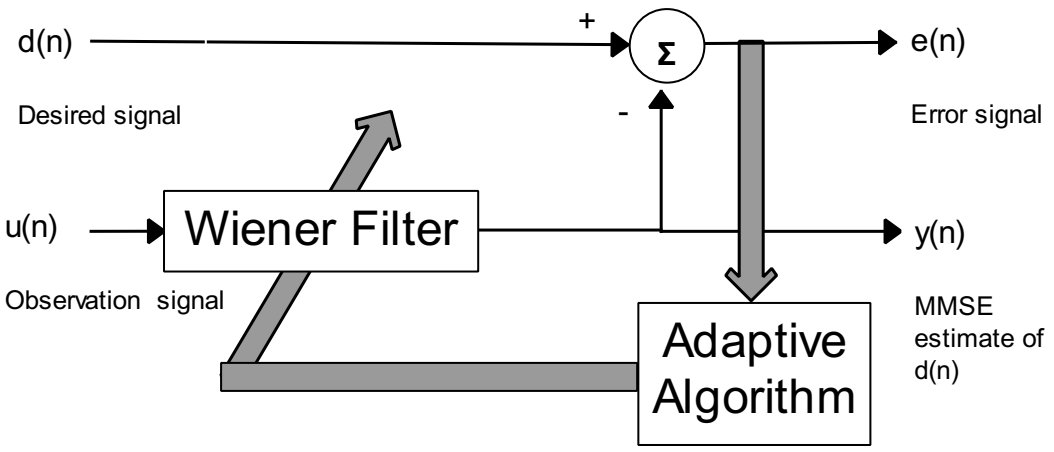
Adaptive filters

The term, adaptive filter, means that the characteristics of a filter is changing in a self automated fashion to obtain the best possible signal quality. Adaptive filters are usually associated with the broader topic of statistical signal processing. An optimum linear filter (FIR or IIR) in the minimum mean square sense can be specifically designed to extract a signal from any noise by minimizing the error signal formed by subtracting the filtered signal from the desired signal. For noisy signals with time varying statistics, this minimization process is often done using an adaptive filter. MatDeck contains the function, lmsadaptive(), which is commonly used for adaptive filtering utilizing the most common LMS algorithm which is explained down below.

A Wiener filter is used as the optimal for stationary input signals. It is defined as a linear FIR filter having a certain number of taps.

$$y(n) = \sum_{i=0}^{M-1} w_i \cdot u(n-i)$$

Here, u(n) is an observation signal potentially corrupted by noise or interference, and w_m are filter taps which are adapted. A block diagram of a adaptive Wiener (FIR) filter is shown below.



In an adaptive Wiener filter the error signal is fed back to the filter weights to adjust them using a steepest-descent algorithm. In a practical situation, implementing this involves estimating the gradient from the available data using the least-mean-square (LMS) algorithm. The LMS algorithm is done using the following steps:

LMS algorithm

Given: - observation signal u(n)
- desired signal d(n)

1. Initialize algorithm : set taps w_m to zero, or to random values.

2. Iterate algorithm for n=0, 1 ... n_max, where n_m is length of given signal.

2.0 Read/generate a new data pair u(n), d(n)

2.1 Filter output $y(n) = \sum_{i=0}^{M-1} w_i \cdot u(n-i)$

2.2 Calculate error e(n)=d(n)-u(n)

2.3 Parameter adaptation $w_m(n+1) = w_m(n) + \mu * u(n-m) * e(n)$

MatDeck Simulation

A simple MatDeck simulation is constructed by using a single sinusoid at a normalized frequency of f_0=0.2 to which we add additive white Gaussian noise. The relation between the amplitude of sinusoid and the standard deviation of the noise is defined as the Signal and Noise Ratio (SNR). The simulation is ran using 1000 samples, SNR equal to 10dB, number of taps is 11, and step size is 0.01.

n_max := 1000 Number of samples of input signal

f0 := 0.1 Normalized frequency of sinusoid

SNR := 10 dB, Signal to Noise Ratio

sigma := sqrt(1/(10^(SNR/10))) Standard deviation of the white noise

u_in := normrandvec(0, sigma^2, n_max) + (sin(2·π·f0·xnodes([0 n_max], n_max-2)))^T

d_in := mat transpose(sin(2·π·f0·xnodes([0 n_max], n_max-2)))

t_in := mat transpose(xnodes([0 n_max], n_max-2)) Time vector

e_out1 := lmsadaptive(u_in, d_in, 0.01, 11) Adaptive filtering using LMS algorithm

