

ECE 6010

Programming Assignment #2

System and Autoregressive Identification

1 Introduction

System identification is the means by which systems are modeled mathematically based on measured data. It is often a precursor to other engineering tasks, such as control or signal processing. The system models are frequently described parametrically. System identification is sometimes done using only measurements of the output of the system. In this exercise, however, it is assumed that both input and output data are available. (Usually having both input and output data significantly simplifies the system identification, often resulting in straightforward linear equations to solve.)

It is usually assumed that the system measurements are made in the presence of noise. A common assumption is that the noise signals are white. However, to make things more interesting, in this assignment the noise is assumed to be from an AR process, whose parameters are also to be identified.

Your task is to take measured data and from it, determine the system and noise parameters.

2 System Model

For brevity, we will employ an operator notation frequently used in the literature. If

$$G(z) = \sum_{i=0}^{L_g} g_i z^{-i}.$$

we will use the notation

$$s(t) = G(z)u(t)$$

as a shorthand for the filtering operation

$$s(t) = \sum_{i=0}^{L_g} g_i u(t-i).$$

A known discrete-time input signal $u(t)$ is applied to a system which is assumed to be FIR, having transfer function

$$G(z) = \sum_{i=0}^{L_g} g_i z^{-i},$$

where we will also assume that the order of the filter L_g is known. The filtered signal $s(t) = G(z)u(t)$ is corrupted by an additive noise signal which is assumed to be autoregressive (AR):

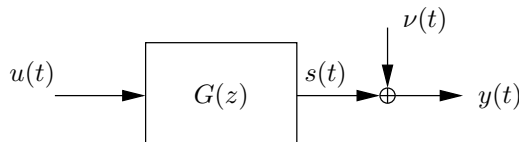
$$\nu(t) = H(z)e(t)$$

where $e(t)$ is a zero-mean, stationary, ergodic, white-noise signal with variance σ_e^2 , and $H(z)$ has the all-pole form

$$H(z) = \frac{1}{1 - \sum_{i=1}^{L_H} a_i z^{-i}}. \quad (1)$$

The measured output signal is thus

$$y(t) = s(t) + \nu(t) = G(z)u(t) + \nu(t) = G(z)u(t) + H(z)e(t).$$



The system identification problem to be addressed here is this: given the input/output measurements $\{u(t), y(t)\}$, estimate $G(z)$ and $H(z)$.

2.1 The AR Model and Its Prediction

The model for the noise $\nu(t)$ can be written another way. The IIR filter $H(z)$ could be written (say, using long division) as

$$H(z) = \sum_{i=0}^{\infty} h_i z^{-i}.$$

with $h_0 = 1$. (That is why we call it IIR — it has an infinite number of coefficients in its impulse response.) So

$$\nu(t) = \left(\sum_{i=0}^{\infty} h_i z^{-i} \right) e(t) = e(t) + \sum_{i=1}^{\infty} h_i e(t-i).$$

For the development below, we will find it convenient to develop a predictor for $\nu(t)$. Given the sequence of measurements $\nu(s)$ for $s \leq t-1$, what is the best estimate of $\nu(t)$? We will denote this estimate by $\hat{\nu}(t|t-1)$. If by “best” we mean best in the minimum mean-squared error sense, we want to find an estimator $\hat{\nu}(t|t-1)$ which minimizes

$$E[(\nu(t) - \hat{\nu}(t|t-1))^2].$$

We know that this will be the conditional mean:

$$\hat{\nu}(t|t-1) = E[\nu(t)|\nu(s), s = t-1, t-2, \dots].$$

Note that if we know all the data $\{\nu(s), s \leq t-1\}$, we equivalently know all the data $\{e(s), s \leq t-1\}$. Thus $\hat{\nu}(t|t-1)$ can equivalently be written

$$\hat{\nu}(t|t-1) = E[\nu(t)|e(s), s = t-1, t-2, \dots].$$

From the form

$$\nu(t) = e(t) + \sum_{i=1}^{\infty} h_i e(t-i)$$

we see immediately that this conditional expectation is

$$\begin{aligned} \hat{\nu}(t|t-1) &= E[e(t) + \sum_{i=1}^{\infty} h_i e(t-i)|e(s), s = t-1, t-2, \dots] \\ &= \sum_{i=1}^{\infty} h_i e(t-i) \end{aligned}$$

since $e(t)$ has zero mean. This can be written in our operator notation as

$$\hat{\nu}(t|t-1) = [H(z) - 1]e(t).$$

Since $e(t) = H^{-1}(z)\nu(t)$, we can write

$$\hat{\nu}(t|t-1) = \frac{[H(z) - 1]}{H(z)}\nu(t) = (1 - H^{-1}(z))\nu(t). \quad (2)$$

