Chapter 2

Theory

2.1 Digital Filters

For many reasons such as flexibility, adaptability, accuracy, stability and cost, the use of digital filters is increasing especially in electronic controllers for active control systems. This section will only focus on the part of digital filters whose understanding will be necessary later in this research. However, for further details concerning digital filters, the reader is referred to many textbooks on the subject [16-18].

Digital filtering is very common in signal processing. It refers to a linear process that modifies in a specified manner the spectral content of an input signal. Filters are therefore designed to have specific magnitude and phase responses that correspond to desired modifications for the input signal. Conventional filters are linear and time-invariant. Performing a constant set of linear operations on a data sequence, x(n), they create a corresponding output, y(n), based on the value of their coefficients. Two different structures are used for digital filters. One is the Finite Impulse Response filter

(FIR) that is always stable and provides a linear phase response since it incorporates only zeros. The other one is the Infinite Impulse Response filter, IIR, that incorporates both zeros and poles and whose stability is therefore not guaranteed. We will essentially be interested in FIR filters since they are the only ones used in the scope of this research.

2.1.1 FIR Filters

A finite impulse response filter (FIR) is a digital filter whose impulse response goes to zero after some finite number of samples. The output *y(n)* of such a filter can be calculated as follow:

$$y(n) = \sum_{i=0}^{I-1} w_i(i) \cdot x(n-i) \tag{2.1}$$

where w_i is the i^{th} coefficient of the I filter coefficients, i=0, 1, …, I-1, and *x(n-i)* is an element of the input data sequence {*x(n), x(n-1), …, x(n-I+1)*}.
A diagram of the implementation of such a filter is shown in Figure 2.1.

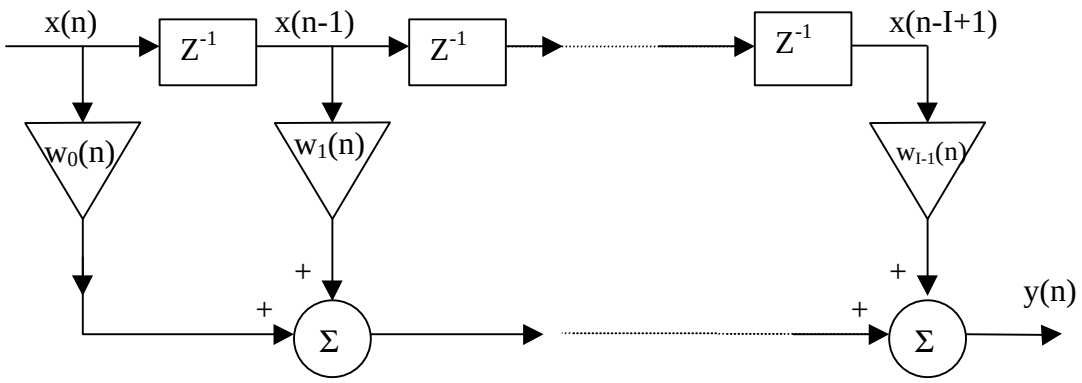

Figure 2.1: Block diagram of digital FIR filter

2.2 Adaptive Systems

Many practical systems are not time-invariant. It is sometimes desirable to filter signals coming from such systems with time-varying filters since the optimal coefficients calculated for a given system, will not necessarily be optimal after the system has changed. For these cases, adaptive systems are required since they have the ability to modify their characteristics automatically to achieve a given objective. Using an adaptive algorithm, the coefficients of the adaptive filter are changed as a function of the changes in the input signal.

2.2.1 Adaptive filters

An adaptive filter consists of two different parts. The first one is a digital filter (e.g.: FIR filter) to perform the desired calculations in order to obtained a desired output signal. The second one is the adaptive algorithm whose task is to adjust the coefficients or weights of the digital filter. A general form of an adaptive filter is illustrated in Figure 2.2.

