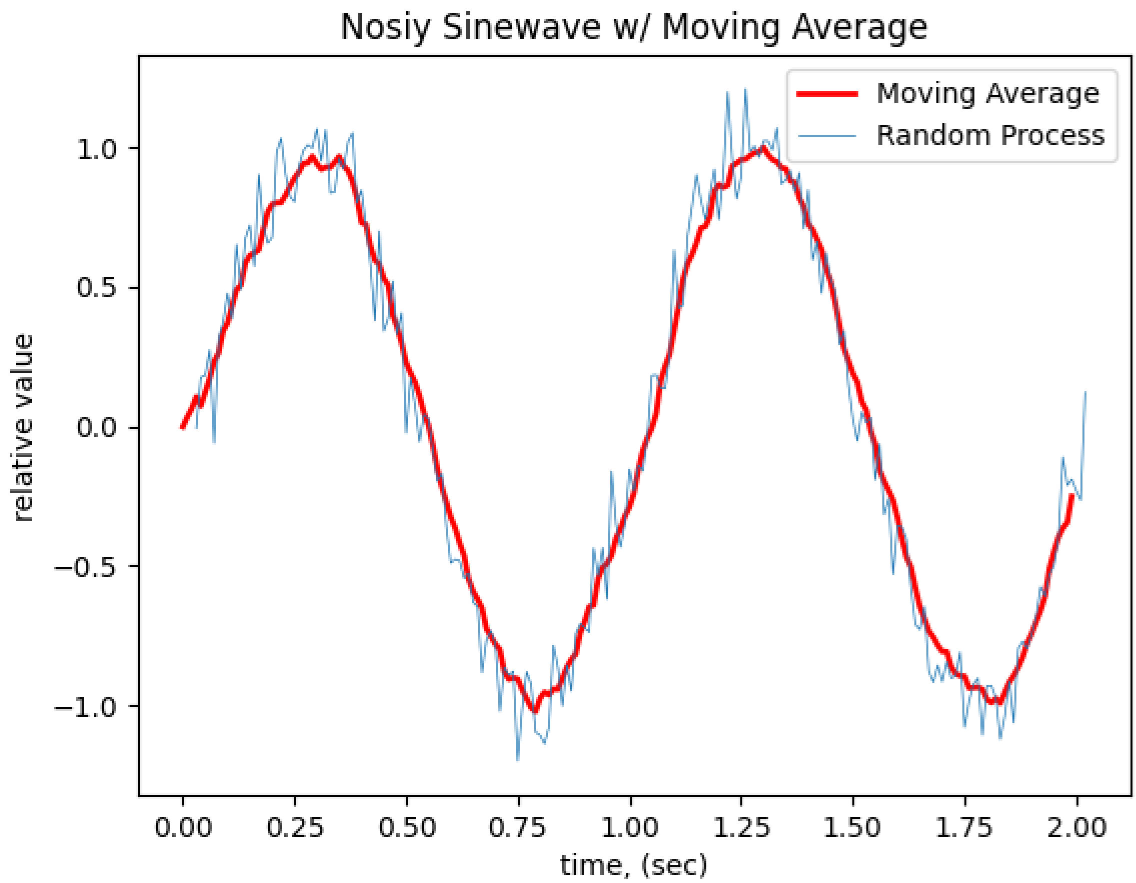


1c. System delay

There is a system delay because the moving average consistes of elements prior to the current location of the signal. Therefore for the moving average to reflect the signal's true location it needs the averaged elements to be centered on that specific location which only occurs in a later frame that is averaged.

In [ ]:
```
t, x, n = a3a.noisysignal(t=0, g=a1b.sinewave, fs=100, tau=0.0, T=2, s=0.1, tscale=
avg = a3b.movingavg(x+n, l=0.8)
print(len(avg))
a3b.plot_movingavg(rand=x+n, avg=avg, t=t, shift=0.03, title="Nosiy Sinewave w/ Mov
```

```
200
```



