Chapter 4

Complementary Material

4.4 Sufficiently Rich Inputs

***Proof of Theorem* 4.4.2.** By definition, $u(k)$ contains at least $\frac{\bar{n}}{2}$ distinct frequencies. Hence the spectral distribution function $F_u(\omega)$ of u is nonzero at (at least) $\bar{n}$ points. Since all the poles of $H(z)$ are within the unit circle, applying the Herglotz theorem [1, 2] to $\phi(k) = H(z)u(k)$, we have

$$P \triangleq \lim_{k\to\infty} \frac{1}{k}\sum_{j=1}^{k} \phi(j)\phi^T(j) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H\left(e^{j\omega}\right) dF_u(\omega) H^*\left(e^{j\omega}\right),$$

where $H^*(e^{j\omega})$ denotes the conjugate transpose of $H(e^{j\omega})$.

Note that P is positive semidefinite by definition. To check positive definiteness of P, consider the equation

$$v^T P v = 0 \tag{4C.1}$$

for an arbitrary vector $v \in \mathcal{R}^n$. (4C.1) implies that

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} \left| v^T H\left(e^{j\omega}\right)\right|^2 dF_u(\omega) = 0.$$

This implies that $v^T H(e^{j\omega}) = 0$ at the points of support of $F_u(\omega)$. But since $H(e^{j\omega_1})$, $H(e^{j\omega_2}),\ldots,H(e^{j\omega_{\bar{n}}})$, where $\bar{n} = n+m+1$ are linearly independent vectors over $\mathcal{C}^n$ $\forall\, \omega_1,\omega_2,\ldots,\omega_{\bar{n}} \in [0,2\pi)$, where $\omega_i \neq \omega_j$ for $i \neq j$, this is possible only if $v = 0$, i.e., there is no nonzero solution (for v) of (4C.1). Hence P is positive definite. This in turn implies that ϕ is weakly PE. □

4.5.1 Projection Algorithm

Proof of parameter convergence for the orthogonalized projection algorithm. First, by induction, we show that $P^2(k) = P(k) = P(0)\cdots P(k)$ for any $k \geq 0$. For $k = 0$, we have $P^2(0) = I = P(0)$. Assuming that $P^2(k-1) = P(k-1) = P(0)\cdots P(k-1)$, we show $P^2(k) = P(k) = P(0)\cdots P(k)$ for an arbitrary $k > 0$. If $\phi^T(k)P(k-1)\phi(k) = 0$, we have

$P(k)=P(k-1)$ and hence $P^2(k)=P^2(k-1)=P(k-1)=P(k)$ and $P^2(k)=P(k-1)P(k)=P(0)\cdots P(k-1)P(k)$. Otherwise, from (4.30), particularly from $P(k)=P(k-1)-\frac{P(k-1)\phi(k)\phi^T(k)P(k-1)}{\phi^T(k)P(k-1)\phi(k)}$, we obtain

$$\begin{aligned}
P^2(k) &= P^2(k-1)-\frac{P^2(k-1)\phi(k)\phi^T(k)P(k-1)}{\phi^T(k)P(k-1)\phi(k)} - \frac{P(k-1)\phi(k)\phi^T(k)P^2(k-1)}{\phi^T(k)P(k-1)\phi(k)} \\
&\quad + \frac{P(k-1)\phi(k)\phi^T(k)P^2(k-1)\phi(k)\phi^T(k)P(k-1)}{\left(\phi^T(k)P(k-1)\phi(k)\right)^2} \\
&= P(k-1)-2\frac{P(k-1)\phi(k)\phi^T(k)P(k-1)}{\phi^T(k)P(k-1)\phi(k)} \\
&\quad + \frac{P(k-1)\phi(k)\left(\phi^T(k)P(k-1)\phi(k)\right)\phi^T(k)P(k-1)}{\left(\phi^T(k)P(k-1)\phi(k)\right)^2} \\
&= P(k-1)- \frac{P(k-1)\phi(k)\phi^T(k)P(k-1)}{\phi^T(k)P(k-1)\phi(k)} \\
&= P(k),
\end{aligned}$$

and hence

$$\begin{aligned}
P(k) &= P^2(k-1)- \frac{P^2(k-1)\phi(k)\phi^T(k)P(k-1)}{\phi^T(k)P(k-1)\phi(k)} \\
&= P(k-1)P(k)=P(0)\cdots P(k-1)P(k).
\end{aligned}$$

Thus, we have proven that

$$P^2(k)=P(k)=P(0)\cdots P(k) \quad \forall k\geq 0\,. \tag{4C.2}$$

Next, again by induction, we show the following three properties of $P(k)x$ that are valid for any $x\in\mathcal{R}^n$ and any $k\geq 1$:

$$\phi^T(i)P(k)x=0 \text{ for } i=1,\ldots,k\,, \tag{4C.3}$$

$$P(k)x \text{ is a linear combination of } \phi(1),\ldots,\phi(k),x\,, \tag{4C.4}$$

$$P(k)x=0 \quad\Leftrightarrow\quad x \text{ is a linear combination of } \phi(1),\ldots,\phi(k)\,. \tag{4C.5}$$

For $k=1$, if $\phi^T(1)P(0)\phi(1)=\phi^T(1)\phi(1)=0$, then $\phi(1)=0$ and $P(1)=P(0)=I$; therefore, all (4C.3)–(4C.5) hold. If $\phi^T(1)P(0)\phi(1)=\phi^T(1)\phi(1)\neq 0$, then

$$P(1)=I-\frac{\phi(1)\phi^T(1)}{\phi^T(1)\phi(1)}$$

and hence (i) $\phi^T(1)P(1)x = \phi^T(1)x - \frac{\phi^T(1)\phi(1)}{\phi^T(1)\phi(1)}\phi^T(1)x = 0$ and (ii) $P(1)x = x - \frac{\phi(1)\phi^T(1)}{\phi^T(1)\phi(1)}x$, which is a linear combination of $\phi(1), x$ (noting that $\frac{\phi^T(1)x}{\phi^T(1)\phi(1)}$ is a scalar) and which is zero if and only if $x = \alpha\phi(1)$ for some $\alpha \in \mathcal{R}$, i.e., (4C.3)–(4C.5) hold. To complete the induction procedure, assuming that (4C.3)–(4C.5) hold for time step $k-1$, we show that they also hold for step k.

By the induction assumption, we have

$$\phi^T(i)P(k-1)x = 0 \text{ for } i = 1,\ldots,k-1, \tag{4C.6}$$

$$P(k-1)x \text{ is a linear combination of } \phi(1),\ldots,\phi(k-1), x, \tag{4C.7}$$

$$P(k-1)x = 0 \;\Leftrightarrow\; x \text{ is a linear combination of } \phi(1),\ldots,\phi(k-1). \tag{4C.8}$$

If $\phi^T(k)P(k-1)\phi(k) = 0$, we have $P(k) = P(k-1)$ and $P(k)\phi(k) = P(k-1)\phi(k) = 0$ (noting that $P(k-1) = Q^TQ$ for some real matrix Q). Therefore, from (4C.6), $\phi^T(i)P(k)x = \phi^T(i)P(k-1)x = 0$ for $i = 1,\ldots,k-1$. Moreover, $P(k)\phi(k) = P(k-1)\phi(k) = 0$ implies that $\phi^T(k)P(k)x = x^TP(k)\phi(k) = 0$. Hence (4C.3) holds. (4C.4) also holds using (4C.7) since $P(k)x = P(k-1)x$. If $P(k)x = P(k-1)x = 0$, x is a linear combination of $\phi(1),\ldots,\phi(k-1)$. If x is a linear combination of $\phi(1),\ldots,\phi(k)$, then $x = x_1 + \alpha\phi(k)$ for some $\alpha \in \mathcal{R}$, where x_1 is a linear combination of $\phi(1),\ldots,\phi(k-1)$. This implies that $P(k)x = P(k-1)x_1 + \alpha P(k)\phi(k) = 0 + 0 = 0$. Therefore, (4C.5) holds as well.

If $\phi^T(k)P(k-1)\phi(k) \neq 0$, (4C.3) follows from (4.30) and (4C.6), observing

$$\phi^T(i)P(k)x = \phi^T(i)P(k-1)x - \frac{\phi^T(i)P(k-1)\phi(k)\phi^T(k)P(k-1)}{\phi^T(k)P(k-1)\phi(k)}x = 0 - 0 = 0$$

for $i = 1,\ldots,k-1$, and

$$\phi^T(k)P(k)x = \phi^T(k)P(k-1)x - \frac{\phi^T(k)P(k-1)\phi(k)}{\phi^T(k)P(k-1)\phi(k)}\phi^T(k)P(k-1)x = 0$$

for $i = k$. Again from (4.30) we have

$$P(k)x = P(k-1)x - P(k-1)\phi(k)\frac{\phi^T(k)P(k-1)x}{\phi^T(k)P(k-1)\phi(k)}.$$

Observing that $\frac{\phi^T(k)P(k-1)x}{\phi^T(k)P(k-1)\phi(k)}$ is scalar, (4C.7) implies (4C.4).

$$P(k)x = \left(I - \frac{P(k-1)\phi(k)\phi^T(k)}{\phi^T(k)P(k-1)\phi(k)} \right) P(k-1)x.$$

If x is a linear combination of $\phi(1),\ldots,\phi(k)$, then $x = x_1 + \alpha\phi(k)$ for some $\alpha \in \mathcal{R}$, where x_1 is a linear combination of $\phi(1),\ldots,\phi(k-1)$. This implies that

$$\begin{aligned} P(k)x = & \left(I - \frac{P(k-1)\phi(k)\phi^T(k)}{\phi^T(k)P(k-1)\phi(k)} \right) P(k-1)x_1 \\ & + \alpha \left(P(k-1)\phi(k) - P(k-1)\phi(k) \frac{\phi^T(k)P(k-1)\phi(k)}{\phi^T(k)P(k-1)\phi(k)} \right) = 0 + 0 = 0. \end{aligned} \tag{4C.9}$$

If x is not a linear combination of $\phi(1),\ldots,\phi(k)$, then $x = x_1 + x_2 + \alpha\phi(k)$ for some $\alpha \in \mathcal{R}$, where x_1 is a linear combination of $\phi(1),\ldots,\phi(k-1)$ and the nonzero vector $x_2 \perp \phi(i)$ for $i = 1,\ldots,k$. Now (4C.9) implies that $P(k)x = P(k)x_2 = P(k-1)x_2 - \frac{\phi^T(k)P(k-1)x_2}{\phi^T(k)P(k-1)\phi(k)} P(k-1)\phi(k)$. (4C.7) and (4C.8) imply that $P(k-1)x_2 = \sum_{i=1}^{k-1} \alpha_i\phi(i) + \alpha_k x_2 \neq 0$ and $\frac{\phi^T(k)P(k-1)x_2}{\phi^T(k)P(k-1)\phi(k)} P(k-1)\phi(k) = \sum_{i=1}^{k} \beta_i\phi(i)$ for some $\alpha_1,\ldots,\alpha_k,\beta_1,\ldots,\beta_k \in \mathcal{R}$. Furthermore, (4C.2), (4C.8) imply that $P(k-1)x_2 = P^2(k-1)x_2 = P(k-1)\left(\sum_{i=1}^{k-1} \alpha_i\phi(i) + \alpha_k x_2 \right) = \alpha_k P(k-1)x_2 \neq 0$, i.e., $\alpha_k \neq 0$. Hence $P(k)x = P(k)x_2 = \sum_{i=1}^{k-1} \alpha_i\phi(i) + \alpha_k x_2 - \sum_{i=1}^{k} \beta_i\phi(i) \neq 0$ since $\alpha_k \neq 0$ and $x_2 \perp \phi(i)$ for $i = 1,\ldots,k$, which completes the proof of (4C.5). We have thus completed the proof (by induction) of (4C.3)–(4C.5).

