**Abstract**

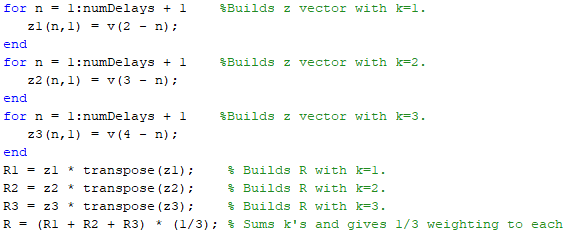
In practical applications, input signals may become corrupted by some noise signal. If the source of the noise (or the noise signal itself) is known, then an adaptive filter may be used to attempt to reconstruct the original, noiseless signal.

This can be accomplished by training the adaptive filter with the noise. Once the adaptive filter is trained against the noise signal, the filter’s output should be approximately the same as the noise added to the original signal. Thus, if the previous statement is true, then the (trained) adaptive filter output subtracted from the corrupted input signal will yield a signal that is approximately the same as the original, uncorrupted input signal.

This project tests how well the adaptive filter can reconstruct an input signal when some known noise is applied. The network is tested using two different noise signals. In both cases, the adaptive filter has a single delay and is trained using the LMS algorithm.

**Method**

First, the input correlation matrix **R** is found:



Which allows the Hessian to be quickly computed ( **A** = 2\***R** ), along with the Eigenvalues and Eigenvectors of the Hessian matrix. From the Eigenvalues of the Hessian, the maximum stable learning rate can be calculated:

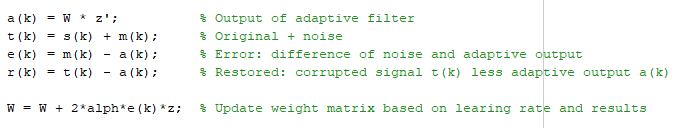


Next, the input/target cross-correlation vector **h** is computed:



Which can be used to calculate the minimum point: **x\*** = **R**\**h** (calculated **x\*** displayed in **Results** section). After, a rough contour is drawn, using **R** and **h** as input data (contour images shown in **Results** section).

Finally, the LMS algorithm is used to train the adaptive filter with 1 delay for noise cancellation. The weight matrix **W** is initialized as [0 0]. Inputs to the adaptive filter (the noise signal) are represented by the vector **z** – for example, at time *k*: **z** = [*v*(k) *v*(k-1)] . Using **W** and **z**, the adaptive filter output a(k) can be found and the LMS algorithm executes as shown below:



Note: Here, alpha is 0.12 as specified by program instruction (a). Using a different alpha is discussed in the **Conclusion** section of this report.

**Results**

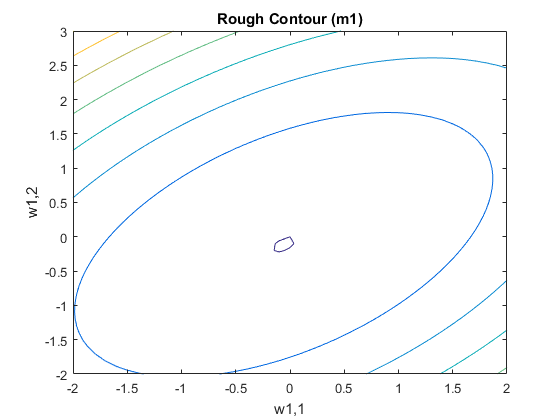
For m1(k) = 0.12sin(2\*pi\*k/3 + pi/2) :

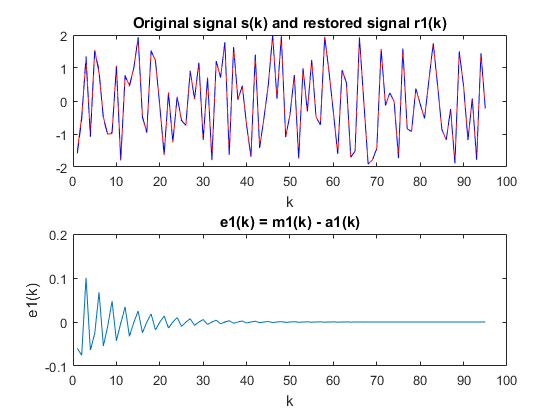
1. Eigenvalues and Eigenvectors of Hessian/Calculated minimum point x1:

eVals = 0.7200 eVecs = -0.7071 -0.7071 x1 = -0.0577

2.1600 -0.7071 0.7071 -0.1155

Rough contour plot:



1. Maximum stable learning rate: 0.9259
2. Plot original (red) and restored (blue) signals:
3. Comment on why restored is not exactly the same as original:

The restored signal differs slightly from the original signal because it takes some time for the adaptive filter to be trained with the noise signal. During this training period, the output of the adaptive filter may not closely resemble the noise signal. Therefore, when the output of the filter is subtracted from the corrupted signal, the resulting signal may not closely resemble the original signal. Note that after the adaptive filter has completed training, the restored signal is almost exactly the same as the original signal. This is easily verified by examining the graph of e1(k) above, where k >= 50.

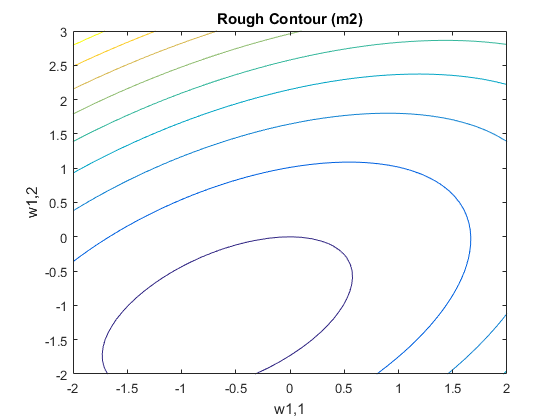
For m2(k) = 1.2sin(2\*pi\*k/3 – 3\*pi/2) :

1. Eigenvalues and Eigenvectors of Hessian/Calculated minimum point x2:

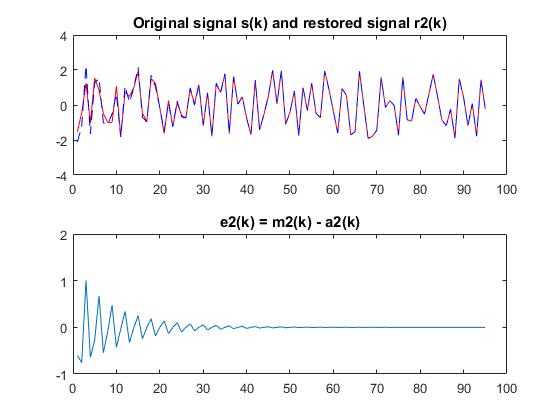
eVals = 0.7200 eVecs = -0.7071 -0.7071 x2 = -0.5774