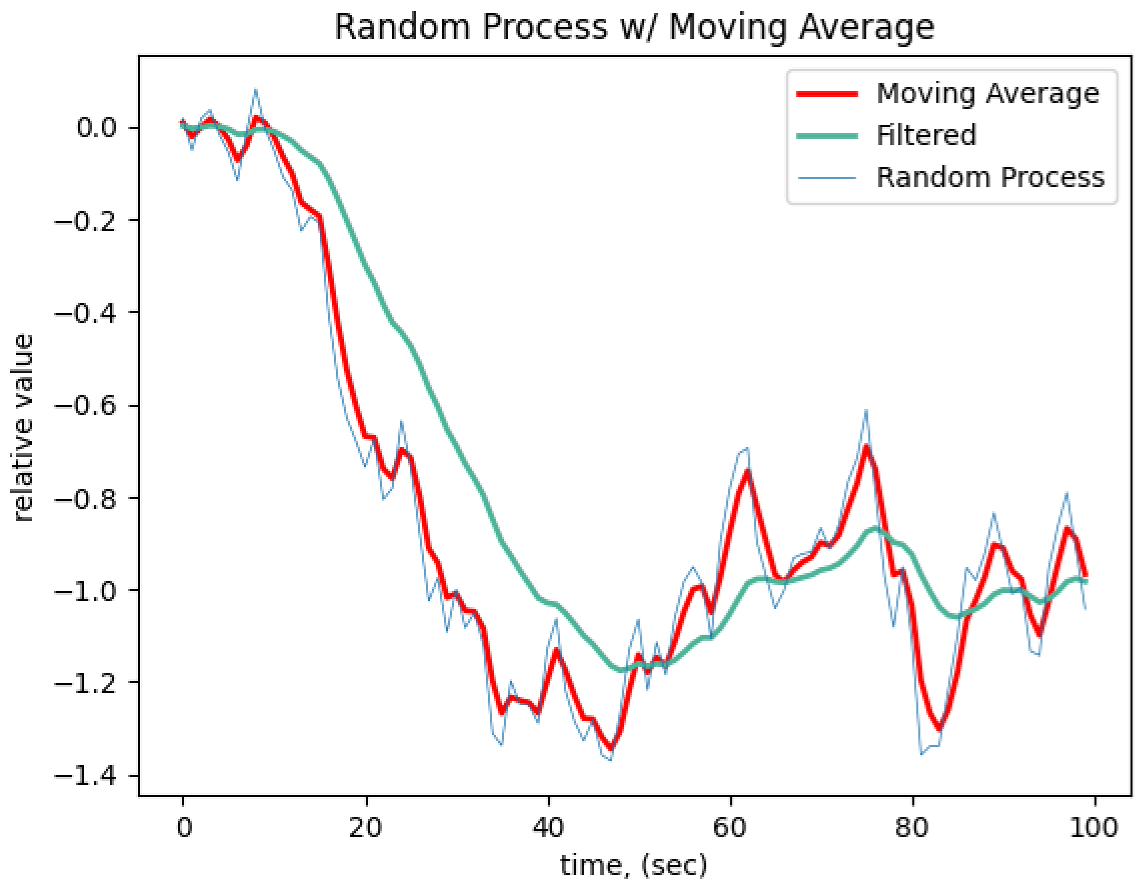
2. IRR Filters

2a. Implementation

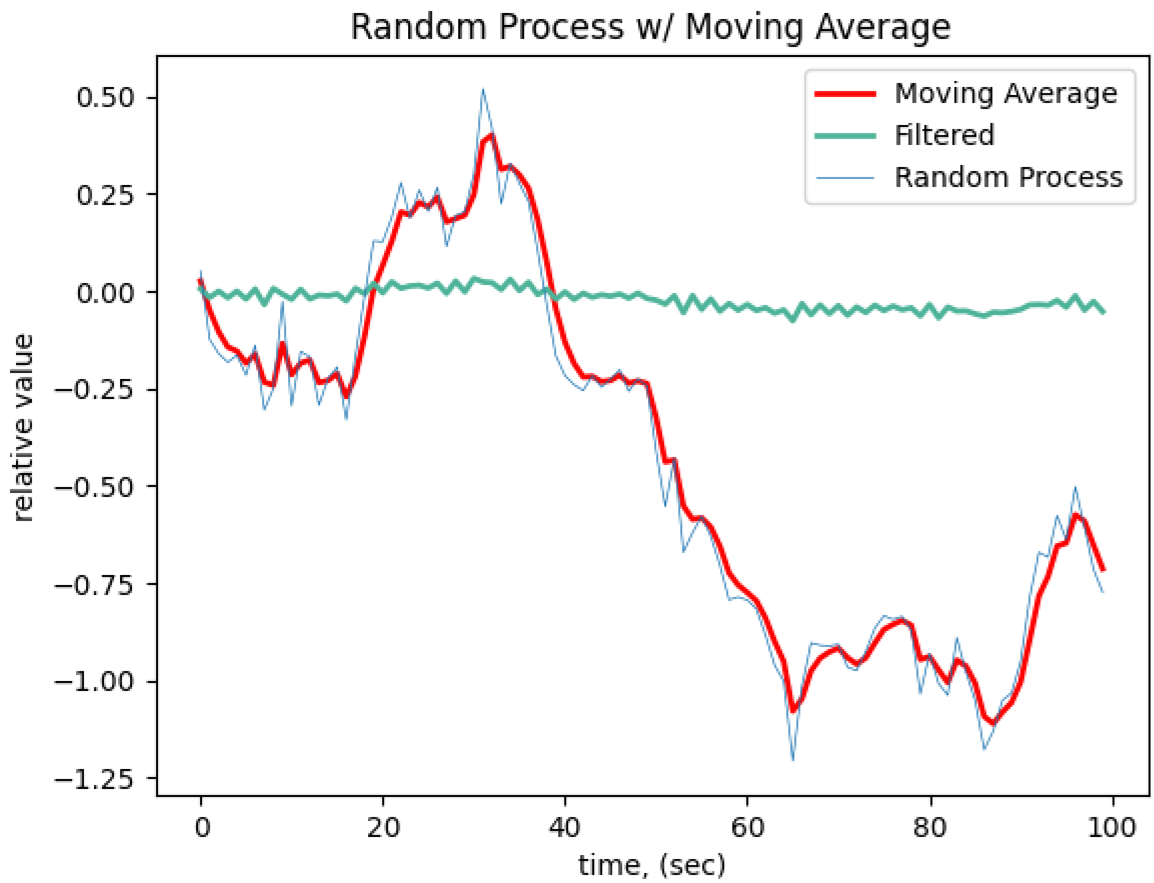
In [ ]:
```
rand = a3b.randprocess(N=100, s=0.1)
a = [-1.265, 0.81]
b = [0.135]
filtered = a3b.filterIIR(x=rand, a=a, b=b)
```