Next, defining $\tilde{\theta}(k) = \theta(k) - \theta^*$, we prove by induction that

$$\tilde{\theta}^T(k)\phi(i) = 0 \quad \forall i \leq k \tag{4C.10}$$

for any $k \geq 1$. For any $k \geq 1$, if $\phi^T(k)P(k-1)\phi(k) = 0$, then

$$\tilde{\theta}^T(k)\phi(i) = \tilde{\theta}^T(k-1)\phi(i) \quad \forall i \leq k, \tag{4C.11}$$

and otherwise (4.30) implies that

$$\begin{aligned} \tilde{\theta}(k) &= \tilde{\theta}(k-1) + P(k-1)\phi(k)\varepsilon(k) \\ &= \tilde{\theta}(k-1) - \frac{P(k-1)\phi(k)\phi^T(k)}{\phi^T(k)P(k-1)\phi(k)} \tilde{\theta}(k-1) \end{aligned}$$

and hence

$$\tilde{\theta}^T(k)\phi(i) = \phi^T(i)\tilde{\theta}(k-1) - \frac{\phi^T(i)P(k-1)\phi(k)}{\phi^T(k)P(k-1)\phi(k)}\phi^T(k)\tilde{\theta}(k-1) \quad \forall i \le k. \qquad (4C.12)$$

For $k=1$, if $\phi^T(1)P(0)\phi(1) = \phi^T(1)\phi(1) = 0$, then $\phi(1) = 0$ and hence $\tilde{\theta}^T(1)\phi(1) = 0$. Otherwise, (4C.12) implies that $\tilde{\theta}^T(1)\phi(1) = 0$. To complete the induction procedure, for an arbitrary $k > 0$, assuming that $\tilde{\theta}^T(k-1)\phi(i) = 0$ for any $i \le k-1$, we show that $\tilde{\theta}^T(k)\phi(i) = 0$ for any $i \le k$. If $\phi^T(k)P(k-1)\phi(k) \ne 0$, then for $i = k$, we have $\tilde{\theta}^T(k)\phi(k) = 0$ from (4C.12). For $i < k$, by the induction assumption, $\phi^T(i)\tilde{\theta}(k-1) = 0$ $\forall i \le k-1$. Hence, (4C.12) also implies that

$$\tilde{\theta}^T(k)\phi(i) = -\phi^T(i)P(k-1)\frac{\phi(k)\phi^T(k)\tilde{\theta}(k-1)}{\phi^T(k)P(k-1)\phi(k)} \quad \forall i \le k-1.$$

But then because of (4C.3), we have $\tilde{\theta}^T(k)\phi(i) = 0 \quad \forall i \le k-1$.

If $\phi^T(k)P(k-1)\phi(k) = 0$, $\tilde{\theta}(k) = \tilde{\theta}(k-1)$. Hence, by the induction assumption we have $\tilde{\theta}^T(k)\phi(i) = \tilde{\theta}^T(k-1)\phi(i) = 0 \quad \forall i \le k-1$. For $i = k$, $\phi^T(k)P(k-1)\phi(k) = 0$ implies that $P(k-1)\phi(k) = 0$ (noting that $P(k-1) = Q^TQ$ for some real matrix Q). But then (4C.5) implies that $\phi(k)$ is a linear combination of $\phi(1),\ldots,\phi(k-1)$ and hence $\tilde{\theta}^T(k)\phi(k) = 0,$ as well. This completes the proof of (4C.10).

If rank $[\phi(1),\ldots,\phi(m)] = n$, (4C.10), i.e., $\tilde{\theta}^T(m)\phi(i) = 0 \quad \forall i \in \{1,\ldots,m\},$ implies that $\tilde{\theta}(m) = 0,$ and hence the result follows. □

4.8 Parameter Identification Based on DPM

Let us consider the following discrete-time DPM:

$$z(k) = W(\mathrm{z})\theta^{*T}\psi(k), \qquad (4C.13)$$

where $z(k), \psi(k)$ are measured, $W(\mathrm{z})$ is a known proper transfer function with stable poles, and θ^* is the parameter vector to be estimated. We rewrite (4C.13) in the form

$$z(k) = W_m(\mathrm{z})\theta^{*T}\phi(k),$$

where $W(\mathrm{z}) = W_m(\mathrm{z})L(\mathrm{z})$, $\phi(k) = L^{-1}(\mathrm{z})\psi(k)$, and $L(z)$ is chosen so that $L^{-1}(z)$ is proper with stable poles and $W_m(\mathrm{z})$ is proper and strictly positive real (SPR). Let $\theta(k)$ be the estimate of θ^* at instant k. Then

$$\hat{z}(k) = W_m(\mathrm{z})\theta^T(k)\phi(k)$$

and the estimation error is defined as

$$\varepsilon(k) = z(k) - \hat{z}(k) - W_m(\mathrm{z})m^2(k)\varepsilon(k),$$

where $m(k) > 0$ is the normalizing signal designed to bound $\phi(k)$ from above. Since $W_m(\mathrm{z})$ is SPR there exist $P = P^T > 0$, $L = L^T > 0$, a constant ν, and a vector q such that

$$\begin{aligned} &A^T PA - P = -qq^T - L, \\ &A^T Pb = c - \nu q, \quad 2d - b^T Pb = \nu^2, \end{aligned} \tag{4C.14}$$

where (A,b,c,d) is a minimal state-space realization of $W_m(\mathrm{z}) = c^T(\mathrm{z}I - A)^{-1}b + d$.

Since

$$\varepsilon(k) = W_m(z)\left[-\tilde{\theta}^T(k)\phi(k) - m^2(k)\varepsilon(k)\right]$$

we can write

$$\begin{aligned} e(k+1) &= Ae(k) + b\left(-\tilde{\theta}^T(k)\phi(k) - m^2(k)\varepsilon(k)\right), \\ \varepsilon(k) &= c^T e(k) + d\left(-\tilde{\theta}^T(k)\phi(k) - m^2(k)\varepsilon(k)\right). \end{aligned}$$

Let us consider the function

$$V(k) = e^T(k)Pe(k) + \tilde{\theta}^T(k)\Gamma^{-1}\tilde{\theta}(k)$$

and compute

$$\begin{aligned} \Delta V(k) &\triangleq V(k+1) - V(k) \\ &= e^T(k+1)Pe(k+1) - e^T(k)Pe(k) + \tilde{\theta}^T(k+1)\Gamma^{-1}\tilde{\theta}(k+1) - \tilde{\theta}^T(k)\Gamma^{-1}\tilde{\theta}(k) \\ &= e^T(k)\left(A^T PA - P\right)e(k) + u^2(k)b^T Pb + 2e^T(k)A^T Pbu(k) \\ &\quad + \tilde{\theta}^T(k+1)\Gamma^{-1}\tilde{\theta}(k+1) - \tilde{\theta}^T(k)\Gamma^{-1}\tilde{\theta}(k), \end{aligned}$$

where $u(k) = -\tilde{\theta}^T(k)\phi(k) - m^2(k)\varepsilon(k)$. Using (4C.14), we obtain

$$\begin{aligned} \Delta V(k) &= -e^T(k)qq^T e(k) - e^T(k)Le(k) + \left(2d - \nu^2\right)u^2(k) + 2e^T(k)cu(k) \\ &\quad - 2e^T(k)q\nu u(k) + \tilde{\theta}^T(k+1)\Gamma^{-1}\tilde{\theta}(k+1) - \tilde{\theta}^T(k)\Gamma^{-1}\tilde{\theta}(k) \\ &= -e^T(k)Le(k) - \left(e^T(k)q + \nu u(k)\right)^2 + 2\varepsilon(k)u(k) + \tilde{\theta}^T(k+1)\Gamma^{-1}\Delta\tilde{\theta}(k) \\ &\quad + \tilde{\theta}^T(k)\Gamma^{-1}\Delta\tilde{\theta}(k) \\ &\le -e^T(k)Le(k) - 2\tilde{\theta}^T(k)\phi(k)\varepsilon(k) - 2\varepsilon^2(k)m^2(k) + \tilde{\theta}^T(k+1)\Gamma^{-1}\Delta\tilde{\theta}(k) \\ &\quad + \tilde{\theta}^T(k)\Gamma^{-1}\Delta\tilde{\theta}(k), \end{aligned}$$

where

$$\Delta\tilde{\theta}(k)=\tilde{\theta}(k+1)-\tilde{\theta}(k)=\theta(k+1)-\theta(k).$$

Choosing

$$\Delta\tilde{\theta}(k)=\theta(k+1)-\theta(k)=\Gamma\varepsilon(k)\phi(k),$$

we have

$$\Delta V(k)\le -e^T(k)Le(k)+\varepsilon^2(k)\phi^T(k)\Gamma\phi(k)-2\varepsilon^2(k)m^2(k)\le 0,$$

and hence $V(k)(k),\theta(k),e(k),\varepsilon(k)\in\ell_\infty$, provided that $\lambda_{\max}(\Gamma)\le 2$. Following the same analysis as in the gradient algorithm case, we can establish that $\varepsilon(k)m(k),\epsilon(k),\Delta\tilde{\theta}(k)\in\ell_2$ and $\varepsilon(k),\varepsilon(k)m(k),\Delta\tilde{\theta}(k)\to 0$ as $k\to\infty$.

Let us summarize the algorithm for estimating θ^* in (4C.13):

$$\begin{aligned}\varepsilon(k)&=z(k)-W_m(z)\left(\theta^T(k)\phi(k)+\varepsilon(k)m^2(k)\right),\\ \theta(k+1)&=\theta(k)+\Gamma\varepsilon(k)\phi(k).\end{aligned}\tag{4C.15}$$

The normalizing signal $m(k)$ is designed to bound $\phi(k)$ from above. Some choices for $m(k)$ include $m^2(k)=1+\phi^T(k)\phi(k)$ or $m^2(k)=\alpha+\phi^T(k)P_0\phi(k)$ for some constant $\alpha>0$ and $P_0=P_0^T>0$.

Theorem 4.8.1 *The PI algorithm* (4C.15) *with* $\lambda_{\max}(\Gamma)\le 2$ *guarantees that*

(i) $\theta(k)\in\ell_\infty$.

(ii) $\varepsilon(k),\varepsilon(k)m(k),\varepsilon(k)\phi(k),|\theta(k)-\theta(k-N)|\in\ell_2\cap\ell_\infty$.

(iii) $\varepsilon(k),\varepsilon(k)m(k),|\varepsilon(k)\phi(k)|,|\theta(k)-\theta(k-N)|\to 0$ *as* $k\to\infty$, *where* $N\ge 0$ *is any finite integer.*

(iv) *If* $\frac{\phi(k)}{m(k)}$ *is PE, then* $\theta(k)\to\theta^*$ *exponentially fast.*

Proof. The proof of (i)–(iii) is given above. The proof of (iv) follows from that for the SPM model and is left as an exercise. □

4.9 Parameter Identification Based on B-SPM

Consider the B-SPM

$$z(k)=\rho^*\left(\theta^{*T}\phi(k)+z_0(k)\right),\tag{4C.16}$$

where $z(k),z_0(k),\phi(k)$ are available for measurement at each time k and ρ^*,θ^* are scalar and vector unknown parameters, respectively. The estimation error is generated as

$$\begin{aligned}\hat{z}(k) &= \rho(k-1)\left(\theta^T(k-1)\phi(k)+z_0(k)\right),\\ \varepsilon(k) &= \frac{z(k)-\hat{z}(k)}{m^2(k)},\end{aligned} \tag{4C.17}$$

where $\rho(k-1),\theta(k-1)$ are the estimates of ρ^*,θ^* generated at time $k-1$, used at time k to construct $\hat{z}(k),\varepsilon(k)$. The normalizing signal $m(k)$ is designed to bound from above $\phi(k),z(k)$. An example for $m(k)$ is

$$m^2(k)=1+\phi^T(k)\phi(k)+z^2(k).$$

Let us consider the cost function

$$\begin{aligned}J(\rho,\theta) &= \frac{\varepsilon^2(k)m^2(k)}{2}=\frac{\left[z(k)-\rho(k-1)\left(\theta^T(k-1)\phi(k)+z_0(k)\right)\right]^2}{2m^2(k)}\\ &= \frac{\left[z(k)-\rho^*\theta^T(k-1)\phi(k)-\rho(k-1)\xi(k)+\rho^*\xi(k)-\rho^*z_0(k)\right]^2}{2m^2(k)},\end{aligned}$$

where

$$\xi(k)=\theta^T(k-1)\phi(k)+z_0(k) \tag{4C.18}$$

is available for measurement at time k. Using the steepest descent method to minimize $J(\rho,\theta)$ with respect to ρ,θ, we obtain

$$\rho(k)=\rho(k-1)-\gamma\varepsilon(k)\xi(k), \tag{4C.19}$$

$$\theta(k)=\theta(k-1)-\Gamma\,\mathrm{sgn}\left(\rho^*\right)\varepsilon(k)\phi(k), \tag{4C.20}$$

where $\gamma>0$ and $\Gamma=\Gamma^T>0$ are the adaptive gains.

Theorem 4.9.1 *If the adaptive gains γ,Γ are designed to satisfy*

$$0<\gamma<1,\quad \rho_0\lambda_{\max}(\Gamma)<1,$$

where $\rho_0>0$ is an upper bound for $|\rho^*|$ and $m(k)$ is chosen as $m^2(k)=1+\gamma\xi^2(k)+\rho_0\phi^T(k)\Gamma\phi(k)$, then

(i) $\theta(k),\rho(k)\in\ell_\infty$.

