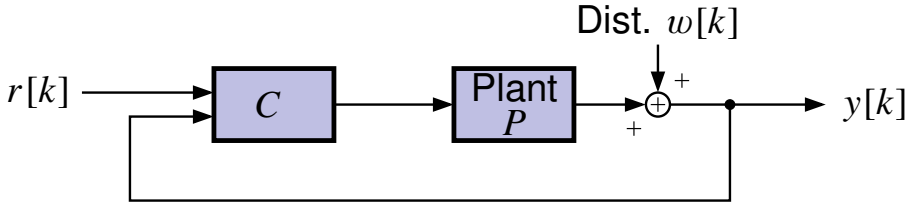
ADAPTIVE INVERSE CONTROL

7.1: The adaptive inverse control concept

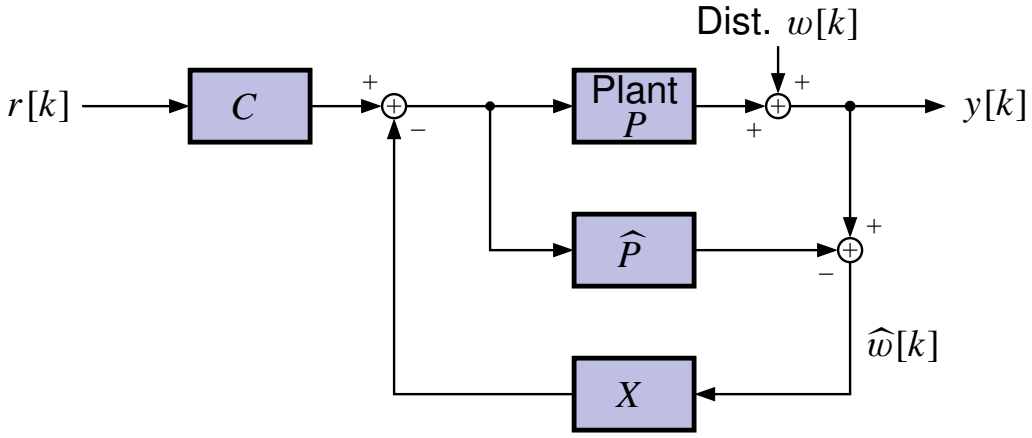
- Some problems with conventional control methods:
  - The plant's dynamic equations are assumed to be known;
  - Disturbance statistics are also assumed to be known (and often assumed Gaussian and white)
  - Performance is sacrificed when the plant model is uncertain;
  - Design is very complicated for large, MIMO systems;
  - Design is *very* complicated for large, nonlinear systems.
- An adaptive design has several benefits:
  - Control of very complicated dynamic systems becomes simple;
  - Better plant modeling gives greater precision of control;
  - The controller is robust to variation in internal plant parameters;
  - Excellent disturbance canceling.
- There are a number of different approaches to adaptive control.
- We briefly consider a method called Adaptive Inverse Control (AIC).
- There are a number of different formulations of AIC, and we will look at one proposed by Widrow and Walach.[1]

[1] Widrow, B., and Walach, E., Adaptive Inverse Control, Reissue Ed., Wiley-IEEE, 2007.

- This is *not* the most common approach, but:
  - It is quite easy to understand, especially if you know some DSP;
  - It can work very well if certain conditions are met;
  - The system ID approach itself is worth studying, and the rest of the method follows directly from that.
- There are some open issues with this formulation; notably:
  - It works only for stable or stabilized plants—if a plant is unstable, it must first be stabilized using conventional means;
  - There is no direct analog to integral control: Adaptation slowly eliminates steady-state error, but if there is plant-model mismatch, the error can take considerable time to decay.
- A conventional control system looks like:



- An adaptive inverse control system looks like:



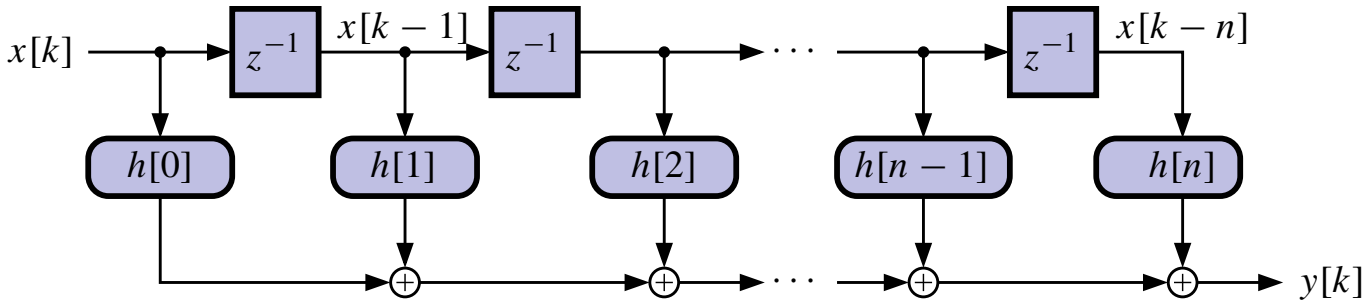
- Adapt $\widehat{P}$ to model the plant P (system identification).
- Adapt C to control the plant P (feedforward control).
- Adapt X to perform disturbance canceling.

7.2: Adaptive linear filters

- We first consider some fixed and adaptive digital filter theory.
- Adaptive filters are the building blocks used for adaptive inverse control, and it is important to understand them thoroughly.
  - First, the structure of an adaptive filter is considered;
  - Second, the optimal solution is derived;
  - Third, an adaptive algorithm is discussed.

Digital filter structure

- The structure of a finite-impulse-response (FIR) digital filter is:



- It comprises a tapped-delay-line connected to the filter input.
- The output is computed to be a weighted sum of the delayed inputs.

$$y[k] = \sum_{i=0}^{n} h[i]x[k-i] \qquad \text{and} \qquad Y(z) = \underbrace{\left[\sum_{i=0}^{n} h[i]z^{-i}\right]}_{H(z)} X(z).$$

- The response to an impulse can be nonzero for at most $n+1$ time steps (which is a finite number, and gives rise to the term “FIR”).
- For a deterministic input, it is possible to compute $y[k]$ using either the difference equation or the transfer function.

- For a random input, it is only possible to pre-compute some statistical properties of the output given some statistical properties of the input.

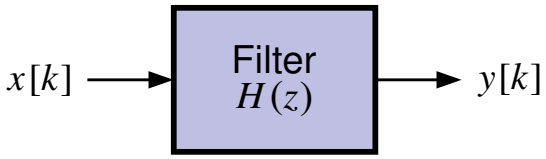
AUTOCORRELATION: We define the autocorrelation function of the signal $x[k]$ to be

$$\phi_{xx}[m] = \mathbb{E}\Big[x[k]x[k+m]\Big].$$

CROSSCORRELATION: We define the crosscorrelation between two signals $x[k]$ and $y[k]$ to be

$$\phi_{xy}[m] = \mathbb{E}\Big[x[k]y[k+m]\Big] = \phi_{yx}[-m].$$

EXAMPLE: Correlations involving one filter



- It is going to be important to be able to calculate the cross-correlation between a filter's input $x[k]$ and output $y[k]$:

$$\begin{aligned}
\phi_{xy}[m] &= \mathbb{E}\Big[x[k]y[k+m]\Big] \\
&= \mathbb{E}\left[x[k]\left(\sum_{i=-\infty}^{\infty} h[i]x[k+m-i]\right)\right] \\
&= \sum_{i=-\infty}^{\infty} \underbrace{\mathbb{E}\Big[x[k]x[k+m-i]\Big]}_{\phi_{xx}[m-i]} h[i] \\
&= \phi_{xx}[m] * h[m].
\end{aligned}$$

- Taking the z-transform, $\Phi_{xy}(z) = \Phi_{xx}(z)H(z)$.
- We continue by finding the autocorrelation function of the filter output

$$\begin{aligned}
\phi_{yy}[m] &= \mathbb{E}\Big[y[k]y[k+m]\Big] \\
&= \mathbb{E}\left[\left(\sum_{i=-\infty}^{\infty} h[i]x[k-i]\right)\left(\sum_{j=-\infty}^{\infty} h[j]x[k+m-j]\right)\right]
\end{aligned}$$

$$= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} h[i]h[j]\mathbb{E}\Big[x[k-i]x[k+m-j]\Big]$$

$$= \sum_{i=-\infty}^{\infty} h[i] \underbrace{\sum_{j=-\infty}^{\infty} h[j]\phi_{xx}[m+i-j]}_{\phi_{xy}[m+i]}$$

$$= \sum_{i=-\infty}^{\infty} h[i]\phi_{xy}[m+i].$$

- This looks a lot like convolution, but the sign on i is wrong in ϕ_{xy}.
- Define $\tilde{h}[i] = h[-i]$

$$\phi_{yy}[m] = \sum_{i=-\infty}^{\infty} \tilde{h}[-i]\phi_{xy}[m+i]$$

$$= \sum_{i'=+\infty}^{-\infty} \tilde{h}[i']\phi_{xy}[m-i'].$$

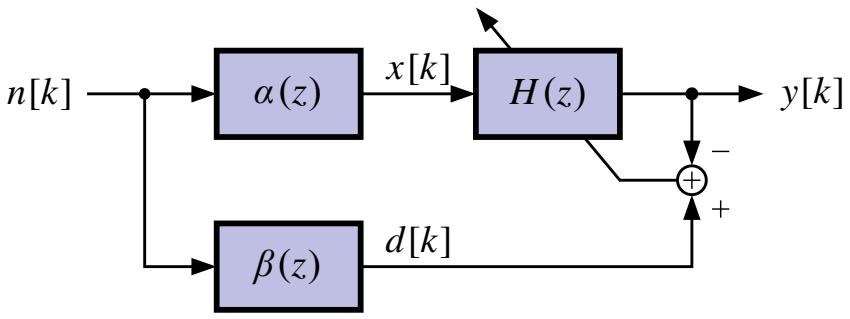
- This is a convolution. So,

$$\phi_{yy}[m] = h[-m] * \phi_{xy}[m] = h[-m] * \phi_{xx}[m] * h[m]$$

$$\Phi_{yy}(z) = H(z^{-1})\Phi_{xx}(z)H(z).$$

EXAMPLE: Correlations involving two filters

- Many times, we must compute $\Phi_{xd}(z)$ and $\Phi_{xx}(z)$ for block diagrams like the one to the right.



- To find $\Phi_{xd}(z)$ and $\Phi_{xx}(z)$ we first compute $\phi_{xd}[m]$.

$$\phi_{xd}[m] = \mathbb{E}\Big[x[k]d[k+m]\Big]$$

$$x_k = \sum_{l=-\infty}^{\infty} \alpha[l]n[k-l]$$

$$d_k = \sum_{i=-\infty}^{\infty} \beta[i]n[k-i]$$

$$\phi_{xd}[m] = \mathbb{E}\left[\sum_{l=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} \alpha[l]n[k-l]n[k+m-i]\beta[i]\right]$$

$$= \sum_{l=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} \alpha[l]\phi_{nn}[m+l-i]\beta[i].$$

- This may be unwrapped by taking the z-transform.

$$\Phi_{xd}(z) = \sum_{l=-\infty}^{\infty} \alpha[l]z^{+l} \sum_{m=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} \phi_{nn}[m+l-i]z^{-(m+l-i)}\beta[i]z^{-i}$$

$$= \sum_{l=-\infty}^{\infty} [\alpha[l]z^{+l}] \left[\sum_{i=-\infty}^{\infty} [\beta[i]z^{-i}] \sum_{m=-\infty}^{\infty} \phi_{nn}[m+l-i]z^{-(m+l-i)}\right]$$

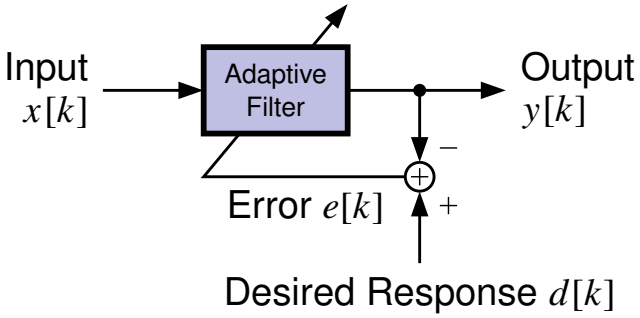
$$= \sum_{l=-\infty}^{\infty} [\alpha[l]z^{+l}]\,[B(z)\Phi_{nn}(z)]$$

$$= A(z^{-1})\Phi_{nn}(z)B(z).$$

- By extension, $\Phi_{xx}(z) = A(z^{-1})\Phi_{nn}(z)A(z)$.

7.3: Optimal two-sided (Wiener) solution

- An adaptive filter is drawn to the right.
- It has an input, an output, and a special input called the desired response.



- Difference between desired and actual filter output is the filter error.
- We wish to find the filter that minimizes the error in some sense.
- A typical "cost function" that is minimized is the mean-squared-error (MSE) of the filter, $\mathbb{E}\left[e[k]^2\right]$.
- Let us solve for the filter that minimizes this error. Note, we allow the filter to be "two sided" (noncausal).

$$e[k] = d[k] - y[k]$$

$$e^2[k] = d^2[k] + y^2[k] - 2d[k]y[k]$$

$$= d^2[k] + \sum_{l=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h[l]h[m]x[k-l]x[k-m]$$

$$-2 \sum_{l=-\infty}^{\infty} h[l]x[k-l]d[k]$$

$$\mathbb{E}\left[e^2[k]\right] = \mathbb{E}\left[d^2[k]\right] + \sum_{l=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h[l]h[m]\mathbb{E}\left[x[k-l]x[k-m]\right]$$

$$-2 \sum_{l=-\infty}^{\infty} h[l]\mathbb{E}\left[x[k-l]d[k]\right]$$

$$= \phi_{dd}[0] + \sum_{l=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h[l]h[m]\phi_{xx}[l-m] - 2\sum_{l=-\infty}^{\infty} h[l]\phi_{xd}[l].$$

- To minimize this expression with respect to the filter weights, we take the derivative of MSE with respect to one specific weight, and set that derivative to zero.

$$\frac{\partial \mathbb{E}\left[e^2[k]\right]}{\partial h[j]} = 0 + 2\sum_{l=-\infty}^{\infty} h^*[l]\phi_{xx}[j-l] - 2\phi_{xd}[j] \equiv 0$$

$$\sum_{l=-\infty}^{\infty} h^*[l]\phi_{xx}[j-l] = \phi_{xd}[j]$$

$$h^*[j] * \phi_{xx}[j] = \phi_{xd}[j]$$

$$H^*(z)\Phi_{xx}(z) = \Phi_{xd}(z)$$

$$H^*(z) = \frac{\Phi_{xd}(z)}{\Phi_{xx}(z)}.$$

- This solution computes the transfer function (and also the weights) of the filter that minimizes the MSE.
  - It is called the "Wiener solution."
  - It is unconstrained, so the result may be noncausal.
- The error may not necessarily be reduced to zero. We can actually compute the minimum MSE:

$$\mathbb{E}\left[e^2[k]\right]_{\min} = \phi_{dd}[0] + \sum_{l=-\infty}^{\infty} h^*[l]\phi_{xd}[l] - 2\sum_{l=-\infty}^{\infty} h^*[l]\phi_{xd}[l]$$

$$= \phi_{dd}[0] - \sum_{l=-\infty}^{\infty} h^*[l]\phi_{xd}[l].$$

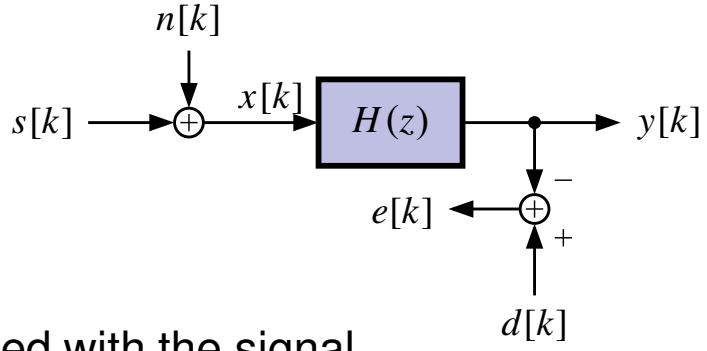
- One useful property of the Wiener solution is

$$\begin{aligned}\mathbb{E}\Big[x[k]e[k+m]\Big] &= \mathbb{E}\Big[x[k](d[k+m]-y[k+m])\Big] \\ &= \mathbb{E}\left[x[k]d[k+m]-\sum_{l=-\infty}^{\infty}h[l]x[k-l+m]x[k]\right] \\ &= \phi_{xd}[m]-\sum_{l=-\infty}^{\infty}h[l]\phi_{xx}[m-l] \\ &= 0 \quad \text{when} \quad h[l]=h^*[l].\end{aligned}$$

- This shows that the crosscorrelation between the filter input and the error is zero for the Wiener solution.
- No linear combination of the weights can reduce the error. This makes sense!

EXAMPLE: This example shows a signal-processing application of Wiener filter theory: Filtering a signal to try to eliminate noise from it.

- Consider the system to the right:
- The desired response is $d[k]=s[k]$.



- Assume that the noise is uncorrelated with the signal, noise is zero mean.

$$H^*(z)=\frac{\Phi_{xd}(z)}{\Phi_{xx}(z)}$$

$$\begin{aligned}\phi_{xd}[m] &= \mathbb{E}\Big[x[k]d[k+m]\Big]=\mathbb{E}\Big[(s[k]+n[k])s[k+m]\Big] \\ &= \mathbb{E}\Big[s[k]s[k+m]\Big]+\underbrace{\mathbb{E}\Big[n[k]s[k+m]\Big]}_{0}\end{aligned}$$

$$= \phi_{ss}[m].$$

$$\phi_{xx}[m] = \mathbb{E}\Big[x[k]x[k+m]\Big] = \mathbb{E}\Big[(s[k]+n[k])(s[k+m]+n[k+m])\Big]$$

$$= \mathbb{E}\Big[s[k]s[k+m]\Big] + \mathbb{E}\Big[n[k]n[k+m]\Big]$$

$$+ \underbrace{\mathbb{E}\Big[s[k]n[k+m]\Big]}_{0} + \underbrace{\mathbb{E}\Big[n[k]s[k+m]\Big]}_{0}$$

$$= \phi_{ss}[m] + \phi_{nn}[m].$$

- Therefore,

$$H^*(z) = \frac{\Phi_{ss}(z)}{\Phi_{ss}(z) + \Phi_{nn}(z)}.$$

  - Frequencies where noise is high results in low gain.
  - Frequencies where noise is low results in near-unity gain.

EXAMPLE: Let $\phi_{ss}[m] = \frac{10}{27}\left(\frac{1}{2}\right)^{|m|}$ and $\phi_{nn}[m] = \frac{2}{3}\delta[m]$.

- First, find $\Phi_{ss}(z)$. (Remember, $\phi_{ss}[k]$ is a two-sided signal.)

$$\Phi_{xd}(z) = \Phi_{ss}(z) = \sum_{k=-\infty}^{\infty} \frac{10}{27}\left(\frac{1}{2}\right)^{|k|} z^{-k}$$

$$= \frac{\frac{10}{27}}{1-\frac{1}{2}z^{-1}} + \frac{\frac{10}{27}\cdot\frac{1}{2}z}{1-\frac{1}{2}z}$$

$$= \frac{\frac{10}{27}(1-\frac{1}{4})}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{2}z)}$$

$$= \frac{5/18}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{2}z)}.$$

- Now, find $\Phi_{xx}(z)$.

$$\Phi_{xx}(z) = \Phi_{ss}(z) + \underbrace{\Phi_{nn}(z)}_{2/3}$$

$$= \frac{20 - 6z - 6z^{-1}}{18(1 - \frac{1}{2}z^{-1})(1 - \frac{1}{2}z)}$$

$$= \frac{(1 - \frac{1}{3}z^{-1})(1 - \frac{1}{3}z)}{(1 - \frac{1}{2}z^{-1})(1 - \frac{1}{2}z)}.$$

- So,

$$H^*(z) = \frac{\Phi_{ss}(z)}{\Phi_{ss}(z) + \Phi_{nn}(z)} = \frac{5/18}{(1 - \frac{1}{3}z^{-1})(1 - \frac{1}{3}z)}$$

  and $h^*[k] = \frac{5}{16}\left(\frac{1}{3}\right)^{|k|}$.