Let the inverse transfer function $H^{-1}(z)$ be written in the form

$$H^{-1}(z) = \sum_{i=0}^{\infty} \tilde{h}_i z^{-i}, \quad (3)$$

where $\tilde{h}_0 = 1$. (Note: It should be clear that $\tilde{h}_i \neq h_i$ in general.) In the case that $H(z)$ is in fact an all-pole filter as in (1), we have

$$H^{-1}(z) = 1 - \sum_{i=1}^{L_H} a_i z^{-1},$$

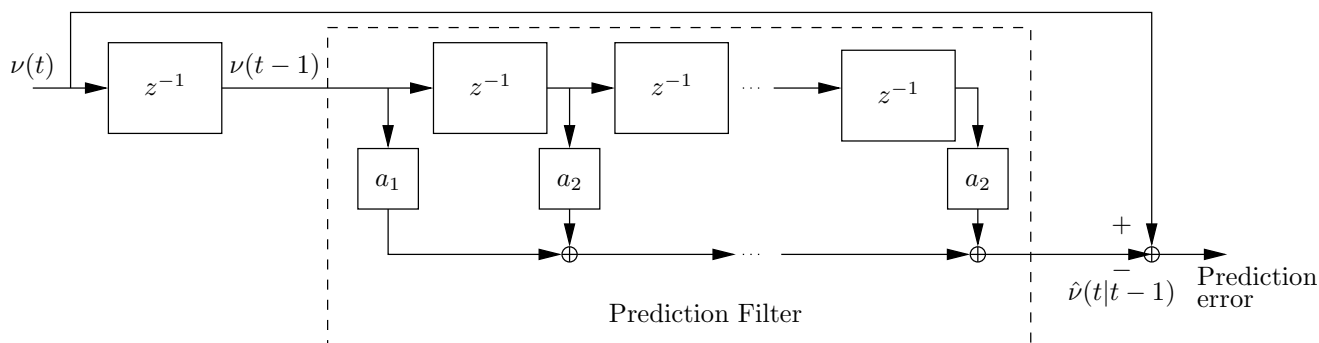
so that $\tilde{h}_i = a_i, i = 1, 2, \dots, L_H$. Substituting (3) into (2) we obtain in the general case

$$\hat{\nu}(t|t-1) = - \sum_{i=1}^{\infty} \tilde{h}_i \nu(t-i).$$

In the case that $H(z)$ is an all-pole filter, we obtain

$$\hat{\nu}(t|t-1) = + \sum_{i=1}^{L_H} a_i \nu(t-i).$$

That is: the best predictor of $\nu(t)$ given previous measurements of $\nu(t)$ has the form of an FIR filter, as shown in the block diagram here:



From a system identification perspective, we would want to choose the coefficients $\{a_i\}$ in this filter so that the prediction error $\nu(t) - \hat{\nu}(t|t-1)$ is as small as possible. That is, we would choose $\{a_1, a_2, \dots, a_{L_H}\}$ so that

$$E[(\nu(t) - \sum_{i=1}^{L_H} a_i \nu(t-i))^2]$$

is as small as possible. We can write this in a vector form as follows. Let

$$\boldsymbol{\nu}(t) = \begin{bmatrix} \nu(t) \\ \nu(t-1) \\ \vdots \\ \nu(t-L_H+1) \end{bmatrix} \quad \text{and} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{L_H} \end{bmatrix}.$$

Then we want to minimize

$$E[(\nu(t) - \mathbf{a}^T \boldsymbol{\nu}(t-1))^2] \tag{4}$$

Exercise 1: Show that minimizing the expression in (4) leads to the Widrow-Hopf normal equations

$$R_h \mathbf{a} = \mathbf{p}_h, \tag{5}$$

where $R_h = E[\boldsymbol{\nu}(t-1)\boldsymbol{\nu}(t-1)^T]$ and $\mathbf{p}_h = E[\nu(t)\boldsymbol{\nu}(t-1)]$.