(ii) $\varepsilon(k),\varepsilon(k)m(k),\varepsilon(k)\phi(k),\left|\theta(k)-\theta(k-N)\right|,\left|\rho(k)-\rho(k-N)\right|\in\ell_2\bigcap\ell_\infty$ *and converge to zero as $k\to\infty$ for any finite integer $N\geq 0$.*

Proof. Define the parameter errors $\tilde{\rho}(k)=\rho(k)-\rho^*$, $\tilde{\theta}(k)=\theta(k)-\theta^*$. The estimation error ε(k) may be expressed in terms of the parameter errors using (4C.16), (4C.17) as

$$\varepsilon(k)=\frac{\rho^*\left[\theta^{*T}\phi(k)+z_0(k)\right]-\rho(k-1)\left(\theta^T(k-1)\phi(k)+z_0(k)\right)}{m^2(k)}$$
$$=\frac{\tilde{\rho}(k-1)\xi(k)+\rho^*\tilde{\theta}^T(k-1)\phi(k)}{m^2(k)},$$

where $\xi(k)=\theta^T(k-1)\phi(k)+z_0(k)$. The adaptive law (4C.19), (4C.20) may also be expressed as

$$\tilde{\rho}(k)=\tilde{\rho}(k-1)-\gamma\varepsilon(k)\xi(k),$$
$$\tilde{\theta}(k)=\tilde{\theta}(k-1)-\Gamma\,\mathrm{sgn}\left(\rho^*\right)\varepsilon(k)\phi(k)$$

by subtracting ρ^* and θ^* from each side of (4C.19), (4C.20), respectively. We consider the positive definite function

$$V(k)=\frac{\tilde{\rho}^2(k)}{2\gamma}+\frac{\left|\rho^*\right|}{2}\tilde{\theta}^T(k)\Gamma^{-1}\tilde{\theta}(k).$$

Then

$$\Delta V(k)\triangleq V(k)-V(k-1)=-\varepsilon^2(k)m^2(k)\left[1-\frac{\left(\gamma\xi^2(k)+\left|\rho^*\right|\phi^T(k)\Gamma\phi(k)\right)}{2m^2(k)}\right].$$

It is clear that if $0<\gamma<1$ and $|\rho^*|\|\Gamma\|\le\rho_0\lambda_{\max}(\Gamma)<1$ or $m^2(k)=1+\gamma\xi^2(k)+\rho_0\phi^T(k)\Gamma\phi(k)$, then

$$\frac{\gamma\xi^2(k)+\left|\rho^*\right|\phi^T(k)\Gamma\phi(k)}{2m^2(k)}<1,$$

which implies that

$$\Delta V(k)\le -c\ \ \varepsilon^2(k)m^2(k)$$

for some constant $0<c<1$. The rest of the analysis follows in the same manner as in the case of linear SPM. □

4.13 Examples Using the Adaptive Control Toolbox

4.13.1 Gradient Algorithms

The MATLAB functions **udgrad** and **dgradl** or the Simulink block **Parameter Estimator** can be used to simulate the gradient algorithm of section 4.5.2, as shown in Examples 4.13.1–4.13.3.

Example 4.13.1 Consider the ARMA model of Example 2.1.2, i.e.,

$$y(t) = 1.9y(t-1) - 0.9y(t-2) + u(t-1) + 0.5u(t-2) + 0.25u(t-3)$$

with the same input $u(t) = \sin(\frac{\pi}{25}t)$ and zero initial conditions. For the corresponding SPM

$$z(t) = \theta^{*T}\phi(t),$$

where

$$z(t) = y(t),$$
$$\phi(t) = \left[u(t-1),\ u(t-2),\ u(t-3),\ -y(t-1),\ -y(t-2)\right]^T,$$
$$\theta^* = \left[b_0,\ b_1,\ b_2,\ a_1,\ a_2\right]^T = \left[1,\ 0.5,\ 0.25,\ -1.9,\ 0.9\right]^T,$$

assume that all the parameters b_0, b_1, b_2, a_1, a_2 are unknown. Now let us apply the gradient algorithm based on instantaneous cost to estimate these parameters. Fixing the initial values of the parameter estimates as zeros and the gain vector as I_5, and having the time histories of ϕ and z, we can obtain the parameter estimates using the following code:

```
tt = 0:t_final;
u = sin(pi/25*t);
Gamma = 1;
theta =
dgradl(y,phi,zeros(1,length(y)),zeros(5,1),Gamma,Gamma,[],[
]);
zhat = sum(theta(:,2:length(tt)).*phi);
epsilon = (z-zhat)./(1+sum(phi.*(Gamma*phi)));
```

The results are plotted in Figures 4C.1 and 4C.2. Output estimation is successful with a fast convergence rate, although the parameter estimates do not converge to the actual values of the parameters exactly. The latter observation can be explained by the phenomena of persistent excitation and dominant richness; i.e., $u(t) = \sin(\frac{\pi}{25}t)$ is not dominantly rich enough for convergence of the parameter estimates. Instead, if we apply $u(t) = \sin(\frac{\pi}{25}t) + 0.5\cos(\frac{\pi}{32}t) - 0.25\sin(\frac{\pi}{20}t + \frac{2\pi}{5})$, which is dominantly richer, we obtain better results in terms of convergence to the actual parameter values, as shown in Figure 4C.3.

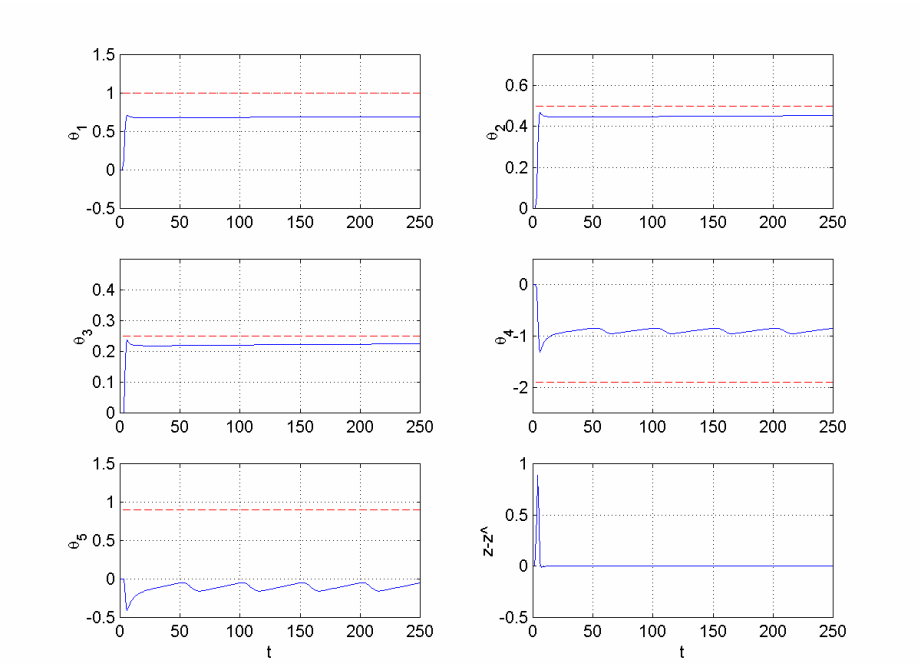


Figure 4C.1 *Time histories of the parameter estimates* θ_i *and the unnormalized estimation error* $z-\hat{z}$ *in Example* 4.13.1 ($u(t)=\sin(\frac{\pi}{25}t)$).

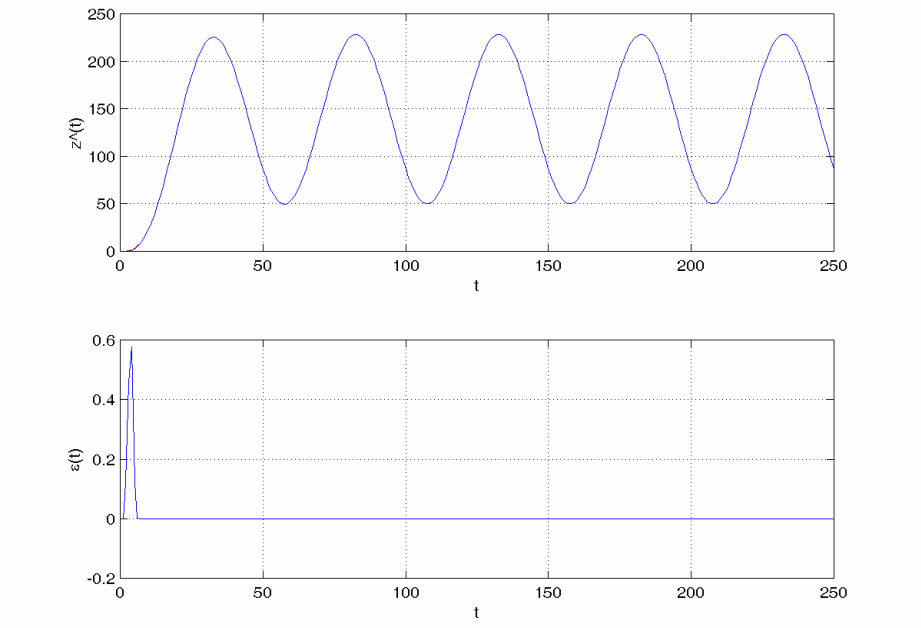


Figure 4C.2 *Time histories of the output estimate* $\hat{z}$ *and the normalized estimation error* ε *in Example* 4.13.1 ($u(t)=\sin(\frac{\pi}{25}t)$).

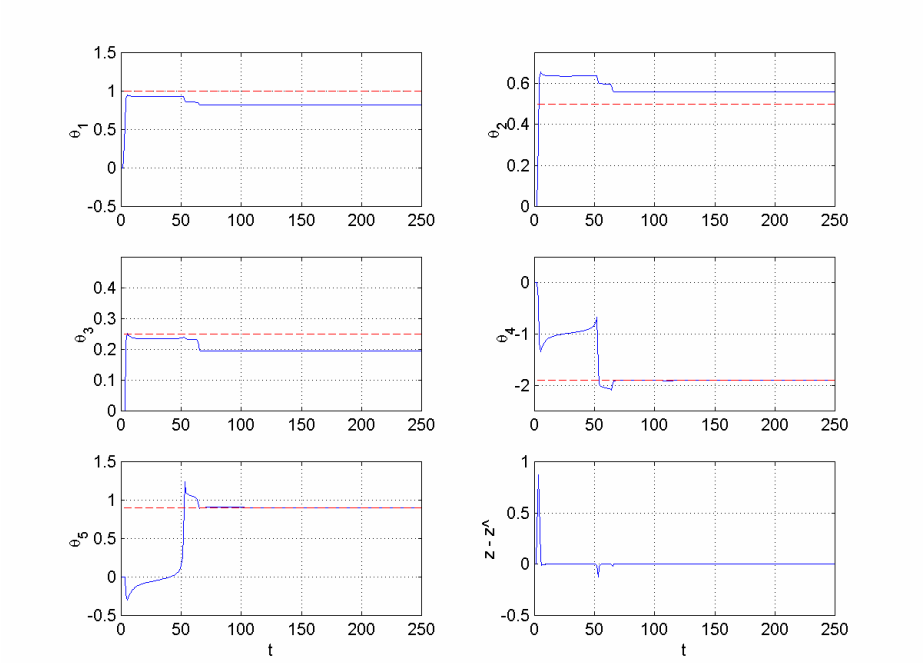
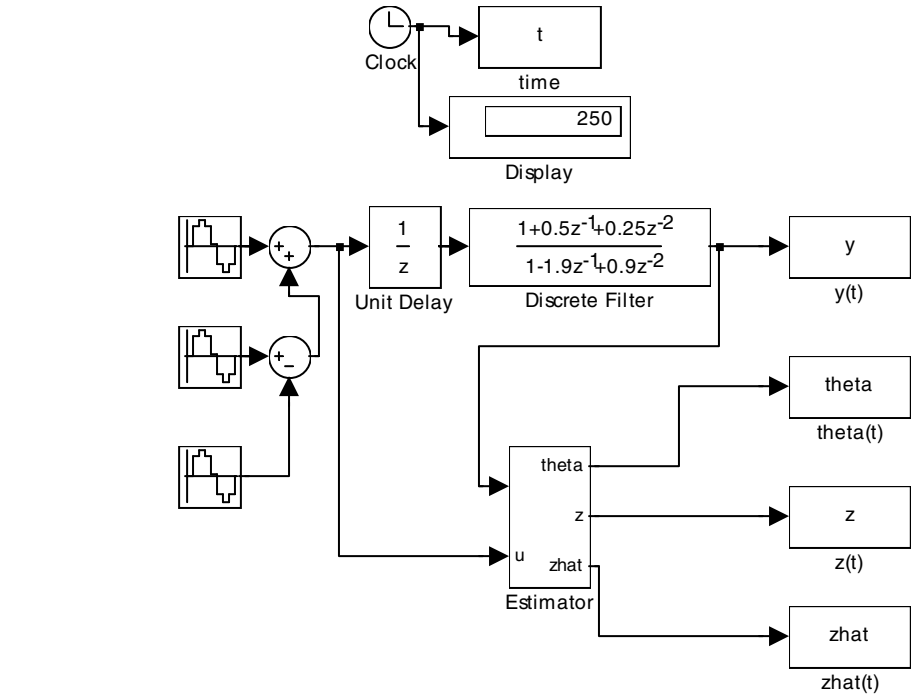


Figure 4C.3 *Time histories of the parameter estimates* θ_i *and the unnormalized estimation error* $z-\hat{z}$ *in Example* 4.13.1 ($u(t)=\sin(\frac{\pi}{25}t)+0.5\cos(\frac{\pi}{32}t)-0.25\sin(\frac{\pi}{20}t+\frac{2\pi}{5})$).