2.1600 -0.7071 0.7071 -1.1547

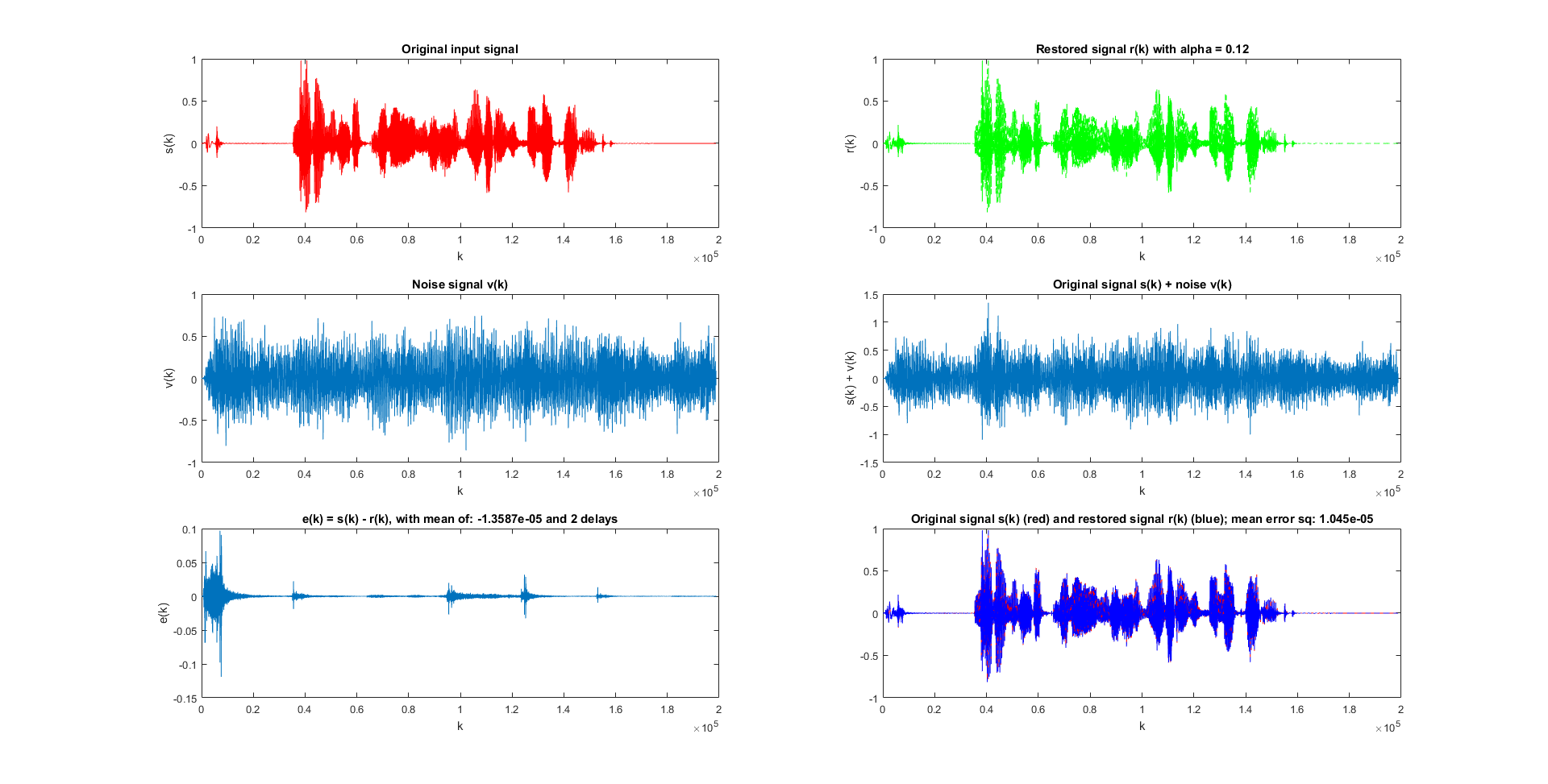
Rough contour plot:



1. Maximum stable learning rate: 0.9259
2. Plot original (red) and restored (blue) signals:

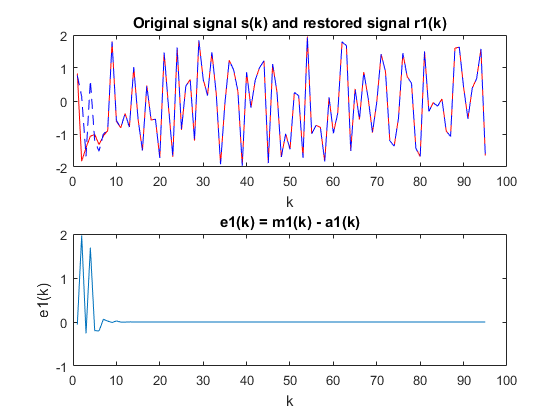


For recorded sound, noise:

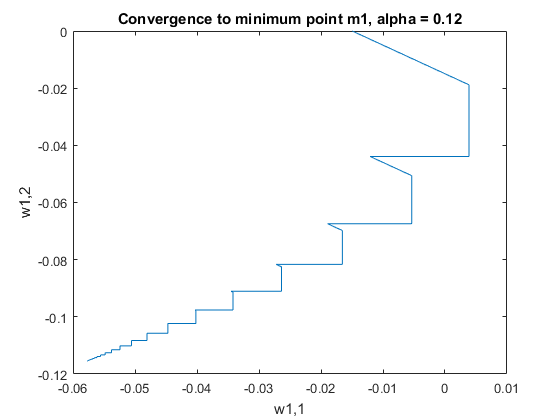
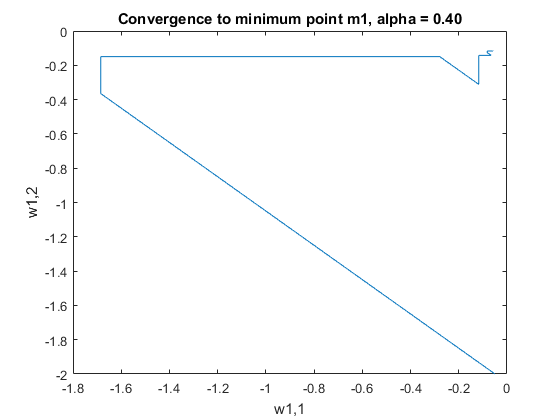
1. Below are the plots obtained from using the program on recorded voice and noise input sound files. These sound files, along with the computed corrupt and filtered signals, can be played on request during the presentation/oral exam for this project. The plot below uses a learning rate of alpha = 0.12 and the adaptive filter has 2 total delays. The effects of changing alpha and the number of delays in the filter is discussed in the **Conclusion** section of this report.

**Conclusion**

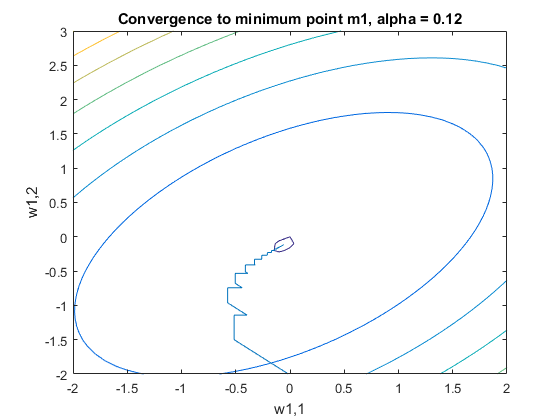
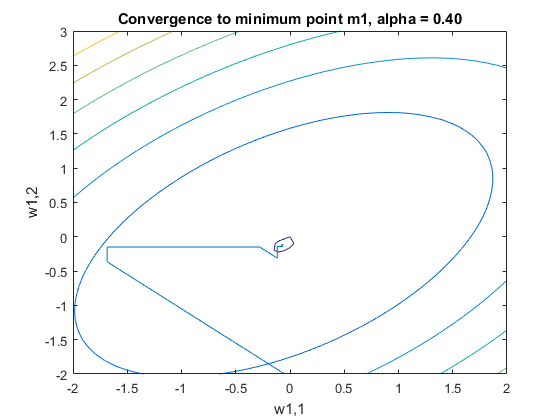
For the first parts, a learning rate of alpha = 0.12 was used as required by the program instructions. However, the maximum stable learning rate is calculated to be alpha = 0.9259 . One may ask the question, “How does increasing the learning rate impact the performance of the algorithm?”. When a learning rate of alpha = 0.40 is used, the following graph is obtained:



Examining the graph for e1(k) above, it can be observed that when alpha = 0.40 the adaptive filter is finished training in the neighborhood of k = 10 (time steps). This convergence is much faster than the results obtained for alpha = 0.12 in part (d) of **Results** (recall that part (d) showed that the filter finishes training in the neighborhood of k = 50). This can quickly be observed by examining the graphs for **W** convergence:



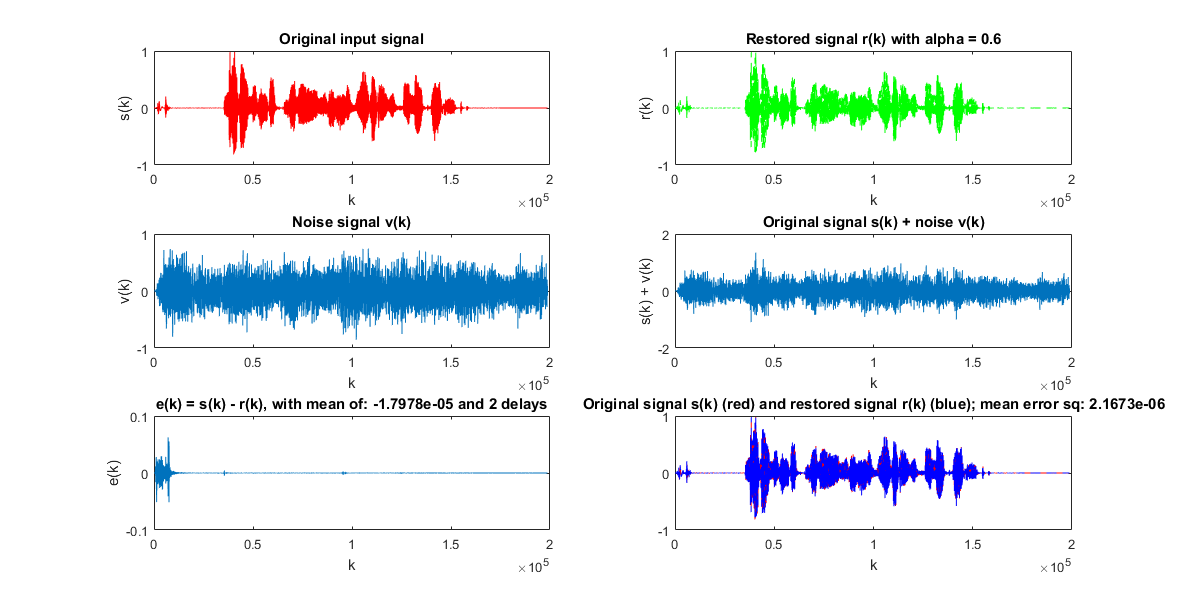
In order to easily visualize this minimum convergence with respect to the contour plot, **W** was initialized as [0 -2] for the following 2 graphs (**W** is normally initialized as [0 0]). Notice that both learning rates converge to the minimum, but alpha = 0.40 converges with significantly less steps.



