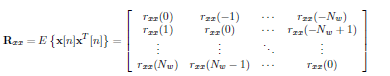
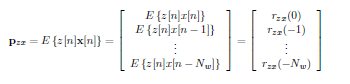
**4. Optimal filtering – fixed and adaptive**

By using MATLAB snr function, it is calculated that the signal-to-noise ratio is around 20 dB. It agrees with our theory since variance of system output is 1 while that of WGN is . So the SNR is dB.

**4.1 Wiener filter**

**4.1.1**

Using the formulas above, we are able to calculate matrix R and P. After solving the Wiener-Hopf equation  **.** The optimal wiener filter coefficients are obtained. The first five coefficients of the optimal Wiener filters are: 1.00, 2.00, 3.00, 2.00 and 1.00 correspondingly, while all other coefficients are small enough to be ignored. It can be seen that the first five coefficients agree with our ‘unknown’ system. Note that when we calculate filter coefficients, filter output is not unit variance, since changing signal variance scales the wiener filter coefficients down.

**4.1.2**

Repeat the experiments with different noise power and investigate its effect on SNR. The relation between variance, SNR and filter coefficients are shown in the following table. Note that filter output variance is fixed at 1 to make comparison convenient. Wiener filter coefficients are scaled down but ratio is kept same.

**Table 1.** Relation between noise variance, SNR and Wiener filter coefficients

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Std\_dev**  **(σ)** | **SNR** | **First Five Optimal Wiener Coefficients** | | | | |
| **1** | **2** | **3** | **4** | **5** |
| 0.1 | 19.8 | 0.24 | 0.47 | 0.71 | 0.47 | 0.23 |
| 2 | -6.06 | 0.27 | 0.43 | 0.56 | 0.29 | 0.26 |
| 4 | -12 | 0.28 | 0.66 | 0.7 | 0.55 | 0.18 |
| 6 | -15.1 | 0.36 | 0.47 | 0.67 | 0.3 | 0.07 |
| 8 | -18 | 0.38 | 0.24 | 0.91 | 0.51 | 0.17 |
| 10 | -20 | 0.18 | 0.38 | 0.78 | 0.47 | 0.55 |

Since unknown system output is fixed at 1, relation between SNR and noise variance fits the equation: . It is observed that, as noise power increases, there are more deviation between calculated wiener filter coefficients and theoretical values.

However, theoretically, noise have no effect on wiener filter coefficients if noise is zero-mean.

, if given noise is zero-mean.

In reality, pure zero mean WGN does not exist, that is why in our test, noise power adds to the error of filter coefficients. In this case, noise window size is chosen to be the maximum, which is 999. Large window size makes mean of noise converge to zero. So more accurate results are obtained. Conversely, smaller window size adds to the influence on optimal Wiener filter coefficients brought by noise power. If Nw is smaller than 5, calculated coefficients are no longer complete and accurate, since ‘unknown’ system has five coefficients.

**4.1.3**

Complexity of Wiener filter solution comes from calculating cross- and auto-correlations as well as solving the Wiener-Hopf equations.

Finding each correlation matrix p requires O(Nw\*N) multiplications and O(Nw\*N) additions. While for matrix R, calculation process requires O(Nw\*Nw\*N) multiplications and O(Nw\*Nw\*N) additions. In total, computational complexity to calculate R and p is O(N\*Nw2 ), this is also the number of multiplications and additions required.

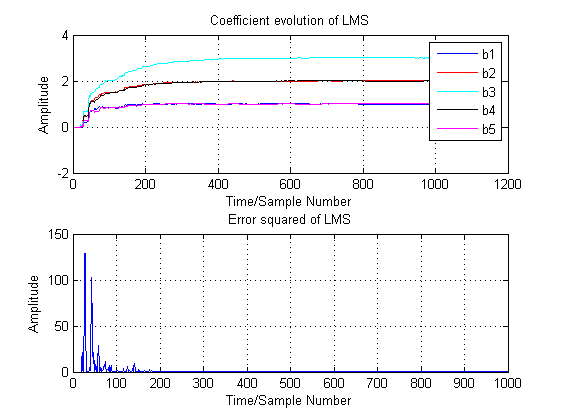
Solving Wiener-Hopf equation, takes O(Nw3) to find its inverse. Multiplication between the two matrix has Nw\*Nw multiplications and additions. In total, Solving Wiener-Hopf equation has complexity O(Nw3).

Combining the two processes, this algorithm has complexity of O(N\*Nw2).