The code above implements the batch (offline) process of the gradient algorithm-based estimation using **`dgradl`**. Instantaneous (online) estimation can be performed using **`udgrad`**. For example, the results above can also be obtained using the following code:

```
n = 2; m = 2; d = 1;
K = [1 0 0 0];
Gamma = 1;
y(1) = 0;
[nstate,x] = ufilt('init',b,[1 a]);
[nstate_lm, x_lm] = uarma2lm('init',[n d m],K);
[z(1), phi(:,1)] = uarma2lm('output',x_lm, [u(1) y(1)], [n
d m],K);
x_lm = uarma2lm('state',x_lm, [u(1) y(1)], [n d m],K);
theta(:,1) = zeros(5,1);

for k = 2:t_final,
   y(k) = ufilt('output', x, u(k-1), b, [1 a],0);
   x = ufilt('state', x, u(k-1), b, [1 a]);
   [z(k), phi(:,k)] = uarma2lm('output',x_lm, [u(k) y(k)], [n d m],K);
   x_lm = uarma2lm('state',x_lm, [u(k) y(k)], [n d m],K);
   theta(:,k) = udgrad(theta(:,k-1),z(k),phi(:,k),[],0,Gamma,[],Gamma);
end;
```



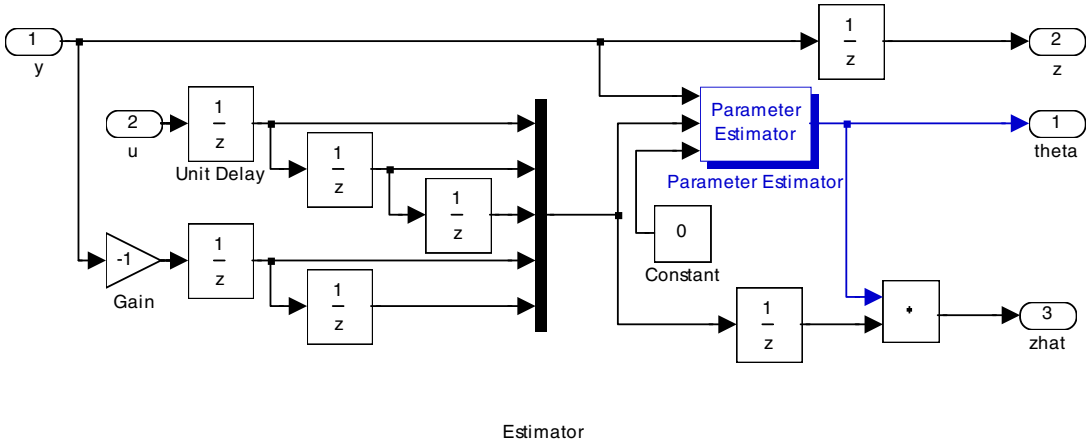


Figure 4C.4 *Simulink scheme for estimating the coefficients of the ARMA model in Example* 4.13.1.

The online estimation process can be simulated using Simulink as well. The results above can be reproduced using the Simulink scheme drawn in Figure 4C.4 and choosing the parameters of the block **`Parameter Estimator`** appropriately via its user interface. ■

Example 4.13.2 Consider the digital broadcast signal

$$y(t) = A\sin(t\omega + \varphi)\,,$$

which has a known frequency ω but an unknown phase φ and an unknown amplitude A. Our objective is to estimate A and φ using the knowledge of ω and measurements of y. We need a linear parametric model for this task. By using the identity

$$A\sin(t\omega+\varphi) = A\cos\varphi\sin t\omega + A\sin\varphi\cos t\omega,$$

we obtain

$$y = \theta^{*T}\phi$$

with $\theta^* = [A_1 \quad A_2]^T$ and $\phi(t) = [\sin t\omega \quad \cos t\omega]^T$, where $A_1 = A\cos\varphi$ and $A_2 = A\sin\varphi$. Now, θ^* can be estimated as $\theta = [\hat{A}_1 \quad \hat{A}_2]^T$ using the gradient algorithm of section 4.5.2. Once θ^* is estimated as $\theta = [\hat{A}_1 \quad \hat{A}_2]^T$, we can obtain the estimates of A and φ as

$$\hat{A} = \sqrt{\hat{A}_1^2 + \hat{A}_2^2} \quad \text{and} \quad \hat{\varphi} = \cos^{-1}\left(\frac{\hat{A}_1}{\hat{A}}\right).$$

Now, assume that $A = 10$, $\varphi = 30^\circ = \frac{\pi}{6}$ rad, and $\omega = 0.05\pi$ rad. Choosing the coefficient matrix Γ as the identity matrix, we can use **udgrad** and **dgradl** to obtain the estimates of the parameters online and offline, respectively. The following code can be used for offline estimation:

```
A = 10;
varphi = pi/6;
omega = 0.05*pi;

t_final = 250;
t = 1:t_final;
tt = 0:t_final;

y = A*sin(t*omega+varphi);
phi = [sin(t*omega); cos(t*omega)];

Gamma = 1;
theta =
dgradl(y,phi,zeros(1,length(y)),zeros(2,1),Gamma,Gamma,[],[
]);
yhat = sum(theta(:,2:length(tt)).*phi);
Ahat = sqrt(theta(1,:).^2+theta(2,:).^2);
phihat = acos(theta(1,:)./Ahat);
```

The results are plotted in Figure 4C.5. ■

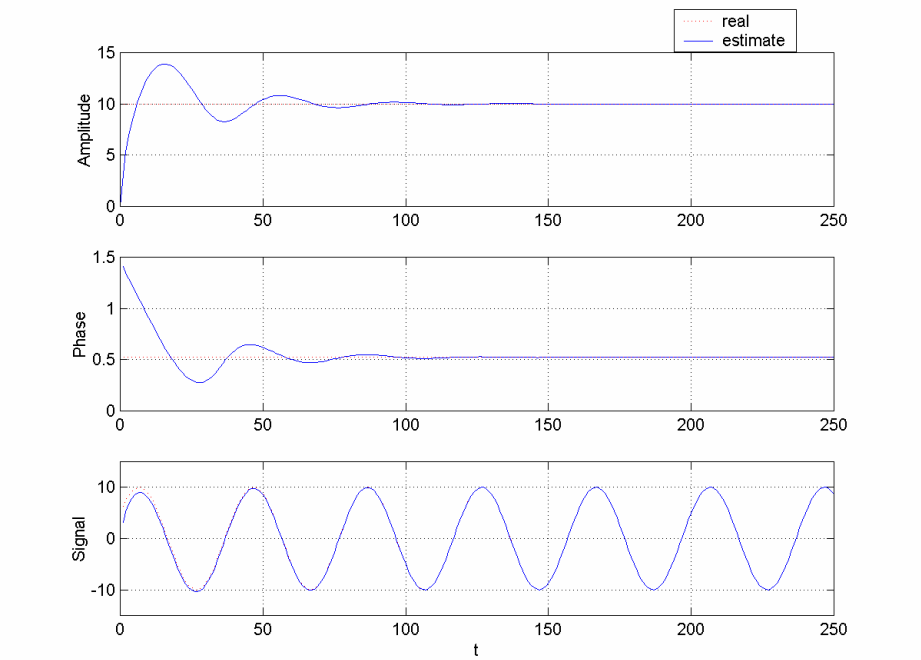


Figure 4C.5 *Time histories of the estimates of the amplitude and the phase of the broadcast signal and the signal itself in Example* 4.13.2.

Example 4.13.3 Consider an unstable system with the I/O relation

$$y(t) = a_1 y(t-1) + a_2 y(t-2) + u(t-1)$$

with $a_1 = -0.5,\ a_2 = 1.5,\ y(0) = 1$, where the coefficients a_1, a_2 are unknown. We want to stabilize the system and to regulate the output y to converge to zero, by choosing the control signal u appropriately. If the coefficients a_1, a_2 of the system were known, we could use

$$u(t) = -a_1 y(t) - a_2 y(t-1)$$

for this task. Since the coefficients are unknown, we need to use their estimates, $\hat{a}_1, \hat{a}_2$:

$$u(t) = -\hat{a}_1(t) y(t) - \hat{a}_2(t) y(t-1)\,.$$

For parameter estimation we can use the parametric model

$$z(t) = \theta^{*T} \phi(t)\,,$$

where

$$\begin{aligned} z(t) &= y(t) - u(t-1), \\ \phi(t) &= \left[y(t-1) \quad y(t-2) \right]^T, \\ \theta^* &= \left[a_1 \quad a_2 \right]^T. \end{aligned}$$

Based on this model, we can obtain the estimate $\theta(t) = [\hat{a}_1(t) \quad \hat{a}_2(t)]^T$ needed for the control law using the gradient algorithm of section 4.5.2. Fixing $\Gamma = I_2$, we can simulate our estimator-controller design using the Simulink scheme drawn in Figure 4C.6 and choosing the parameters of the block **`Parameter Estimator`** appropriately via its user interface. The simulation results are shown in Figure 4C.7. ■

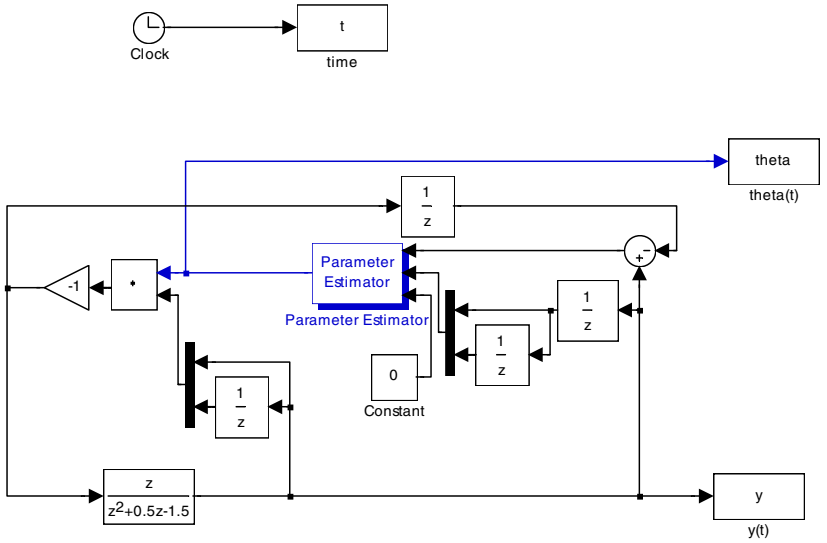


Figure 4C.6 *Simulink scheme for estimating the coefficients of the system model in Example* 4.13.3.

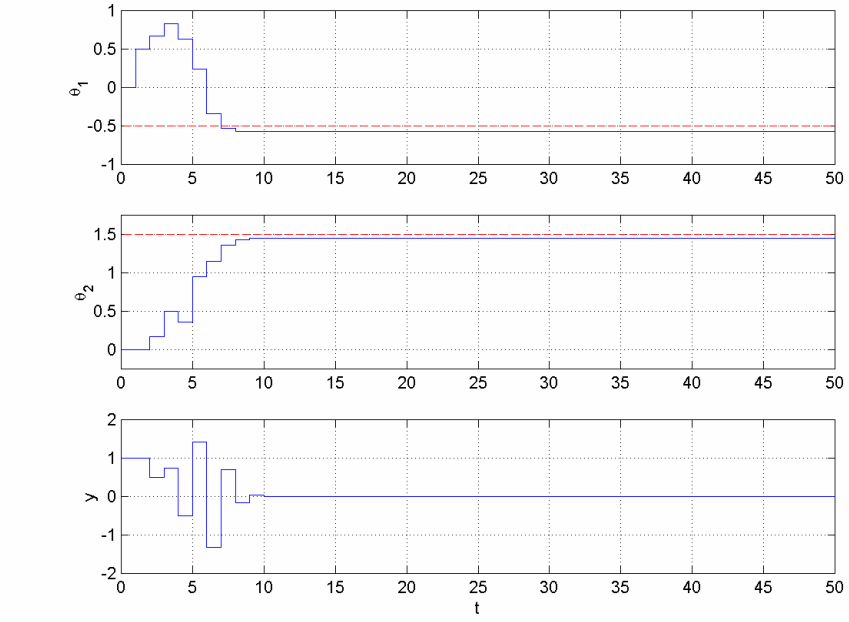


Figure 4C.7 *Time histories of the parameter estimates* θ_i *and the output* y *in Example* 4.13.3.

The projection algorithm (4.26), the modified projection algorithm (4.29), and the orthogonalized projection algorithm (4.30) can be implemented using the MATLAB functions **udproj** and **dprojpure** or the Simulink block **Parameter Estimator** as demonstrated in Examples 4.13.4–4.13.6.

