

OPTIMAL ESTIMATION University of Florida
Mechanical and Aerospace Engineering

HW 7

Issued: October 12, 2009, Due: October 19, 2009 (in class)

Problem 1.

[30 pt]

write a MATLAB script to simulate and perform a *recursive* least squares estimation of the parameters a_d, b_d of the following system

$$\begin{aligned}x[k+1] &= a_d x[k] + b_d(u[k] + w[k]) \\ y[k] &= x[k] + n[k]\end{aligned}$$

where $w[k]$ and $n[k]$ are sequences of zero-mean Normally distributed random variables (noise) that are mutually uncorrelated (i.e., $\text{cov}(w[i], n[j]) = 0$ for all i, j). In addition, $w[k]$ and $n[k]$ are *white noise*, meaning that $\text{cov}(w[k]w[j]) = 0$ if $k \neq j$ and $\text{cov}(w[k]w[j]) = \sigma_w^2$ and $\text{cov}(n[k]n[j]) = 0$ if $k \neq j$ and $\text{cov}(n[k]n[j]) = \sigma_n^2$. Use the following values: $a_d = 0.998$, $b_d = 0.4995$, $\sigma_n^2 = 0.5$, $\sigma_w^2 = 0.1$. You can assume that the output of `randn`, when properly used, ensures whiteness. See the script `systemID_example1.m` for an example on how to simulate the system and perform *batch* least squares parameter estimation.

ARX model identification Read the following before attempting the next problem.

Consider the following n -th order discrete-time single-input-single-output system

$$\frac{Y(z)}{U(z)} = H(z) = \frac{\alpha_m z^m + \alpha_{m-1} z^{m-1} + \dots + \alpha_1 z + \alpha_0}{z^n + \beta_{n-1} z^{n-1} + \dots + \beta_1 z + \beta_0}$$

This means that in time-domain the output $y[k]$ is related to the input $u[k]$ with the following difference equation:

$$\begin{aligned}y[k+n] + \beta_{n-1}y[k+n-1] + \beta_{n-2}y[k+n-2] + \dots + \beta_1y[k+1] + \beta_0y[k] \\ = \alpha_m u[k+m] + \alpha_{m-1}u[k+m-1] + \dots + \alpha_1u[k+1] + \alpha_0u[k].\end{aligned}\quad (1)$$

We re-write (1) to see that the output $y[\cdot]$ can be computed recursively from the inputs and past outputs:

$$\begin{aligned}y[k+n] &= -\beta_{n-1}y[k+n-1] - \beta_{n-2}y[k+n-2] - \dots - \beta_1y[k+1] - \beta_0y[k] \\ &\quad + \alpha_m u[k+m] + \alpha_{m-1}u[k+m-1] + \dots + \alpha_1u[k+1] + \alpha_0u[k]\end{aligned}\quad (2)$$

or, equivalently,

$$\begin{aligned}y[k] &= -\beta_{n-1}y[k-1] - \beta_{n-2}y[k-2] - \dots - \beta_1y[k-(n-1)] - \beta_0y[k-n] \\ &\quad + \alpha_m u[k-(n-m)] + \alpha_{m-1}u[k-(n-m+1)] + \dots + \alpha_1u[k-(n-1)] + \alpha_0u[k-n]\end{aligned}$$

Note that the output $y[k]$ at time k depends only on the past outputs $y[k-1], y[k-2], \dots, y[k-n+1], y[k-n]$ and inputs $u[k-n+m], u[k-n+m-1], \dots, u[k-n+1], u[k-n]$.