2b. First order low- and high-pass IIR filters

In [ ]:
```
rand = a3b.randprocess(N=100, s=0.1)
avg = a3b.movingavg(x=rand, l=0.5)
a = [-0.9]
b = [0.1]
filtered = a3b.filterIIR(x=rand, a=a, b=b)
a3b.plot_movingavg(rand=rand, avg=avg, filtered=filtered)
```

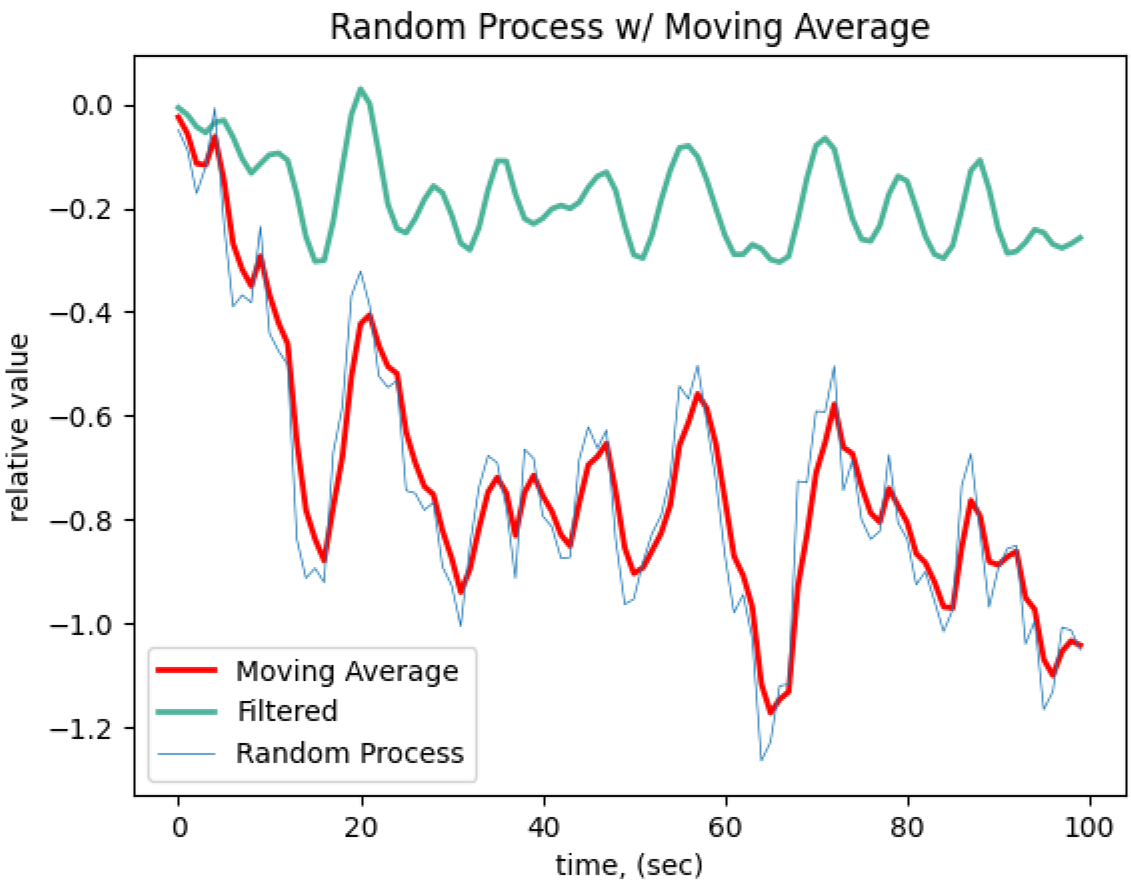


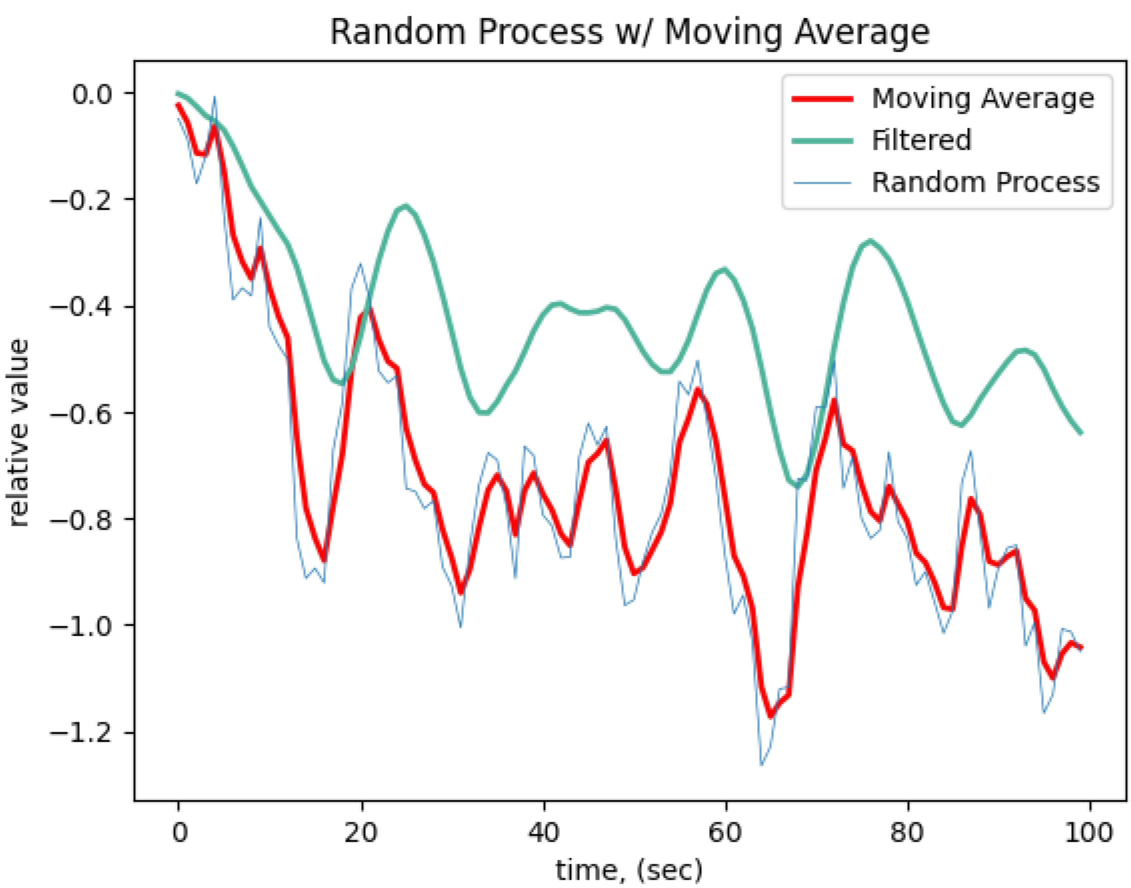
In [ ]:
```
rand = a3b.randprocess(N=100, s=0.1)
avg = a3b.movingavg(x=rand, l=0.5)
a = [0.9]
b = [0.1]
filtered = a3b.filterIIR(x=rand, a=a, b=b)