Example 4.13.4 Consider the ARMA model of Example 4.13.1, with the same initial conditions, the same initial parameter estimate values, and the dominantly rich input $u(t) = \sin(\frac{\pi}{25}t) + 0.5\cos(\frac{\pi}{32}t) - 0.25\sin(\frac{\pi}{20}t + \frac{2\pi}{5})$. Assume that all of the five parameters b_0, b_1, b_2, a_1, a_2 are unknown. Let us apply the projection algorithm (4.26), the modified projection algorithm (4.29), and the orthogonalized projection algorithm (4.30) to estimate these parameters, one by one. We will choose $\alpha = 0.1$ and $c = 1$ for the modified projection algorithm. Having the histories of ϕ and z, we can obtain the parameter estimates using the following code lines appropriately, instead of the corresponding **dgradl** and **udgrad** lines in the codes of Example 4.13.1:

(Pure projection algorithm based offline estimation)
```
theta = dprojpure(y,phi,zeros(5,1));
```

(Pure projection algorithm based online estimation)
```
theta(:,k) = udproj('pure',theta(:,k-1),z(k),phi(:,k));
```

(Modified projection algorithm based offline estimation)
```
alpha = 0.5; c = 1;
...
theta = dprojmod(y,phi,alpha,c,zeros(5,1));
```

(Modified projection algorithm based online estimation)
```
alpha = 0.5; c = 1;
...
theta(:,k) = udproj('mod',theta(:,k-1),z(k),phi(:,k),alpha,c);
```

(Orthogonalized projection algorithm based offline estimation)
```
theta = dprojorth(y,phi,zeros(5,1));
```

(Orthogonalized projection algorithm based online estimation)
```
P = eye(5);
x_p = [theta(:,1); P(:)];
...
x_p = udproj('orth',x_p,z(k),phi(:,k));
theta(:,k) = x_p(1:5);
```

The results are plotted in Figures 4C.8–4C.10. As can be seen in Figure 4C.10, orthogonalized projection-based estimation has led to perfect results with fast convergence rates in both the output estimation and the parameter estimation. These results could also be obtained using the Simulink block **Parameter Estimator** with appropriate parameters. ■

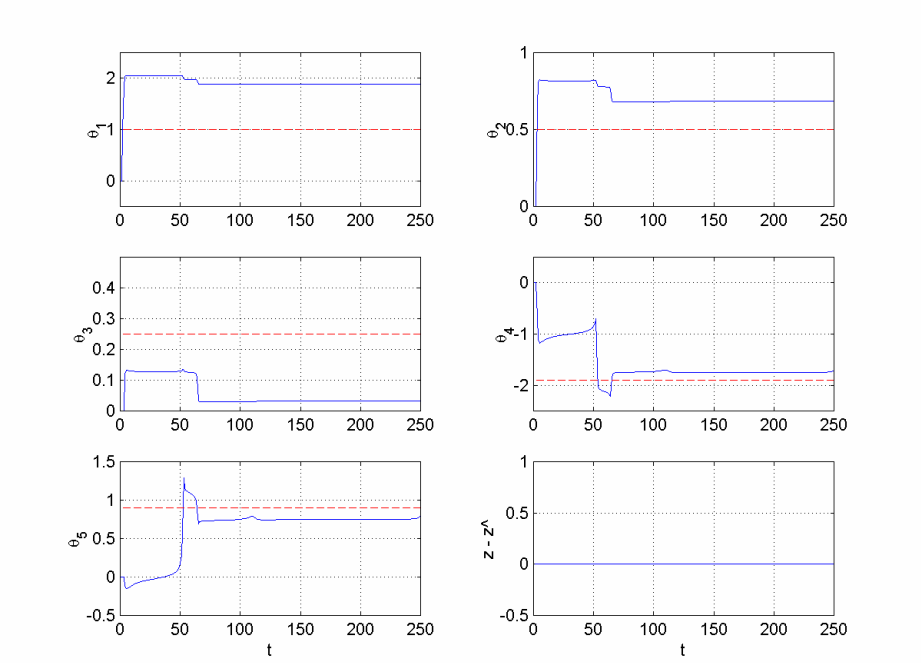


Figure 4C.8 *Time histories of the parameter estimates* θ_i *and the unnormalized estimation error* $z-\hat{z}$ *in Example* 4.13.4 *for pure projection algorithm-based estimation.*

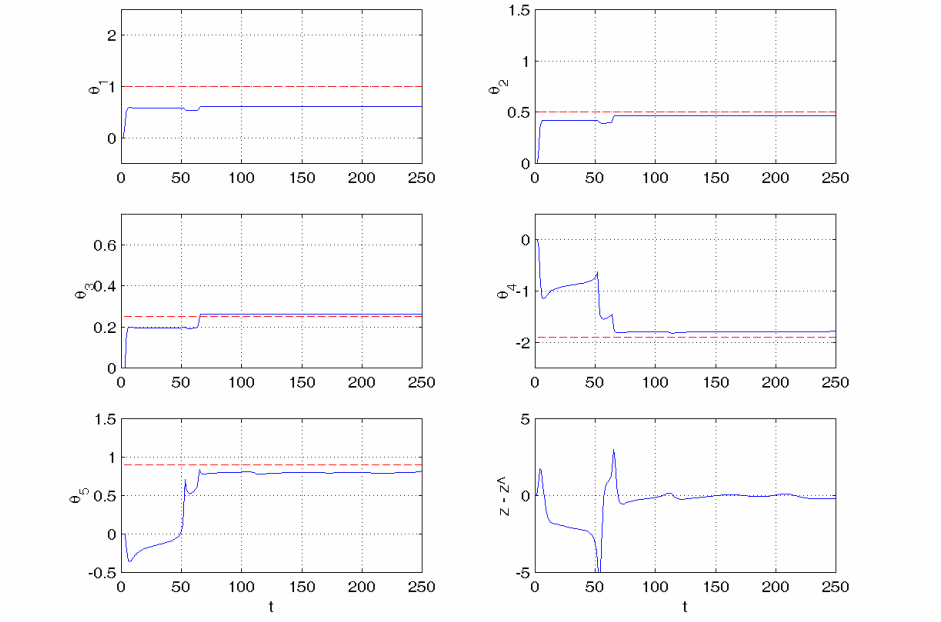


Figure 4C.9 *Time histories of the parameter estimates* θ_i *and the unnormalized estimation error* $z-\hat{z}$ *in Example* 4.13.4 *for modified projection algorithm-based estimation.*

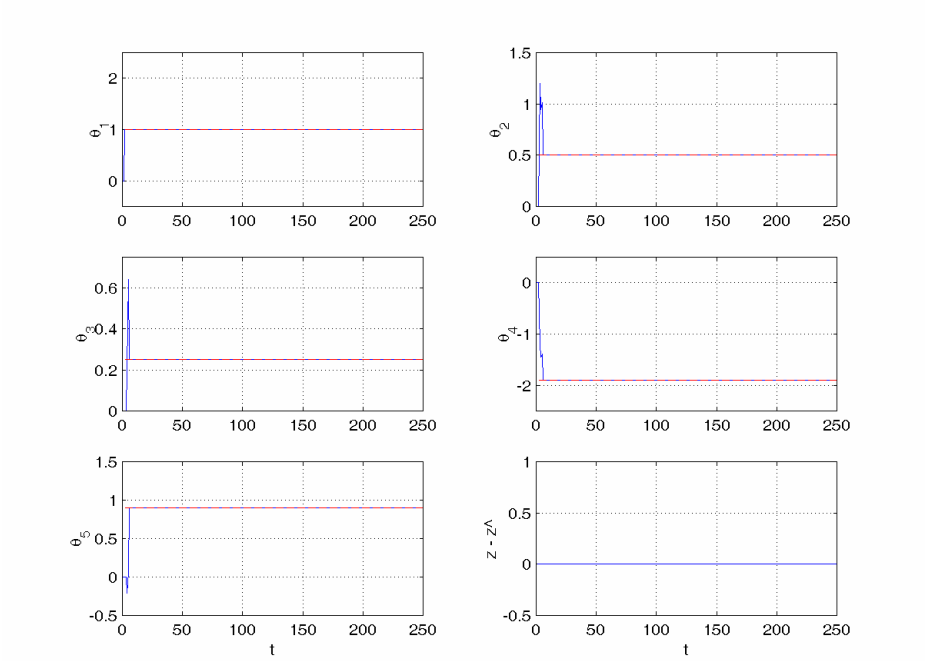


Figure 4C.10 *Time histories of the parameter estimates* θ_i *and the unnormalized estimation error* $z-\hat{z}$ *in Example* 4.13.4 *for orthogonalized projection algorithm-based estimation.*

Example 4.13.5 Consider the unstable system

$$y(t)=a_1y(t-1)+a_2y(t-2)+u(t-1),$$

with unknown coefficients a_1,a_2, of Example 4.13.3. Let us use the same control law,

$$u(t)=-\hat{a}_1(t)y(t)-\hat{a}_2(t)y(t-1),$$

for stabilization and regulation, but obtain the necessary parameter estimates $\hat{a}_1,\hat{a}_2$ using the projection algorithm instead of the gradient algorithm. We can use the parametric model

$$z(t)=\theta^{*T}\phi(t),$$

where

$$\begin{aligned} z(t)&=y(t)-u(t-1),\\ \phi(t)&=\begin{bmatrix}y(t-1) & y(t-2)\end{bmatrix}^T,\\ \theta^*&=\begin{bmatrix}a_1 & a_2\end{bmatrix}^T, \end{aligned}$$

and obtain the estimate $\theta(t)=[\hat{a}_1(t)\quad \hat{a}_2(t)]^T$ using (4.25). Fixing $\Gamma=I_2$, we can simulate our estimator-controller design using the Simulink scheme drawn previously in Figure 4C.6 with appropriate modifications in the parameters of the block **`Parameter Estimator`**. The simulation results are shown in Figure 4C.11. ■

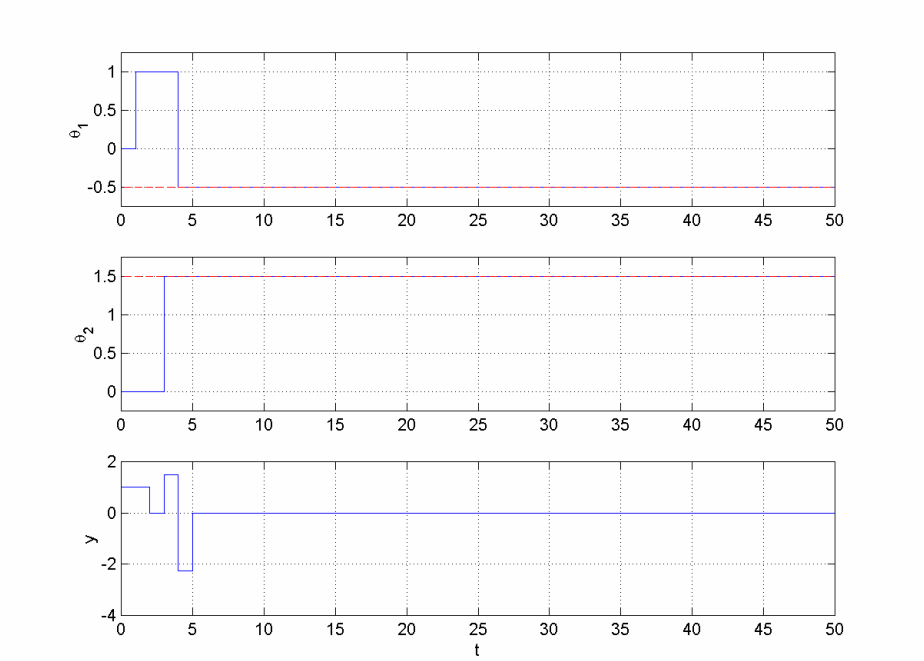


Figure 4C.11 *Time histories of the parameter estimates* θ_i *and the output* y *in Example* 4.13.5.

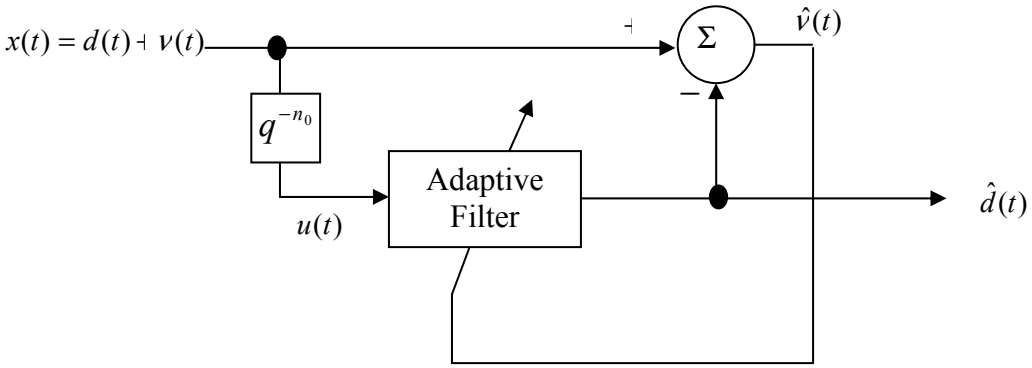


Figure 4C.12 *Adaptive noise cancellation without a reference noise signal.*

Example 4.13.6[1] The diagram in Figure 4C.12 shows a widely used scheme for adaptive noise cancellation without a reference noise signal [3,4]. The aim here is to estimate the process signal $d(t)$ from a noise corrupted observation $x(t) = d(t) + v(t)$, where $v(t)$ is the interfering white noise uncorrelated to the signal $d(t)$. Use of the modified projection (NLMS) algorithm for this task is very popular in stochastic signal processing, since it has good mean-square convergence properties [4,5].