- The minimum MSE can be computed

$$\mathbb{E}\left[e^2[k]\right]_{\min} = \phi_{dd}[0] - \sum_{l=-\infty}^{\infty} h^*[l]\phi_{xd}[l]$$

$$= \frac{10}{27} - \sum_{l=-\infty}^{\infty} \left(\frac{5}{16}\right)\left(\frac{1}{3}\right)^{|l|}\left(\frac{10}{27}\right)\left(\frac{1}{2}\right)^{|l|}$$

$$= \frac{10}{27} - \frac{25}{216}\sum_{l=-\infty}^{\infty}\left(\frac{1}{6}\right)^{|l|} = \frac{10}{27} - \frac{25}{216}\cdot\frac{1 - (\frac{1}{6})^2}{(1 - \frac{1}{6})^2}$$

$$= \frac{80}{216} - \frac{35}{216} = \frac{5}{24}.$$

7.4: Optimal one-sided (Shannon–Bode) solution

- In the previous example, the solution was a non-causal filter.
- Usually, we are concerned with causal results, so will modify the method to include a causality constraint.
- Before, to find the $h^*[n]$ that minimizes MSE, we differentiated and set the derivatives to zero. We got the result

$$\sum_{l=-\infty}^{\infty} h^*[l]\phi_{xx}[j-l] = \phi_{xd}[j].$$

- Now, we restrict $h^*[l]$ to be causal

$$\sum_{l=0}^{\infty} h^*_{\text{causal}}[l]\phi_{xx}[j-l] = \phi_{xd}[j], \qquad j \geq 0$$

$$h^*_{\text{causal}}[j] = 0, \qquad j < 0.$$

- This is not convolution, so we need to be clever when solving it.
- Let us try a simple case when $\phi_{xx}[m] = \delta[m]$. Then

$$h^*_{\text{causal}}[j] = \begin{cases} \phi_{xd}[j], & j \geq 0; \\ 0, & j < 0. \end{cases}$$

- With the same $\phi_{xx}[m]$, but without the causality constraint on $h^*[j]$, we found that $h^*[j] = \phi_{xd}[j]$ for all j.
- So, for "white" inputs, the causal Wiener filter is the same as the noncausal filter, except that the noncausal part is set to zero.
- This is the key to solving the problem.

1. "Whiten" the input.

2. Compute the unconstrained solution.
3. Retain only the causal part.

1) WHITENING THE INPUT: Assuming that the input signal $x[k]$ is some filtered white signal, we can always write its autocorrelation transform as

$$\Phi_{xx}(z) = \frac{A(1-az^{-1})(1-az)(1-bz^{-1})(1-bz)\cdots}{(1-\alpha z^{-1})(1-\alpha z)(1-\beta z^{-1})(1-\beta z)\cdots}$$

where a, b, α, β (etc) have magnitude less than 1.

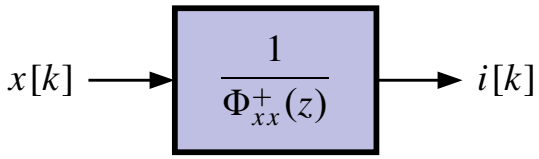
- We can factor this as

$$\Phi_{xx}(z) = \Phi_{xx}^{+}(z)\Phi_{xx}^{-}(z)$$

$$\Phi_{xx}^{+}(z) = \sqrt{A}\frac{(1-az^{-1})(1-bz^{-1})\cdots}{(1-\alpha z^{-1})(1-\beta z^{-1})\cdots}$$

$$\Phi_{xx}^{-}(z) = \sqrt{A}\frac{(1-az)(1-bz)\cdots}{(1-\alpha z)(1-\beta z)\cdots}.$$

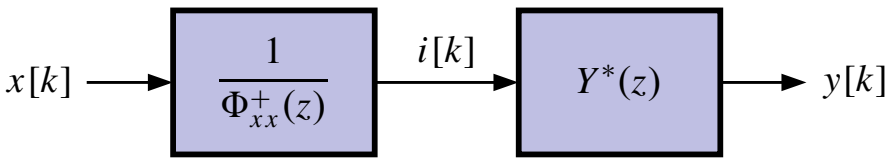
- All poles and zeros of $\Phi_{xx}^{+}(z)$ are inside the unit circle, and all poles and zeros of $\Phi_{xx}^{-}(z)$ are outside the unit circle.
- Let $H_{\text{whitening}}(z) = \dfrac{1}{\Phi_{xx}^{+}(z)}$.



- Then, $\Phi_{ii}(z) = \dfrac{1}{\Phi_{xx}^{+}(z)}\dfrac{1}{\Phi_{xx}^{+}(z^{-1})}\Phi_{xx}(z) = \dfrac{1}{\Phi_{xx}^{+}(z)}\dfrac{1}{\Phi_{xx}^{-}(z)}\Phi_{xx}(z) = 1.$
- Therefore, $\phi_{ii}[m] = \delta[m]$ and the filter output signal is white.

2) UNCONSTRAINED SOLUTION WITH $i[k]$ WHITE: The remaining, noncausal, solution is

$$Y^*(z) = \frac{\Phi_{xd}(z)}{\Phi_{xx}(z)}\Phi^+_{xx}(z) = \frac{\Phi_{xd}(z)}{\Phi^-_{xx}(z)}.$$



3) CONSTRAIN: We now constrain the solution to be causal

$$Y^*_{\text{causal}}(z) = \left[\frac{\Phi_{xd}(z)}{\Phi^-_{xx}(z)}\right]_+$$

$$H^*_{\text{causal}}(z) = \frac{1}{\Phi^+_{xx}(z)}\left[\frac{\Phi_{xd}(z)}{\Phi^-_{xx}(z)}\right]_+.$$

EXAMPLE: Recall the earlier noise-cancelling example, where

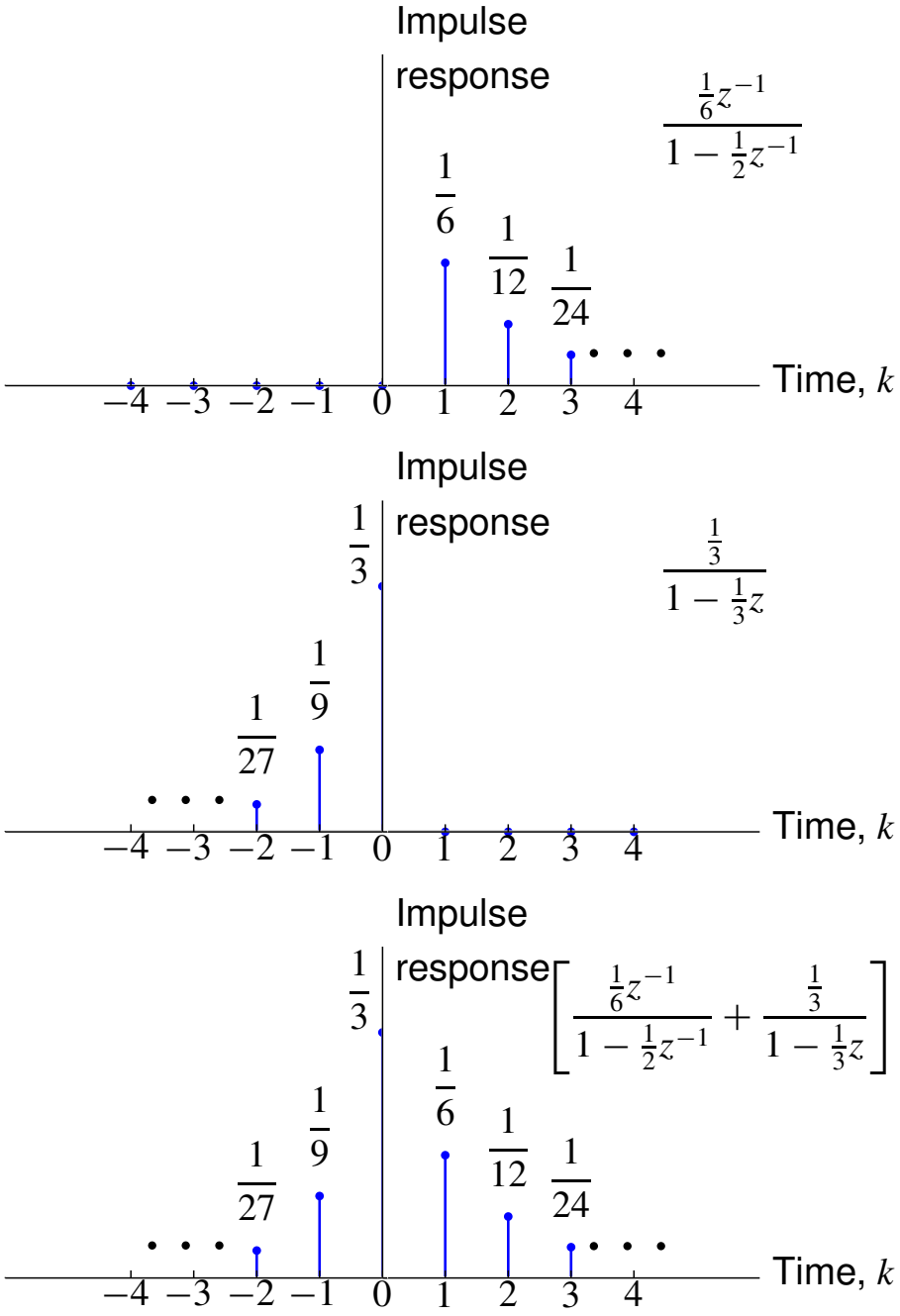
$$\Phi_{xx}(z) = \frac{(1-\frac{1}{3}z^{-1})(1-\frac{1}{3}z)}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{2}z)} \quad \text{and} \quad \Phi_{xd}(z) = \frac{\frac{5}{18}}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{2}z)}.$$

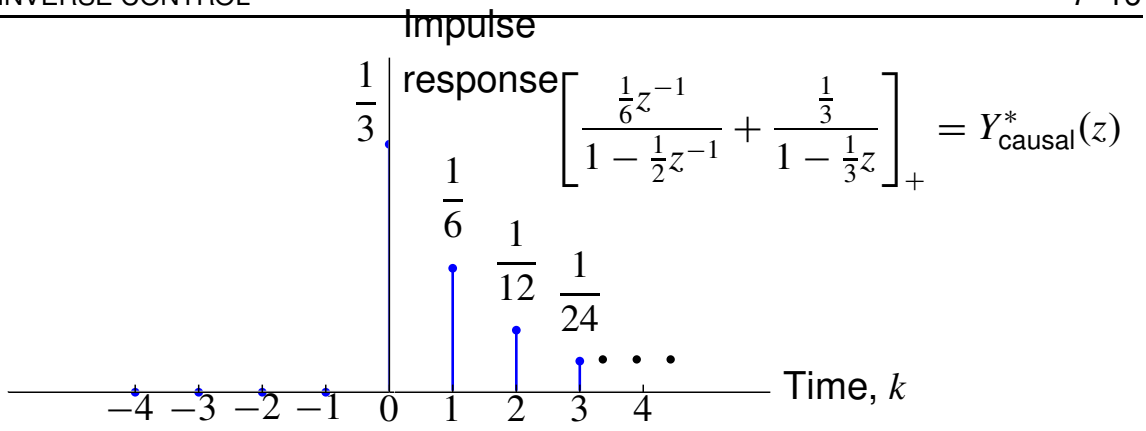
- We factor $\Phi_{xx}(z)$

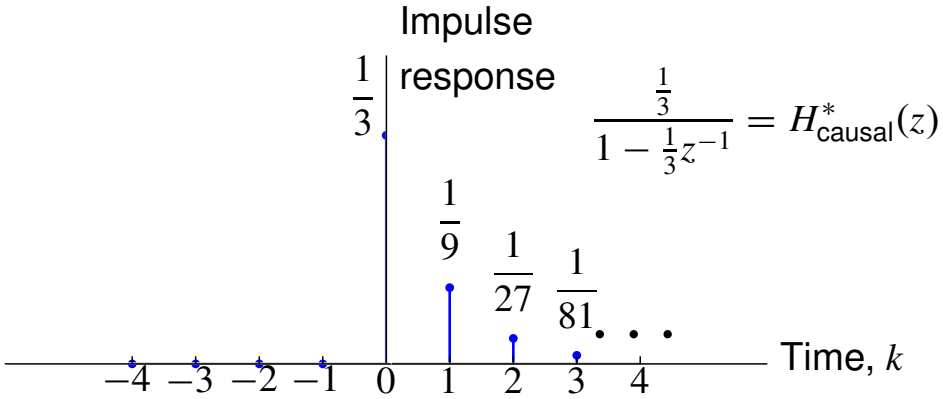
$$\Phi^+_{xx}(z) = \frac{(1-\frac{1}{3}z^{-1})}{(1-\frac{1}{2}z^{-1})} \quad \text{and} \quad \Phi^-_{xx}(z) = \frac{(1-\frac{1}{3}z)}{(1-\frac{1}{2}z)}.$$

- So, the remaining noncausal solution is

$$Y^*(z) = \frac{\Phi_{xd}(z)}{\Phi^-_{xx}(z)} = \frac{\frac{5}{18}}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{3}z)} = \left[\frac{\frac{1}{6}z^{-1}}{1-\frac{1}{2}z^{-1}} + \frac{\frac{1}{3}}{1-\frac{1}{3}z}\right].$$







- Then

$$H^*_{\text{causal}}(z) = \frac{(1-\frac{1}{2}z^{-1})}{(1-\frac{1}{3}z^{-1})} \cdot \frac{\frac{1}{3}}{(1-\frac{1}{2}z^{-1})} = \frac{\frac{1}{3}}{1-\frac{1}{3}z^{-1}}$$

$$\mathbb{E}\left[e^2[k]\right]_{\min} = \frac{10}{27} - \sum_{l=0}^{\infty}\left(\frac{10}{27}\right)\left(\frac{1}{3}\right)\left(\frac{1}{2}\right)^l = \frac{6}{27} > \frac{5}{24}.$$

- The causal filter does worse, as expected.

7.5: Adapting linear filters

- We have studied the structure of linear adaptive filters and the optimal solution.
- Now, we see how to adapt the weights.
- We assume that the filter is FIR (no feedback connections).
- It has input vector $X[k] = \begin{bmatrix} x[k] & x[k-1] & \cdots & x[k-n] \end{bmatrix}^T$.
- It has a weight vector $W = \begin{bmatrix} w[0] & w[1] & \cdots & w[n] \end{bmatrix}^T$.
- For any given set of weights in the weight vector, we compute the filter output as $y[k] = W^T X[k]$.
- The filter MSE can be computed as

$$e[k] = d[k] - y[k] = d[k] - W^T X[k] = d[k] - X^T[k]W$$

$$e^2[k] = d^2[k] - 2d[k]X^T[k]W + W^T X[k]X^T[k]W$$

$$\mathbb{E}\left[e^2[k]\right] = \mathbb{E}\left[d^2[k]\right] - 2P^T W + W^T RW,$$

  where we define $P = \mathbb{E}\left[X[k]d^T[k]\right]$ and $R = \mathbb{E}\left[X[k]X^T[k]\right]$.
- The MSE, as a function of the weights, is a parabolic bowl.
- There is one minima, and it may be found by gradient descent.
- We update the weights of the filter in the direction of the negative gradient

$$\nabla = \begin{bmatrix} \partial \text{MSE}/\partial w[0] \\ \partial \text{MSE}/\partial w[1] \\ \vdots \\ \partial \text{MSE}/\partial w[n] \end{bmatrix} = -2P + 2RW.$$

- At the minimum, $\nabla = 0$, which gives $P = RW^*$ or $W^* = R^{-1}P$ if R^{-1} exists. Then,

$$\text{MSE}_{\min} = \mathbb{E}\left[d^2[k]\right] - P^T W^*$$

$$\text{MSE} = \text{MSE}_{\min} + (W - W^*)^T R (W - W^*)$$

The LMS algorithm

- To perform gradient descent:

$$W[k+1] = W[k] - \mu\nabla[k] = W[k] - \mu\left[2RW[k] - 2P\right].$$

- This does not help much, since we need to know R and P.
- So, we try a different approach. We let

$$\widehat{\nabla}[k] = \begin{bmatrix} \partial e^2[k]/\partial w[0] \\ \partial e^2[k]/\partial w[1] \\ \vdots \\ \partial e^2[k]/\partial w[n] \end{bmatrix} = 2e[k] \begin{bmatrix} \partial e[k]/\partial w[0] \\ \partial e[k]/\partial w[1] \\ \vdots \\ \partial e[k]/\partial w[n] \end{bmatrix}.$$

- Recall that $e[k] = d[k] - X^T[k]W$. Therefore, $\widehat{\nabla}[k] = -2e[k]X[k]$.
- Note that $\mathbb{E}\left[\widehat{\nabla}[k]\right] = \nabla[k]$. Using $\widehat{\nabla}[k]$ instead of $\nabla[k]$ is unbiased but noisy.
- So, our weight-update algorithm, the "LMS Algorithm," is very simply

$$W[k+1] = W[k] + 2\mu e[k]X[k].$$

- It can be shown that the LMS algorithm is stable for $0 < \mu < \dfrac{2}{\|X[k]\|^2}$ when all values of k are considered.
- An adaptive learning rate can also be used

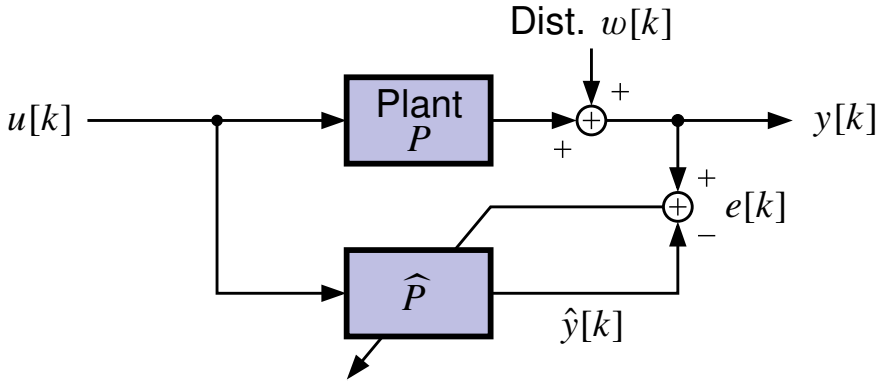
$$\mu[k] = \min_{0 \le j \le k} \frac{1}{\|X[j]\|^2}.$$

7.6: System identification

- Conventional control techniques need models of the plant in order to be able to design controllers for them.
- Most adaptive-control techniques (incl. AIC) also need plant models.
- Process of constructing a plant model is called system identification.
- Lennart Ljung defines system ID to be "the art and methodology of building mathematical models of dynamical systems based on input–output data" (and prior knowledge).

Adaptive system identification

- It is not important what kind of system identification is done to be able to do adaptive inverse control.
- There is an advantage to adaptive system identification, however: If the plant is slowly time varying, an adaptive model reflects those changes and a controller can adapt to control the varying plant.
- Adaptive system identification is represented as shown:



- The plant model is placed in parallel with the plant.
- It receives the same input signal as the plant, and it predicts what the plant output will be.
- The difference between the actual plant output and the plant model output is used to adapt the plant model

- The LMS algorithm may be used directly. The plant itself has a certain impulse response vector

$$\mathbf{P} = \begin{bmatrix} p[0] & p[1] & p[2] & \cdots & p[l] & \cdots \end{bmatrix}^T.$$

- An FIR plant model has its impulse response vector

$$\widehat{\mathbf{P}}[k] = \begin{bmatrix} \hat{p}[0] & \hat{p}[1] & \hat{p}[2] & \cdots & \hat{p}[n] \end{bmatrix}^T.$$

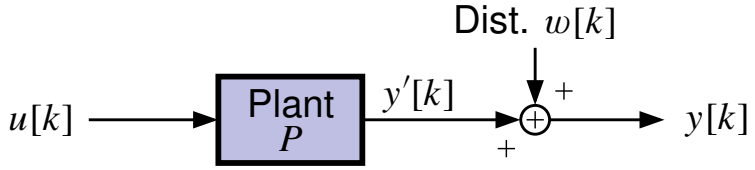
- There are $n+1$ weights and each weight is a function of time, k.
- When the weights converge or are fixed, the filter becomes linear.
- Weights converge to "Wiener solution" when minimizing MSE.
- $\widehat{\mathbf{P}}[k]$ may converge very closely to $\mathbf{P}$. However, there are three sources of "mismatch" between them.