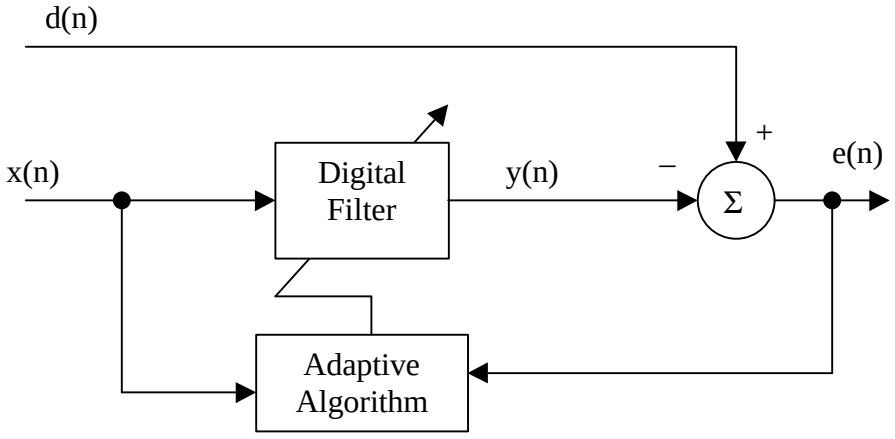

Figure 2.2: Block diagram of adaptive filter

In this block diagram, the objective of the adaptive filter is for the output signal, $y(n)$ to match the desired response $d(n)$, also called "primary" input signal. The error signal $e(n)$ calculated as the difference between $d(n)$ and $y(n)$ indicates the system performance (i.e.: how accurately the true output $y(n)$ matches the desired output $d(n)$). With this error signal and the input signal, $x(n)$, also called the "reference" signal, the adaptive algorithm progressively adjusts the coefficients of the digital filters, on a sample-by-sample basis to minimize the error signal.

If the digital filter is an FIR filter, equation (2.1) is still valid, but the coefficients $w_i(n)$ are now time-varying and updated by the adaptive algorithm. We define the input vector $X(n)$ at time n as:

$$X(n)=[x(n)\ \ x(n-1)\ \ \ldots\ \ x(n-I+1)]^T \tag{2.2}$$

and the weight vector at time n as:

$$W(n)=[w_0(n)\ \ w_1(n)\ \ \ldots\ \ w_{I-1}(n)]^T \tag{2.3}$$

where the superscript T denotes transpose. Then, the output signal $y(n)$ in equation (2.1) can be rewritten with both vectors $X(n)$ and $W(n)$ as follow:

$$y(n)=W^T(n).X(n) \tag{2.4}$$

The error signal $e(n)$ can be expressed as a function of the same two vectors:

$$e(n)=d(n)-y(n)=d(n)-\ W^T(n).X(n) \tag{2.5}$$

2.2.2 Adaptive Algorithms

The objective of an adaptive algorithm is to adaptively determine the weight vector that minimizes a cost function that is usually chosen to be the mean square value of the error signal (Cf: equation (2.5)). Instead of the mean square value of the error, the error signal itself could have been chosen as a cost function. However, this is not suitable in many cases because a "large" negative error is often as detrimental as a "large" positive error. The absolute value of the error could also be chosen as the cost function but this often leads to intractable mathematical problems when results have to be analyzed. Therefore, it is preferred to use the measure of the square error, $e^2(n)$, since it lends itself much better to analytical work and is related to the power of the error signal [19]. Finally, for stochastic reasons (e.g.: random noise being added to the signals), it is natural to choose, as a performance measure, the statistical property of the squared error signal which is its mean. The Mean Squared Error, MSE, will, therefore, be defined as the ensemble average or expectation of the squared error signal:

$$e(n)=E[e^2(n)] \tag{2.6}$$

where $E[\ .\ ]$ is the expectation operator.

As the adaptive filter "adapts", $e(n)$ is the indicator of convergence behavior. The variable n in $e(n)$ is analogous to the number of adaptive filter iterations that have been performed.

2.2.2.1 MSE Surface

Let us first assume for the moment that both signal $d(n)$ and $x(n)$ are statistically stationary. If we consider a FIR filter, combining equation (2.5) into equation (2.6), with the temporary assumption that the weight vector $W(n)$ is a deterministic sequence we can calculate the MSE as a function of the I filter coefficients:

$$e(n)=E[d^2(n)]-2P^TW(n)+W^T(n)RW(n) \quad (2.7)$$

where P is the cross-correlation vector of size I between $d(n)$ and $X(n)$:

$$P=E[d(n)X(n-j)] \text{ for } j=0\rightarrow I-1 \quad (2.8)$$

and R is the input auto-correlation matrix of size I by I:

$$R=E[X(n)X(n-j)] \text{ for } j=0\rightarrow I-1 \quad (2.9)$$

From equation (2.7) we can see that the performance function of a causal FIR filter is a quadratic function of the given weights since these coefficients only appear at the first and second degrees. Furthermore, as for each value of the filter coefficient vector, $W(n)$, corresponds a value (scalar) of MSE. The MSE values associated with $W(n)$ form then an (I+1)-dimensional space, commonly called the MSE *surface* [19].