a3b.plot_movingavg(rand=rand, avg=avg, filtered=filtered)
```



2c. Second order bandpass filters

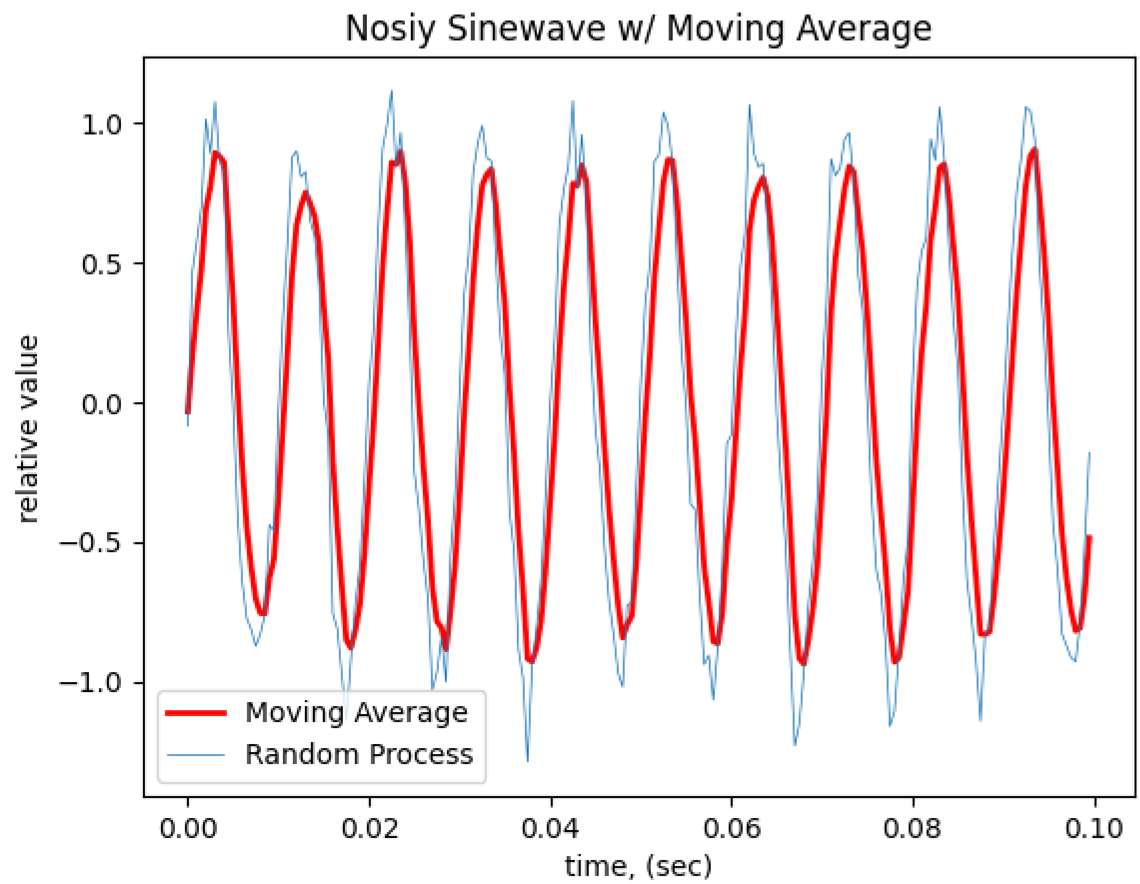
In [ ]:
```
#np.random.seed(10)
rand = a3b.randprocess(N=100, s=0.1)
avg = a3b.movingavg(x=rand, l=0.5)
a = [-1.265, 0.81]
b = [0.135]
filtered = a3b.filterIIR(x=rand, a=a, b=b)
a3b.plot_movingavg(rand=rand, avg=avg, filtered=filtered)
a = [-1.702, 0.81]
b = [0.063]
filtered = a3b.filterIIR(x=rand, a=a, b=b)
a3b.plot_movingavg(rand=rand, avg=avg, filtered=filtered)
```