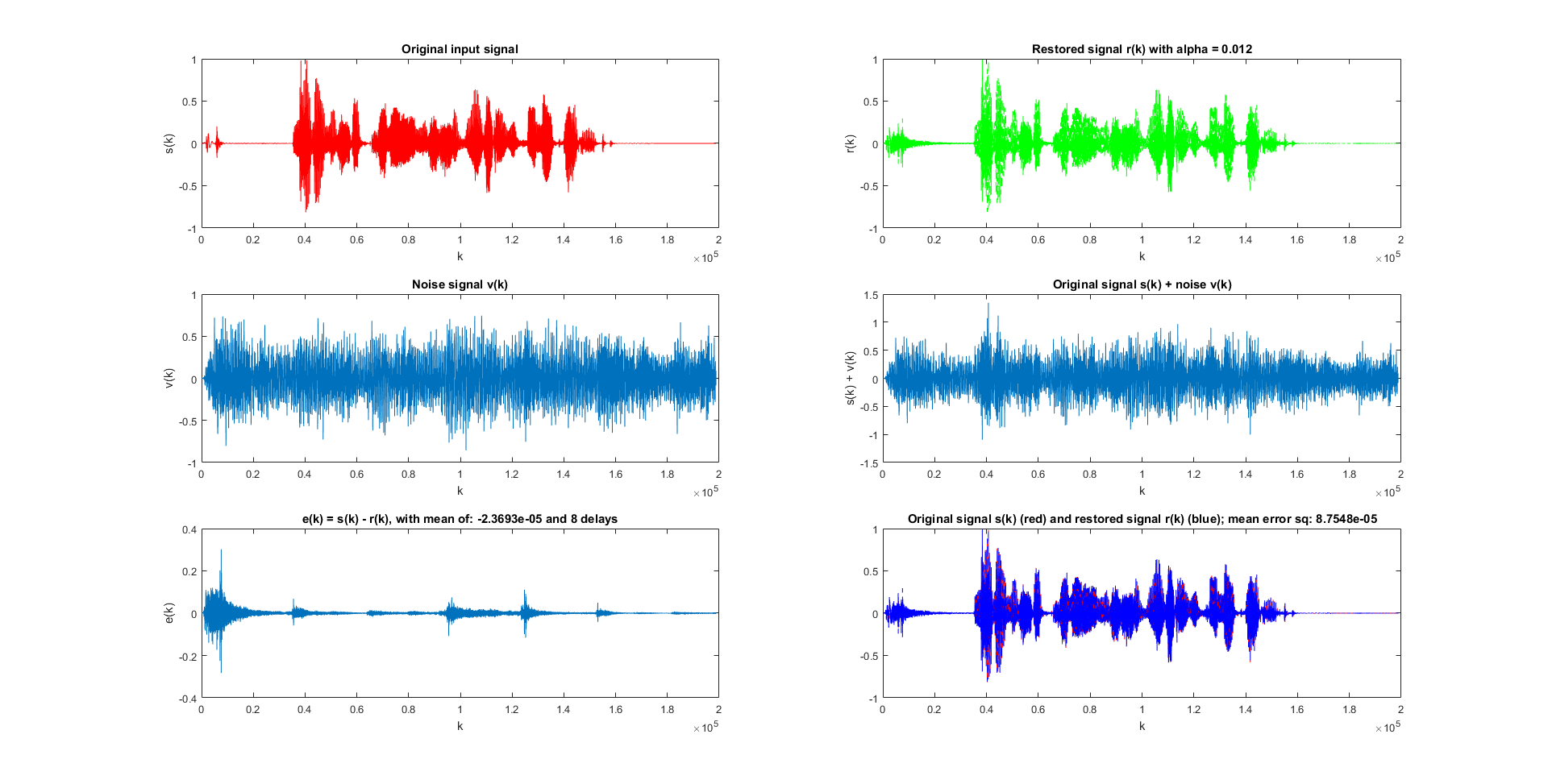
**For the last part, part (g)**, distinct learning rates of alpha = {0.012; 0.12; 0.24; 0.48; 0.6; 0.8} were tested on adaptive filter networks with the number of delays = {2; 4; 8} (in other words, each value in alpha{} was tested in a network, for each value in delays{}). The mean value of the error signal e(k) squared was then used to compare performance of the different networks, where a mean-squared value approaching 0 represents the error signal e(k) approaching 0.

Observations:

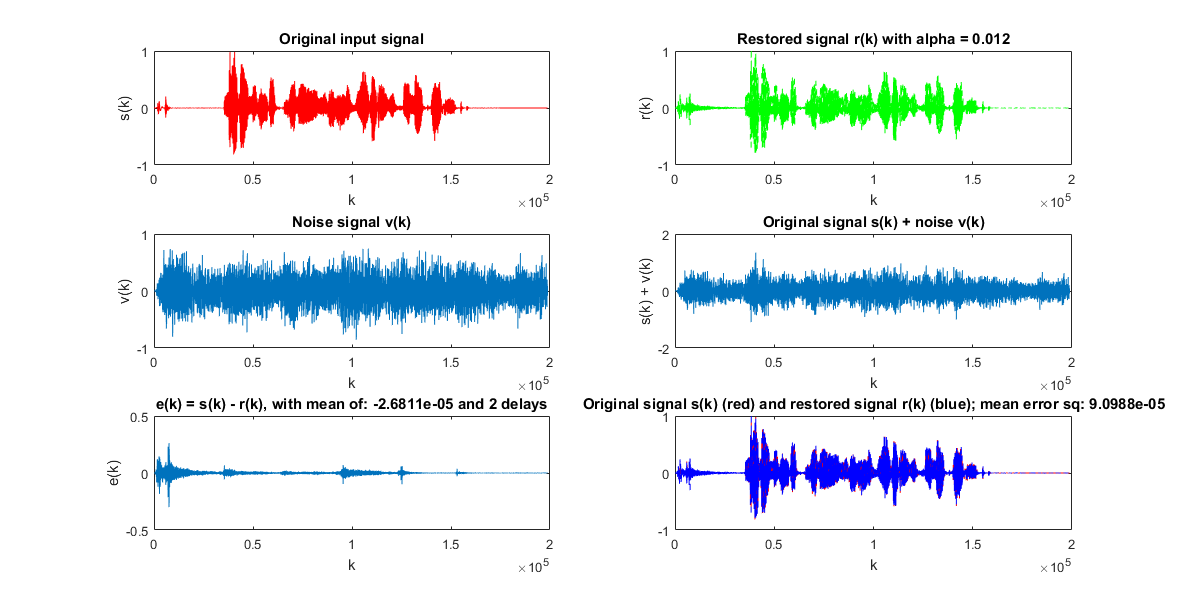
* As alpha becomes larger, additional delays cause the mean of the squared-error to **increase**
* For very small alpha (0.012), mean-square-error **decreased** as delays increased from 2->8
* For alpha >= 0.48, using 8 or more delays **prevents** original signal restoration (r(k) diverges)
* For alpha >= 0.6, using 4 or more delays **prevents** original signal restoration (r(k) diverges)
* Of all alpha/delay pairs tested, alpha = 0.6/2-delays produced the lowest mean-square-error



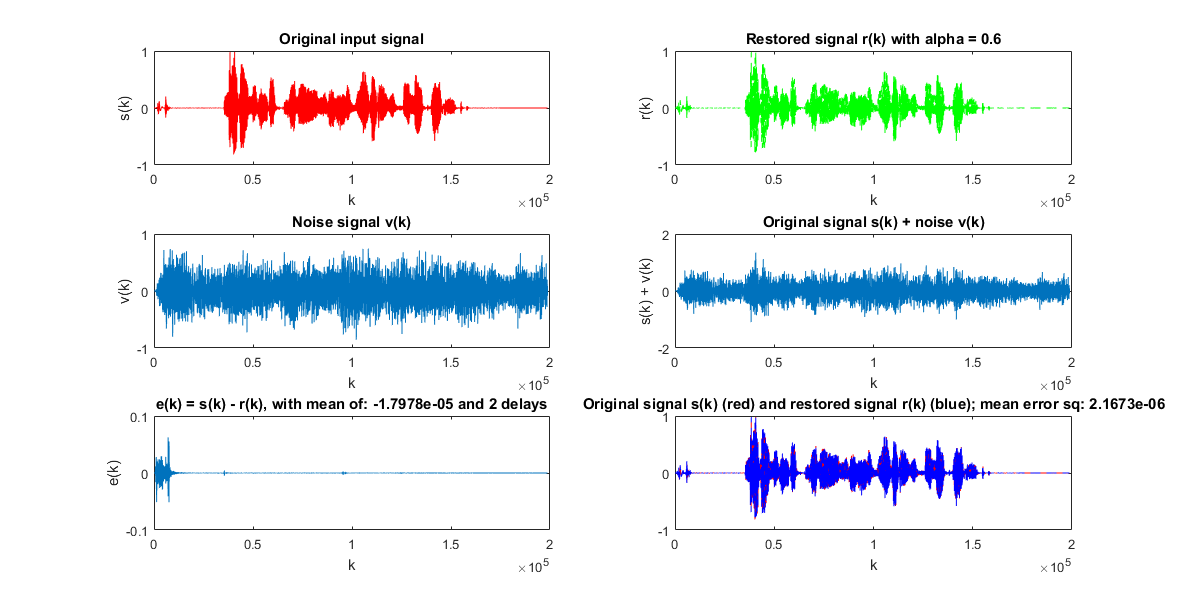
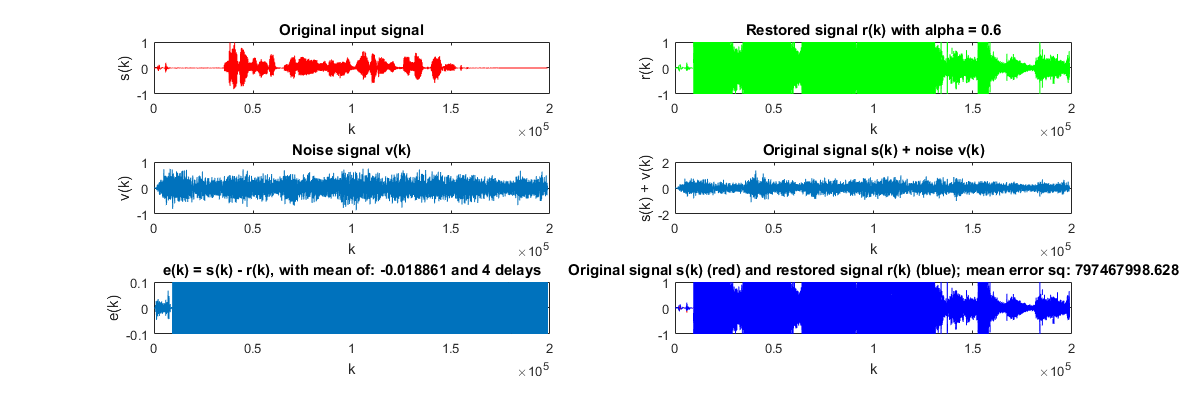
Lowest mean-square-error when a = 0.6 and delays = 2