For a better visualization of the signification of the MSE surface, let us consider the case where I=2. In that case, the error surface is represented in a three-dimensional space (Cf: Figure 2.3). The height of $e(n)$ corresponds to the power of the error signal $e(n)$ that results from filtering the signal $x(n)$ with the coefficients $W(n)$. If the coefficients change, the power of the error signal will change too, as indicated by the changing height on the surface above the w_0-w_1 plane. $W^0=[w_0^0\ w_1^0]^T$ is the optimal coefficient vector that gives the minimum mean-square error, e_{min}. As the error surface function is quadratic, the optimal coefficient is unique.

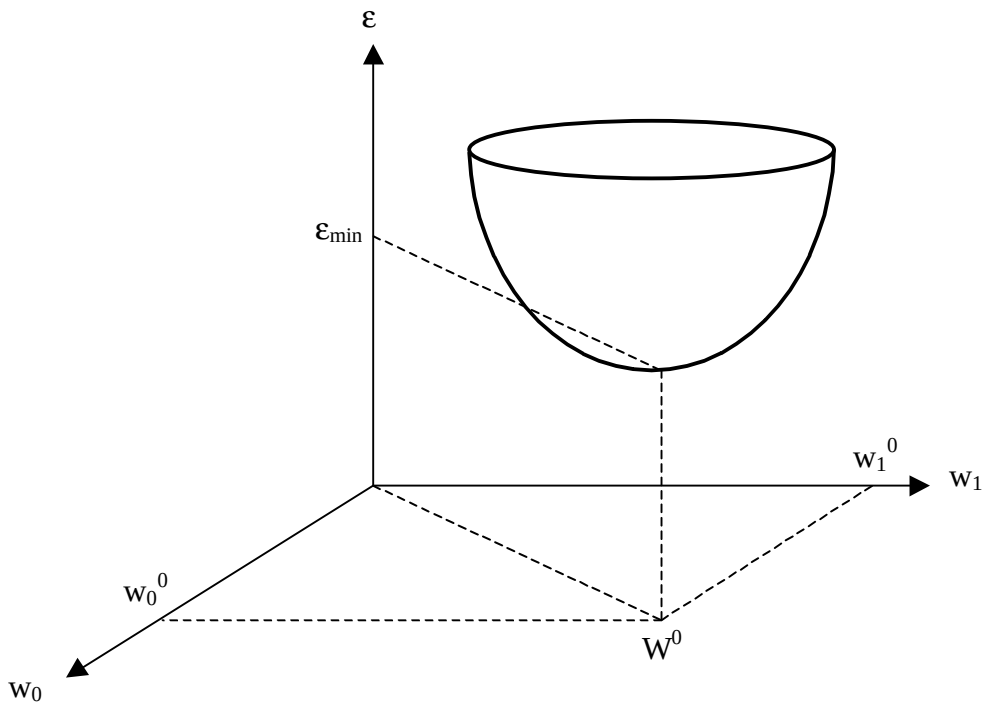

Figure 2.3: Three-dimensional performance surface, I=2 case

Since W^0 minimizes the MSE cost function $e(n)$, the vector differentiation of equation (2.7) gives W^0 as the solution to:

$$RW^0=P \text{ or } W^0=R^{-1}P \tag{2.10}$$

In principle, the solution to the adaptive filtering problem is given with this equation, but this solution requires a continuous estimation of both the cross-correlation vector and the auto-correlation matrix. Furthermore, in the case where the signals are non-stationary, computation of R and P becomes very restricting. However, the most important property of the MSE surface is its concave shape that gives it a unique global minimum. Indeed, expressing the minimum MSE from equations (2.7) and (2.10), we have:

$$e_{min}(n)=E[d^2(n)]-P^TW^0 \tag{2.11}$$

Substituting this expression of $e_{min}(n)$ into equation (2.7) with the relation (2.10) we get:

$$e(n)= e_{min}(n)+(W(n)-W^0)^T R(W(n)-W^0)= e_{min}(n)+V^T(n)RV(n) \qquad (2.12)$$

where $V(n)$ is the vector difference between the weigh vector and the optimal solution. Therefore, from equation (2.12) it appears that the MSE is a quadratic function of the weights. As a consequence, any change in the weight vector from its optimal form will increase the error above is minimum value. This property is fundamental and it has been the leading idea for most adaptive algorithms such as the LMS algorithm [18-19].