Random Process w/ Moving Average
relative value
time, (sec)
Moving Average
Filtered
Random Process
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
0
20
40
60
80
100

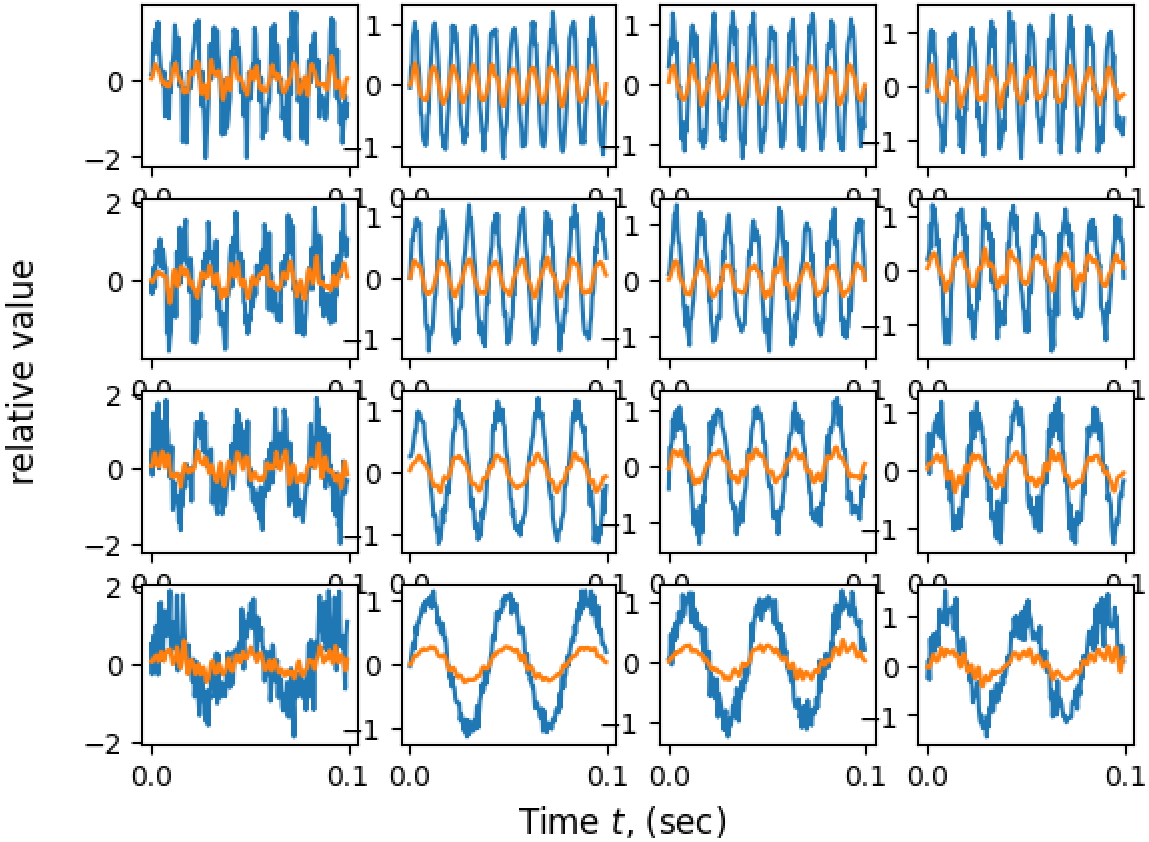

Random Process w/ Moving Average
relative value
time, (sec)
Moving Average
Filtered
Random Process
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
0
20
40
60
80
100

2d. Characterizing the filter response

In [ ]:
```
t, x, n = a3a.noisysignal(t=0, g=a1b.sinewave, fs=2000, tau=0, T=0.1, s=0.1, tscale
avg = a3b.movingavg(x+n, l=0.6)
a3b.plot_movingavg(rand=x+n, avg=avg, t=t, title="Nosiy Sinewave w/ Moving Average"
```



In [ ]:
```
g = a1b.sinewave
a = [-1.265, 0.81]
b = [0.135]
rows = 4
cols = 4
fs = 2000
tau = 0
T = 0.1
tscale = 1.0
s = [0.5, 0.1, 0.15, 0.2]
f = [100, 75, 50, 25]

a3b.plot_filter_grid(g=g, a=a, b=b, rows=rows, cols=cols, fs=fs, tau=tau, T=T, tsca
```