u_in - input signal with noise
d_in - desired signal

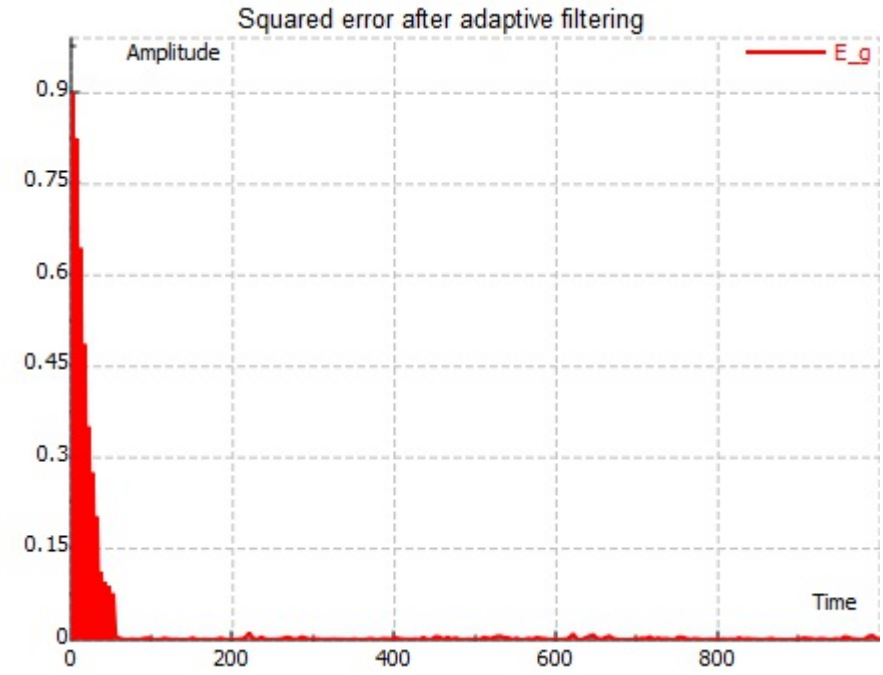
U_g := join mat cols(t_in, u_in)

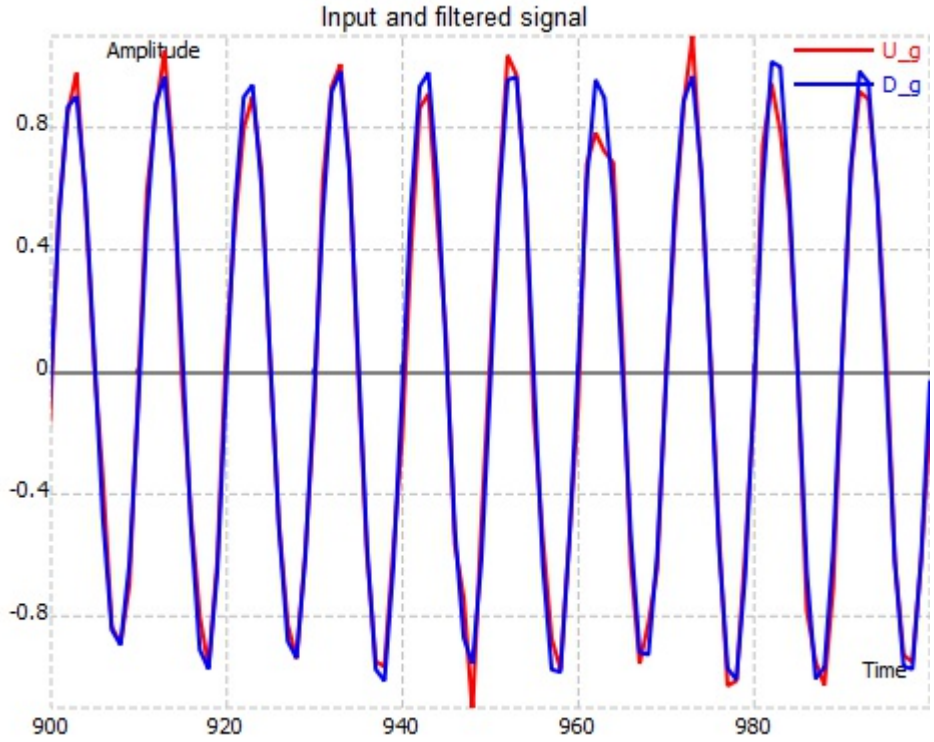
D_g := join mat cols(t_in, e_out1[0])

E_g := join mat cols(t_in, (e_out1[2])^2)

Results

MatDeck's function, lmsadaptive(), takes a input signal and the desired signal, together with Wiener filter length and step size for the LMS algorithm. The function then returns the filtered signal, adaptive filter coefficients, and the error signal. Filtered signals and error signals are of the same length as the input signal. In the next segment, we give the graph of the input signal and filtered signal. In a separate figure, we give the graph of the squared error. Analyzing the graph of the squared error, we can see that the adaptive filter minimizes the error in a steepest descent sense as expected.


Squared error after adaptive filtering
Amplitude
E_g
0.9
0.75
0.6
0.45
0.3
0.15
0
Time
0
200
400
600
800


Input and filtered signal
Amplitude
U_g
D_g
0.8
0.4
0
-0.4
-0.8
Time
900
920
940
960
980