**4.2 The least mean square (LMS) algorithm**

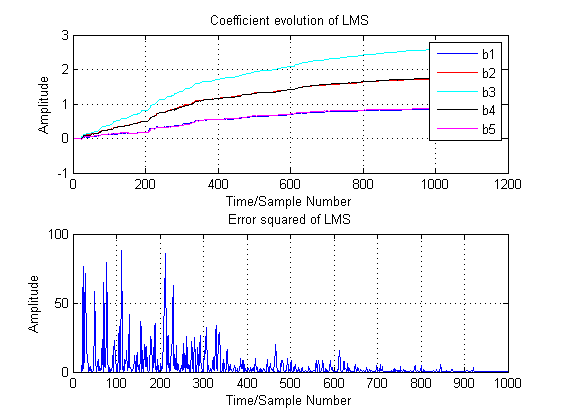
**4.2.1**

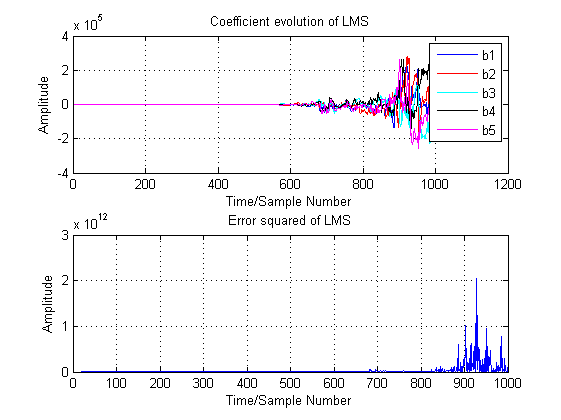
Wiener filter solution cannot be used for time-varying waves. Hence, we use least mean square algorithm to approach the right coefficients iteratively. We start from one segment of input signal, keep tracking the error between filter output and theoretical output, and adjust the weights of past inputs based on error signal. Weights are updated iteratively using the formula:.

Result using least mean square algorithm is quite close to wiener filter solution for the case in 4.1 where signal is stationary. The first five coefficients are 1.0104, 1.9991, 2.9939, 2.0118 and 1.0183, while other coefficients are small enough to be neglected.

**4.2.2**

**Figure 1.** Coefficients evolution and error squared LMS algorithm, u=0.01





**Figure 2.** Coefficients evolution and error squared LMS algorithm, u=0.002

**Figure 3.** Coefficients evolution and error squared LMS algorithm, u=0.1

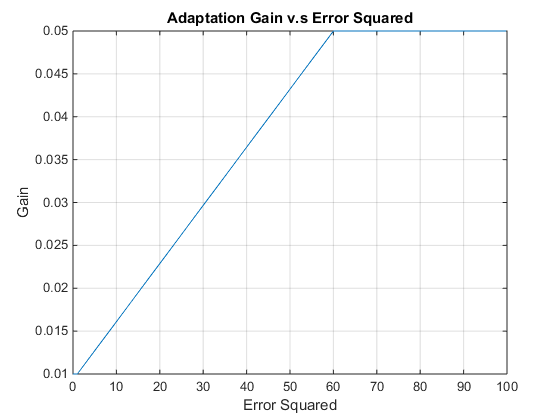
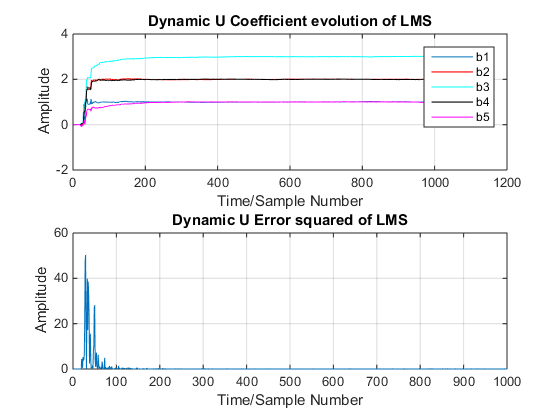
Figure 1, 2 and 3 show the evolution of LMS algorithm and error squared with different adaptation gain, which determines how fast estimated coefficients changes according to error signal. Larger adaptation gain means larger steps size, which makes coefficients quickly approach theoretical value but may results in inaccuracy. Figure 3 shows the example of large adaptation gain and error signal is becoming large at large sample number. Figure 2 shows the case when adaptation gain is small, coefficients change so slow that theoretical values may not be reached. If given a suitable adaptation gain and signal is stationary, adaptive filter converge to the Wiener solution within sample length.

**4.2.3**

The computational complexity of the LMS algorithm is composed of iterative computation of output y, error signal e and weights. Assuming N\_w is window size, N is total sample length.