3. The impulse response function

3a. Deriving the impulse response function

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

$$\mathcal{H}\left(x[n]\right) = \mathcal{H}\left(\sum_{k=-\infty}^{\infty} x[k]\delta[n-k]\right)$$

$$\mathcal{H}\left(x[n]\right) = \sum_{k=-\infty}^{\infty} \mathcal{H}\left(x[k]\delta[n-k]\right)$$

Need linearity to factor out constant $x[k]$ from $\mathcal{H}\left(x[k]\delta[n-k]\right)$.

$$\mathcal{H}\left(x[n]\right) = \sum_{k=-\infty}^{\infty} x[k]\mathcal{H}\left(\delta[n-k]\right)$$

Need time invariance since the delta function operates with and without a time delay. Therefore $\mathcal{H}\left(\delta[n-k]\right) = h[n-k]$.

$$\mathcal{H}\left(x[n]\right) = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

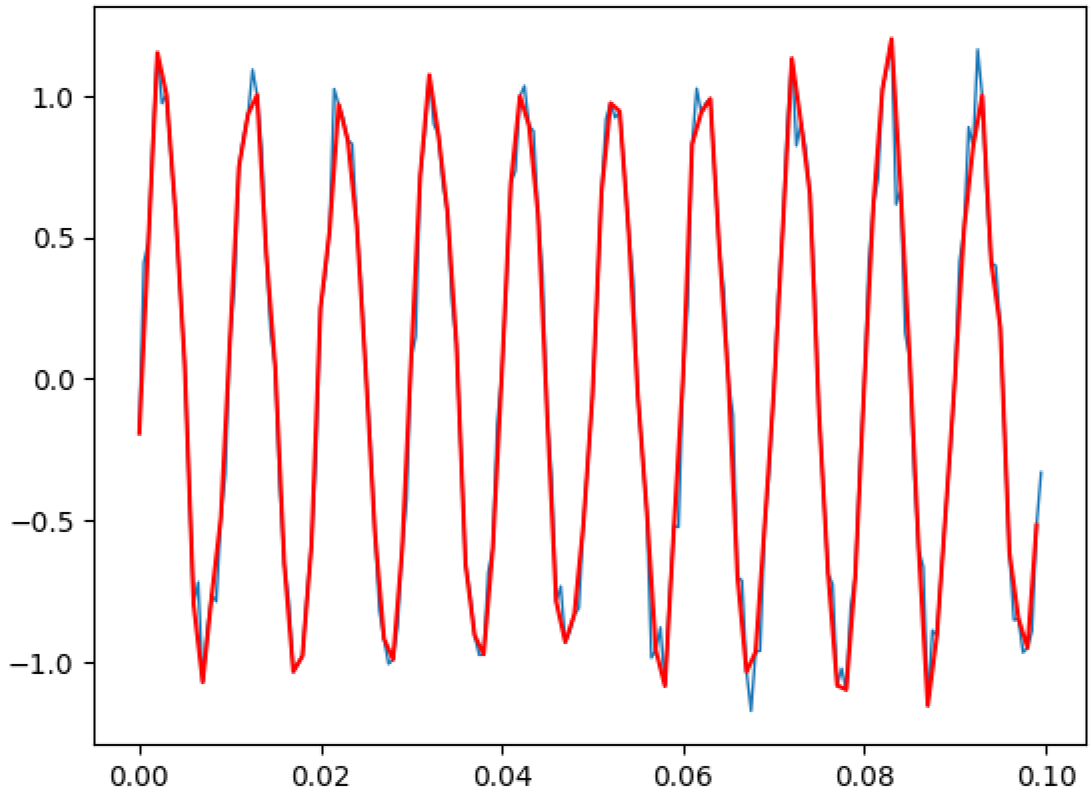
$$ h[n]=

$$\begin{cases} \text{undefined} & n = 0 \\ 0 & n \neq 0 \end{cases}$$

3b. Impulse responses

In [ ]:
```
import matplotlib.pyplot as plt

def impulse(x, fs, tscale, f):
    y = []
    t = []
    idx = 0
    while idx < len(x):
        t.append(idx/len(x)/fs)
        y.append(x[int(idx)])
        idx += f/fs*tscale
    return t, y

def plot_impulse(t, y, t0=None, x0=None):
    if x is not None and t0 is not None:
        plt.plot(t0, x0, '#1f77b4', linewidth=1)
    plt.plot(t, y, 'r')
    '''
    for i in range(len(y)):
        plt.plot([t[i], t[i]], [0, y[i]], 'r')
        plt.scatter(t[i], y[i], c='r', s=10, zorder=10)
    '''
    plt.show()
```

In [ ]:
```
t0, x0, n0 = a3a.noisysignal(t=0, g=a1b.sinewave, fs=2000, tau=0, T=0.1, s=0.1, tsc
avg = a3b.movingavg(x0+n, l=0.6)
#a3b.plot_movingavg(rand=x0+n0, avg=avg, t=t0, title="Nosiy Sinewave w/ Moving Aver

t, y = impulse(x=x0+n0, fs=10, tscale=0.01, f=2000)
plot_impulse(t=t, y=y, t0=t0, x0=x0+n0)
```