1. $\widehat{\mathbf{P}}[k]$ is of finite length, while $\mathbf{P}$ may be infinitely long. By choosing n "large enough" this problem is reduced.
2. If the input to the plant $u[k]$ does not adequately cover the input space of the plant, then the model will not be very general. $u[k]$ needs to be "persistently exciting."
   For a linear plant, the input needs to cover all frequencies of interest. If it does not, some small amount of white noise may be added to the input signal. This process is called "dither." It helps make the modeling process solid, but adds additional disturbance to the control system.
3. The adaptation process itself adds noise to the weights of $\widehat{\mathbf{P}}[k]$ ("misadjustment"). Finite amounts of data are used to determine $\widehat{\mathbf{P}}[k]$. Only if infinite amounts of data are used, and the adaptation is done infinitely slowly, will there be no weight noise.

The usual solution is to adapt quickly to get an approximate model with high noise, and then adapt slowly to reduce the weight noise.

Ideal modeling performance

- First, we assume that the plant model is not necessarily causal or finite length.



- Then, the Wiener solution for the plant model is $\widehat{P}(z) = \dfrac{\Phi_{uy}(z)}{\Phi_{uu}(z)}$ where

$$\begin{aligned}\phi_{uy}[k] &= \mathbb{E}\big[u[j]y[j+k]\big] \\ &= \mathbb{E}\big[u[j](y'[j+k] + w[j+k])\big] \\ &= \mathbb{E}\big[u[j]y'[j+k]\big] + \mathbb{E}\big[u[j]w[j+k]\big] \\ &= \phi_{uy'}[k] + \phi_{uw}[k].\end{aligned}$$

- Assume $u[k]$ and $w[k]$ are uncorrelated, and $\mathbb{E}\big[w[k]\big] = 0$. Then,

$$\begin{aligned}\phi_{uy}[k] &= \phi_{uy'}[k] \\ \Phi_{uy}(z) &= \Phi_{uy'}(z) \\ \widehat{P}(z) &= \frac{\Phi_{uy'}(z)}{\Phi_{uu}(z)} = P(z).\end{aligned}$$

- Note, the input signal is assumed to be "persistently exciting." If not, then $\Phi_{uu}(e^{j\omega})$ will be zero at some value of ω and the division fails.
- A plant model adapted using LMS will not "blow up," but it will not be correct, either.

7.7: Sources of modeling errors

Mismatch due to FIR models

- Linear plants usually have exponential impulse response behavior and so have an infinitely long impulse response.
- However, we usually model a plant as FIR, so there is some mismatch.
- The FIR Wiener solution is

$$\widehat{\mathbf{P}} = [\phi_{uu}]^{-1}[\phi_{uy}]$$

$$\begin{bmatrix} \hat{p}[0] \\ \hat{p}[1] \\ \vdots \\ \hat{p}[n-1] \end{bmatrix} = \begin{bmatrix} \phi_{uu}[0] & \phi_{uu}[1] & \cdots & \phi_{uu}[n-1] \\ \phi_{uu}[1] & \phi_{uu}[0] & \cdots & \phi_{uu}[n-2] \\ \vdots & & & \\ \phi_{uu}[n-1] & \phi_{uu}[n-2] & \cdots & \phi_{uu}[0] \end{bmatrix}^{-1} \begin{bmatrix} \phi_{uy}[0] \\ \phi_{uy}[1] \\ \vdots \\ \phi_{uy}[n-1] \end{bmatrix}.$$

- Again, assuming that $\mathbb{E}[w_k] = 0$ and that $u[k]$ and $w[k]$ are uncorrelated, $\phi_{uy}[k] = \phi_{uy'}[k]$, which gives $\widehat{\mathbf{P}} = [\phi_{uu}]^{-1}[\phi_{uy'}]$.
- This is exact, but hard to interpret. Assume, for example, that $u[k]$ is white

$$[\phi_{uu}]^{-1} = \begin{bmatrix} \phi_{uu}[0] & \cdots & 0 \\ \vdots & \phi_{uu}[0] & \\ 0 & & \ddots \end{bmatrix}^{-1} = \frac{1}{\phi_{uu}[0]} I.$$

- Also,

$$[\phi_{uy'}] = \begin{bmatrix} p[0] \\ p[1] \\ \vdots \\ p[n] \end{bmatrix} \phi_{uu}[0], \quad \text{so} \quad \widehat{\mathbf{P}} = \begin{bmatrix} \hat{p}[0] \\ \hat{p}[1] \\ \vdots \\ \hat{p}[n] \end{bmatrix} = \begin{bmatrix} p[0] \\ p[1] \\ \vdots \\ p[n] \end{bmatrix}.$$

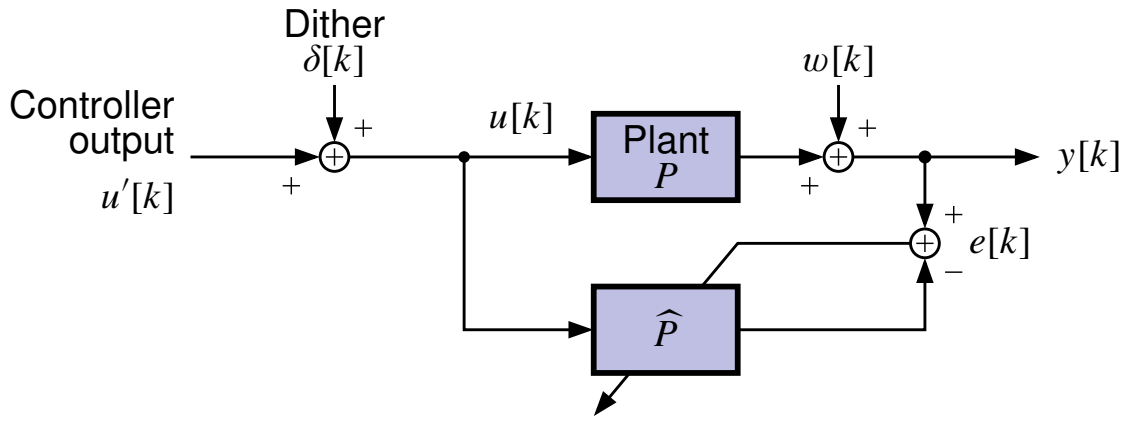
- The FIR model is exactly the same as the first n samples of the plant's impulse response.
- Since the impulse response of a stable system decays to zero, then the plant mismatch goes to zero as $n \to \infty$.
- In practice, n is chosen so that the error is "negligible."

Mismatch due to inadequacies of $u[k]$

- If the plant input (the controller output) does not contain all frequencies, the plant model will not converge to the correct solution.
- If the controller output is nonstationary, there are similar problems.
- One solution is to add a white noise (dither) signal to $u[k]$ to help the plant modeling process.
- This will cause an increase in the plant output disturbance, but much better plant modeling will be done.
- We consider three dither schemes. Here is "scheme A."

$$Y(z) = W(z) + P(z)\big(\Delta(z) + U'(z)\big)$$

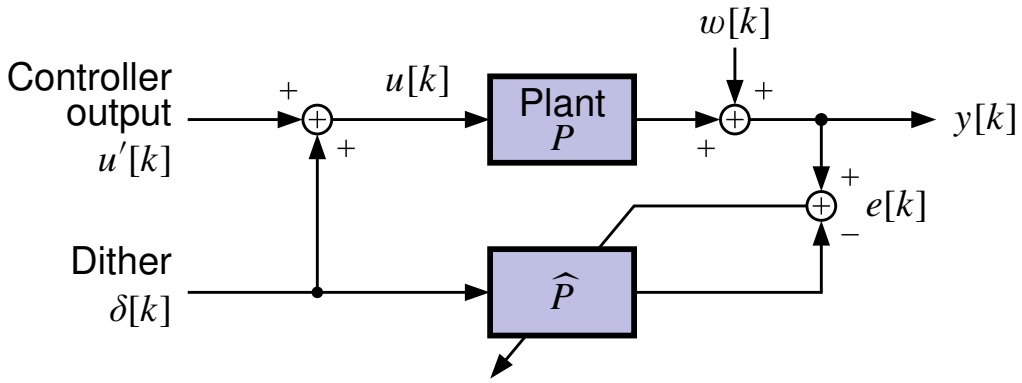
$$\widehat{Y}(z) = \widehat{P}(z)\big(\Delta(z) + U'(z)\big).$$



- In this scheme, dither is added directly to the controller output.
- The basis of plant modeling, however, is still $u'[k]$.
- If $u'[k]$ is nonstationary, there can be problems. Enter "scheme B."

$$Y(z) = W(z) + P(z)(\Delta(z) + U'(z))$$

$$\widehat{Y}(z) = \widehat{P}(z)\Delta(z).$$

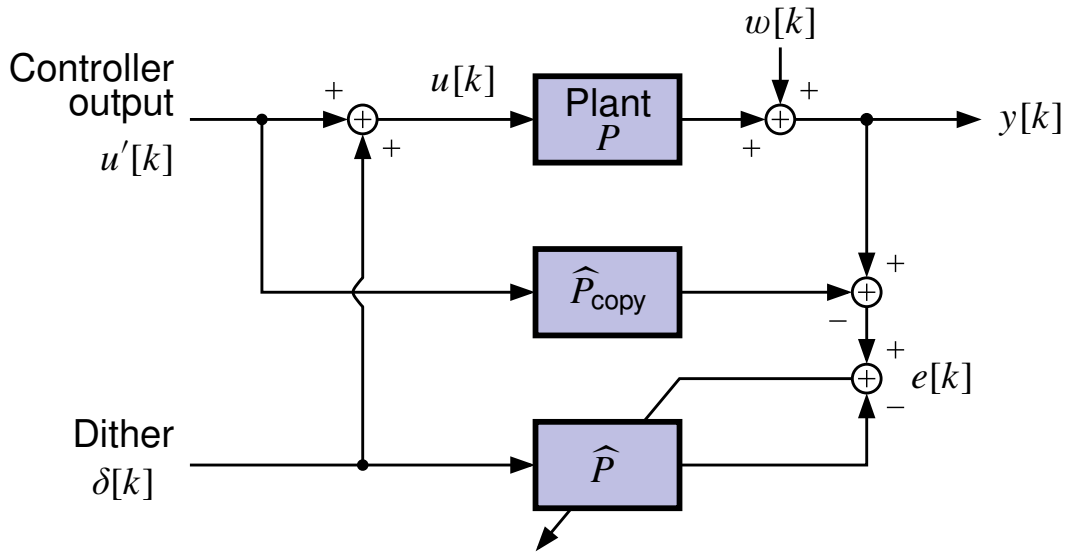


- In scheme B, the basis of the model is the dither part of the signal.
- Since the dither is assumed to be uncorrelated with $u'[k]$, the plant response to $u'[k]$ is uncorrelated with the plant response to $\delta[k]$.
- Therefore, the response to $u'[k]$ is treated as disturbance in the plant modeling process. Because this "disturbance" is so large, the noise in the weights of $\widehat{P}$ will also be large.
- With a bit more complexity, this can be corrected with "scheme C."

$$Y(z) = W(z) + P(z)(\Delta(z) + U'(z))$$

$$D(z) = W(z) + P(z)\Delta(z)$$

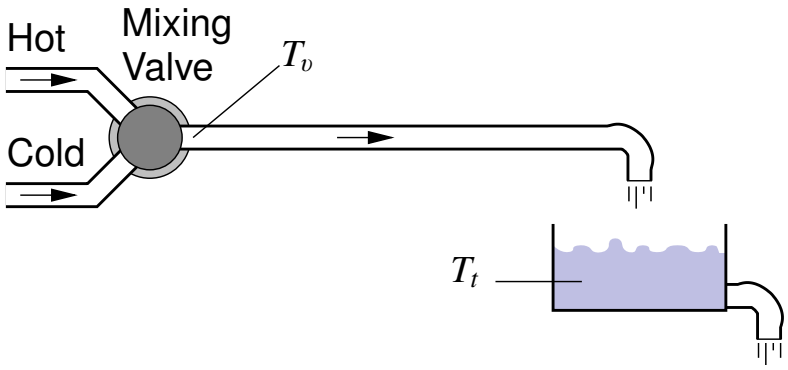
$$\widehat{Y}(z) = \widehat{P}(z)\Delta(z).$$



- Here, the plant response to $u'[k]$ is subtracted out of the adaptation signal in order to reduce the weight noise in $\widehat{P}$.
- Scheme C is the best, but the most complex too.

7.8: Example of tank temperature-control problem

- Consider the following system



- Hot and cold water are mixed at a valve. The goal is to control the temperature of a tank of water.

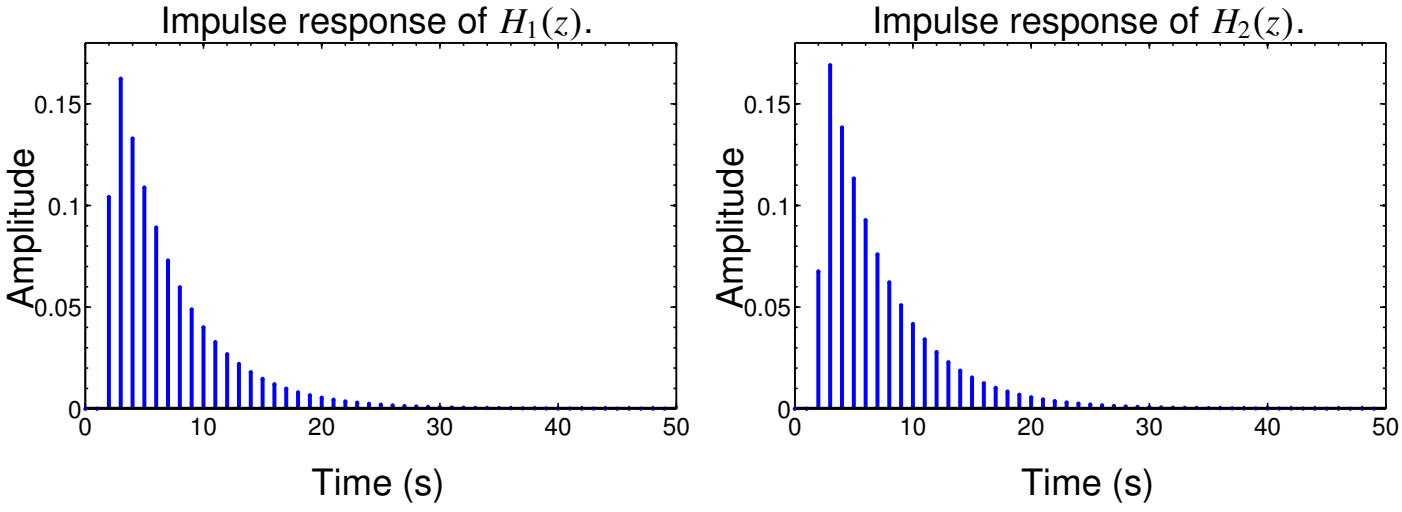
$$H_1(z) = 0.1042\frac{z+0.7402}{z^3-0.8187z^2} \quad \text{and} \quad H_2(z) = 0.0676\frac{z+1.6813}{z^3-0.8187z^2}.$$

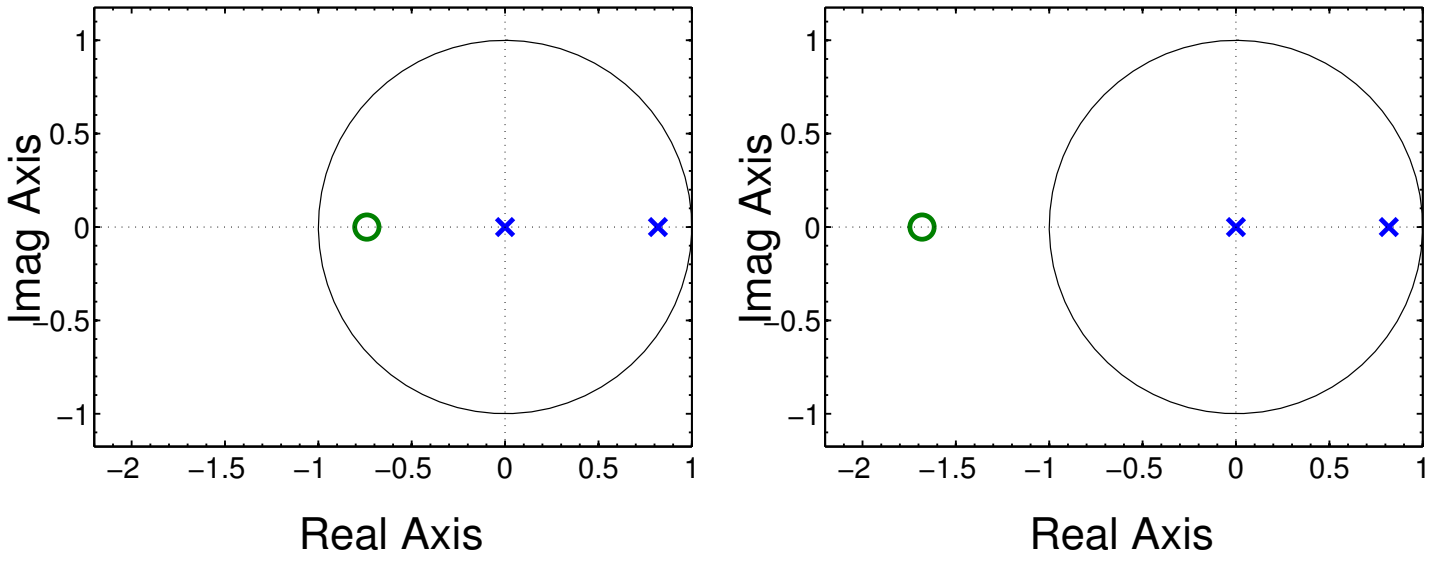
- These two transfer functions differ only in the length of the pipe (transport delay). Notice that $H_1(z)$ is minimum phase and $H_2(z)$ is nonminimum phase.

- The first system, for example, may be implemented as

$$y[k] = 0.8187y[k-1] + 0.1042u[k-2] + (0.1042)(0.7402)u[k-3].$$

- Impulse responses and pole-zero plots of these two plants are:





- With a sufficiently small μ, the estimated impulse response is indistinguishable from the true impulse response.

Sample code for system identification using LMS

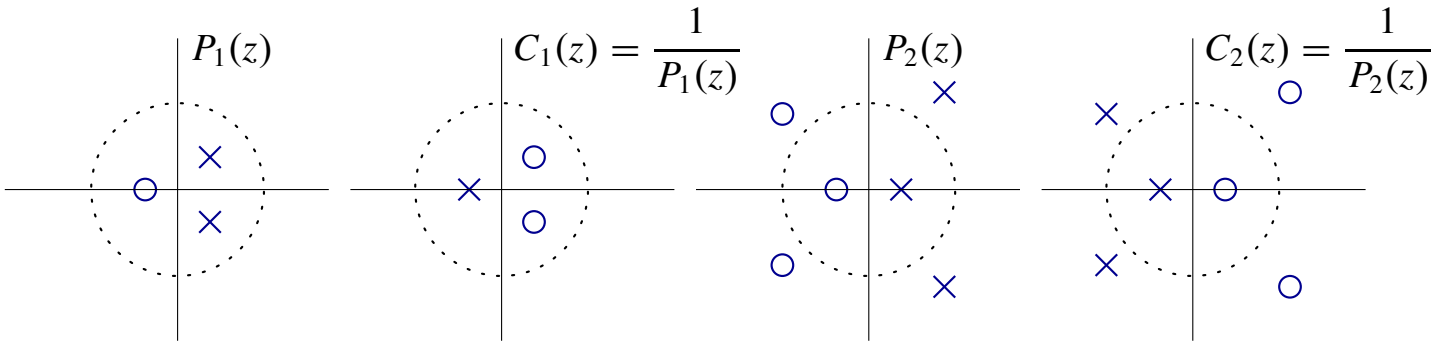
```
numtaps = 50; mu = 0.01;
X = zeros([numtaps 1]);
P = zeros([numtaps 1]);
uk = zeros([4 1]); yk = 0;
numiter = 500;

for k = 1:numiter,
  uk = [randn(1); uk(1:end-1)]; % simulate actual plant
  yk = 0.8187*yk + 0.1042*uk(3) + 0.1042*0.7402*uk(4);

  X = [uk(1); X(1:end-1)]; % simulate plant model
  yhat = P'*X;

  mu = min(mu,1/(X'*X)); % adaptive learning rate
  P = P + 2*mu*(yk-yhat)*X; % adapt P
end
stem(P,'.');
```

7.9: Linear SISO adaptive feedforward control (minimum phase)

- We now start looking at the control problem. We first consider inverse control of minimum-phase plants, then of nonminimum-phase plants.
- The technique is extended to include model-reference control, and effects of disturbance and weight noise will be considered.