Also show that the minimum mean squared prediction error is

$$E(\nu^2(t)) - \mathbf{a}^T E[\nu(t)\boldsymbol{\nu}(t-1)],$$

where \mathbf{a} is the solution to (5). This can be used as an estimate of the variance of the noise $e(t)$, σ_e^2 .

2.2 Prediction of $y(t)$ and System Identification

An important step in the overall system identification process is a one-step-ahead predictor for $y(t)$. Suppose that $y(s)$ is known for $s \leq t-1$. What is the best estimate (prediction) that can be made for $y(t)$ using all of this information? We will denote this predictor as $\hat{y}(t|t-1)$. Since the input $u(t)$ is known, this prediction must be made on the best prediction of $\nu(t)$ given the information up to time $t-1$, which we have denoted as $\hat{\nu}(t|t-1)$:

$$\hat{y}(t|t-1) = G(z)u(t) + \hat{\nu}(t|t-1).$$

We have seen above that $\hat{\nu}(t|t-1)$ can be written as $(1 - H^{-1}(z))\nu(t)$. We thus have

$$\hat{y}(t|t-1) = G(z)u(t) + (1 - H^{-1}(z))\nu(t) = G(z)u(t) + (1 - H^{-1}(z))(y(t) - G(z)u(t))$$

or

$$\hat{y}(t|t-1) = H^{-1}(z)G(z)u(t) + (1 - H^{-1}(z))y(t).$$

The prediction error is

$$y(t) - \hat{y}(t|t-1) = -H^{-1}(z)G(z)u(t) + H^{-1}(z)y(t)$$

Exercise 2 Show that the prediction error can be written as

$$y(t) - \hat{y}(t|t-1) = e(t).$$

The results of this exercise are very important: *The prediction error for the optimal predictor form a white noise sequence.* The signal $e(t)$ that cannot be predicted from past observations represents new information. For this reason, $e(t)$ is called the **innovation** at time t .

Let us take the prediction error and write it in two ways. We have

$$e(t) = H^{-1}(z)(y(t) - G(z)u(t)) \tag{6}$$

We can also write

$$e(t) = (H^{-1}(z)y(t)) - G(z)(H^{-1}u(t)). \tag{7}$$

Let us consider what we can learn from each of these.

2.2.1 Representation 1

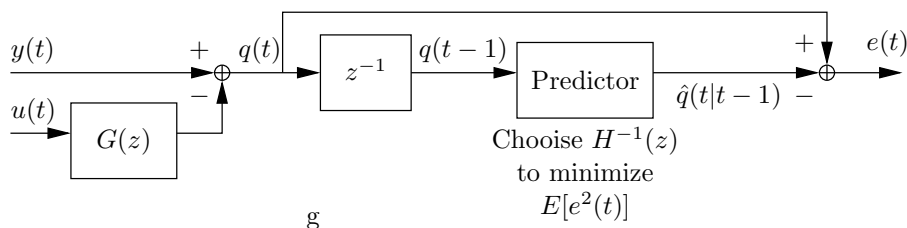
Suppose that $G(z)$ were known. Then in (6) we could form the signal

$$q(t) = y(t) - G(z)u(t),$$

With the input $u(t)$ known and the output $y(t)$ known, $q(t)$ can be computed. Now rewrite (6) as

$$e(t) = H^{-1}(z)q(t) = q(t) - a_1q(t-1) - a_2q(t-2) - \dots - a_{L_H}q(t-L_H) \tag{8}$$

The identification problem at this point is then: Determine the coefficients of the linear predictor with coefficients $\{a_i\}$ that minimizes $E[e(t)^2]$. The idea is suggested by the following figure.