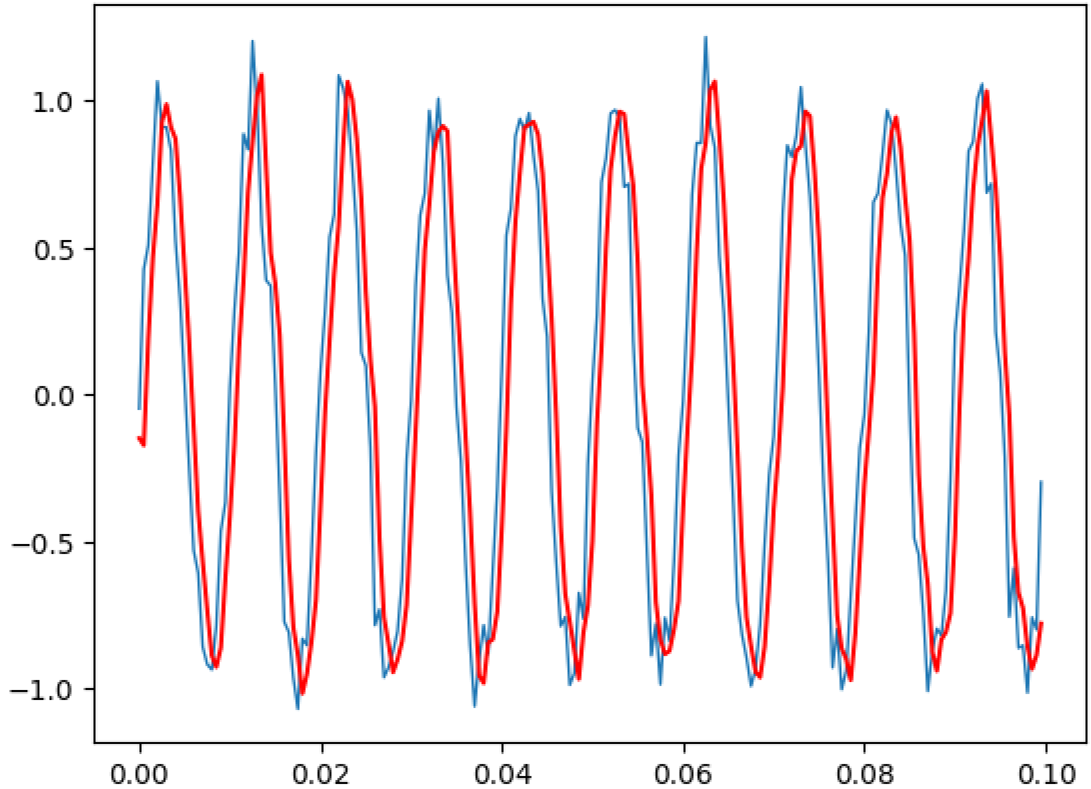


4. Filtering with convolution

4a. Implementing convolution

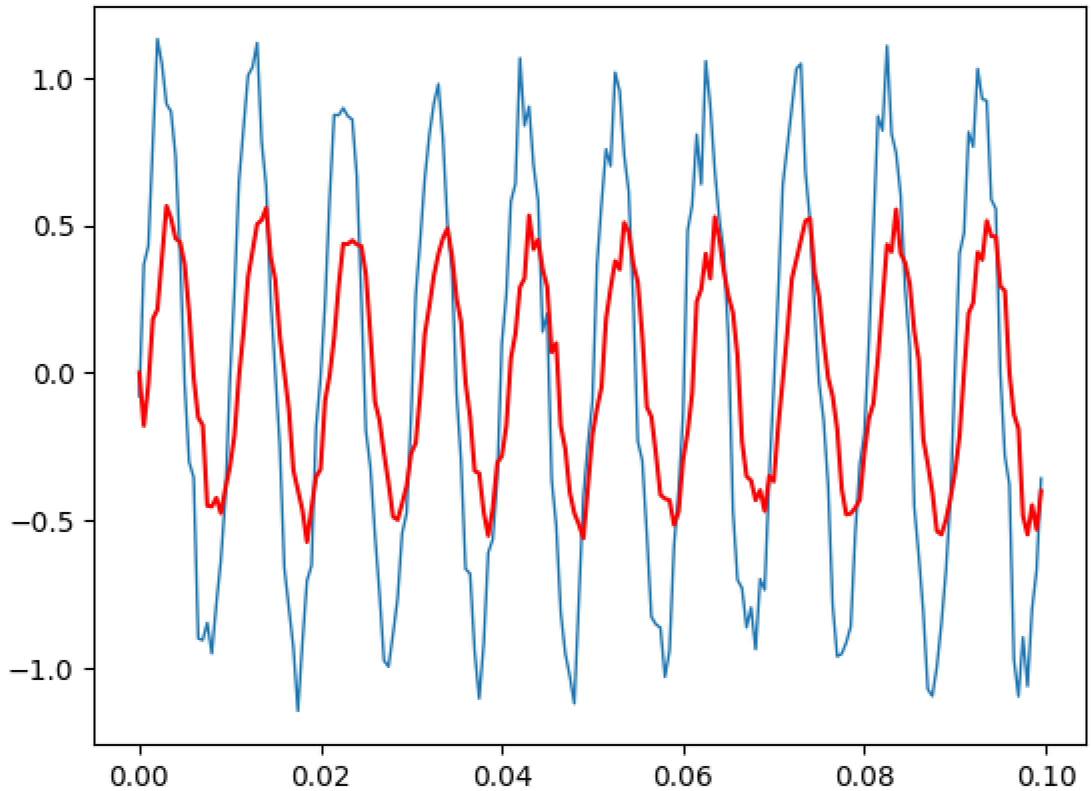
In [ ]:
```
def d(x):
    return 1 if x == 0 else 0

def convolve(x, h, h0:int=1):
    y = []
    if h0 == 1: # causal filter: x[k<=n]
        for n in range(0, len(x)):
            y.append(0)
            for k in range(len(h)):
                if n-k >= 0:
                    y[-1] += x[n-k]*h[-1-k]

    elif h0 == 0: # non causal filter: x[k>n]
        for n in range(0, len(x)):
            y.append(0)
            for k in range(len(h)):
                if n-k >= 0:
                    y[-1] += x[n-k-len(h)//2]*h[-1-k]
    return y
```