We can select the adaptive filter in Figure 4C.12 as the MA model

$$d(t) = \sum_{n=0}^{N} a_n(t)u(t-n) = \sum_{n=0}^{N} a_n(t)x(t-n-n_0)\,,$$

[1] The reader is referred to [3,4] for the theory behind the construction of the scheme and the selection of the adaptive law in this example.

where N is the order of the MA model. We can rewrite the model equation in the linear parametric form as

$$\begin{aligned} d(t) &= \theta^{*T}(t)\phi(t), \\ \phi(t) &= \left[x(t-n_0) \quad \cdots \quad x(t-N-n_0)\right]^T, \\ \theta^*(t) &= \left[a_0(t) \quad \cdots \quad a_N(t)\right]^T \end{aligned}$$

and obtain the estimate $\hat{d}(t)$ of the process signal using

$$\hat{d}(t) = \theta(t)\phi(t),$$

where $\theta(t)$ is the estimate of the parameter vector $\theta^*(t)$. Finally, to produce the parameter estimate $\theta(t)$ we can use the NLMS algorithm in the form

$$\theta(t) = \theta(t-1) + \frac{\alpha\phi(t)}{c + \phi^T(t)\phi(t)}\left(x(t) - \theta^T(t-1)\phi(t)\right),$$

which emulates $\theta(t) = \theta(t-1) + \frac{\alpha\phi(t)}{c+\phi^T(t)\phi(t)}\hat{\nu}(t)$.

Now, assume that the process signal is given as

$$d(t) = \sin(t\omega_0 + \varphi),$$

with $\omega_0 = 0.04\pi$ and $\varphi = \frac{\pi}{6}$, and the noise dynamics is given by

$$\nu(t) = 0.7\nu(t-1) + g(t),$$

where $g(t)$ is a zero-mean white noise with variance $\sigma_\nu^2 = 0.1$. Let us apply the estimation scheme above with $n_0 = 30, N = 14, \alpha = 0.25,$ and $c = 0.5$ to get $\hat{d}(t)$. We can use the following code for this task:

```
omega0=0.04*pi;
varphi = pi/6;

t_final = 1000;
t = 0:t_final;
tt = -44:t_final;
g_tt = sqrt(0.1)*randn(1,t_final+45);
d_tt(1:45) = sin([-44:0]*omega0 + varphi);
dhat_tt(1:45) = zeros(1,45);
nu_tt(1) = g_tt(1);
x_tt(1) =d_tt(1)+nu_tt(1);
for k = 2:45,
   nu_tt(k)=0.7*nu_tt(k-1)+g_tt(k);
   x_tt(k) =d_tt(k)+nu_tt(k);
end

theta_tt(:,1:44) = zeros(15,44);
alpha = 0.25;
```

```
c = 0.5;
phi = x_tt(15:-1:1)';
theta_tt(:,45) =
udproj('mod',theta_tt(:,44),x_tt(44),phi,alpha,c);

for k = 46:t_final+45,
   d_tt(k) = sin(k*omega0 + varphi);
   nu_tt(k)=0.7*nu_tt(k-1)+g_tt(k);
   x_tt(k) =d_tt(k)+nu_tt(k);
   phi = x_tt(k-30:-1:k-44)';
   theta_tt(:,k) = udproj('mod',theta_tt(:,k-1),x_tt(k),phi,alpha,c);
   dhat_tt(k) = theta_tt(:,k)'*phi;
end;

theta = theta_tt(:,45:t_final+45);
d = d_tt(45:t_final+45);
dhat = dhat_tt(45:t_final+45);
x = x_tt(45:t_final+45);
```

The results are plotted in Figure 4C.13. ■

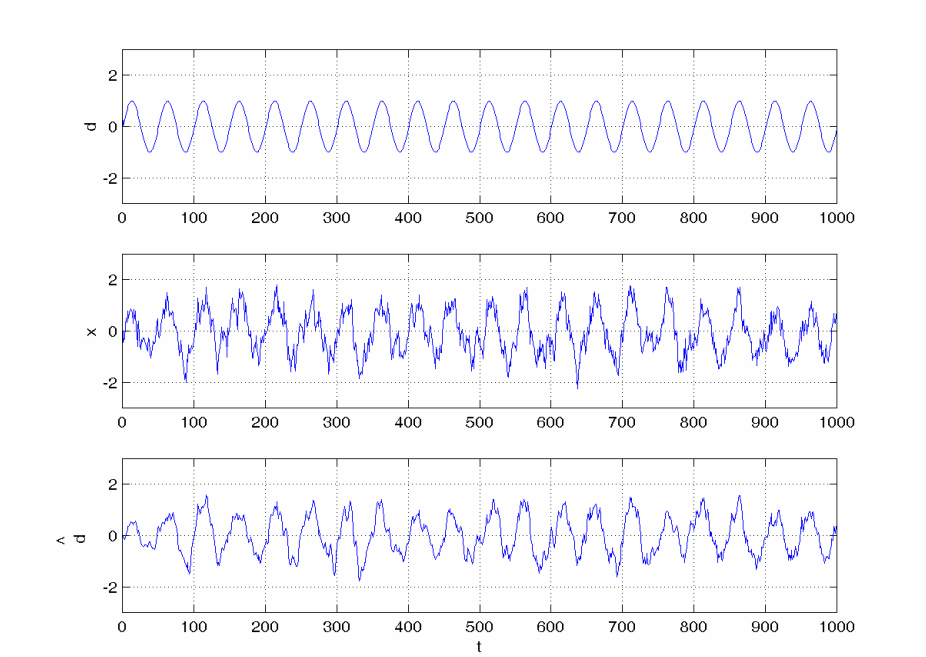


Figure 4C.13 *The process signal* $d(t)$*, the corrupted signal* $x(t)$*, and the process signal estimate* $\hat{d}(t)$ *in Example* 4.13.6.

4.13.2 LS Algorithms

The MATLAB functions **`drls`**, **`udrls`**, and **`urlsarg`** can be used to simulate any of the LS algorithms presented in sections 4.6 and 4.7 as shown in Examples 4.13.7–4.13.9.

Example 4.13.7 Consider the ARMA model of Example 4.13.1, with the same initial conditions, the same initial parameter estimate values, and the dominantly rich input

$u(t) = \sin(\frac{\pi}{25}t) + 0.5\cos(\frac{\pi}{32}t) - 0.25\sin(\frac{\pi}{20}t + \frac{2\pi}{5})$. Assume that all of the five parameters b_0, b_1, b_2, a_1, a_2 are unknown. Let us apply the pure LS algorithm, the LS algorithm with modified weighting, and the LS algorithm with start-up forgetting to estimate these parameters one by one. We will choose $\delta = 0.1$ and $k = 1$ for the modified weighting, and $\beta(0) = 0.7$ and $\beta_1 = 0.99$ for start-up forgetting. Having the histories of ϕ and z, we can obtain the parameter estimates offline for the pure LS algorithm-based estimation, using the following code:

```
ArgLS = urlsarg('pure');
theta =
drls(y,phi,zeros(1,length(y)),zeros(5,1),eye(5),ArgLS);
zhat = sum(theta(:,2:length(tt)).*phi);
```

The same task can be performed online using the following code:

```
ArgLS = urlsarg('pure');
theta(:,1) = zeros(5,1);
P = eye(5);
x_p = [theta(:,1); P(:)];

for k = 2:t_final,
   y(k) = ufilt('output', x, u(k-1), b, [1 a],0);
   x = ufilt('state', x, u(k-1), b, [1 a]);
   [z(k), phi(:,k)] = uarma2lm('output',x_lm, [u(k) y(k)],
[n d m],K);
   x_lm = uarma2lm('state',x_lm, [u(k) y(k)], [n d m],K);
   x_p = udrls(x_p,z(k),phi(:,k),0,ArgLS);
   theta(:,k) = x_p(1:5);
end;
```

We need to change the corresponding lines above to

```
ArgLS = urlsarg('fuchs',0.1,1);
```

for modified (Fuchs) weighting, and to

```
ArgLS = urlsarg('startup forgetting',0.7,0.99);
...
x_p = [theta(:,1); P(:);0.7];
```

for start-up forgetting.

The results are plotted in Figures 4C.14–4C.16. As can be seen in these figures, start-up forgetting in particular improves the parameter convergence and output tracking properties of the LS algorithm. These results could also be obtained using the Simulink block **`Parameter Estimator`** with appropriate parameters. ■

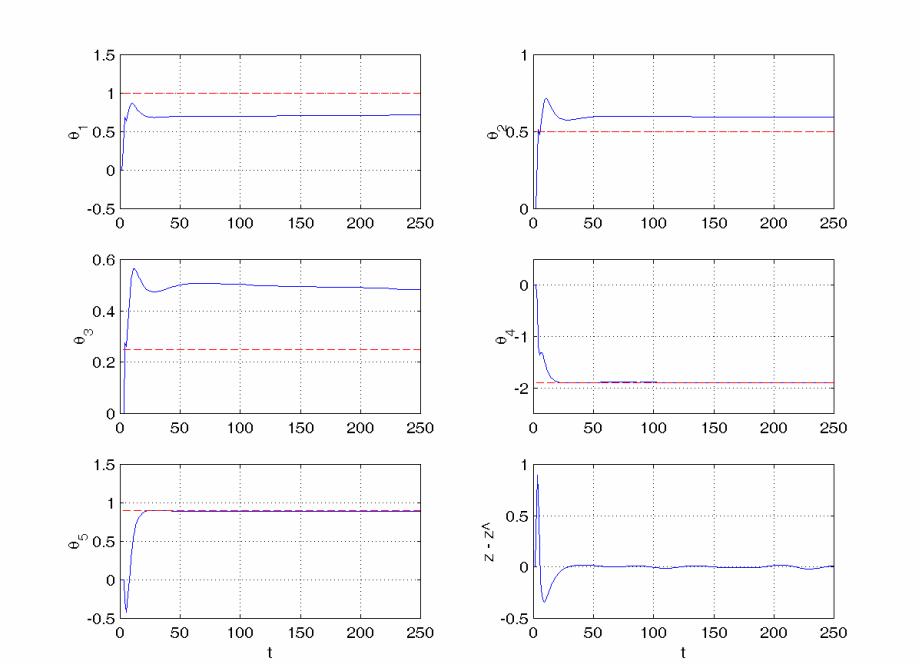


Figure 4C.14 *Time histories of the parameter estimates* θ_i *and the unnormalized estimation error* $z-\hat{z}$ *in Example* 4.13.7 *for the pure LS algorithm.*

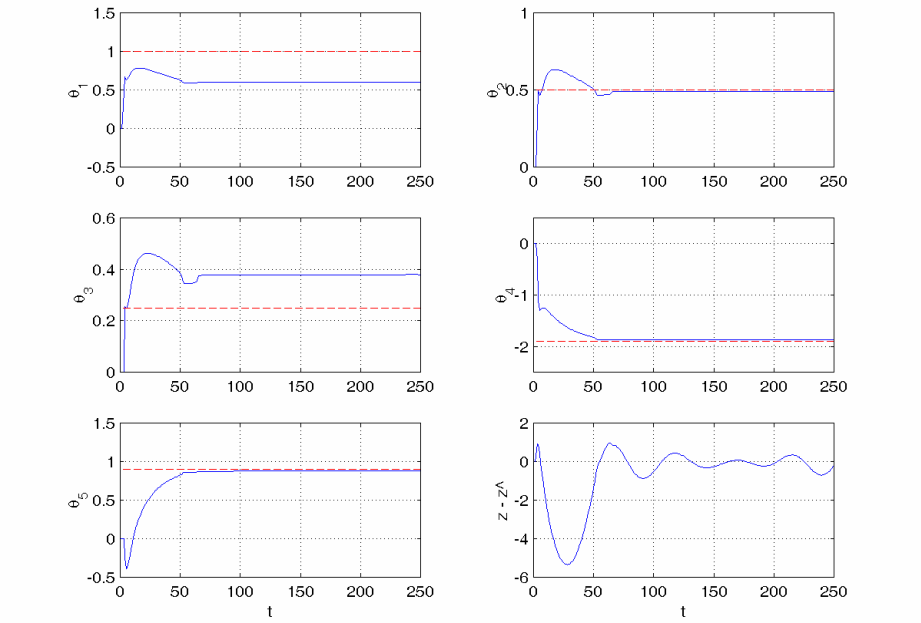


Figure 4C.15 *Time histories of the parameter estimates* θ_i *and the unnormalized estimation error* $z-\hat{z}$ *in Example* 4.13.7 *for the LS algorithm with modified weighting.*