Minimum-phase and nonminimum-phase

- Linear systems may be either minimum-phase or nonminimum-phase.
- If we consider a minimum-phase system, it has all of its poles and zero inside the unit circle in the z-plane.
- A nonminimum-phase system has zeros outside the unit circle.

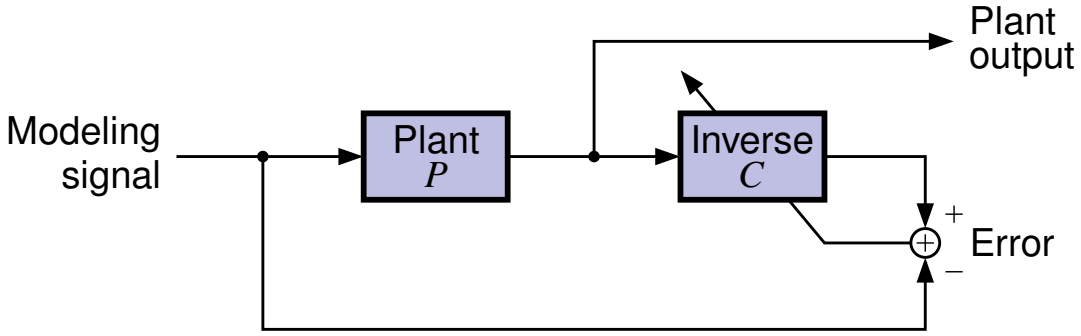


- Let $C_1(z) = 1/P_1(z)$ and let $C_2(z) = 1/P_2(z)$.
- The zeros of $P_1(z)$ become the poles of $C_1(z)$ and the poles of $P_1(z)$ become the zeros of $C_1(z)$. (Same for $C_2(z)$.)
  - $C_1(z)$ may be implemented without difficulty. It is stable and causal.
  - $C_2(z)$ is a problem. It has poles outside the unit circle.
  - $C_2(z)$ is either stable but noncausal, or unstable and causal.
- $P_1(z)$ may be controlled using the controller $C_1(z)$. The overall transfer function of the system is $H(z) = P_1(z)C_1(z) = 1$.

- The output $y[k]$ perfectly follows the input $r[k]$.
- However, since $P_2(z)$ is nonminimum-phase, we have more difficulty controlling it. We will see how soon...

Inverses of minimum-phase plants

- Consider the following system



- Because linear SISO systems commute, $P(z)C(z) = C(z)P(z)$.
- Therefore, the block $P(z)$ followed by $C(z)$ must have the same transfer function as $C(z)$ followed by $P(z)$.
- We use this fact to adapt $P(z)C(z) = 1$.
- We look at the Wiener (unconstrained) solution

$$\Phi_{yy}(z) = \Phi_{uu}(z)P(z)P(z^{-1})$$

$$\Phi_{yd}(z) = \Phi_{uu}(z)P(z^{-1})$$

$$C(z) = \frac{\Phi_{uu}(z)P(z^{-1})}{\Phi_{uu}(z)P(z)P(z^{-1})} = \frac{1}{P(z)}.$$

- Therefore, minimizing the MSE provides the correct inverse.

7.10: Linear SISO adaptive feedforward control (nonmin. phase)

Inverses of nonminimum-phase plants

- The inverse of a nonminimum-phase plant cannot be found exactly this way.
- For example, consider

$$P(z) = \frac{1 + 2z^{-1}}{1 - z^{-1} + \frac{3}{4}z^{-2}}.$$

- This plant is causal and stable, but nonminimum-phase because of the zero at $z = -2$.
- The inverse is

$$\frac{1}{P(z)} = \frac{1 - z^{-1} + \frac{3}{4}z^{-2}}{1 + 2z^{-1}},$$

which has a pole at $z = -2$ (outside the unit circle).

- We can expand this:

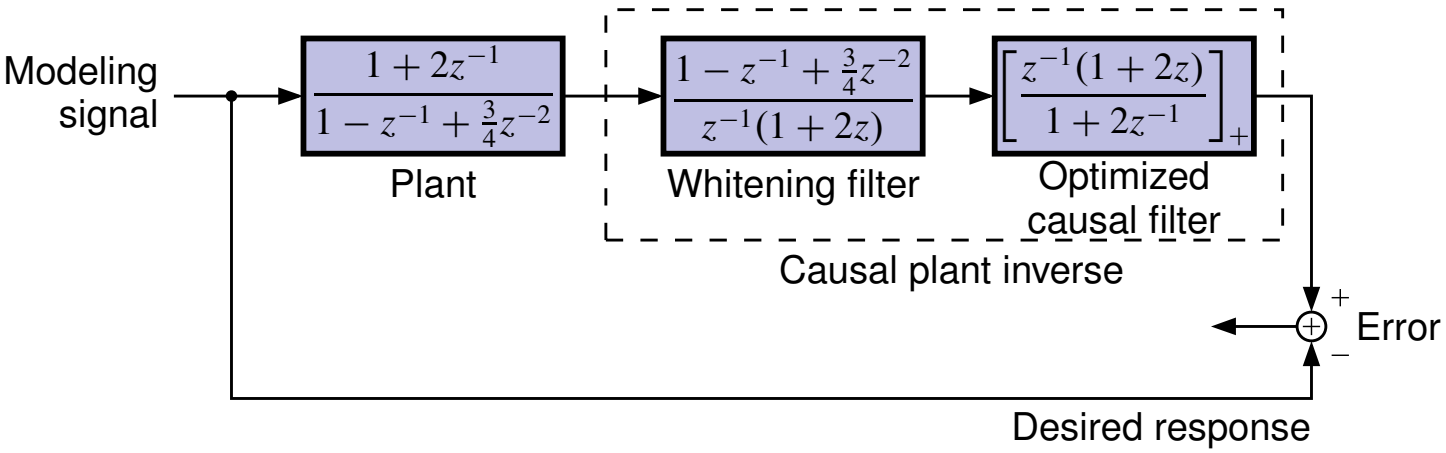
$$\frac{1}{P(z)} = 1 - 3z^{-1} + \frac{27}{4}z^{-2} - \frac{27}{2}z^{-3} + 27z^{-4} \cdots \qquad \text{unstable}$$

or

$$\frac{1}{P(z)} = \frac{3}{8}z^{-1} - \frac{11}{16} + \frac{27}{32}z - \frac{27}{64}z^2 + \frac{27}{128}z^3 \cdots \qquad \text{noncausal}$$

- We see that the inverse can be either stable or causal, but not both!
- If we tried to generate the inverse adaptively, neither of these solutions would arise:
  1. LMS minimizes MSE and will result in a stable system;
  2. Our filters are causal.
- Thus, an imperfect inverse must be generated.

- Consider the Shannon-Bode solution (The mathematical steps used to arrive at this solution are left as an exercise for the reader(!)).



- Consider only the second part of the causal "inverse"

$$\frac{z^{-1}(1+2z)}{1+2z^{-1}} = \frac{1}{2} + \frac{3}{4}z - \frac{3}{8}z^2 + \frac{3}{16}z^3 - \cdots$$

- So, the causal part is

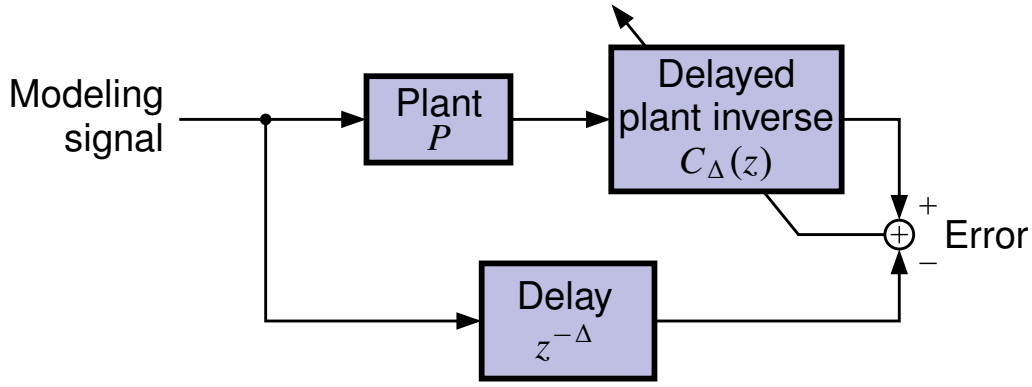
$$\left[\frac{z^{-1}(1+2z)}{1+2z^{-1}}\right]_+ = \frac{1}{2}.$$

- The transfer function of the "inverse" is then $\dfrac{1-z^{-1}+\frac{3}{4}z^{-2}}{4(1+\frac{1}{2}z^{-1})}$.
- This works, but not very well.
- The MSE of the output is $3/4$ versus 1.0 if the transfer function were simply $C(z) = 1$.
- The problem is that by constraining the solution to be causal, we lose a lot of accuracy.

A delayed inverse

- The quality of the inverse may be improved if a delay is allowed.

- That is, instead of $C(z) = 1/P(z)$, use $C_\Delta(z) = z^{-\Delta}/P(z)$.



- For the Shannon-Bode solution, the whitening filter remains the same, but the optimized causal filter changes.
- Without regard to causality, it becomes

$$z^{-\Delta}\left(\frac{z^{-1}(1+2z)}{1+2z^{-1}}\right) = z^{-\Delta}\left(\frac{1}{2}+\frac{3}{4}z-\frac{3}{8}z^{-2}+\frac{3}{16}z^{-3}-\cdots\right),$$

  and the number of causal terms is $\Delta + 1$.
- The MSE when $\Delta = 4$ is about 0.003, which is very good!

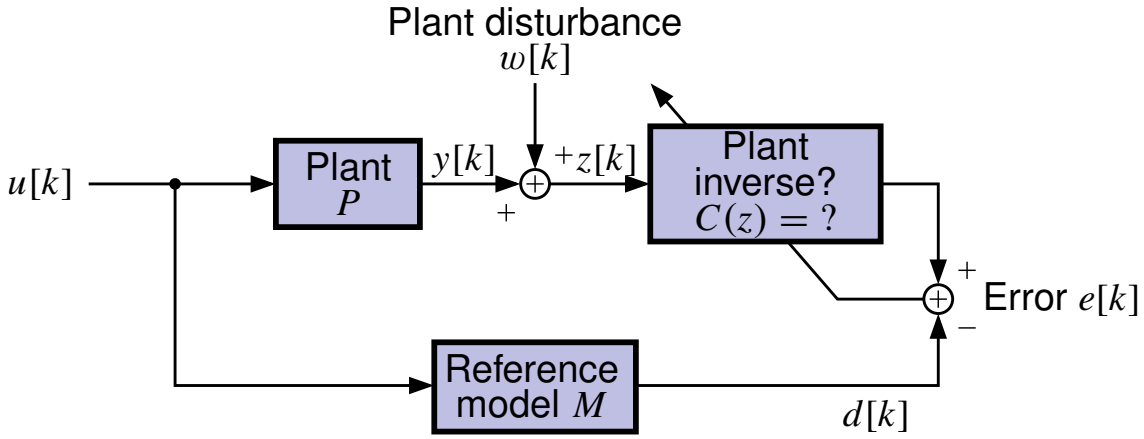
A model-reference based inverse

- We can also find a model-reference based inverse.
- To find $C(z) = M(z)/P(z)$, replace $z^{-\Delta}$ with the more general $M(z)$.
- Adapting will lead to the minimum MSE solution.

7.11: Linear SISO adaptive feedforward control (disturbance)

Inverses of Plants with Disturbances

- This method must be modified slightly if the plant has disturbances.
- If it is used directly, the solution will be incorrect. Consider:



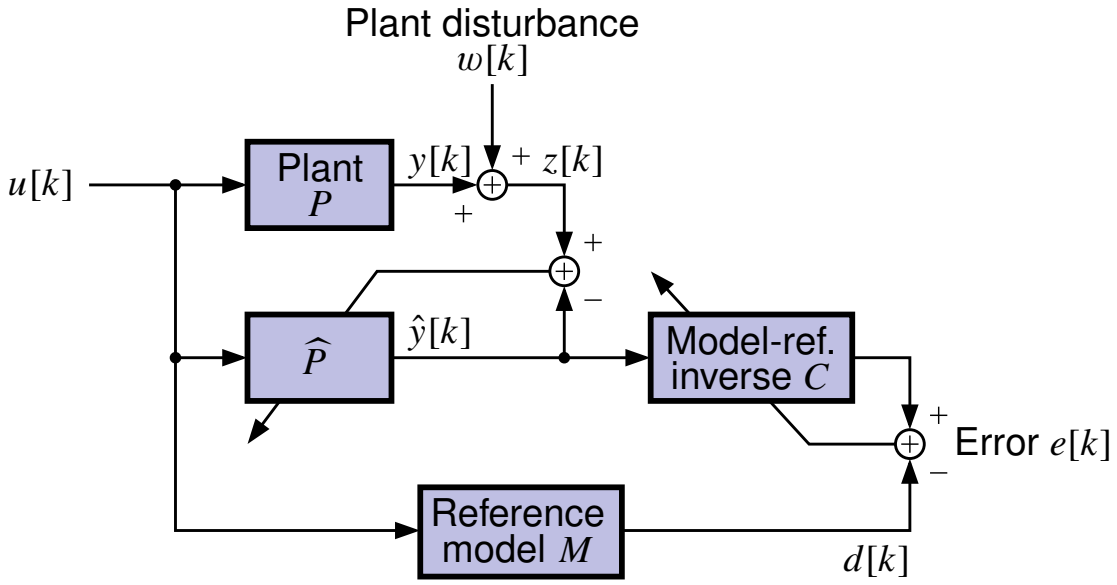
- Without disturbance, the solution would be

$$C(z) = \frac{\Phi_{yd}(z)}{\Phi_{yy}(z)} = \frac{M(z)}{P(z)}.$$

- With disturbance, the solution is

$$C(z) = \frac{\Phi_{zd}(z)}{\Phi_{zz}(z)} = \frac{\Phi_{yd}(z)}{\Phi_{yy}(z) + \Phi_{ww}(z)} \neq \frac{M(z)}{P(z)}.$$

- The solution is to make a plant model $\widehat{P}(z)$.
- Then, let $C(z)$ be the inverse to $\widehat{P}(z)$ and not directly to $P(z)$.
- $\widehat{P}(z)$ has no disturbance, and if it is approximately equal to $P(z)$ then $C(z) \approx M(z)/P(z)$.



- Note that if $\widehat{P}(z) = P(z) + \Delta P(z)$, and we know that $C^*(z) = M(z)/P(z)$ but $C(z) = M(z)/\widehat{P}(z)$, then

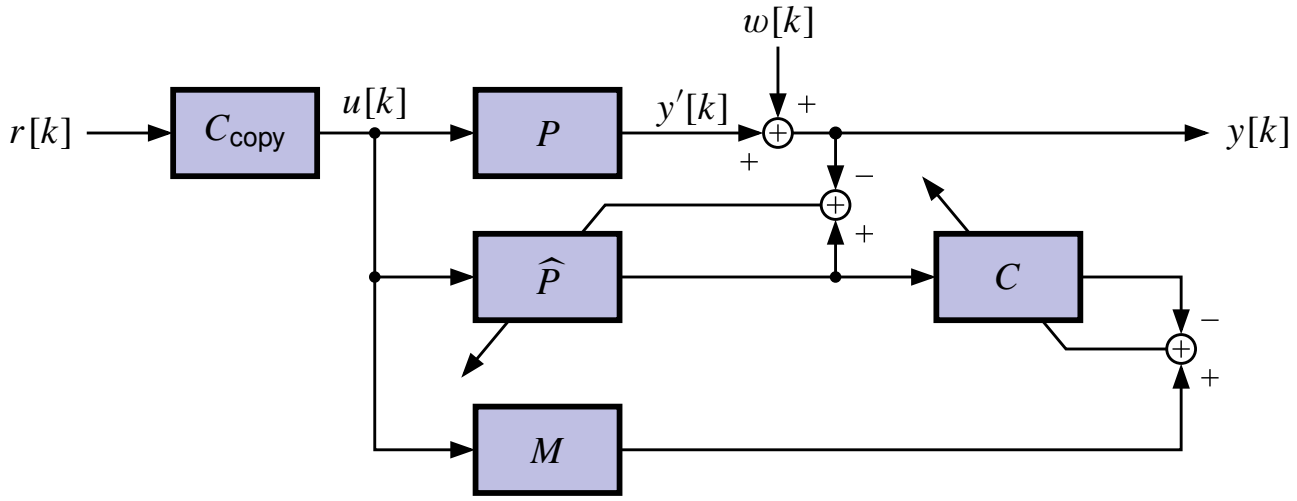
$$C(z) = C^*(z) + \Delta C(z)$$

$$\Delta C(z) = C(z) - C^*(z) = \frac{M(z)}{P(z) + \Delta P(z)} - \frac{M(z)}{P(z)} \approx -\frac{M(z)\Delta P(z)}{P^2(z)}.$$

- So, if $\Delta P(z)/P^2(z)$ is "small", then $\Delta C(z)$ will be small.

SISO Feedforward Linear Control

- The (model-reference/delayed) inverse is used as a controller.
- The weights from the adaptive inverse are copied into the controller



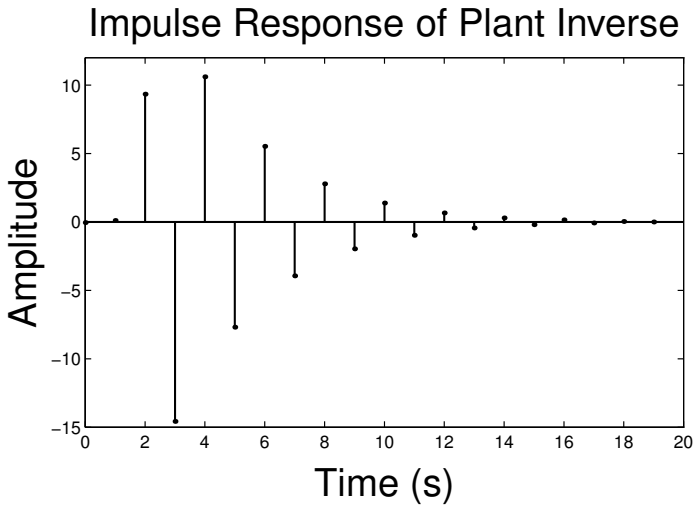
7.12: Initializing the controller weights

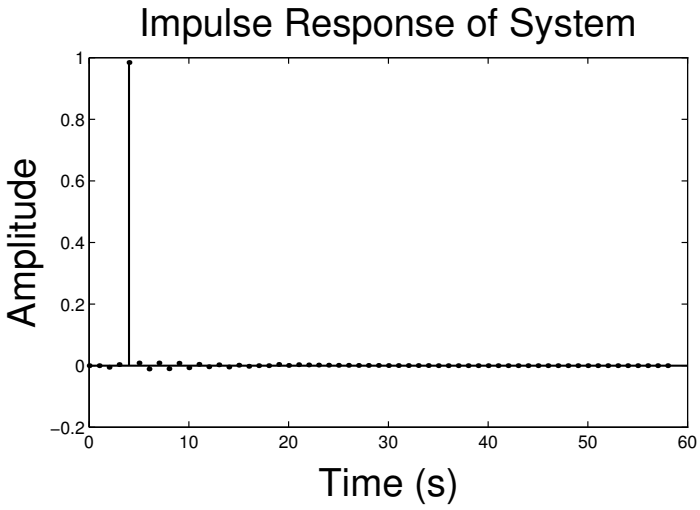
A singularity in the solution

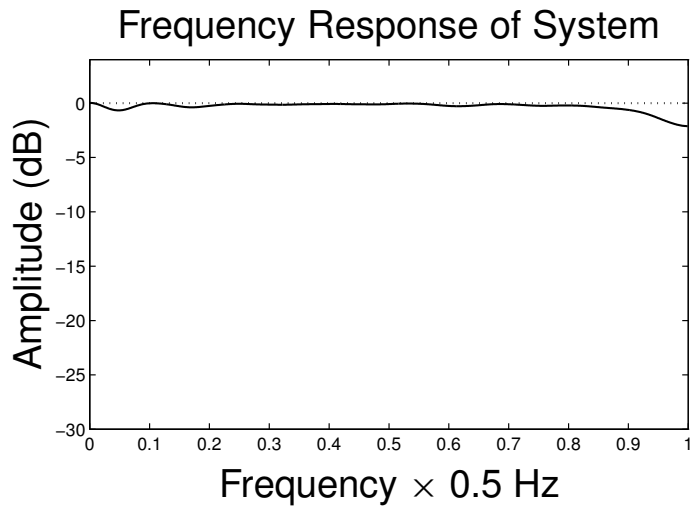
- Notice that the adaptive solution minimizes $e_{\text{eqn}}[k] = C_{\text{copy}}\{r[k]\} - C\{P\{C_{\text{copy}}\{r[k]\}\}\}$ instead of $e_{\text{sys}}[k] = r[k] - P\{C\{r[k]\}\}$.
- If the weights of C are zero, then $e_{\text{eqn}} = 0$ but $e_{\text{sys}} \neq 0$.
- So, by reversing the order of C and P in order to adapt C, we have introduced a new "solution" which is incorrect.
- If you use this method, do not initialize the controller weights to zero!
- If we have an initial model $\widehat{P}(z)$ we can avoid this problem and also speed up controller training by orders of magnitude.
- Use the FFT algorithm to compute $M(e^{j2\pi n/N})$ and $\widehat{P}(e^{j2\pi n/N})$.
- Divide these two vectors pointwise, and take the inverse FFT to get approximate weights for C.