In [ ]:
```
t0, x0, n0 = a3a.noisysignal(t=0, g=a1b.sinewave, fs=2000, tau=0, T=0.1, s=0.1, tsc
y = convolve(x=x0+n0, h=[0.5, 0.5], h0=0)
plot_impulse(t=t0, y=y, t0=t0, x0=x0+n0)
```

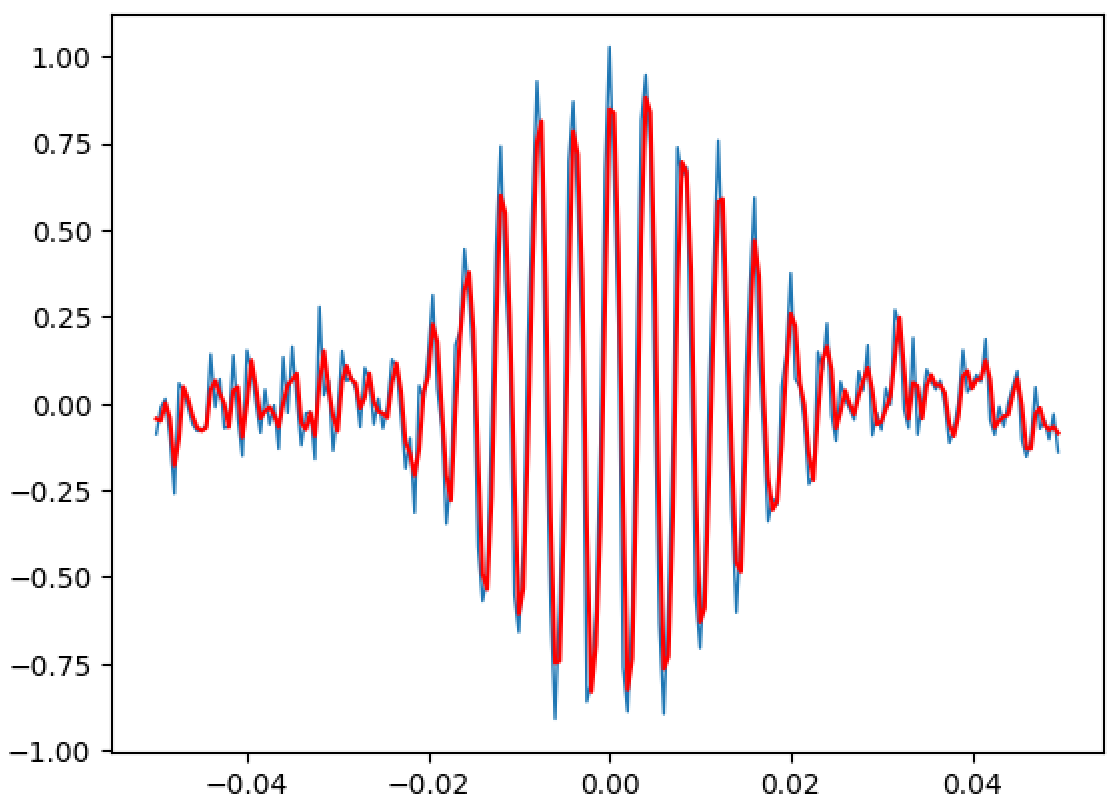


4b. FIR filtering

In [ ]:
```
t0, x0, n0 = a3a.noisysignal(t=0, g=a1b.sinewave, fs=2000, tau=0, T=0.1, s=0.1, tsc
y = convolve(x=x0+n0, h=[0.5, 0], h0=0)
plot_impulse(t=t0, y=y, t0=t0, x0=x0+n0)
```



In [ ]:
```
t0, x0, n0 = a3a.noisysignal(t=-0.05, g=a1b.gabore, fs=2000, tau=0.0, T=0.05, s=0.1
y = convolve(x=x0+n0, h=[0.5, 0.5], h0=1)
plot_impulse(t=t0, y=y, t0=t0, x0=x0+n0)
```



4c. Using matched filters to detect signals in noise

In [ ]:
```
t0, x0, n0 = a3a.noisysignal(t=-0.05, g=a1b.gabore, fs=2000, tau=0.0, T=0.05, s=0.1
y = convolve(x=x0+n0, h=[1/250, 100/250], h0=1)
plot_impulse(t=t0, y=y, t0=t0, x0=x0+n0)
```