Exercise 3 Let

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{L_H} \end{bmatrix}.$$

Show that the predictor which minimizes $E[e(t)^2]$ in (8) has coefficients determined by

$$R_q \mathbf{a} = \mathbf{r}_q \tag{9}$$

where

$$R_q = E[\mathbf{q}(t)\mathbf{q}(t)^T]$$

is the autocorrelation matrix of $\mathbf{q}(t)$, and where

$$\mathbf{r}_q = E[q(t)\mathbf{q}(t-1)]$$

is the cross-correlation vector between $q(t)$ and $\mathbf{q}(t-1)$, where

$$\mathbf{q}(t) = \begin{bmatrix} q(t) \\ q(t-1) \\ \vdots \\ q(t-L_H+1) \end{bmatrix}.$$

2.2.2 Representation 2

Now let us look at the representation in (7). Suppose that $H^{-1}(z)$ were known, and let

$$\tilde{y}(t) = H^{-1}(z)y(t)$$

$$\tilde{u}(t) = H^{-1}(z)u(t).$$

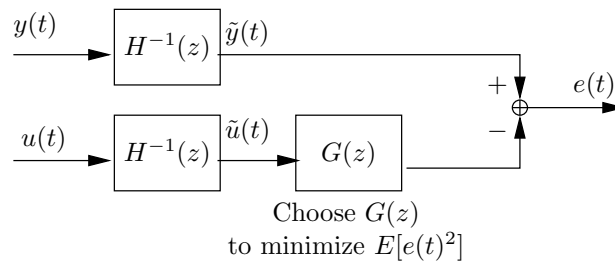
Then (7) can be written as

$$e(t) = \tilde{y}(t) - G(z)\tilde{u}(t). \tag{10}$$

A system identification problem can be stated: Find $G(z)$ to minimize

$$E[e^2(t)].$$

The idea is suggested by the following figure.



Exercise 4 Let

$$\mathbf{g} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ L_g \end{bmatrix}$$

be the vector of coefficients in $G(z)$ and let

$$\tilde{\mathbf{u}}(t) = \begin{bmatrix} \tilde{u}(t) \\ \tilde{u}(t-1) \\ \vdots \\ \tilde{u}(t-L_g) \end{bmatrix}.$$

Show that the filter $G(z)$ which minimizes $E[e^2(t)]$, with $e(t)$ as in (10), has coefficients determined by

$$R_u \mathbf{g} = \mathbf{r}_g \quad (11)$$

where

$$R_u = E[\tilde{\mathbf{u}}(t)\tilde{\mathbf{u}}(t)^T]$$

is the autocorrelation matrix of $\tilde{\mathbf{u}}(t)$ and where

$$\mathbf{r}_g = E[\tilde{y}(t)\tilde{\mathbf{u}}(t)].$$

is the cross-correlation matrix between $\tilde{y}(t)$ and $\tilde{\mathbf{u}}(t)$.

2.2.3 Putting the pieces together

The above sections suggest that:

- Knowing $G(z)$, we can estimate $H^{-1}(z)$, and
- Knowing $H^{-1}(z)$, we can estimate $G(z)$.

However, to begin with we don't know either $G(z)$ or $H^{-1}(z)$. Nevertheless, all is not lost. We start with a guess of $G(z)$, then iterate as follows:

Pick an initial $G(z)$.

1. Solve for an updated $H^{-1}(z)$ using (9).
2. Solve for an updated $G(z)$ using (11).
3. Repeat from step 1 until convergence.

A suggested $G(z)$ is something like

$$G(z) = 0.1z^{-L_g/2}$$

(put a 0.1 in the middle of the impulse response).

3 Estimating the expectations

The algorithm described above requires calculation of the autocorrelation matrices and the cross-correlation vectors. In practice, these are frequently estimated from sample averages. (In other words, we invoke an ergodic assumption.)