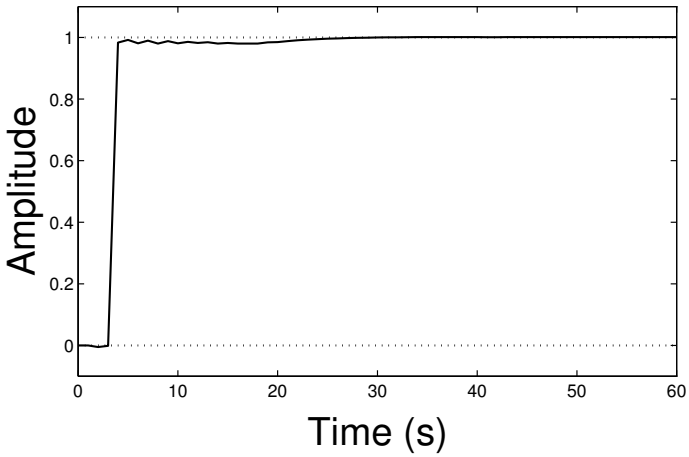
```
% Define reference model: this plant needs delay of 2
M = [0; 0; dimpulse(1,[1 -0.9],510)];
padP = zeros([512 1]); padP(1:length(P)) = P;
fftM = fft(M); fftP = fft(padP); fftC = zeros([512 1]);
ind = find(fftP ~= 0); fftC(ind) = fftM(ind)./fftP(ind);
C = ifft(fftC); C = C(1:25);
stem(C,'.');
```

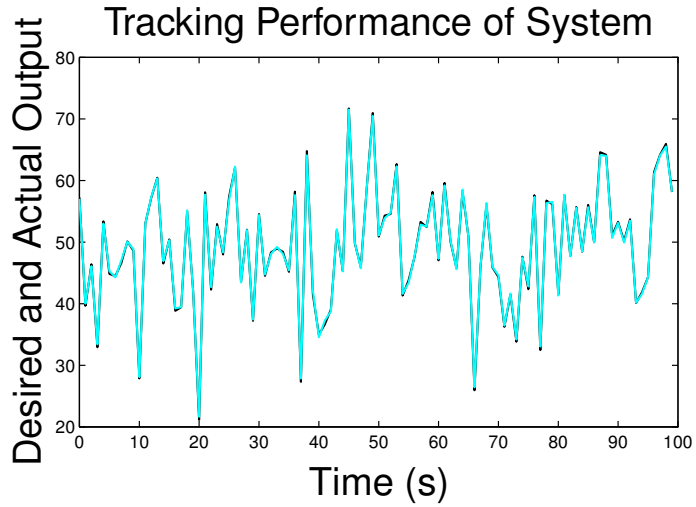
- There are some shortcomings in this approach, but it gives a good approximate control design to begin adaptation with.
- The following plots are from the minimum-phase "tank" example.









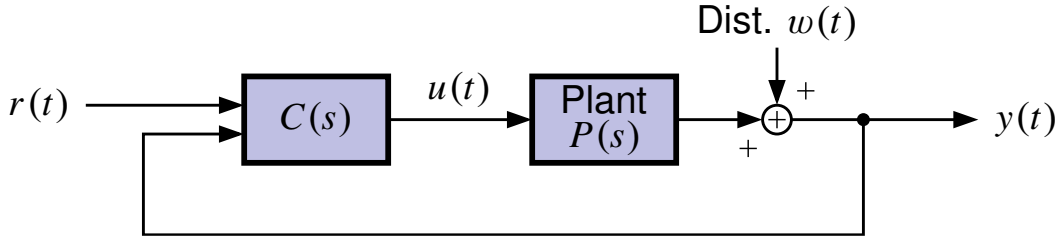


7.13: Disturbance canceling

- We have now considered plant identification and feedforward control.
- We have seen that the adaptive plant model is not biased by disturbance. Also, the controller is not biased by disturbance.
- To put this another way, the plant model converges to a model of the undisturbed plant, and the controller converges such that the undisturbed plant output is the desired output.
- However, nothing has been done to reduce the effects of the disturbance. This is the topic we now address.

Output feedback

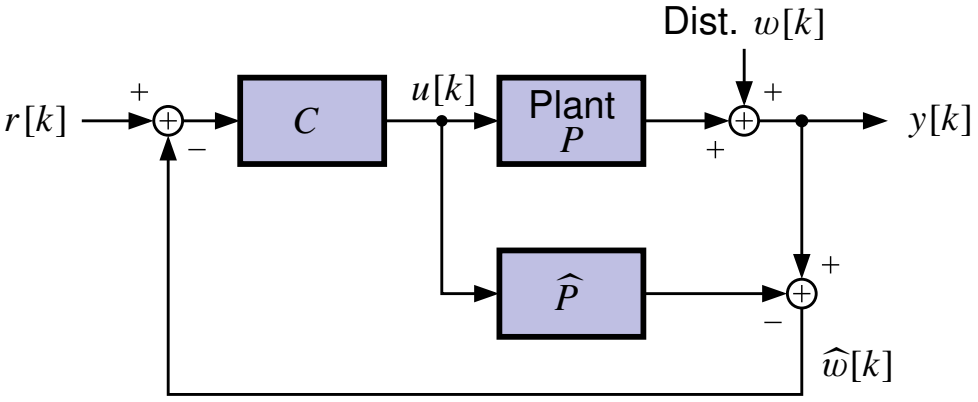
- Classical control techniques use feedback of the plant output to perform disturbance rejection:



- The feedback does three jobs:
  - Stabilizes the system by changing system dynamics;
  - Controls the system by changing system dynamics;
  - Reject disturbance.
- This architecture is not appropriate for use with AIC.
- The "closed loop" transfer function is different from the "open-loop" transfer function, which we design using inverse control.
- There are other problems with this scheme, which we will see later.

Error feedback

- Instead of feeding back the output of the system, we can feed back the error at the output—an estimate of the disturbance.
- This scheme is well known in the chemical process literature as "Internal Model Control."



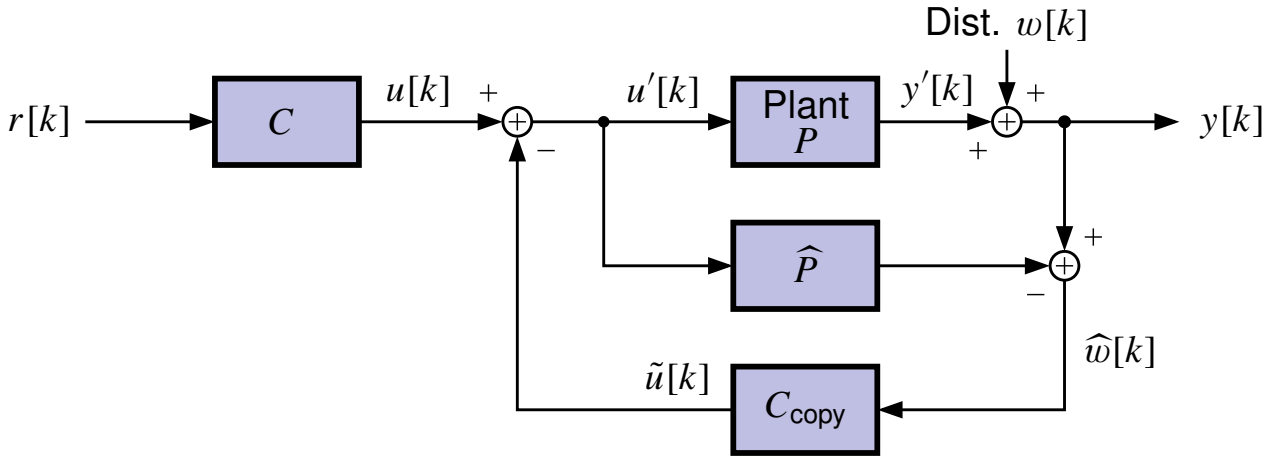
- If $\widehat{P}(z) \approx P(z)$, then the transfer function of this system is

$$Y(z)/R(z) = P(z)C(z).$$

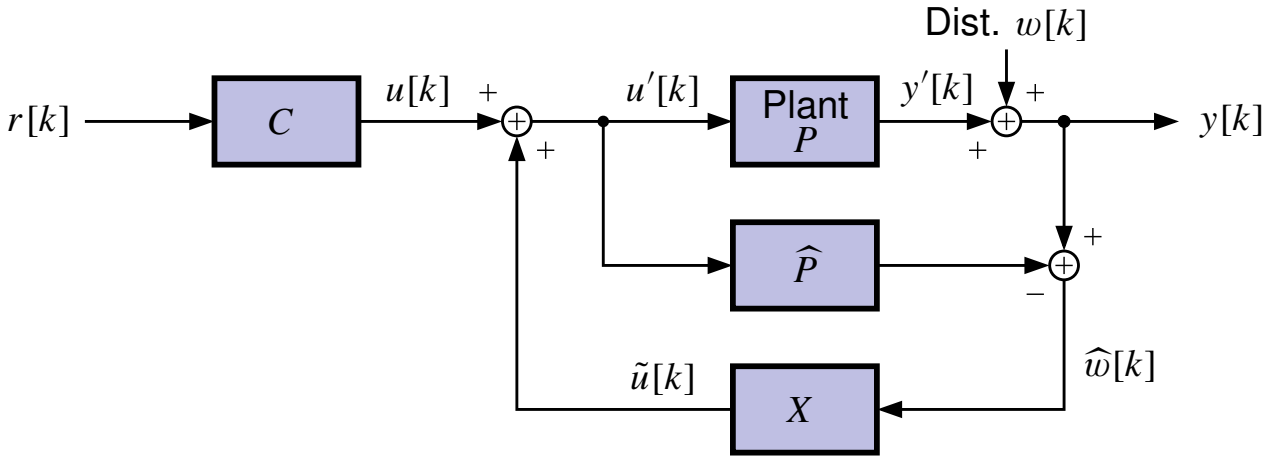
- The feedback has not changed the dynamics of the system.
- We will soon see that even this system is not suitable for use with adaptive inverse control.

Controller feedback

- We modify the last diagram slightly, assuming a linear system, by "pushing" the controller back through the summation.



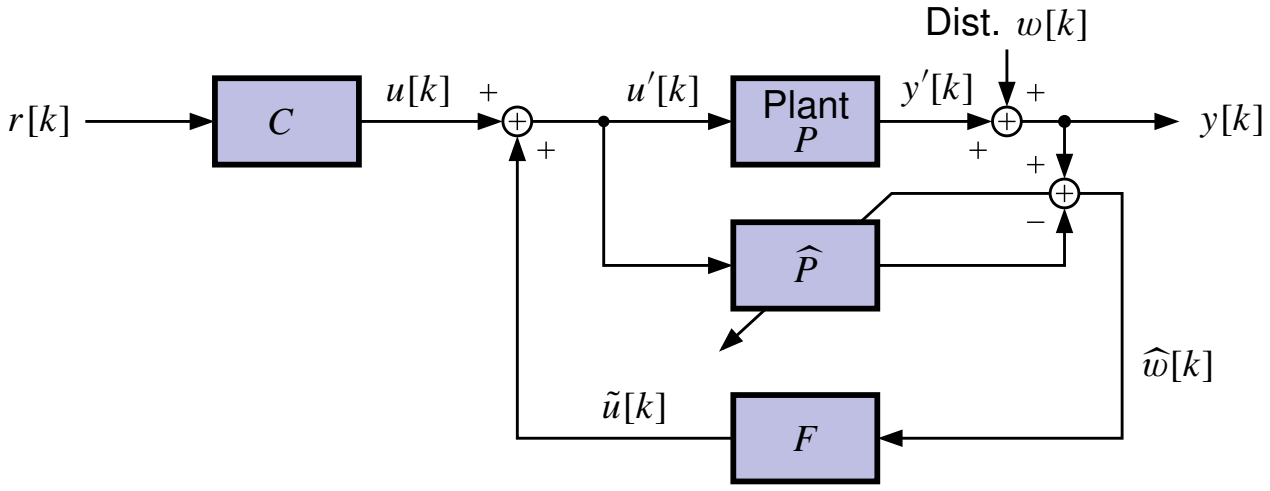
- This is almost the structure used for adaptive inverse control.
- This will work, but we can actually do better if the disturbance canceller is not equal to the controller.



7.14: Analysis of disturbance-canceling architectures

BE CAREFUL!

- We will now see why several of the previous architectures are not useful for adaptive inverse control, and why we still need to be very careful when using the last one.
- To cancel disturbance, we need to add some disturbance to the plant input—and we need to do it very carefully in order not to make the problem worse!
- Therefore, the plant input signal, $u'[k]$ must be correlated with the disturbance.
- This causes problems with adapting the plant model—the model becomes biased.
- Since the plant model is biased, the controller will be biased, and poor control will result.
- To see this, consider the very general system below, where the disturbance estimate is fed back through some filter F and added to the plant input.



- If $F = -C$ this is equivalent to internal model control, for example.
- To analyze, we assume perfect plant modeling, so $\widehat{w}[k] = w[k]$.
- The plant input is $u'[k] = u[k] + w[k] * f[k]$.
- The disturbed plant output is

$$y[k] = u[k] * p[k] + w[k] * p[k] * f[k] + w[k].$$

- Using these results, we can proceed to find the unconstrained Wiener solution for $\widehat{P}(z) = \Phi_{u'y}(z)/\Phi_{u'u'}(z)$.

$$\begin{aligned}
\phi_{u'y}[n] &= \mathbb{E}\Big[u'[k]y[k+n]\Big] \\
&= \mathbb{E}\Big[\big(u[k] + w[k] * f[k]\big)\big(u[k+n] * p[k+n] \\
&\qquad + w[k+n] * f[k+n] * p[k+n] + w[k+n]\big)\Big] \\
&= \mathbb{E}\Big[u[k]\big(u[k+n] * p[k+n]\big) \\
&\qquad + \big(w[k] * f[k]\big)\big(w[k+n] * f[k+n] * p[k+n]\big) \\
&\qquad + \big(w[k] * f[k]\big)\big(w[k+n]\big)\Big] \\
&= p[n] * \phi_{uu}[n] + f[-n] * f[n] * p[n] * \phi_{ww}[n] + f[-n] * \phi_{ww}[n]
\end{aligned}$$

$$\Phi_{u'y}(z) = P(z)\Phi_{uu}(z) + F(z^{-1})F(z)P(z)\Phi_{ww}(z) + F(z^{-1})\Phi_{ww}(z).$$

- Similarly,

$$\begin{aligned}
\Phi_{u'u'}(z) &= \Phi_{uu}(z) + F(z)F(z^{-1})\Phi_{ww}(z) \\
\widehat{P}(z) &= P(z) + \frac{F(z^{-1})\Phi_{ww}(z)}{\Phi_{uu}(z) + F(z)F(z^{-1})\Phi_{ww}(z)} \\
&= P(z) + \Delta(z).
\end{aligned}$$

- Analysis of this expression does not give much insight except to notice that there is no bias if $F = 0$.

- It is also not clear what will happen if the plant model is constrained to be causal:

$$\begin{aligned}
\widehat{P}(z) &= \frac{1}{\Phi^+_{u'u'}(z)} \left[\frac{\Phi_{u'y}(z)}{\Phi^-_{u'u'}(z)} \right]_+ \\
&= \frac{1}{\Phi^+_{u'u'}(z)} \left[\frac{P(z)\Phi_{u'u'}(z) + F(z^{-1})\Phi_{ww}(z)}{\Phi^-_{u'u'}(z)} \right]_+ \\
&= \frac{1}{\Phi^+_{u'u'}(z)} \left[P(z)\Phi^+_{u'u'}(z) + \frac{F(z^{-1})\Phi_{ww}(z)}{\Phi^-_{u'u'}(z)} \right]_+ \\
&= P(z) + \frac{1}{\Phi^+_{u'u'}(z)} \left[\frac{F(z^{-1})\Phi_{ww}(z)}{\Phi^-_{u'u'}(z)} \right]_+ \\
&= P(z) + \Delta(z).
\end{aligned}$$

- We will see two examples. One is a case where the bias $\Delta(z) = 0$ and the other is a case where $\Delta(z) \neq 0$.

7.15: Examples of disturbance-canceling architectures

Example 1

- Let $r[k]$ and $w[k]$ both be white, with unit strength.
- Let $F(z) = -z^{-\Delta}C(z)$, a delayed controller.
- Assume that C and P are both minimum phase
- Then,

$$\begin{aligned}\Phi_{ww}(z) &= 1\\ \Phi_{uu}(z) &= C(z)C(z^{-1})\\ \Phi_{u'u'}(z) &= \Phi_{uu}(z) + F(z)F(z^{-1})\Phi_{ww}(z)\\ &= C(z)C(z^{-1}) + z^{-\Delta}C(z)z^{+\Delta}C(z^{-1})\\ &= 2C(z)C(z^{-1}).\end{aligned}$$

- We factor the spectrum

$$\Phi^{+}_{u'u'}(z) = \sqrt{2}C(z) \qquad \text{and} \qquad \Phi^{-}_{u'u'}(z) = \sqrt{2}C(z^{-1}).$$

- This gives,

$$\begin{aligned}\widehat{P}(z) &= P(z) - \frac{1}{\sqrt{2}C(z)}\left[\frac{z^{\Delta}C(z^{-1})}{\sqrt{2}C(z^{-1})}\right]_{+}\\ &= P(z) - \frac{1}{2C(z)}\left[z^{\Delta}\right]_{+}.\end{aligned}$$

- So, if $\Delta > 0$ the plant model is exact.