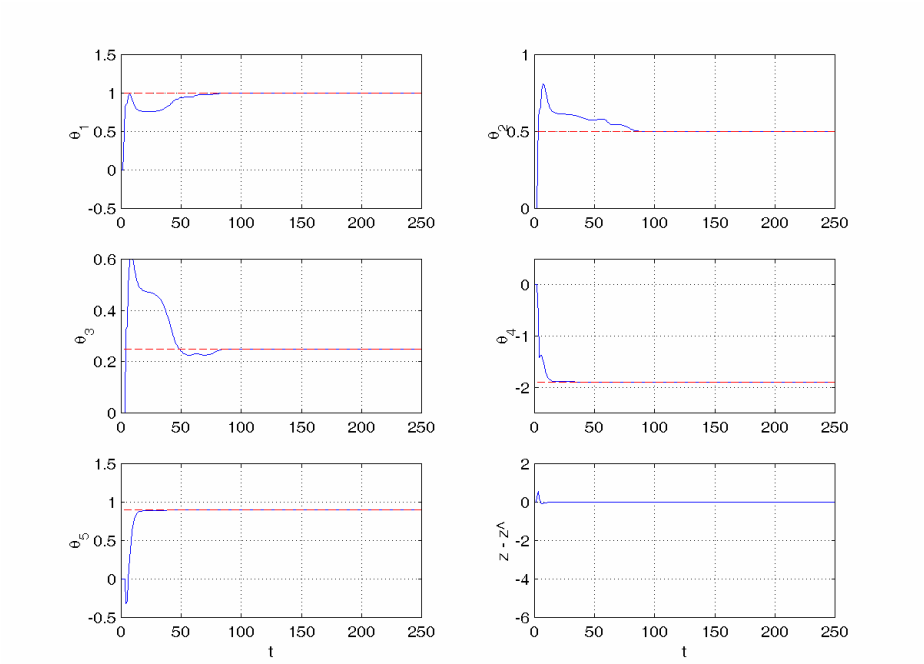


Figure 4C.16 *Time histories of the parameter estimates* θ_i *and the unnormalized estimation error* $z-\hat{z}$ *in Example* 4.13.7 *for the LS algorithm with start-up forgetting.*

Example 4.13.8 Consider the unstable system

$$y(t)=a_1y(t-1)+a_2y(t-2)+u(t-1)$$

with unknown coefficients a_1,a_2, of Examples 4.13.3 and 4.13.5. Let us use the control law

$$u(t)=-\hat{a}_1(t)y(t)-\hat{a}_2(t)y(t-1)$$

for stabilization and regulation as before, using the LS algorithm to obtain the necessary parameter estimates $\hat{a}_1,\hat{a}_2$ this time. In the estimation of $\theta(t)=[\hat{a}_1(t)\quad \hat{a}_2(t)]^T$, we will use the same parametric model, i.e.,

$$z(t)=\theta^{*T}\phi(t),$$

where

$$\begin{aligned} z(t)&=y(t)-u(t-1),\\ \phi(t)&=\big[y(t-1)\quad y(t-2)\big]^T,\\ \theta^*&=\big[a_1\quad a_2\big]^T. \end{aligned}$$

We will apply constant forgetting, constant weighting, and least-size resetting modifications, with the modification variables $a=\beta=0.5, p_0=1, p_{\min}=0.1$. Simulating our estimator-controller design using the Simulink scheme drawn previously

in Figure 4C.6 with appropriate modifications in the parameters of the block **`Parameter Estimator`**, the results shown in Figure 4C.17 are obtained. ■

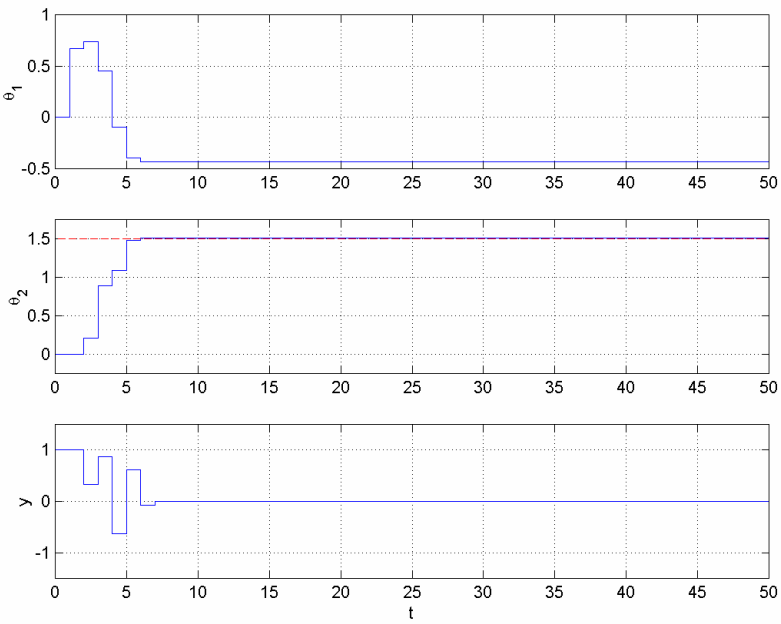


Figure 4C.17 *Time histories of the parameter estimates* θ_i *and the output* y *in Example* 4.13.8.

Example 4.13.9 Consider the adaptive noise cancellation system of Example 4.13.6 with the same system parameters. Let us repeat the estimation process done in Example 4.13.6 using the LS algorithm instead of the NLMS algorithm, keeping the estimation parameters the same. We will apply the LS algorithm with constant weighting ($a = 0.3$) and start-up forgetting ($\beta_0 = 0.7, \beta_1 = 0.99$). For simulation we can use the code of Example 4.13.6 after adding the lines

```
ArgLS = [urlsarg('pure',0.3) urlsarg('startup',0.7,0.99)];
theta0 = zeros(15,1);
P0 = eye(15);
beta0 = 0.7;
xp = [theta0(:); P0(:); beta0];
```

and changing the corresponding lines to

```
xp = udrls(xp,x_tt(44),phi,0,ArgLS);
theta_tt(:,45) = xp(1:15);
...
xp = udrls(xp,x_tt(k),phi,0,ArgLS);
theta_tt(:,k) = xp(1:15);
```

The results are shown in Figure 4C.18. ■

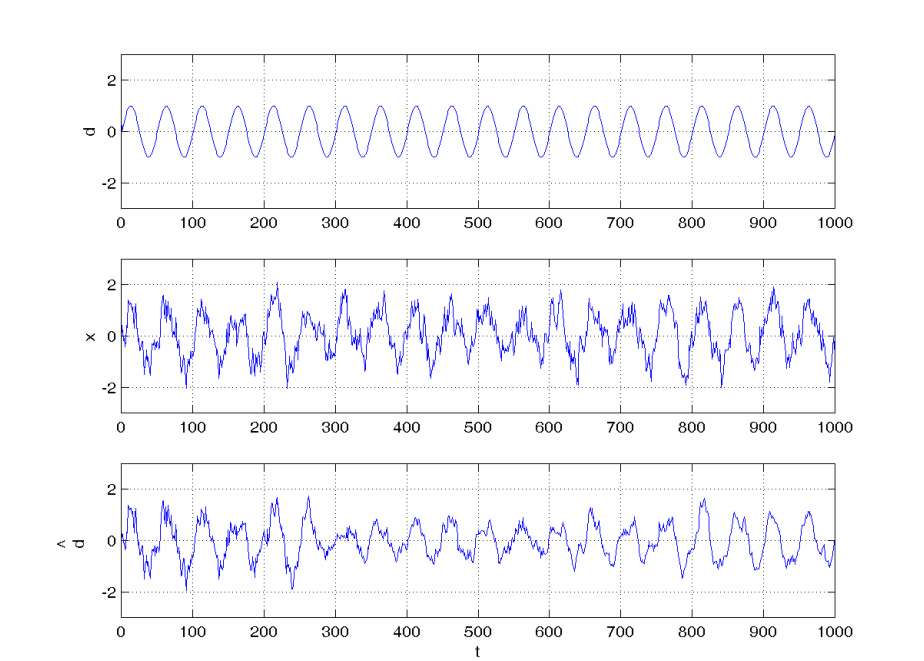


Figure 4C.18 *The process signal* $d(t)$*, the corrupted signal* $x(t)$*, and the process signal estimate* $\hat{d}(t)$ *in Example* 4.13.9.

4.13.3 Parameter Projection

The MATLAB function **`uparproj`** can be incorporated into the parameter estimation functions introduced in sections 4.13.1 and 4.13.2 to implement the PI algorithms with parameter projection as shown in Example 4.13.10.

Example 4.13.10[2] Consider the unstable system of Example 4.13.3 with the same parameter values, but in the existence of a special type of disturbance ν; i.e., let

$$y(t) = a_1 y(t-1) + a_2 y(t-2) + u(t-1) + \nu(t),$$
$$\nu(t) = \frac{t^{3/2} - (t-1)^{3/2} - 1}{(t-1)^{1/2}}$$

with $a_1 = -0.5,\ a_2 = 1.5,\ y(0) = 1$, where the coefficients a_1, a_2 are unknown. We want to stabilize the system and to keep the output y as close as possible to zero, by choosing the control signal u appropriately. We will use the same control law as in Example 4.13.3, i.e.,

$$u(t) = -\hat{a}_1(t) y(t) - \hat{a}_2(t) y(t-1),$$

where $\hat{a}_1, \hat{a}_2$ are estimates of a_1, a_2, respectively. If we apply the pure projection algorithm, without any modification, to generate the parameter estimates $\hat{a}_1, \hat{a}_2$, i.e., if we use

[2] It may be beneficial to see the similar examples in [6,7].

$$
\begin{aligned}
&z(t) = y(t) - u(t-1),\\
&\phi(t) = \begin{bmatrix} y(t-1) & y(t-2) \end{bmatrix}^T,\\
&\theta(t) = \theta(t-1) + \frac{\phi(t)}{\phi^T(t)\phi(t)}\left(z(t) - \theta^T(t-1)\phi(t)\right),\\
&\hat{a}_1(t) = \theta_1(t),\\
&\hat{a}_2(t) = \theta_2(t)
\end{aligned}
$$

with the initial estimate vector $\theta(0) = [0 \quad 0]^T$, we get the results plotted in Figure 4C.19, which are poor in terms of both parameter estimation and output regulation. Now assume that we know $|a_1| \le 1$ and $|a_2| \le 2$. We can use this information applying the parameter projection law given in Example 4.10.3 in our design, i.e., using

$$
\hat{a}_1(t) = \begin{cases} \theta_1(t) & \text{if} \quad |\theta_1(t)| \le 1, \\ -1 & \text{if} \quad \theta_1(t) < -1, \\ 1 & \text{if} \quad \theta_1(t) > 1, \end{cases}
$$

$$
\hat{a}_2(t) = \begin{cases} \theta_2(t) & \text{if} \quad |\theta_2(t)| \le 2, \\ -2 & \text{if} \quad \theta_2(t) < -2, \\ 2 & \text{if} \quad \theta_2(t) > 2. \end{cases}
$$

We can simulate our design using the previous Simulink scheme with appropriate modifications in the parameters of the block **`Parameter Estimator`**. The simulation results are shown in Figure 4C.20. As can be seen in these plots, both the parameter estimation and the output regulation are improved. ■

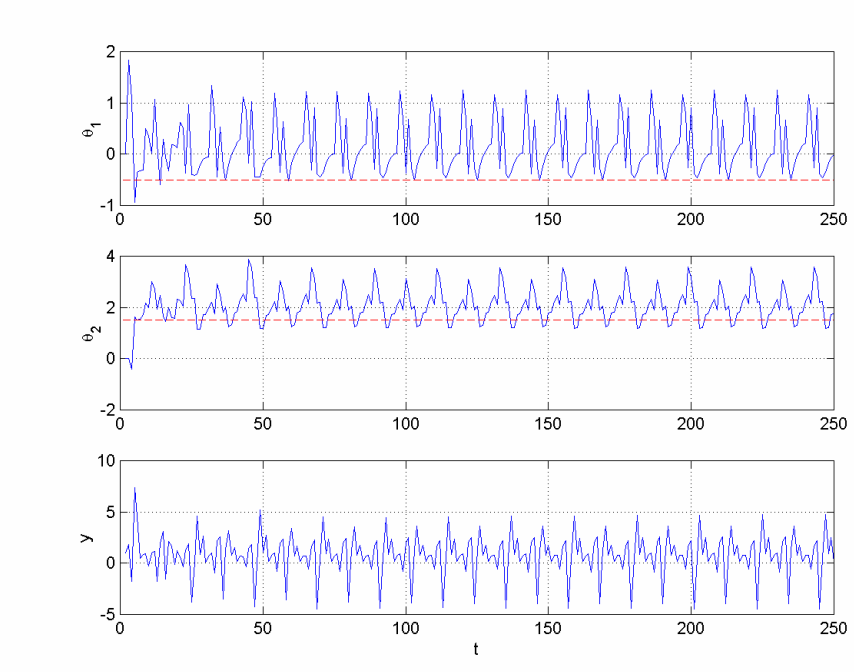


Figure 4C.19 *Time histories of the parameter estimates* θ_i *and the output* y *in Example* 4.13.10 *without parameter projection.*