2.2.2.2 Method of Steepest Descent

Because of the specific bowl shape of the MSE, adjusting the weights to minimize the error is equivalent to descending along this surface until reaching the bottom (i.e.: minimum). Therefore, the method of steepest descent follows this idea, but reaches the minimum descending along the greatest rate of decrease (direction of the tangent) of the MSE surface.

Choosing an arbitrarily initial filter setting $W(0)$ of error $e(0)$, the direction of the greatest rate of decrease at this point [$W(0)$, $e(0)$] of the MSE surface is given by the error gradient with respect to $W(n)$, $\tilde{N}e(n)$, defined as the vector of the directional derivatives $\P e(n)/\P w_I$ for i=0, 1, …, I-1. From this concept, the following algorithm can be easily implemented:

$$W(n+1) = W(n) - \frac{m}{2}\nabla e(n) \qquad (2.13)$$

where m is a convergence factor that controls the rate of descent (the larger the value of m the faster the speed of descent) but also the stability (a too large value of m can make the algorithm diverge). The negative sign on the adaptive weight vector specifies the

negative gradient direction (down the bowl). The arbitrary factor 2 is included to simplify the final version of this algorithm.

From equation (2.7) we can calculate the error gradient:

$$\nabla e(n) = -2P + 2RW(n) \tag{2.14}$$

Then, substituting equation (2.14) into equation (2.13), the final version of the steepest descent algorithm becomes:

$$W(n+1) = W(n) + m[P - RW(n)] \tag{2.15}$$

When the minimum is reached, the error gradient is equal to zero, the algorithm converges to the solution $W(n)=W^0$.

2.2.2.3 LMS Algorithm

Since it assumes exact knowledge of the error gradient vector at each iteration, the steepest descent method cannot be used for practical cases where the statistics of *d(n)* and *X(n)* are unknown. Widrow [20], proposed the LMS algorithm, which uses the instantaneous squared error, $e^2(n)$, to estimate the mean squared error, as a solution to this problem.

$$\hat{e}(n) = e^2(n) \tag{2.16}$$

The gradient estimate used by the LMS algorithm is then, simply the instantaneous gradient of a single squared error sample:

$$\nabla \hat{e}(n) = 2[\nabla e(n)]e(n) \tag{2.17}$$

From equation (2.5), the calculation for the gradient of the instantaneous error with respect with *W(n)* is given by:

$$\nabla e(n) = -X(n) \tag{2.18}$$

and the gradient estimate (Cf: equation (2.17)) becomes:

$$\nabla \hat{e}(n) = -2X(n)e(n) \tag{2.19}$$

Substituting this estimate into equation (2.15) of the steepest descent algorithm finally gives us the equation of the LMS algorithm:

$$W(n+1) = W(n) + \mu X(n)e(n) \tag{2.20}$$

The convergence behavior of this algorithm is determined by the eigenvalues of the auto-correlation matrix of the input signal and the size of the filter. The maximum value for μ to prevent the algorithm from diverging can then be approximated in function of the mean square value of the input signal vector *X* and the number of filter coefficients, I, as follow [21]:

$$\mu_{\max} \approx \frac{1}{\overline{X^2} I} \tag{2.21}$$

This algorithm is very easy to implement as it does not require squaring, averaging or differentiating.

2.2.2.4 Filtered-X LMS Algorithm

Because it is easy to implement in a program and because it is an efficient algorithm, the classical LMS algorithm, described in the previous section, is the most widely used algorithm for practical applications. However, many modified versions of it have been designed to improve convergence speed, implementation simplicity, …etc. For the purpose of this work the Filtered-X LMS algorithm will be the only LMS variant used. First proposed by Morgan [22] in 1980, it was implemented in feedforward control by Widrow [23] in 1981 (see section 2.3.1).

The only difference with the filtered-X LMS algorithm is that the input signal *X(n)* is filtered prior the weight update procedure of the LMS algorithm. The filter used can either be a FIR or an IIR filter. Equation (2.20) of the LMS algorithm becomes then:

$$W(n+1) = W(n) + mFX(n)e(n) \tag{2.22}$$

where *F* is the digital filter used to filter the input signal.