Calculating output y requires N\_w\*(N-N\_w+1) multiplications and additions, which is O(N\*N\_w). Error signal requires N\_w steps of addition. Calculating weights requires 2\*N\_w+(N-N\_w+1) multiplications and additions, which is O(N\_w). In total, its complexity is O(N\*N\_w).

**4.3 Gear shifting**

Instead of using a fixed adaptation gain, we use a time-varying gain instead. For sample with large error, we produce a large adaptation gain to adjust the weight more rapidly. For sample with small error, we apply small adaptation gain in order to obtain accurate final result, since small step size leads to higher accuracy.

**Figure 4.** Relation between adaptation gain and error squared

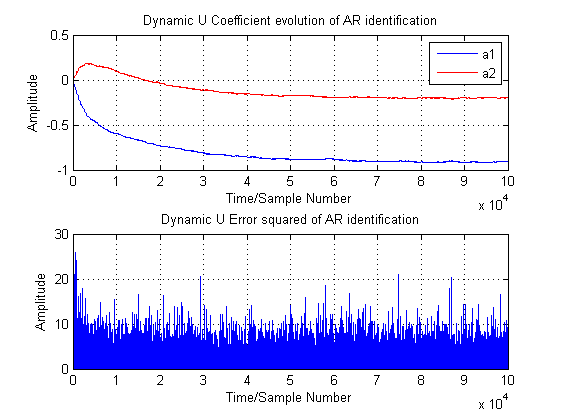
**Figure 5.** Evolution of coefficients and error square Dynamic u

Figure 4 shows the method I use to implement dynamic adaptation gain. When error square is between a certain ranges, adaptation gain is linearly proportional to error squared. When error square exceeds a maximum or minimum, a boundary adaptation gain is applied. Parameters are carefully selected to perform fast rise in the beginning and accurate prediction afterwards.

The result is shown in Figure 5. Compared to figure 1, where u is fixed at 0.01, coefficients prediction approaches to the theoretical value more rapidly. In the second case, coefficients reaches theoretical level in first 200 samples, while the first method requires around 400 samples. Predictions from both methods obtain reasonably good result.

**4.4 Identification of AR processes**

**4.41**



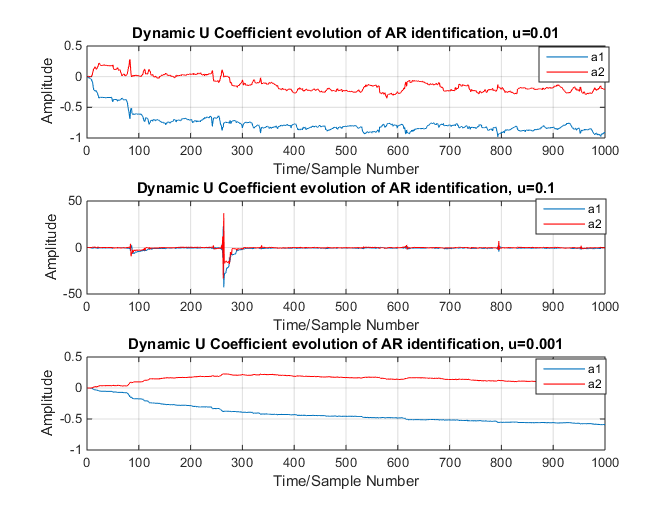
**Figure 6.** Coefficients and error evolution of AR process identification

In AR identification, we aim to predict AR coefficients from system output. Given an AR2 system with equation: , we use adaptation algorithm and iteratively compare the output generated from the past two samples and the current sample. Predicted coefficients are then adjusted to approach theoretical coefficients.

After implementing the algorithm, we obtain Figure 6, which shows that estimated coefficients are around -0.2 and -0.9 correspondingly. We are given the AR filter has b=1 and a= [1 0.9 0.2]. Filter response can be represented as: . It can be written in another form of . As a result, the estimated coefficients are close to expected values.