Example 2

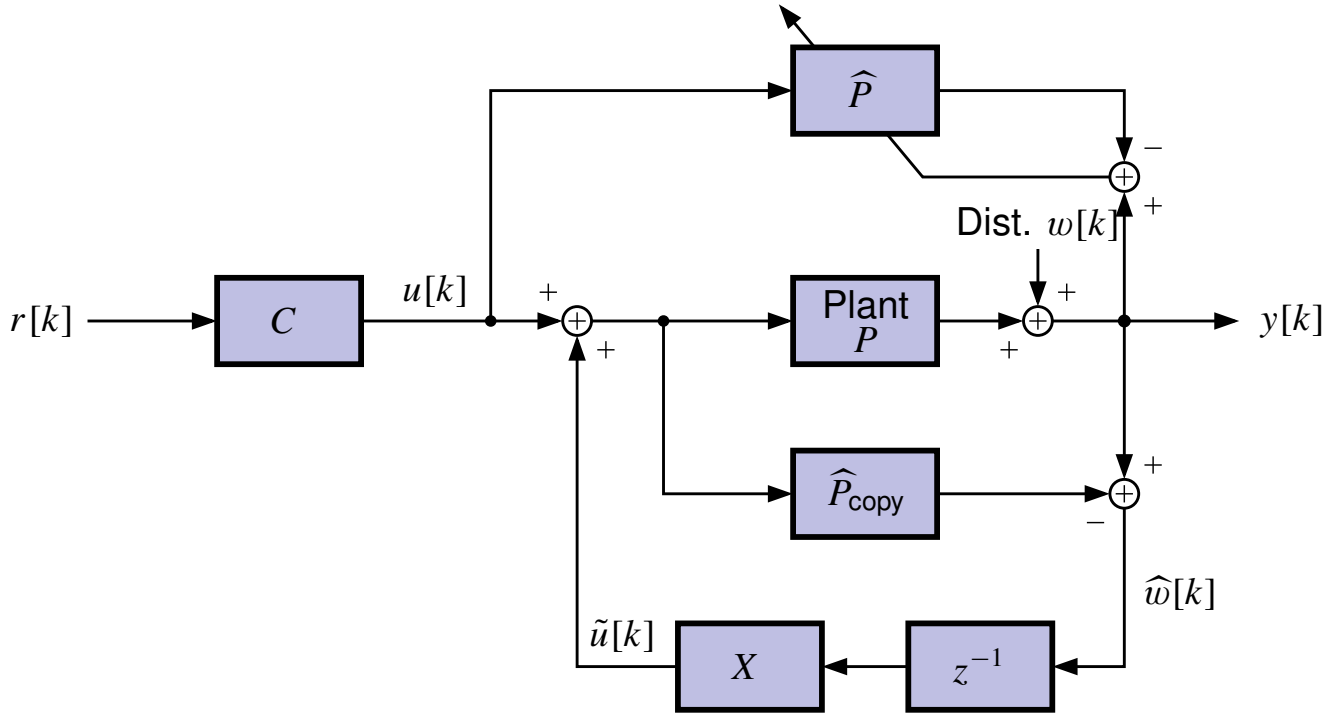
- Let $r[k]$ and $w[k]$ have the same autocorrelation functions, but let them be filtered white noise,

$$\Phi_{rr}(z) = \Phi_{ww}(z) = H(z)H(z^{-1}).$$

- Assume that H, P, and C are all causal and minimum phase. $F(z) = -z^{-\Delta}C(z)$ as before.

$$\begin{aligned}
\Phi_{uu}(z) &= C(z)C(z^{-1})H(z)H(z^{-1}) \\
\Phi_{u'u'}(z) &= \Phi_{uu}(z) + F(z)F(z^{-1})\Phi_{ww}(z) \\
&= 2C(z)C(z^{-1})H(z)H(z^{-1}) \\
\Phi^{+}_{u'u'}(z) &= \sqrt{2}C(z)H(z) \\
\Phi^{-}_{u'u'}(z) &= \sqrt{2}C(z^{-1})H(z^{-1}) \\
\widehat{P}(z) &= P(z) - \frac{1}{\sqrt{2}C(z)H(z)}\left[\frac{z^{\Delta}C(z^{-1})H(z)H(z^{-1})}{\sqrt{2}C(z^{-1})H(z^{-1})}\right]_{+} \\
&= P(z) - \frac{1}{2C(z)H(z)}\left[z^{\Delta}H(z)\right]_{+}.
\end{aligned}$$

- If Δ is small and positive, there will be a bias.
- If $\Delta \gg 0$, the bias approaches zero, but the added delay in the feedback makes disturbance canceling poor.
- So, we look for a better solution. There exist two.
  - Use either “dither scheme B” or “dither scheme C” to model the plant. These schemes work even if the plant input is correlated with the disturbance.
  - Use the following clever idea:



- The plant modeling is now done with the $u[k]$ signal and not the $u'[k]$ signal.
- $u[k]$ is "clean"—it is not correlated with the disturbance.
- The disturbed plant output is now equal to

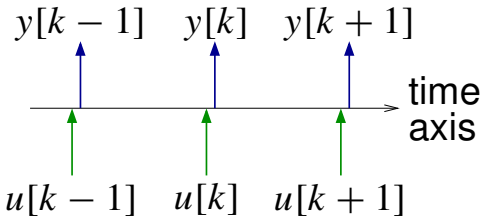
$$\begin{aligned} y[k] &= P\{u'[k]\} + w[k] \\ &= P\{u[k] + X\{w[k]\}\} + w[k] \\ &= P\{u[k]\} + P\{X\{w[k]\}\} + w[k] \\ &= P\{u[k]\} + w'\{k\}. \end{aligned}$$

- So, the plant modeling scheme of the diagram will work, and not be biased by the disturbance or the feedback.
- We have now justified the structure of the disturbance canceller.
- We now proceed to show how to adapt it.

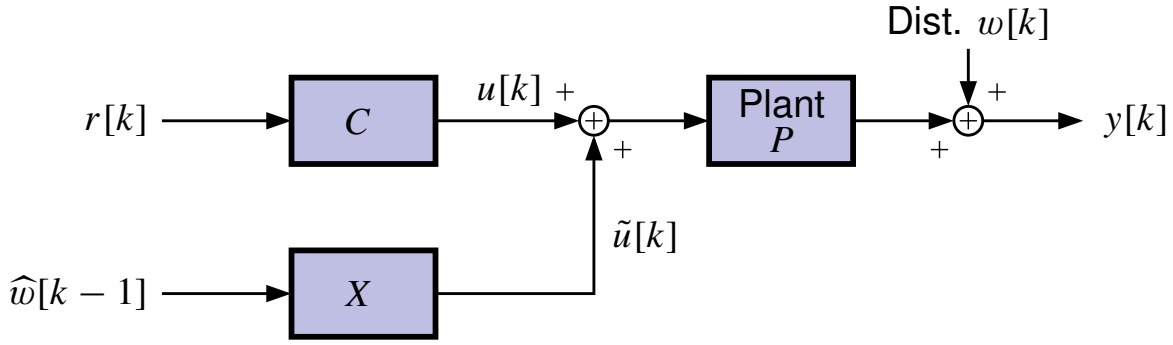
7.16: Properties of the disturbance canceler

Properties of X

- First, we consider using an adaptive linear filter to cancel disturbance.
- It is instructive to think about the job that the disturbance canceler needs to do. Consider the time line:



- The discrete-time nature of our control system means that we cannot read and cancel disturbance at exactly the same moment.
- There is a delay in the system of at least one sample.
- If there are delays in the plant, then the total delay is one sample more than the plant delay.
- This means that disturbance canceling will never be perfect.
- The following discussion shows some properties of the solution. It isn't really helpful in deriving an algorithm but it helps us understand the solution.
- We have a system like this:



- We want to train X so that $y[k] = M\{r[k]\}$.

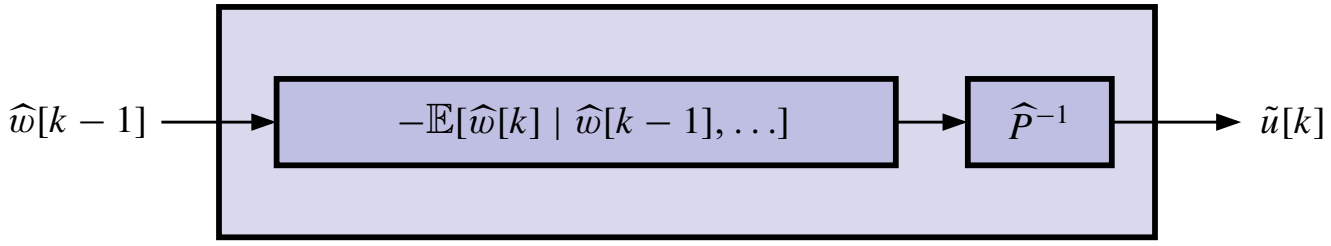
$$y[k] = w[k] + P\{C\{r[k]\} + X\{\widehat{w}[k-1]\}\}$$

$$y[k] - w[k] = P\{C\{r[k]\} + X\{\widehat{w}[k-1]\}\}$$

$$P^{-1}\{M\{r[k]\} - w[k]\} - C\{r[k]\} = X_{\text{des}}\{\widehat{w}[k-1]\}$$

$$P^{-1}\{-w[k]\} = X_{\text{des}}\{\widehat{w}[k-1]\}.$$

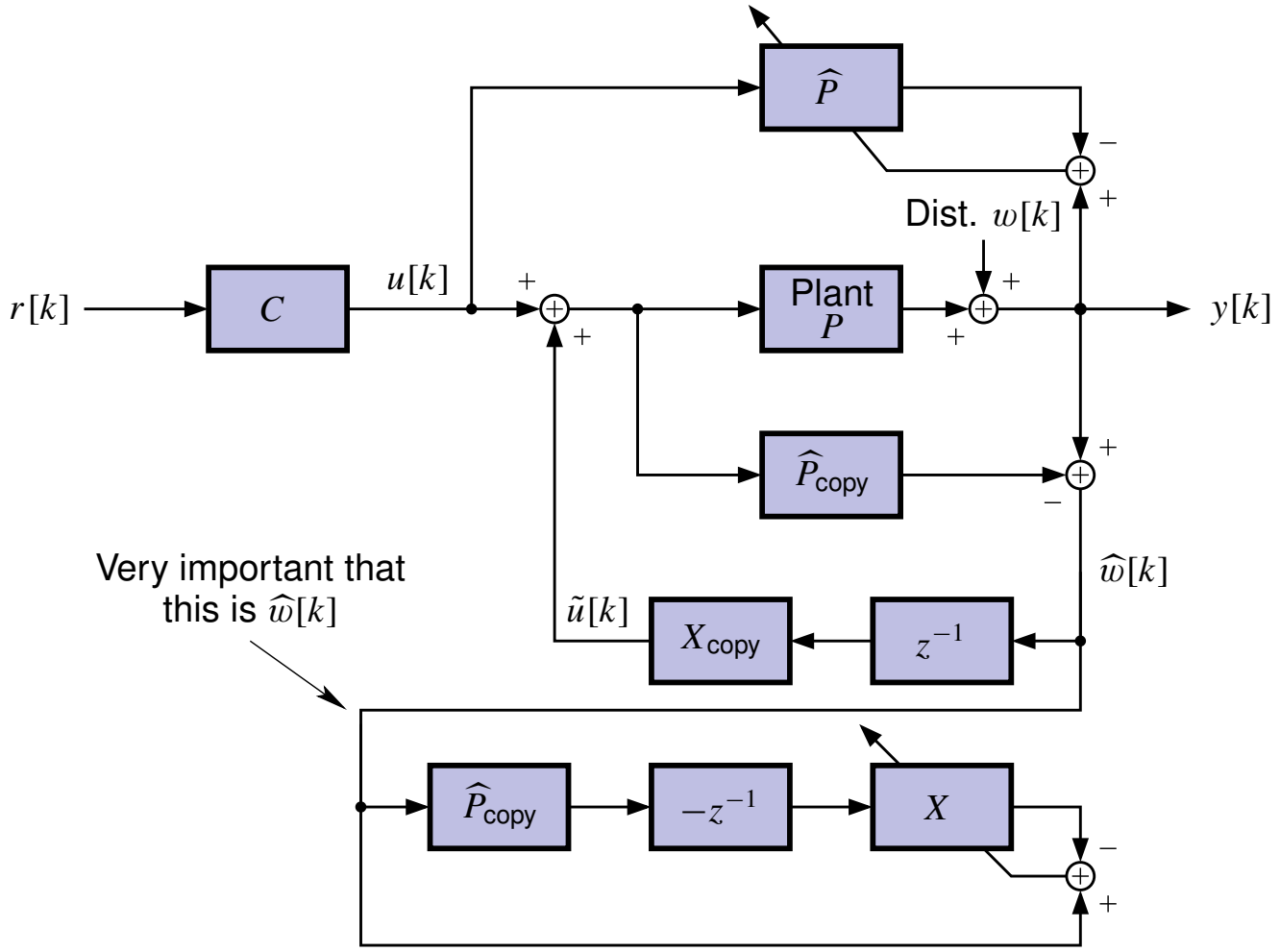
- Re-arranging, we get that the desired response for the X block can be represented as follows:



- Part of the job of the X block is to predict future disturbance values, and the other part computes the required control signal.
- Since $\mathbb{E}[w[k] \mid \widehat{w}[k-1], \ldots]$ is in general nonlinear, the optimal X is a nonlinear adaptive filter, even when the plant is linear!
- A linear filter may still be used for X, but in general, the results will be sub-optimal.

Adapting X

- Now that we see what the solution for X must look like, we consider adapting a linear X for a linear SISO plant



- The z^{-1} in the feedback loop is usually assumed, but is drawn explicitly now, in order to be very precise.
- It is caused by the time between sampling $\widehat{w}[k-1]$ and computing a disturbance cancelling control signal to cancel $w[k]$.
- At the bottom, we see a circuit that is a combination of the plant-inverse circuitry we have seen often already, and the predictor circuit.
- It does both—predict the disturbance one sample into the future, and passes it through the plant-inverse to cancel the disturbance.
- The z^{-1} in the figure must be included.

- It is possible to compute the transfer function of the system:

$$H(z) = \frac{P(z)C(z)}{1 + z^{-1}(P(z) - \widehat{P}(z))X(z)}.$$

- So, if $\widehat{P} \approx P$, then the transfer function is equal to $C(z)P(z)$, and the feedback does not bias the solution.
- The transfer function from $w[k]$ to $y[k]$ is

$$H_{yw}(z) = \frac{1 - z^{-1}\widehat{P}(z)X(z)}{1 + z^{-1}(P(z) - \widehat{P}(z))X(z)}.$$

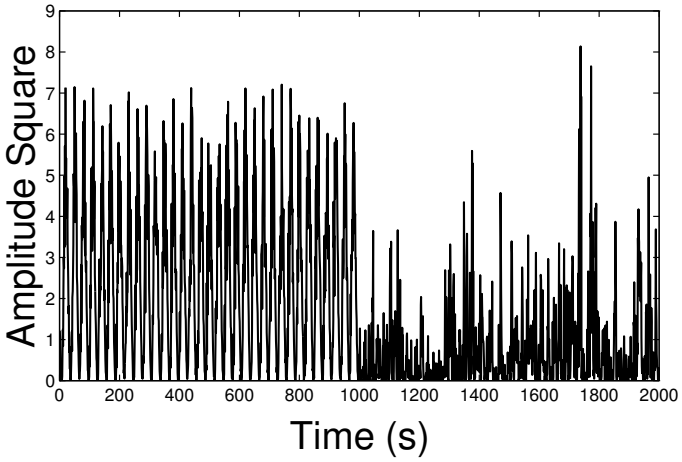
- Again, if $\widehat{P} \approx P$, and $X(z) \approx z^{+1}\widehat{P}^{-1}(z)$, (a predictor plus plant inverse, combined) then the disturbance goes away.

EXAMPLE: The minimum-phase tank, with disturbance input

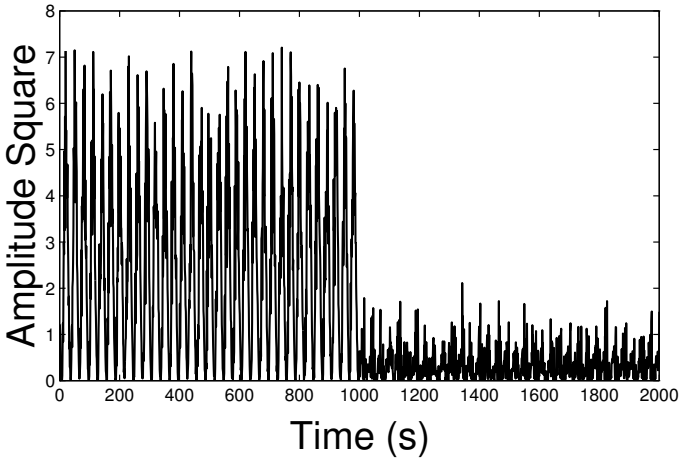
$$w[k] = \left(\frac{u[k] - 5}{18}\right) \sin(2\pi k/60 + \phi).$$

- This is interesting since (1) it is nonlinear, and (2) it is statistically dependent on $u[k]$.

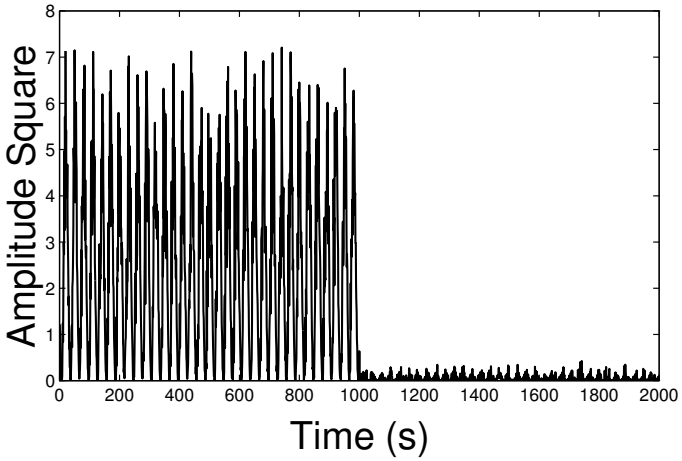
Sys. err., “closed-loop” system



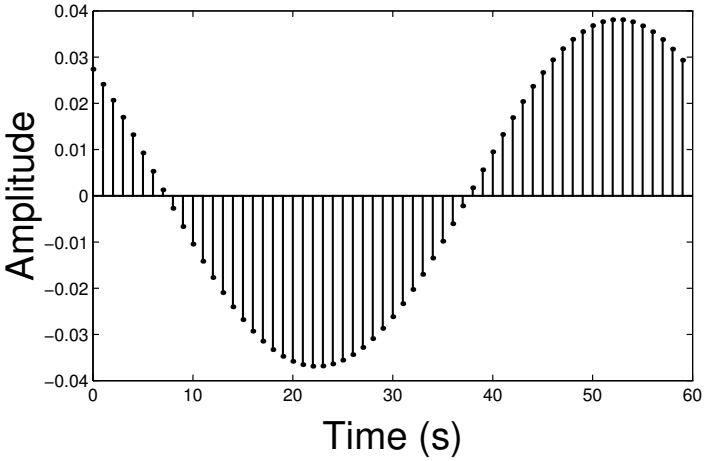
Sys. err., “ctrlr.-feedback” system



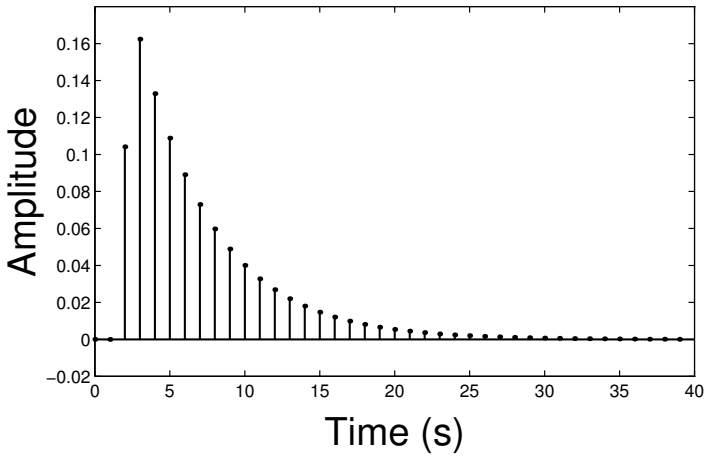
Sys. err., cancel with X



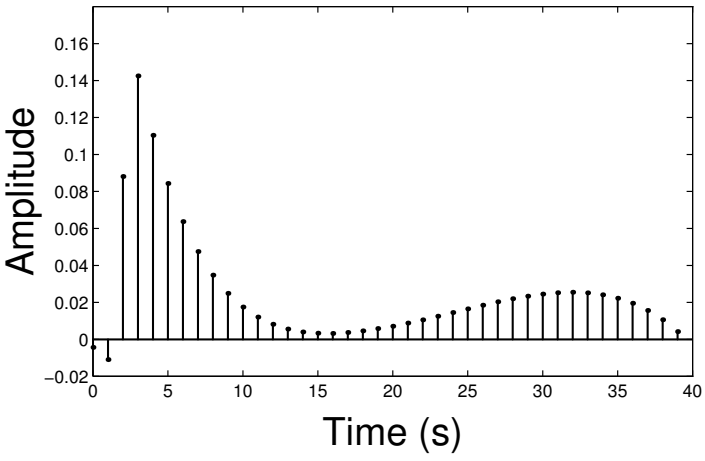
Impulse response of X



Correct impulse response of $\widehat{P}$



Impulse response of $\widehat{P}$, “closed-loop”



7.17: Analysis of AIC for MIMO linear systems

- For MIMO systems, the input to the filter is the vector signal $\left[x[k]\right]$ and the output of the filter is the vector signal $\left[y[k]\right]$.
- The desired response must have the same dimension as $\left[y[k]\right]$; it is labeled $\left[d[k]\right]$.
- A MIMO filter has a transfer-function matrix, such that (for an example 2×2 system)

$$\begin{bmatrix} Y_1(z) \\ Y_2(z) \end{bmatrix} = \begin{bmatrix} H_{11}(z) & H_{12}(z) \\ H_{21}(z) & H_{22}(z) \end{bmatrix} \begin{bmatrix} X_1(z) \\ X_2(z) \end{bmatrix}.$$