To illustrate this idea, suppose that we have a vector random process $\mathbf{x}(t)$, and that we have N vectors of measured sample data which we stack into columns as

$$[\mathbf{x}(1) \quad \mathbf{x}(2) \quad \mathbf{x}(3) \quad \dots \quad \mathbf{x}(N)].$$

Then the autocorrelation matrix $R_x = E[\mathbf{x}(t)\mathbf{x}(t)^T]$ may be estimated as

$$\hat{R}_x = \frac{1}{N-1} \sum_{i=1}^N \mathbf{x}(i)\mathbf{x}(i)^T. \quad (12)$$

Let $y(t)$ be a random process, with measured sample values $y(1), y(2), \dots, y(n)$. The cross correlation vector between a random process $y(t)$ and $\mathbf{x}(t)$, $\mathbf{r} = E[y(t)\mathbf{x}(t)]$ can be estimated as

$$\hat{\mathbf{r}} = \frac{1}{N} \sum_{i=1}^N y(i)\mathbf{x}(i).$$

Exercise 5 Let

$$\mathbf{X} = [\mathbf{x}(1) \quad \mathbf{x}(2) \quad \mathbf{x}(3) \quad \dots \quad \mathbf{x}(N)].$$

show that the estimated autocorrelation matrix R_x in (12) can be written as

$$\hat{R}_x = \frac{1}{N-1} \mathbf{X}\mathbf{X}^T.$$

4 Programming Assignment Part 1

The file `sysid1` on the class website contains input/output data for an unknown system. The first column contains input $u(t)$; the second column contains output $y(t)$. It is known that the filter has five coefficients,

$$G(z) = g_0 + g_1z^{-1} + g_2z^{-2} + g_3z^{-3} + g_4z^{-4}$$

(that is, $L_g = 4$) and the AR process has

$$H^{-1}(z) = 1 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3}$$

(that is, $L_h = 3$).

Write a MATLAB program that reads in the data and, using the iterative procedure outlined in section 2.2.3, estimates $G(z)$ and $H^{-1}(z)$. Also, determine an estimate for σ_e^2 .

Hints:

- You are strongly encouraged to choose your own $G(z)$ and $H^{-1}(z)$ filters and create your own input/output data for debugging purposes. You can verify that your program correctly identifies the parameters in your simulated system, then use your program on the unknown data.
- Helpful MATLAB commands: `filter`, `rand`, `randn`.

5 Adaptive filters

Solving for the unknown filter coefficients requires setting up the estimated autocorrelation and cross-correlation information, then solving a set of linear equations. These will work provided that the system is stationary, that sufficient data are available to make accurate estimates, and that the process does not take too long. (In real-time circumstances, the computation time might exceed the available computational resources.)

Overcoming these demands can be accomplished in part by the use of **adaptive filters**. These are filters that adapt themselves to minimize some mean-squared error criterion. Here we provide a very brief introduction to the topic, by introducing what are known as LMS adaptive filters. (LMS stands for least mean squares.)

Let $u(t)$ be the input to an FIR filter, with output

$$x(t) = \sum_{i=0}^L w_i u(t-i).$$

The coefficients w_i are the FIR filter coefficients, also referred to as filter weights. This filter operation can be represented as

$$x(t) = \mathbf{w}^T \mathbf{u}(t)$$

where

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_L \end{bmatrix}$$

is the vector of filter coefficients and

$$\mathbf{u}(t) = \begin{bmatrix} u(t) \\ u(t-1) \\ \vdots \\ u(t-L) \end{bmatrix}$$

is the vector of filter inputs/memory.

Suppose that some *desired* signal $d(t)$ is available, and we wish to filter the input signal $u(t)$ so that the filter output $x(t)$ matches the desired signal as closely as possible. That is, we form

$$e(t) = x(t) - d(t),$$

and choose \mathbf{w} to minimize $e(t)$. More precisely, we desire to minimize the mean-squared error in $e(t)$:

$$\min_{\mathbf{w}} E[e^2(t)].$$

Exercise 5 Show that $E[e^2(t)]$ is minimized when

$$R\mathbf{w} = \mathbf{p}$$

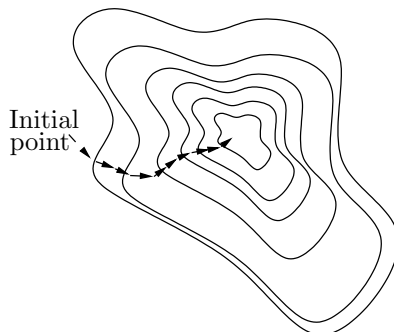
where $R = E[\mathbf{u}(t)\mathbf{u}(t)^T]$ and $\mathbf{p} = E[d(t)\mathbf{u}(t)]$.