**4.4.2**

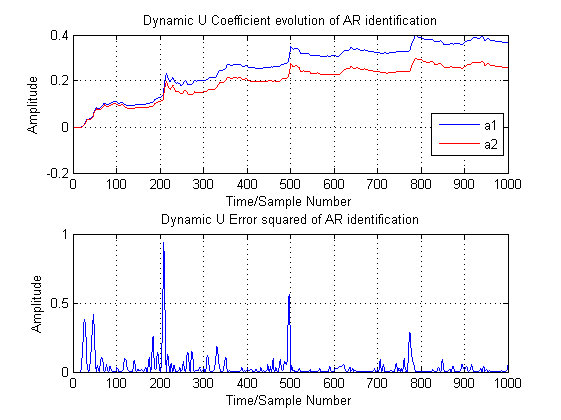
Figure 7 shows the effect of different step size on predicted coefficients. It is observed that when adaptation gain is 0.01, coefficients approach theoretical values relatively fast and then oscillates around the correct value. For the case when adaptation gain is 0.1, step size is so large that the coefficients just oscillate around the actual values. When the adaptation gain is 0.001, coefficients approach the actual values slowly. In figure 8, sample number is not large enough for coefficients to converge to -0.9 and -0.2.



**Figure 7.** Coefficients and error evolution for various adaptation gain

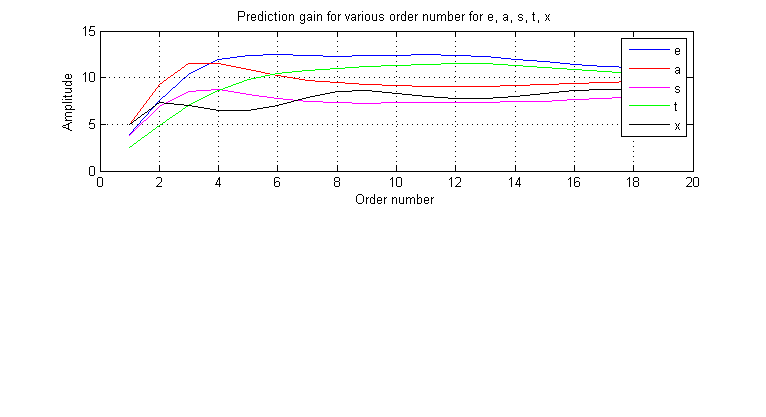
**4.5 Speech recognition**

**4.5.1**

After recording the letters with sampling frequency 44100 Hz and taking 1000 samples in the valid region, various adaptation gains are used.

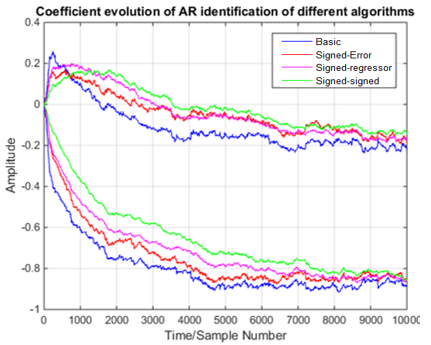
**Figure 8.** Coefficients and error evolution for various adaptation gain

Figure 8 shows the coefficients evolution when u is 0. 1. From the error evolution graph, 0.1 is a good option. Gear shifting requires different parameter setting from previous exercise and coefficients rise faster than fixed adaptation gain. If we observe the weights calculated using adaptive filter, coefficients are becoming negligible after order 3.

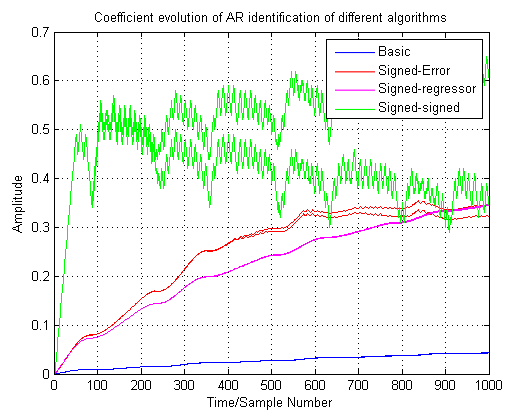
**4.5.2**

**Figure 9.** Prediction gain for e, a, s, t and x

Figure 9 is the example of using prediction gain to determine filter performance for different letters, using the formula: In general, when order number rise above five, optimal performance is achieved. However, the best order is different for different letters. And prediction gain only measures performance and ignore the cost of computational complexity.

**4.6 Dealing with computational complexity: sign algorithms**

**Figure 8.** Coefficients and error evolution of AR identification using different algorithms

Figure 8 compares the performance of different algorithms. The basic algorithm rise fastest while the signed-error algorithm rise slowest among the four algorithms. It is observed that these algorithms obtain reasonably good result as the basic algorithm, while computational complexity is reduced.

**Figure 9.** Coefficients and error evolution of AR identification using different algorithms

The above figure is the performance for different algorithms applied on speech signal when u=0.01, it can be observed that sign algorithm has the shortest reaction time among them and a reasonably good result can be obtained. While other algorithms either rise very slowly or produce result with large error for non-stationary signals.