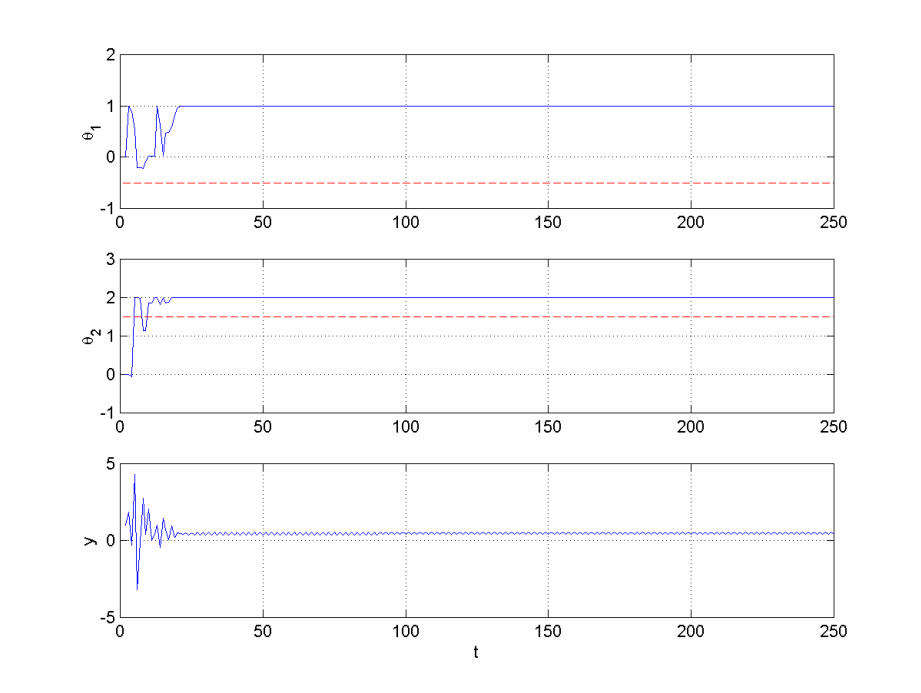


Figure 4C.20 *Time histories of the parameter estimates* θ_i *and the output* y *in Example* 4.13.10 *with parameter projection.*

4.13.4 Robust Parameter Identification

The MATLAB function **`urobust`** can be incorporated into the parameter estimation functions introduced in sections 4.13.1–4.13.3 to simulate the robust PI algorithms as demonstrated in Example 4.13.11.

Example 4.13.11[3] Consider the unstable system of Example 4.13.10 with the same parameters and the same disturbance. We want to stabilize the system and to keep the output y as close as possible to zero, using the same control law as before, i.e.,

$$u(t) = -\hat{a}_1(t)y(t) - \hat{a}_2(t)y(t-1),$$

where $\hat{a}_1, \hat{a}_2$ are estimates of a_1, a_2, respectively. Let us first use the parameter estimator of Example 4.13.3, i.e., the gradient algorithm based on instantaneous cost, without modification to see the effect of the disturbance. One can perform the simulations using the Simulink scheme of Figure 4C.6 without changing anything except the addition of the disturbance. The result pictured in Figure 4C.21 shows that the disturbance affects both the parameter estimation and the output regulation a lot. Now, to reduce the effect of the disturbance, let us modify our parameter estimator using the fixed σ-modification method described in section 4.11 as well as the ε-modification and the dead zone modification in the literature. Application of the fixed σ-modification results in the adaptive law

[3] It may be beneficial to see the similar examples in [6,7].

$$\theta(t) = \theta(t-1) + \Gamma\left[\phi(t)\frac{z(t) - \theta^T(t-1)\phi(t)}{1 + \phi^T(t)\Gamma\phi(t)} - s_0\theta(t-1)\right].$$

Selecting $s_0 = 0.25$, we can simulate our estimator-controller design using the same Simulink scheme with appropriate modifications in the parameters of the block **`Parameter Estimator`**. ε and dead zone modifications can be implemented similarly. The simulation results are shown in Figures 4C.22–4C.24. ■

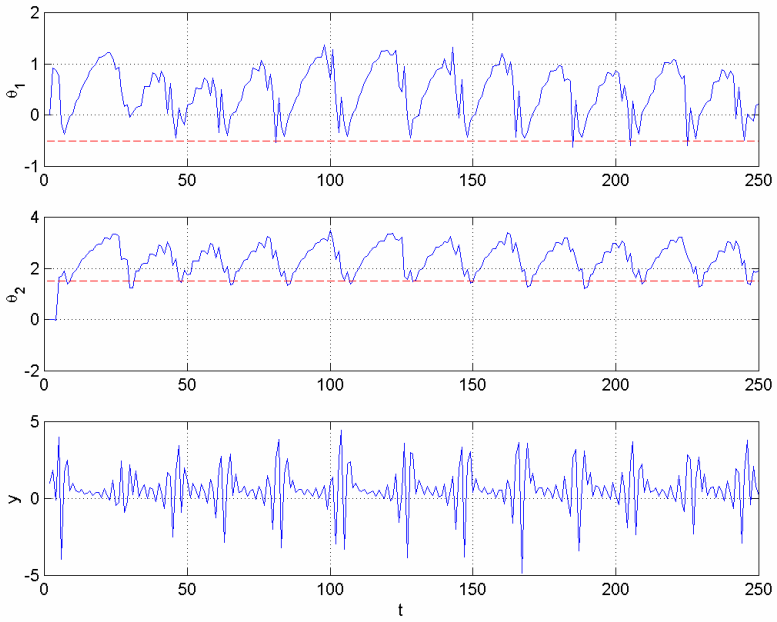


Figure 4C.21 *Time histories of the parameter estimates* θ_i *and the output* y *in Example* 4.13.11 *without any robustness modification*.

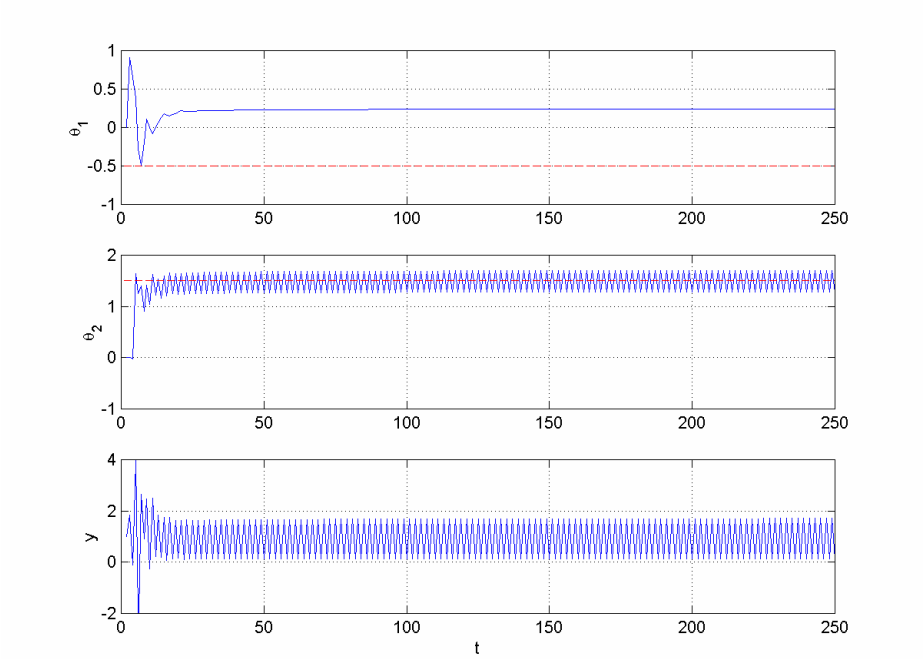


Figure 4C.22 *Time histories of the parameter estimates* θ_i *and the output* y *in Example* 4.13.11 *with fixed* σ *-modification.*

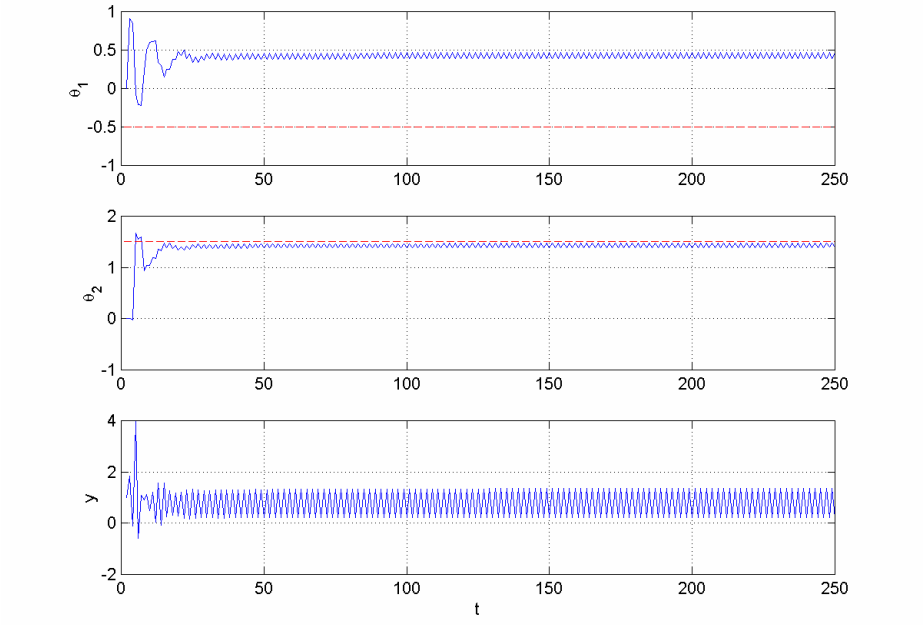


Figure 4C.23 *Time histories of the parameter estimates* θ_i *and the output* y *in Example* 4.13.11 *with* ε *-modification.*

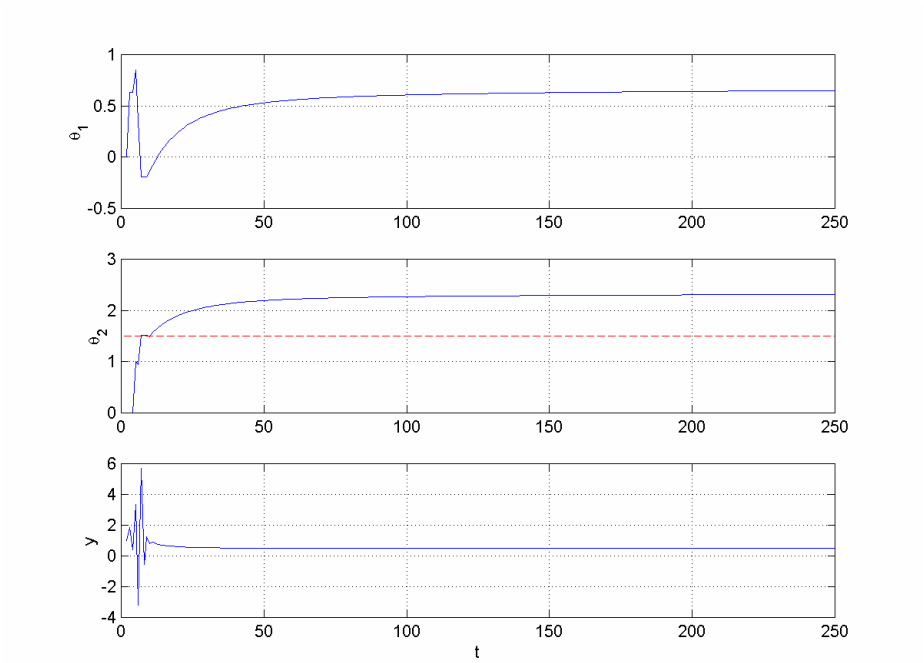


Figure 4C.24 *Time histories of the parameter estimates* θ_i *and the output* y *in Example* 4.13.11 *with dead zone modification.*

Bibliography

[1] C.W. BURILL, *Measure, Integration and Probability,* Mc Graw-Hill, New York, 1973.

[2] G. C. GOODWIN AND K. S. SIN, *Adaptive Filtering Prediction and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

[3] M. H. HAYES, *Statistical Digital Signal Processing and Modeling*, John Wiley, New York, 1996.

[4] S. HAYKIN, *Adaptive Filter Theory,* Prentice-Hall, Upper Saddle River, NJ, 2001.

[5] A. WEISS AND D. MITRA, *Digital adaptive filters: Conditions for convergence, rates of convergence, effects of noise and errors arising from the implementation*, IEEE Trans. Info. Theory, 25 (1979) pp. 637–652.

[6] K. J. ASTROM AND B. WITTENMARK, *Adaptive Control,* Addison-Wesley, Reading, MA, 1995.

[7] B. EGARDT, *Stability of Adaptive Controllers,* Lecture Notes in Control and Info. Sci. 20, Springer-Verlag, New York, 1979.