By this point, this form of the optimal solution should be familiar.

So far, the filter is not adaptive. We make an adaptive filter as possible. We form an error measure as a function of the filter coefficients:

$$J(\mathbf{w}) = E[(d(t) - \mathbf{w}\mathbf{u}(t))^2]$$

Rather than minimizing all in one step, as before, we compute the gradient of $J(\mathbf{w})$ with respect to \mathbf{w} , and update the current \mathbf{w} by moving in the direction of the negative gradient. That is, we “slide downhill” on the surface of $J(\mathbf{w})$. The idea of sliding downhill is conveyed in the following figure:



This figure shows the contours of a function (of two variables). At each point in the plane, the contours are orthogonal to the direction of the gradient, with the gradient pointing in the direction of greatest increase. Thus, starting at an initial point and moving some small distance from the point in the direction of the negative gradient at that point decreases the function value. Then starting at the new point and moving in the direction of the negative gradient at that point again decreases the function value. A series of such steps will eventually reach a point of local minimum. An update rule such as this is referred to as *steepest descent*.

We denote the gradient of $J(\mathbf{w})$ with respect to \mathbf{w} as $\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w})$. Based on this, an update rule can be written as

$$\mathbf{w}^{[k+1]} = \mathbf{w}^{[k]} - \frac{\mu}{2} \frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}) \Big|_{\mathbf{w}^{[k]}}.$$

That is, the filter weights at the next time around, $\mathbf{w}^{[k+1]}$, are obtained by moving from the current weights, $\mathbf{w}^{[k]}$, in the direction of the negative gradient, evaluated at the current weights, $-\frac{\partial}{\partial J(\mathbf{w})} \Big|_{\mathbf{w}^{[k]}}$. The quantity $\mu/2$ is a “step size,” indicating how far the move should be.

Exercise 6 Show that

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w})$$

can be written as

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}) = 2E[(x(t) - d(t))\mathbf{u}(t)] = 2E[e(t)\mathbf{u}(t)].$$

Hence the weight update rule is

$$\mathbf{w}^{[k+1]} = \mathbf{w}^{[k]} - \mu E[e(t)\mathbf{u}(t)].$$

The last exercise describes how to update the coefficients of an adaptive filter in such a way that the filter $x(t)$ will become as close as possible to $d(t)$ (given enough time). Eventually, the solution will converge to the exact MMSE solution.

However, there is a practical problem with the filter to this point. It requires computing $E[e(t)\mathbf{u}(t)]$. That is, the *expected value* must be computed, which requires (theoretically) some kind of probability information, or some kind of ensemble to average over. This is problematic in practice. In the LMS filter, the *stochastic gradient* approximation is used:

Assume that, for every instance (draw) of a random variable, that that instance is equal to the mean value of the random variable.

This seems somewhat reasonable. By the Chebyshev inequality, we would expect that a randomly-drawn value would be close to the mean value. On the other hand, it is not exact: We don't expect *any* toss of a die (taking outputs in the range $\{1, 2, 3, 4, 5, 6\}$) to have the value of the mean, which is 3.5!

Under the stochastic gradient approximation, we therefore assume that

$$E[e(t)\mathbf{u}(t)] \approx e(t)\mathbf{u}(t).$$

While we don't get precise equality at every time step, over a number of runs, or over a number of time steps, we will be right on average. Under this approximation, the gradient descent does not run strictly downhill in the steepest direction, but it does proceed downhill *on average*.