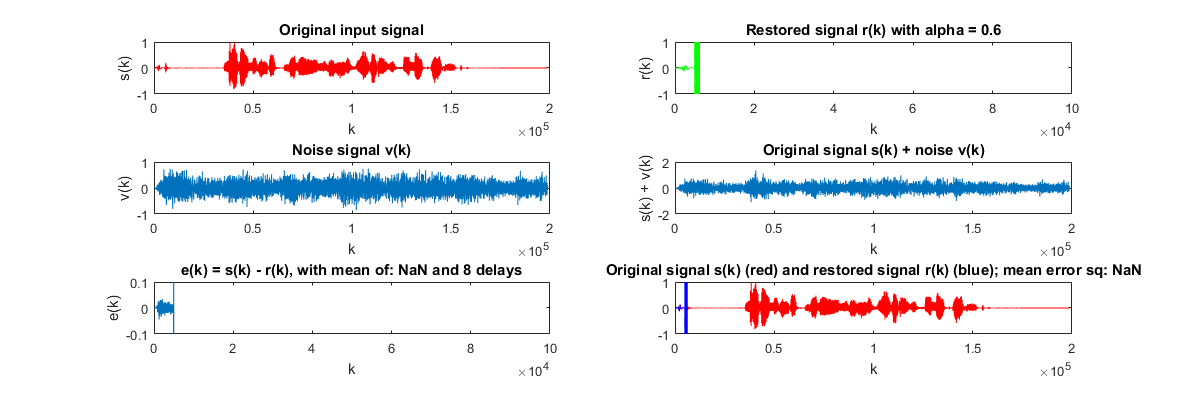
Alpha = 0.012; Delays = 8

Note that both the mean of e(k) and the mean of e(k)^2 **decrease** as delays increase from 2->8 for alpha = 0.012 .

Alpha = 0.012; Delays = 2



Alpha = 0.6; Delays = 2

Alpha = 0.6; Delays = 4

Alpha = 0.6; Delays = 8

On the other hand, the means **increase** as delays increase from 2->4 for alpha = 0.6, and the original signal becomes unrestorable when 8 delays are present.