According to the same criteria as with the LMS algorithm, the maximum value of the convergence coefficients to ensure convergence is no longer a function of the reference signal itself, but of the mean square value of the filtered reference signal, *Q* (*Q(n)=F.X(n)*):

$$m_{\max} \approx \frac{1}{\overline{Q^2} I} \tag{2.23}$$

Motivations for this evolution of the LMS algorithm will become more obvious and will be explained in section 2.3.1, where the filtered-X LMS algorithm is implemented in feedforward control.

2.3 Feedforward Control

Feedforward control is less widely used than feedback control, since it requires some knowledge of the disturbance (i.e.: a reference signal). However, due to the development of the filtered X-LMS algorithm, feedforward control methods have been widely used for the active control of sound and vibration [4], [24-25]. This section will introduce feedforward control, but a more detailed presentation of feedforward control is given by Fuller *et al.* [11] and Nelson *et al.* [18].

To introduce the concept of feedforward control and the associated vocabulary, let us start by considering the following Single Input Single Output (SISO) control problem: we have a structure (in the case of a vibration problem), or a system, S, excited by a primary disturbance signal, d, due to a reference source signal or excitation signal, X. The signal X is the voltage input to an actuator, assumed fully active in our case (e.g.: electromagnetic shaker, piezoelectric ceramic, … etc), that generates the force, d, exciting the system. The transfer function between the output force, d, and the input signal, X, is commonly referred to as the primary path.

We want to cancel the effect of the disturbance on the system measured with a sensor located on the structure. This sensor output is called the error signal, e. Knowing the reference signal, the aim of the control is, then, to use a second actuator to create a secondary disturbance on the system that destructively interferes with the primary disturbance, and brings the error signal to zero. To achieve this, the control signal or secondary disturbance is generated with a controller, H, whose input is an estimate of the reference signal (Cf: Figure 2.3). It is then assumed that the overall excitation of the structure is proportional to the difference between the primary and secondary excitations (d-u).

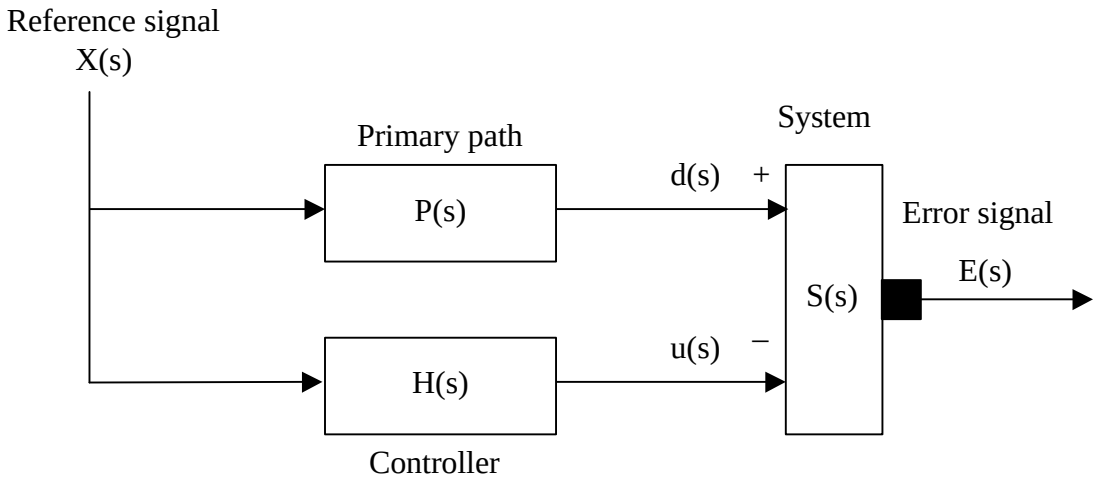

Figure 2.4: Block diagram of feedforward control

In the Laplace domain, we can easily derive an expression for the error signal, *E(s)*, as a function of the system, *S(s)*, the primary path, *P(s)*, the controller, *H(s)* and the reference signal, *X(s)*:

$$E(s) = S(s)[P(s) - H(s)]X(s) \tag{2.24}$$

To cancel the error signal (*E(s)=0*), equation (2.24) directly shows that we have to derive *H(s)* equal to *P(s)* (*H(s)=P(s)*). Feedforward control relies on a perfect matching between d and (-u). Therefore, the magnitude and phase of H have to be carefully adjusted to provide the exact expected response. Even though the problem is relatively simple for single frequency signals, in experimental cases, reference signals and primary paths may slowly change with time. It is thus necessary for H to be able to both implement the desired frequency response and also performing necessary adjustment over time. For this reason, feedforward control requires the use of an adaptive digital filter that

is capable of performing the adjustments necessary to track the slight changes of the system over time.