- In other words,

$$Y_1(z) = H_{11}(z)X_1(z) + H_{12}(z)X_2(z)$$

$$Y_2(z) = H_{21}(z)X_1(z) + H_{22}(z)X_2(z).$$

- In the time domain, we have

$$y_1[k] = h_{11}[k] * x_1[k] + h_{12}[k] * x_2[k]$$

$$y_2[k] = h_{21}[k] * x_1[k] + h_{22}[k] * x_2[k].$$

- This can be written as using summations

$$y_1[k] = \sum_{m=-\infty}^{\infty} h_{11}[m]x_1[k-m] + \sum_{m=-\infty}^{\infty} h_{12}[m]x_2[k-m]$$

$$y_2[k] = \sum_{m=-\infty}^{\infty} h_{21}[m]x_1[k-m] + \sum_{m=-\infty}^{\infty} h_{22}[m]x_2[k-m].$$

- The summations may be combined such that

$$y_1[k] = \sum_{m=-\infty}^{\infty} (h_{11}[m]x_1[k-m] + h_{12}[m]x_2[k-m])$$

$$y_2[k] = \sum_{m=-\infty}^{\infty} \left(h_{21}[m]x_1[k-m] + h_{22}[m]x_2[k-m]\right).$$

- Furthermore, this can be written in matrix form:

$$\begin{bmatrix} y_1[k] \\ y_2[k] \end{bmatrix} = \sum_{m=-\infty}^{\infty} \begin{bmatrix} h_{11}[m] & h_{12}[m] \\ h_{21}[m] & h_{22}[m] \end{bmatrix} \begin{bmatrix} x_1[k-m] \\ x_2[k-m] \end{bmatrix},$$

or

$$\big[y[k]\big] = \sum_{m=-\infty}^{\infty} \big[h[m]\big]\big[x[k-m]\big].$$

Vector autocorrelation and crosscorrelation

- Before deriving the MIMO Wiener proof, we define the following quantities for vector signals:

AUTOCORRELATION: We define the autocorrelation function of the vector signal $[x[k]]$ to be

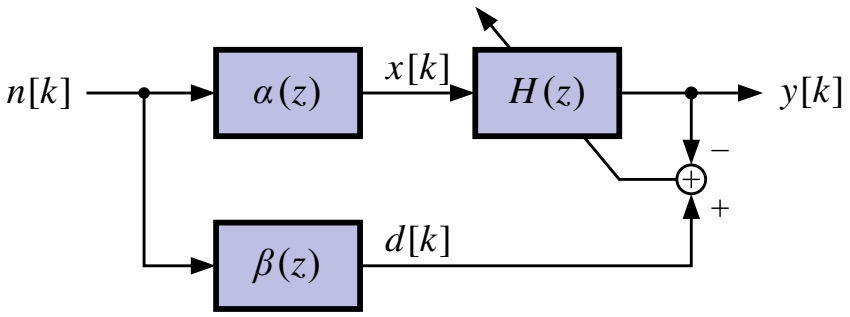
$$[\phi_{xx}[m]] = \mathbb{E}\Big[[x[k]][x^T[k+m]]\Big].$$

CROSSCORRELATION: We define the crosscorrelation between two vector signals $[x[k]]$ and $[y[k]]$ to be

$$[\phi_{xy}[m]] = \mathbb{E}\Big[[x[k]][y^T[k+m]]\Big] = [\phi_{yx}^T[-m]].$$

EXAMPLE: Correlations in a generic adaptive context

- Many times, we must compute the quantities $[\Phi_{xd}(z)]$ and $[\Phi_{xx}(z)]$ for block diagrams like the following figure



- To find $[\Phi_{xd}(z)]$ and $[\Phi_{xx}(z)]$ we first compute $[\phi_{xd}[m]]$.

$$[\phi_{xd}[m]] = \mathbb{E}\left[\big[x[k]\big]\big[d[k+m]\big]^T\right]$$

$$\big[x[k]\big] = \sum_{l=-\infty}^{\infty} \big[\alpha[l]\big]\big[n[k-l]\big]$$

$$\big[d[k]\big] = \sum_{i=-\infty}^{\infty} \big[\beta[i]\big]\big[n[k-i]\big]$$

$$[\phi_{xd}[m]] = \mathbb{E}\left[\sum_{l=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} \big[\alpha[l]\big]\big[n[k-l]\big]\big[n[k+m-i]\big]^T\big[\beta[i]\big]^T\right]$$

$$= \sum_{l=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} \big[\alpha[l]\big]\,[\phi_{nn}[m+l-i]]\,\big[\beta[i]\big]^T.$$

- This may be unwrapped by taking the z-transform.

$$[\Phi_{xd}(z)] = \sum_{l=-\infty}^{\infty} \big[\alpha[l]\big]z^{+l} \sum_{m=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} [\phi_{nn}[m+l-i]]z^{-(m+l-i)}\big[\beta[i]\big]^T z^{-i}$$

$$= \sum_{l=-\infty}^{\infty} \big[\big[\alpha[l]\big]z^{+l}\big]\left[\sum_{i=-\infty}^{\infty} \big[\big[\beta[i]\big]z^{-i}\big]\sum_{m=-\infty}^{\infty} \phi_{nn}^T[m+l-i]z^{-(m+l-i)}\right]^T$$

$$= \sum_{l=-\infty}^{\infty} \big[\big[\alpha[l]\big]z^{+l}\big]\,\big[B(z)\Phi_{nn}^T(z)\big]^T$$

$$= [A(z^{-1})][\Phi_{nn}(z)][B(z)^T].$$

- By extension, $[\Phi_{xx}(z)] = [A(z^{-1})][\Phi_{nn}(z)][A(z)^T]$.

7.18: Unconstrained Wiener solution for MIMO systems

- Note that $\big[e[k]\big] = \big[d[k]\big] - \big[y[k]\big]$.
- We wish to minimize the expected 2-norm of $\big[e[k]\big]$:

$$\big\|\big[e[k]\big]\big\|^2 = \big[e[k]\big]^T\big[e[k]\big]$$

$$= \big[d[k]\big]^T\big[d[k]\big] + \sum_{l=-\infty}^{\infty}\sum_{u=-\infty}^{\infty}\Big(\big[h[l]\big]\big[x[k-l]\big]\Big)^T\big[h[u]\big]\big[x[k-u]\big]$$

$$-2\sum_{l=-\infty}^{\infty}\Big(\big[h[l]\big]\big[x[k-l]\big]\Big)^T\big[d[k]\big]$$

$$= \big[d[k]\big]^T\big[d[k]\big] + \sum_{l=-\infty}^{\infty}\sum_{u=-\infty}^{\infty}\big[x[k-l]\big]^T\big[h[l]\big]^T\big[h[u]\big]\big[x[k-u]\big]$$

$$-2\sum_{l=-\infty}^{\infty}\big[x[k-l]\big]^T\big[h[l]\big]^T\big[d[k]\big].$$

- Each composite quantity is a scalar, so $(\cdot) = \mathrm{Tr}(\cdot)$. Furthermore, $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ for all $A \in \mathbb{C}^{n\times m}$ and $B \in \mathbb{C}^{m\times n}$.

$$\big\|\big[e[k]\big]\big\|^2 = \big[d[k]\big]^T\big[d[k]\big]$$

$$+\sum_{l=-\infty}^{\infty}\sum_{u=-\infty}^{\infty}\mathrm{Tr}\Big(\big[h[l]\big]^T\big[h[u]\big]\big[x[k-u]\big]\big[x[k-l]\big]^T\Big)$$

$$-2\sum_{l=-\infty}^{\infty}\mathrm{Tr}\Big(\big[h[l]\big]^T\big[d[k]\big]\big[x[k-l]\big]^T\Big)$$

$$\mathbb{E}\Big(\big\|\big[e[k]\big]\big\|^2\Big) = \mathbb{E}\Big(\big[d[k]\big]^T\big[d[k]\big]\Big)$$

$$+\sum_{l=-\infty}^{\infty}\sum_{u=-\infty}^{\infty}\mathrm{Tr}\Big(\big[h[l]\big]^T\big[h[u]\big]\mathbb{E}\Big(\big[x[k-u]\big]\big[x[k-l]\big]^T\Big)\Big)$$

$$-2\sum_{l=-\infty}^{\infty}\text{Tr}\left(\left[h[l]\right]^T\mathbb{E}\left(\underbrace{\left[d[k]\right]\left[x[k-l]\right]^T}_{\phi_{dx}[-l]=\phi_{xd}[l]^T}\right)\right)$$

$$= \text{Tr}\left([\phi_{dd}[0]]\right) + \sum_{l=-\infty}^{\infty}\sum_{u=-\infty}^{\infty}\text{Tr}\left(\left[h[l]\right]^T\left[h[u]\right]\left[\phi_{xx}[l-u]\right]\right) - 2\sum_{l=-\infty}^{\infty}\text{Tr}\left(\left[h[l]\right]^T\left[\phi_{xd}[l]\right]^T\right).$$

- To find the optimal filter weights, we take the derivative of $\mathbb{E}\left(\|e[k]\|^2\right)$ and set it to zero. The following identities help:

$$\frac{\partial\text{Tr}(AX^TB)}{\partial X} = BA \qquad \frac{\partial\text{Tr}(AXB)}{\partial X} = A^TB^T$$

$$\frac{\partial\text{Tr}(AXBX^T)}{\partial X} = A^TXB^T + AXB.$$

- So,

$$\frac{\partial\mathbb{E}\left(\left\|\left[e[k]\right]\right\|^2\right)}{\partial h_j} = 0 + \frac{\partial(\sum\sum)}{\partial h_j} - 2\frac{\partial(\sum)}{\partial h_j} \equiv 0.$$

- Now,

$$\frac{\partial(\sum)}{\partial h_j} = \frac{\partial\text{Tr}\left(\left[h[j]\right]^T\left[\phi_{xd}[j]\right]^T\right)}{\partial h_j} = \left[\phi_{xd}[j]\right]^T,$$

since all of the summation terms not containing $\left[h[j]\right]$ go away under the differentiation, and we let $A = I$ and $B = \left[\phi_{xd}[j]\right]^T$.

- Also, the remaining partial may be computed for different values of summation indices. If $l \neq j$ and $u \neq j$

$$\frac{\partial(\sum\sum)}{\partial h_j} = 0.$$

- If $l = j$ and $u \neq j$ then

$$\frac{\partial(\sum\sum)}{\partial h_j} = \frac{\partial \mathrm{Tr}\left(\left[h[j]\right]^T \left[h[u]\right]\left[\phi_{xx}[j-u]\right]\right)}{\partial h_j}$$
$$= \left[h[u]\right]\left[\phi_{xx}[j-u]\right].$$

- If $l \neq j$ and $u = j$ then

$$\frac{\partial(\sum\sum)}{\partial h_j} = \frac{\partial \mathrm{Tr}\left(\left[h[l]\right]^T \left[h[j]\right]\left[\phi_{xx}[l-j]\right]\right)}{\partial h_j}$$
$$= \left[h[l]\right]\left[\phi_{xx}[l-j]\right]^T = \left[h[l]\right]\left[\phi_{xx}[j-l]\right].$$

- If $l = j$ and $u = j$ then

$$\frac{\partial(\sum\sum)}{\partial h_j} = \frac{\partial \mathrm{Tr}\left(\left[h[j]\right]^T \left[h[j]\right]\left[\phi_{xx}[0]\right]\right)}{\partial h_j}$$
$$= \frac{\partial \mathrm{Tr}\left(\left[h[j]\right]\left[\phi_{xx}[0]\right]\left[h[j]\right]^T\right)}{\partial h_j} = 2\left[h[l]\right]\left[\phi_{xx}[0]\right].$$

- Putting all of the above together, we have

$$\frac{\partial \mathbb{E}\left(\left\|\left[e[k]\right]\right\|^2\right)}{\partial h_j} = 2\sum_{l=-\infty}^{\infty}\left[h[l]\right]\left[\phi_{xx}[j-l]\right] - 2\left[\phi_{xd}[j]\right]^T \equiv 0.$$

- Take z-transforms of the above, realizing that the summation is a convolution,

$$\left[H(z)\right]\left[\Phi_{xx}(z)\right] = \left[\Phi_{xd}(z)\right]^T$$
$$\left[H(z)\right] = \left[\Phi_{xd}(z)\right]^T\left[\Phi_{xx}(z)\right]^{-1}.$$

- This simplifies to the familiar SISO solution when all terms are scalars: $H(z) = \dfrac{\Phi_{xd}(z)}{\Phi_{xx}(z)}$.

7.19: Causal Wiener solution for MIMO systems

- In order to use the unconstrained Wiener filter solution to find the causal Wiener filter solution, we can use the causal part of the unconstrained solution if we first "whiten" the input to that causal filter.
- We want to whiten x_k and to do so we suppose that x_k was generated via filtering white noise through a minimum-phase filter $[G(z)]$.
- Then, (using the autocorrelation result in the next section), we can write

$$\Phi_{xx}(z) = [G(z^{-1})][I][G(z)]^T = [G(z^{-1})][G(z)]^T.$$

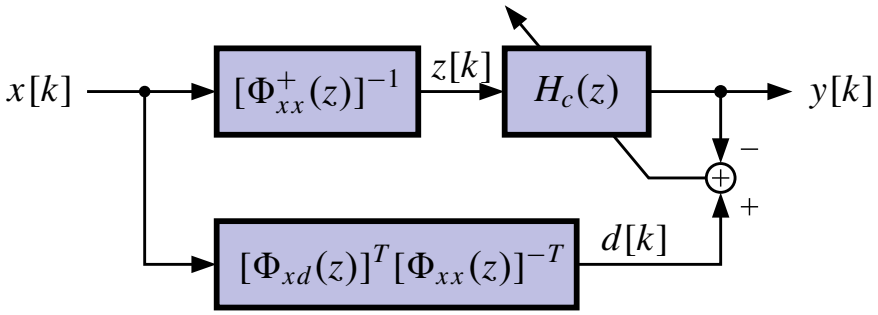
- Let the whitening filter be $H_w(z) = [G(z)]^{-1}$. Note that if $[G(z)]$ is minimum-phase,

$$\Phi_{xx}^{+}(z) = [G(z)]$$

$$\Phi_{xx}^{-}(z) = [G(z^{-1})].$$

  and $\Phi_{xx}(z) = \Phi_{xx}^{-}(z)\Phi_{xx}^{+}(z)^T$.

- Now, consider the following figure:



- Recall that the autocorrelation of $z[k]$ can be found as

$$\Phi_{zz}(z) = A(z^{-1})\Phi_{xx}(z)B(z)^T$$

$$= [\Phi_{xx}^{+}(z^{-1})]^{-1}[\Phi_{xx}^{-}(z)][\Phi_{xx}^{+}(z)^T][\Phi_{xx}^{+}(z)]^{-T}$$

$$= [\Phi_{xx}^{-}(z)]^{-1}[\Phi_{xx}^{-}(z)][\Phi_{xx}^{+}(z)^T][\Phi_{xx}^{+}(z)]^{-T} = I,$$

so, $z[k]$ is indeed white.

- We can also verify that the cross-correlation between $x[k]$ and $d[k]$ is what we expect

$$\begin{aligned}\Phi_{xd}(z) &= A(z^{-1})\Phi_{xx}(z)B(z)^T \\ &= I\ \Phi_{xx}(z)\left(\Phi_{xd}^T(z)\Phi_{xx}^{-T}(z)\right)^T \\ &= \Phi_{xx}(z)\Phi_{xx}^{-1}(z)\Phi_{xd}(z) = \Phi_{xd}(z).\end{aligned}$$

- So, we can also find the cross-correlation between $z[k]$ and $d[k]$

$$\begin{aligned}\Phi_{zd}(z) &= A(z^{-1})\Phi_{xx}(z)B(z)^T \\ &= [\Phi_{xx}^{+}(z^{-1})][\Phi_{xx}(z)]\left([\Phi_{xd}^T(z)][\Phi_{xx}^{-T}(z)]\right)^T \\ &= [\Phi_{xx}^{+}(z^{-1})][\Phi_{xd}(z)] = [\Phi_{xx}^{-}(z)][\Phi_{xd}(z)].\end{aligned}$$

- Then, the unconstrained Wiener solution for $H_c(z)$ is

$$\begin{aligned}H_c^*(z) &= [\Phi_{zd}(z)]^T \underbrace{[\Phi_{zz}(z)]^{-1}}_{I} \\ &= [\Phi_{xd}(z)]^T[\Phi_{xx}^{-}(z)]^T.\end{aligned}$$

- The constrained Wiener solution is the causal part of $H_c(z)$ preceded by the whitening filter, or

$$[H(z)]_{\text{causal}} = \left[[\Phi_{xd}(z)]^T[\Phi_{xx}^{-}(z)]^{-T}\right]_{+}\left[\Phi_{xx}^{+}(z)\right]^{-1}.$$

7.20: Matrix RLS instead of LMS to promote speed

Matrix RLS

- The *Recursive Least Squares* (RLS) algorithm is a fast and efficient method for adapting the weights of an adaptive linear filter.
- Matrix-RLS is used to adapt a linear MIMO filter, whereas regular RLS adapts a SISO filter.
- Matrix-RLS may be derived in the same way as standard RLS.
- The steps are omitted here. The final form of a numerically-stable version of the algorithm is:

$$\begin{aligned}
\pi(k) &= X(k)^T \Phi_{xx}^{-1}(k-1) \\
r(k) &= \frac{1}{\lambda + \pi(k)X(k)} \\
K(k) &= r(k)\pi(k) \\
\zeta(k) &= d(k) - \mathbf{W}(k-1)X(k) \\
\mathbf{W}(k) &= \mathbf{W}(k-1) + \zeta(k)K(k) \\
\Phi_{xx}^{-1}(k) &= \frac{1}{\lambda}\text{tri}\left\{\Phi_{xx}^{-1}(k-1) - \pi(k)^T K(k)\right\},
\end{aligned}$$

where

- $\text{tri}\{\cdot\}$ takes the upper (or lower) triangular part of a matrix and replicates it in the lower (or upper) part to preserve symmetry.
- λ is a forgetting constant, and is set slightly less than 1.
- $X(k)$ is the tap-delay-line input to the filter at time k, such that the filter output is $\mathbf{W}(k)X(k)$.
- $\Phi_{xx}^{-1}(k)$ is initialized to a diagonal matrix with large (order of 10,000) entries.

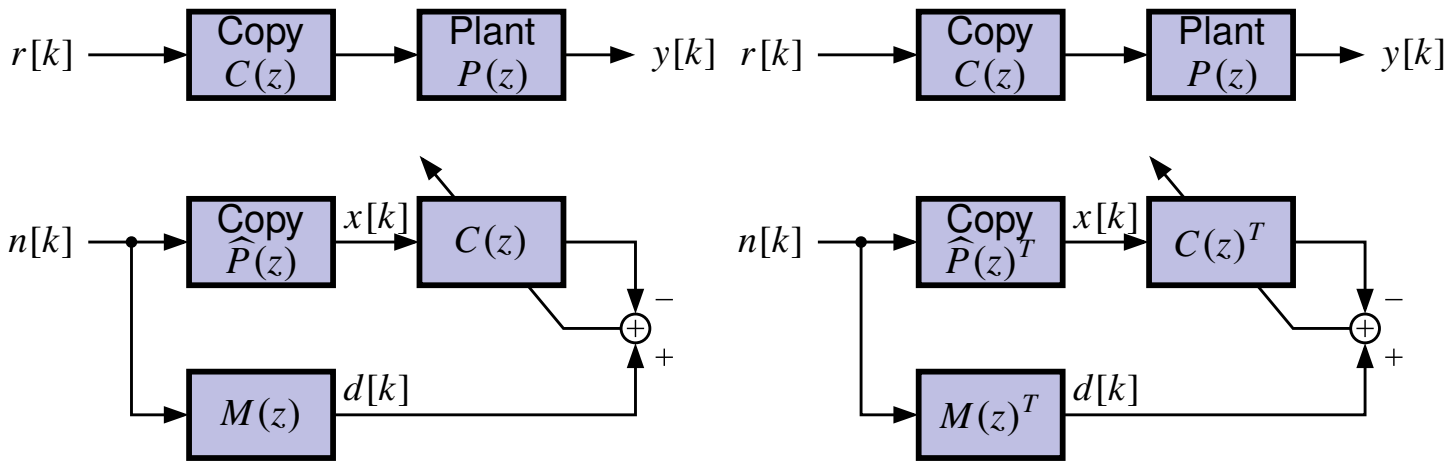
Convergence

- There are two convergence issues to deal with.
- First, convergence of the plant model, and then convergence of the controller/disturbance canceler.
- The plant model needs to converge before either the controller or disturbance canceler can converge.
- Here we assume that matrix-RLS is used to adapt the two filters.
- RLS converges in about two times as many iterations as there are taps in the FIR filter.
- So, for example, if a matrix-FIR filter has five taps per sub-filter, convergence is achieved in about ten iterations.
- Similarly, convergence of the controller or disturbance canceler occurs in about twice as many iterations as there are taps in each filter.
- However, since the controller- and canceler adaptation process is done offline, it can be done quickly in the background and entire system convergence occurs can occur in about twice as many iterations as there are taps in the plant-model FIR filter.
- A note re. using matrix-LMS. The convergence speed of matrix-LMS is related to the eigenvalue spread of the autocorrelation matrix R of the filter input.
- The eigenvalue spread in the R matrix in the example presented next is fourteen orders of magnitude!
- Matrix-LMS converges very slowly.