Under this approximation the filter weight update rule is written

$$\mathbf{w}^{[k+1]} = \mathbf{w}^{[k]} - \mu e(t)\mathbf{u}(t).$$

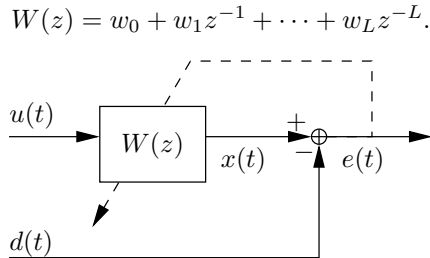
Putting all the pieces together, we obtain the following algorithm for the LMS adaptive filter: Starting from some initial filter coefficients $\mathbf{w}^{[0]}$, and $k = 0$:

```

Given inputs  $u(t)$  and  $d(t)$ :
Form  $\mathbf{u}(t) = [u(t), u(t-1), \dots, u(t-L)]^T$ .
 $x(t) = \mathbf{u}(t)^T \mathbf{w}^{[k]}$  (compute filter output)
 $e(t) = x(t) - d(t)$  (compute error between output and desired)
 $\mathbf{w}^{[k+1]} = \mathbf{w}^{[k]} - \mu e(t)\mathbf{u}(t)$  update filter coefficients)
 $k \leftarrow k + 1$  (increment iteration number)
repeat

```

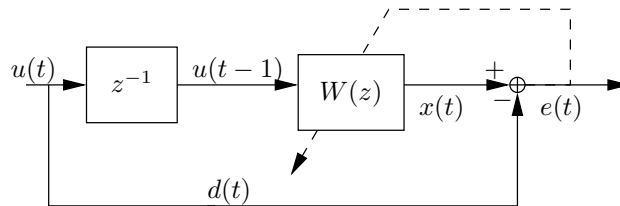
The adaptive filter configuration is often portrayed as in the figure here, where $W(z)$ represents the adaptive filter,



The dashed line feeding back through the filter box is intended to suggest variability, such as in a variable resistor.

5.1 Adaptive predictors

The adaptive filter can be used as an adaptive predictor, that is, a predictor which learns the taps that it needs to predict its input signal.



In this configuration, the input to the filter is $u(t - 1)$. The output of the filter is interpreted as

$$x(t) = \hat{u}(t|t - 1),$$

that is, the prediction of $u(t)$ based on previous information. The desired signal is $d(t) = u(t)$: The filter adapts itself until $d(t)$ is as close as possible to $u(t)$, using the information in $u(t - 1), u(t - 2), \dots, u(t - L)$.

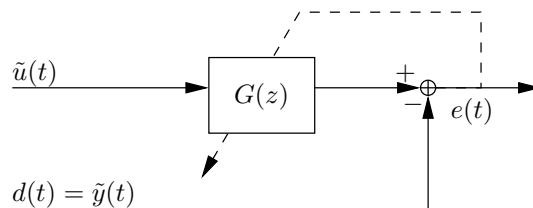
Thus, the predictor in the figure in section 2.2.1 can be replaced with an adaptive filter.

5.2 Adaptive identification

The problem of identifying $G(z)$ to minimize the error in the representation

$$e(t) = \tilde{y}(t) - G(z)\tilde{u}(t)$$

can also be expressed using an adaptive filter, as shown here:



6 Programming Assignment Part 2

- Implement an LMS adaptive filter function in MATLAB. You will want to write it so that the filter memory is passed in (so that you may have multiple matched filters in your code. It might be implement it so that it is declared as

```
[x,e, uvect,w] = function lms(u,d,uvectin,win,mu)
```

where:

- `u` and `d` represent the filter input and desired signal, respectively,
 - `uvectin` represents $\mathbf{u}(t-1)$ (the memory contents of the filter from the last time,
 - `win` represents $\mathbf{w}^{[k]}$, the filter weights from the last time,
 - `mu` represents μ , the adaptive filter step size,
 - `x` and `e` represent the filter output and filter error, respectively,
 - `uvect` represents $\mathbf{u}(t)$ (to be used next time around)
 - `w` represents $\mathbf{w}^{[k+1]}$ (to be used next time around).
- Use your adaptive filter function to replace the prediction block and filtering block of the system identification. (You will need two adaptive filters.) Try various values of μ to see which give good convergence.
 - Make a plot of the filter output error as a function of time for the two filters for various values of μ .

7 What to turn in

- A discussion of what you did, what you learned, and what troubles you had.
- Printouts of your program listings.
- Results of the system identification for the two different programming exercises. (That is, what are the filter taps? What is σ_e^2 ?)
- Plots as appropriate and necessary.

Careful writing is important, as is getting the right answer! (That is, make sure you identify the system correctly!)