2.3.1 Adaptive Feedforward Control Using the Filtered-X LMS Algorithm

In real systems with dynamics such as Figure 2.3, the presence of the system S means that the filtered-X LMS algorithm must be used instead of the classical LMS algorithm. Indeed, the filtered-X LMS algorithm is used in cases where there is a transfer function in the secondary path (i.e.: between u and e). The filtered-X LMS algorithm uses an estimate S^* of the true control path S to filter the reference signal before it is used to adapt the controller coefficients. This technique produces a control system that can track the disturbance signal (adaptive filtering) and ensures rapid convergence.

The system S^* is represented by a finite length FIR filter whose coefficients we determine prior to control during a phase called "system identification". In this procedure, the disturbance actuator is turned off and only the control actuator is driven with the desired single frequency signal (the scope of this work does not cover broadband control). The classical LMS algorithm is used to identify the secondary path, computing the coefficients for the FIR filter S^*. The final equation for the filtered-X algorithm applied to this control case, after equation (2.22) is:

$$W(n+1) = W(n) + mS^* X(n)e(n) \tag{2.25}$$

A block diagram of the implementation of the filtered-X LMS algorithm in the feedforward control is presented in Figure (2.4).

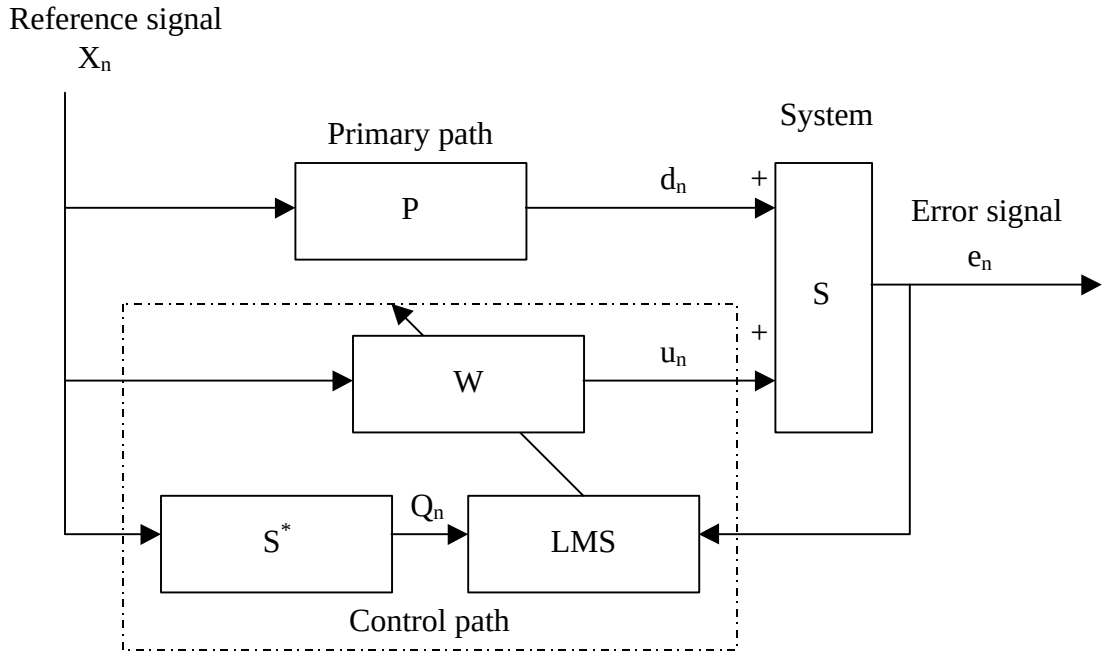

Figure 2.5: Block diagram of adaptive feedforward control with filtered-X LMS algorithm

The above methodology will be used in later work to actively control impedances for actuator testing.