The equation (1) can be compactly written as

$$y[k] = \varphi[k]^T \boldsymbol{\theta} \quad (3)$$

where $\boldsymbol{\theta}$ is a vector containing the parameters $\alpha_i, i = 0, \dots, m$ and $\beta_j, j = 0, \dots, n-1$, and $\varphi[k]$ contains past inputs and outputs:

$$\boldsymbol{\theta} \triangleq \begin{bmatrix} -\beta_{n-1} \\ \vdots \\ -\beta_1 \\ -\beta_0 \\ \alpha_m \\ \alpha_{m-1} \\ \vdots \\ \alpha_1 \\ \alpha_0 \end{bmatrix} \quad \varphi[k] \triangleq \begin{bmatrix} y[k-1] \\ \vdots \\ y[k-n] \\ u[k-n+m] \\ \vdots \\ u[k-n+1] \\ u[k-n] \end{bmatrix}$$

The vector $\varphi[k]$ is called the *regressor vector* and the equation (3) is called an ARX model (Auto-Regression model with eXogeneous inputs).

Eq. (3) is a linear equation of the form $y = Ax$, but there are $n + m + 1$ unknowns and only one equation. By collecting the values of the input and the output signals for N time steps, where $N \geq n + m + 1$, we can construct the following system of linear equations with at least as many equations as there are unknowns:

$$\mathbf{Z} = H\boldsymbol{\theta} \quad (4)$$

where

$$H \triangleq \begin{bmatrix} \varphi^T[1] \\ \varphi^T[2] \\ \vdots \\ \varphi^T[N] \end{bmatrix} = \begin{bmatrix} y[0] & \dots & y[1-n] & u[1-n+m] & \dots & u[1-n] \\ y[1] & \dots & y[2-n] & u[2-n+m] & \dots & u[2-n] \\ \vdots & & \vdots & & & \vdots \\ y[N-1] & \dots & y[N-n] & u[N-n+m] & \dots & u[N-n] \end{bmatrix}, \quad \mathbf{Z} \triangleq \begin{bmatrix} y[1] \\ y[2] \\ \vdots \\ y[N] \end{bmatrix} \quad (5)$$

In practice, when we construct \mathbf{Z} and H from noisy data, there may not be a solution to (4). In that case, we compute the least squares solution to $H\boldsymbol{\theta} = \mathbf{Z}$:

$$\hat{\boldsymbol{\theta}} = (H^T H)^{-1} H^T \mathbf{Z}, \quad (6)$$

which is the *least squares estimate* of $\boldsymbol{\theta}$.

To summarize, least squares identification of the parameters of an ARX model consists of the following steps:

1. Choose an $N > n + m + 1$.

2. Apply a probe input signal $u[k]$, $k \in \{1, 2, \dots, N\}$ to the system. The probe signal should be sufficiently “rich”.
3. Measure the corresponding output $y[k]$, $k \in \{1, 2, \dots, N\}$.
4. Construct the vector \mathbf{Z} and matrix H described in (5).
5. Determine the value of the parameter vector $\boldsymbol{\theta}$ by computing the least squares solution to the equation $H\boldsymbol{\theta} = \mathbf{Z}$.

An example is the following. Suppose the continuous time plant (the ratio of the Laplace transform of the output $y(t)$ to the Laplace transform of the input $u(t)$) is

$$\frac{Y(s)}{U(s)} = H_c(s) = \frac{100}{s^2 + 0.2s + 100}$$

A discrete-time approximation to this plant when the the input and the output are sampled at T second intervals is obtained by replacing s by $\frac{2}{T} \frac{z-1}{z+1}$ (the so-called Tustin approximation). This can be done in MATLAB as follows ($T = 0.001$ is used):

```
Hc = tf(100,[1 0.2 100]);
Hd = c2d(Hc,0.001,'tustin')
Transfer function:
2.5e-05 z^2 + 4.999e-05 z + 2.5e-05
-----
z^2 - 2 z + 0.9998
```

See the MATLAB script `secondorder_simulation.m` for a simulation example.

Problem 2.

[30 pt]

Simulate the following system with an input signal of your choice and estimate the plant parameters (coefficients of the numerator and denominator polynomial) using least squares¹.

$$H_d = \frac{2s + 200}{s^2 + 0.05s + 100}.$$

¹For this exercise, a batch least squares processing is fine, you dont have to do a recursive least squares estimate. In addition, no need to add noise to the input or output. Perform estimation with noise free data.