7.21: The fast "transpose" method

Feedforward control

- While linear SISO systems have transfer functions, linear MIMO systems have transfer function matrices, denoted for example as $[P(z)]$.
- Matrix multiplication is not in general commutative; therefore, $[P(z)][C(z)] \neq [C(z)][P(z)]$ and we can not use the same simple method as with the SISO system.
- Here, we present a very simple and fast method to adapt a linear MIMO controller.
- It uses the fact that $[[P(z)][C(z)]]^T = [C(z)]^T[P(z)]^T$.
- In the left figure below, we see a diagram suitable for adapting a SISO controller.
- In the right diagram, we see a system suitable for adapting a MIMO controller.

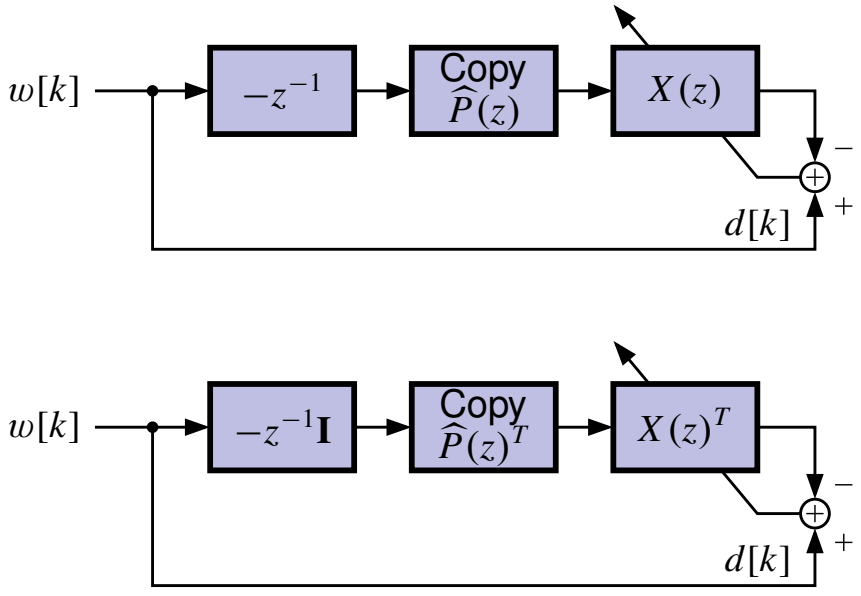


- The entire operation depends on being able to take the "transpose" of an adaptive filter representing a transfer function matrix.

- These filters are actually stored as impulse-response matrices, and the transpose operation is simply a re-organization of the components of the impulse-response matrices.
- The mechanics of taking a filter transpose are discussed later.
- As with the SISO linear case, convergence using the RLS algorithm occurs within twice as many iterations as there are taps in the longest impulse response in the MIMO controller filter C.
- This is an improvement of many orders of magnitude when compared with the other known cited methods.

Disturbance canceling

- We can adapt a disturbance canceler for a linear MIMO system using the same equation as for the SISO case, and adopting the transpose method used when adapting C for the MIMO system.
- The block diagram of the adaptation method is shown below: SISO on top; MIMO on bottom.



7.22: Finding the filter transpose

- To train the controller, we need the transposed filter $[\widehat{P}(z)]^T$.
- The weight matrix for this filter is not the same as $[\mathbf{W}_{\widehat{P}}]^T$.
- To find the correct weight matrix for an arbitrary transposed filter $[H(z)]^T$ consider first the weight matrix for some arbitrary filter $[H(z)]$.
- For the sake of example we will assume $[H(z)]$ has two inputs, two outputs and $N = 4$ so that the impulse response is five samples long.
- Then, the H-matrix has entries as shown below

$[h_{11}]_0$	$[h_{12}]_0$	$[h_{11}]_1$	$[h_{12}]_1$	$[h_{11}]_2$	$[h_{12}]_2$	$[h_{11}]_3$	$[h_{12}]_3$	$[h_{11}]_4$	$[h_{12}]_4$
$[h_{21}]_0$	$[h_{22}]_0$	$[h_{21}]_1$	$[h_{22}]_1$	$[h_{21}]_2$	$[h_{22}]_2$	$[h_{21}]_3$	$[h_{22}]_3$	$[h_{21}]_4$	$[h_{22}]_4$

- Four impulse responses are embedded in H: h_{11}, h_{12}, h_{21} and h_{22}.
- When we take the z-transform and make the transfer-function matrix for this filter, we get

$$[H(z)] = \begin{bmatrix} H_{11}(z) & H_{12}(z) \\ H_{21}(z) & H_{22}(z) \end{bmatrix}$$

which has transpose

$$[H(z)]^T = \begin{bmatrix} H_{11}(z) & H_{21}(z) \\ H_{12}(z) & H_{22}(z) \end{bmatrix}.$$

- So, the transpose of the weight filter has entries as shown below

$[h_{11}]_0$	$[h_{21}]_0$	$[h_{11}]_1$	$[h_{21}]_1$	$[h_{11}]_2$	$[h_{21}]_2$	$[h_{11}]_3$	$[h_{21}]_3$	$[h_{11}]_4$	$[h_{21}]_4$
$[h_{12}]_0$	$[h_{22}]_0$	$[h_{12}]_1$	$[h_{22}]_1$	$[h_{12}]_2$	$[h_{22}]_2$	$[h_{12}]_3$	$[h_{22}]_3$	$[h_{12}]_4$	$[h_{22}]_4$

- Computing the transpose of the filter involves a simple reordering of the weights in the weight matrix of the filter.
- The following MATLAB code performs the transpose operation.

```
function fout=ftranspose(fin,numtaps);
% This function computes the filter transpose of the input
% MIMO linear filter. The input filter is a matrix of
% dimension (numout)x(numin*numtaps).
%
% e.g. The input filter has two outputs, three inputs and
% two taps.
%
%   +--------+--------+--------+--------+--------+--------+
%   | f11(1) | f12(1) | f13(1) | f11(2) | f12(2) | f13(2) |
%   +--------+--------+--------+--------+--------+--------+
%   | f21(1) | f22(1) | f23(1) | f21(2) | f22(2) | f23(2) |
%   +--------+--------+--------+--------+--------+--------+
%
% The algorithm first reshapes into a 3-D matrix:
%
%   +--------+--------+--------+    +--------+--------+--------+
%   | f11(1) | f12(1) | f13(1) |    | f11(2) | f12(2) | f13(2) |
%   +--------+--------+--------+    +--------+--------+--------+
%   | f21(1) | f22(1) | f23(1) |    | f21(2) | f22(2) | f23(2) |
%   +--------+--------+--------+    +--------+--------+--------+
%
% Then, it reorders the dimensions:
%
%   +--------+--------+    +--------+--------+
%   | f11(1) | f21(1) |    | f11(2) | f21(2) |
%   +--------+--------+    +--------+--------+
%   | f12(1) | f22(1) |    | f12(2) | f22(2) |
%   +--------+--------+    +--------+--------+
%   | f13(1) | f23(1) |    | f13(2) | f23(2) |
%   +--------+--------+    +--------+--------+
%
```



```
% Then it puts everything back together:
%
%   +--------+--------+--------+--------+
%   | f11(1) | f21(1) | f11(2) | f21(2) |
%   +--------+--------+--------+--------+
%   | f12(1) | f22(1) | f12(2) | f22(2) |
%   +--------+--------+--------+--------+
%   | f13(1) | f23(1) | f13(2) | f23(2) |
%   +--------+--------+--------+--------+
%
% Copyright (c) 2000. Dr. Gregory L. Plett. Please distribute
% with this copyright message.
%
[numout numin]=size(fin); numin=numin/numtaps;
fout=reshape(permute(reshape(fin,[numout,numin,numtaps]), [2 1 3]), [numin
  numout*numtaps]);
```

7.23: Wiener solution for MIMO AIC

- There are three cases to consider when computing the Wiener solution for the controller weight matrix when the plant is linear MIMO.
  - The plant has more outputs than inputs;
  - The plant has fewer outputs than inputs;
  - The plant has an equal number of outputs and inputs.
- An important mathematical result relating to z-transforms of impulse response matrices must first be developed.
- Assume that unit intensity white noise is filtered by $H(z)$. A prior result is that the z-transform of the autocorrelation of the filter output is then $\Phi_{yy}(z) = H(z^{-1})H(z)^T$.
- Since autocorrelation functions are symmetric, then we have the important results which hold for any $H(z)$

$$H(z^{-1})H(z)^T = H(z)H(z^{-1})^T$$

and (now letting the filter be $H(z)^T$)

$$H(z^{-1})^T H(z) = H(z)^T H(z^{-1}).$$

MORE OUTPUTS THAN INPUTS: If the plant has more outputs than inputs, then $\widehat{P}(z)^T \widehat{P}(z)$ is generally invertible.

- The solution for the controller will be

$$\widehat{C}^T(z) = [\Phi_{xd}(z)]^T [\Phi_{xx}(z)]^{-1}$$

$$\widehat{C}^T(z)[\Phi_{xx}(z)] = [\Phi_{xd}(z)]^T$$

$$\widehat{C}^T(z)\left(\widehat{P}^T(z^{-1})\widehat{P}(z)\right) = M^T(z)\widehat{P}(z^{-1}).$$

- Transposing both sides to write in terms of $\widehat{C}(z)$

$$\widehat{P}^T(z)\widehat{P}(z^{-1})\widehat{C}(z) = \widehat{P}^T(z^{-1})M(z).$$

- We need to get rid of the "z^{-1}" terms. Multiply on the left by $\widehat{P}(z)$

$$\widehat{P}(z)\underbrace{\widehat{P}^T(z)\widehat{P}(z^{-1})}_{\widehat{P}^T(z^{-1})\widehat{P}(z)}\widehat{C}(z) = \underbrace{\widehat{P}(z)\widehat{P}^T(z^{-1})}_{\widehat{P}(z^{-1})\widehat{P}^T(z)}M(z)$$

$$\underbrace{\widehat{P}(z)\widehat{P}^T(z^{-1})}_{\widehat{P}(z^{-1})\widehat{P}^T(z)}\widehat{P}(z)\widehat{C}(z) = \widehat{P}(z^{-1})\widehat{P}^T(z)M(z)$$

$$\widehat{P}(z^{-1})\widehat{P}^T(z)\widehat{P}(z)\widehat{C}(z) = \widehat{P}(z^{-1})\widehat{P}^T(z)M(z).$$

- Multiply on the left by $\left[\widehat{P}(z^{-1})^T\widehat{P}(z^{-1})\right]^{-1}\widehat{P}(z^{-1})^T$

$$\left[\widehat{P}(z^{-1})^T\widehat{P}(z^{-1})\right]^{-1}\widehat{P}(z^{-1})^T\widehat{P}(z^{-1})\widehat{P}^T(z)\widehat{P}(z)\widehat{C}(z)$$
$$= \left[\widehat{P}(z^{-1})^T\widehat{P}(z^{-1})\right]^{-1}\widehat{P}(z^{-1})^T\widehat{P}(z^{-1})\widehat{P}^T(z)M(z).$$

- When terms cancel we are left with

$$\widehat{P}(z)^T\widehat{P}(z)C(z) = \widehat{P}(z)^TM(z)$$
$$C^{(opt)}(z) = \left[\widehat{P}(z)^T\widehat{P}(z)\right]^{-1}\widehat{P}(z)^TM(z),$$

  which is the pseudo-inverse of $\widehat{P}(z)$ in cascade with the reference model.

- This is the desired solution (although it would be better if the solution had $[P(z)^TP(z)]^{-1}P(z)^TM(z)$ instead).

MORE INPUTS THAN OUTPUTS: If the plant has more inputs than outputs, some matrix cancellations in the above method will no longer be valid.

- Instead, $\widehat{P}(z)\widehat{P}(z)^T$ will generally be invertible. Again, we start with

$$\underbrace{\widehat{P}(z)^T\widehat{P}(z^{-1})}_{\widehat{P}^T(z^{-1})\widehat{P}(z)}C(z) = \widehat{P}^T(z^{-1})M(z).$$

- Multiply on the left by $\left[\widehat{P}(z^{-1})\widehat{P}(z^{-1})^T\right]^{-1}\widehat{P}(z^{-1})$. We are left with

$$\widehat{P}(z)C(z) = M(z).$$

- We notice (by substitution) that one possible solution for $C(z)$ is

$$C^{(opt)}(z) = \widehat{P}(z)^T\left[\widehat{P}(z)\widehat{P}(z)^T\right]^{-1}M(z).$$

- This is the minimum-norm solution.
- Generally, when there are more inputs than outputs there will be many solutions to $C(z)$ which all achieve zero error.
- The minimum-norm solution is one of these, and uses the least amount of control effort (in a mean-squared sense).
- It is not a guaranteed solution, however.

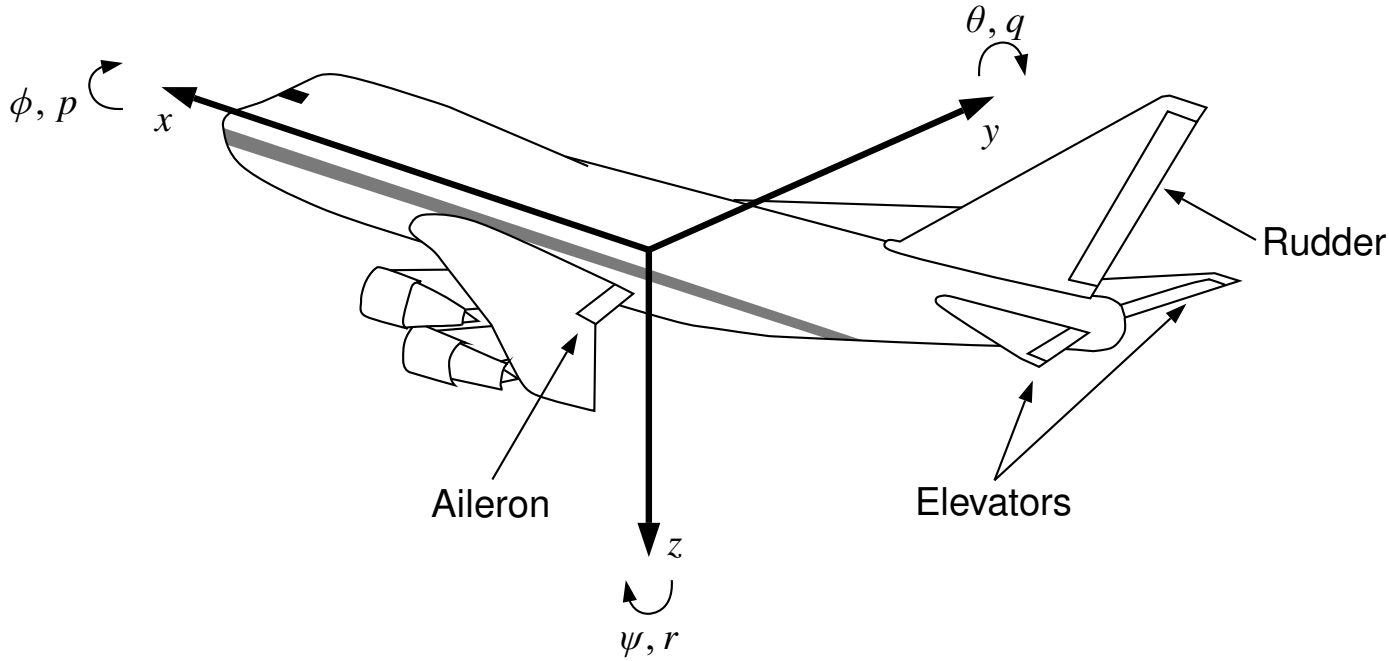
EQUAL NUMBER OF INPUTS AND OUTPUTS: If the plant has an equal number of inputs and outputs, and if the plant is invertible, then both solutions become

$$C^{(opt)}(z) = \widehat{P}(z)^{-1}M(z).$$

- We note that all three Wiener solutions are based on the inverse of the plant model and not on the inverse of the plant.
- So, the efficacy of this method is dependent on an accurate plant model.

7.24: MIMO example: No disturbance

- Two aspects of flight control for a Boeing 747 aircraft were selected to demonstrate linear, MIMO control.
- The dynamics of the airplane have been approximated by a linear model around an equilibrium point.
- In the case at hand, the equilibrium "point" is level flight at 40,000 ft and a nominal forward speed of Mach 0.8 (774 ft/sec).
- The resulting linearized equations of motion are eighth-order, but they may be separated into two fourth-order sets representing the perturbations in longitudinal and lateral motion.
- Here, we wish to control the aircraft's yaw-rate (r) and bank-angle (ϕ).



x, y, z = position coordinates

p, q, r = roll, pitch and yaw *rates*

ϕ, θ, ψ = roll (bank), pitch and yaw *angles*

- The dynamics of the system are most compactly represented in "state-space" form.

- When converted to discrete-time, we define

$$\big[u[k]\big] = \begin{bmatrix} \text{Rudder angle in degrees} \\ \text{Aileron angle in degrees} \end{bmatrix}$$

$$\big[y[k]\big] = \begin{bmatrix} \text{Yaw rate, } r[k], \text{ in radians/second} \\ \text{Bank angle, } \phi[k], \text{ in radians} \end{bmatrix},$$

and,

$$\big[x[k]\big] = \begin{bmatrix} \text{Sideslip angle, } \beta[k], \text{ in radians} \\ \text{Yaw rate, } r[k], \text{ in radians/second} \\ \text{Roll rate, } p[k], \text{ in radians/second} \\ \text{Bank angle, } \phi[k], \text{ in radians} \end{bmatrix}.$$

- Then,

$$\big[x[k+1]\big] = A_d\big[x[k]\big] + B_d\big[u[k]\big]$$

$$\big[y[k]\big] = C_d\big[x[k]\big],$$

where,

$$A_d = \begin{bmatrix} 0.8876 & -0.3081 & 0.0415 & 0.0198 \\ 0.2020 & 0.3973 & -0.0046 & 0.0024 \\ -1.2515 & 0.5106 & 0.7617 & -0.0139 \\ -0.3313 & 0.1510 & 0.4407 & 0.9976 \end{bmatrix},$$

$$B_d = \begin{bmatrix} 0.4806 & -0.0013 \\ -1.5809 & 0.3887 \\ 0.0599 & 4.8390 \\ 0.0390 & 1.2585 \end{bmatrix}.$$

- The output matrix C_d was chosen to be either

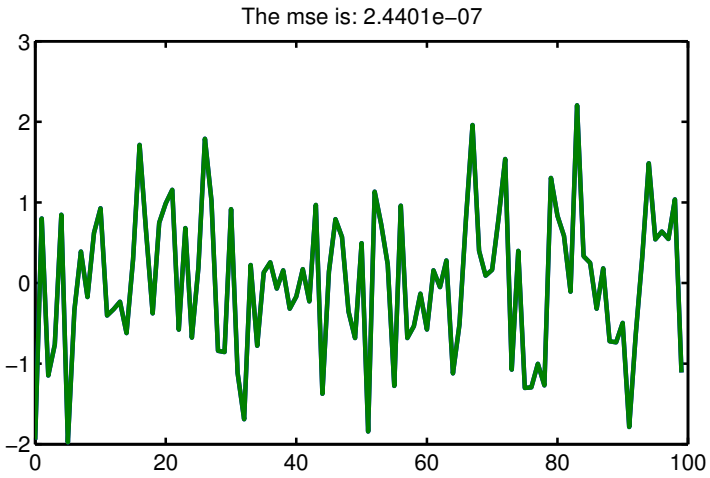
$$C_d = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}, \quad C_d = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad C_d = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

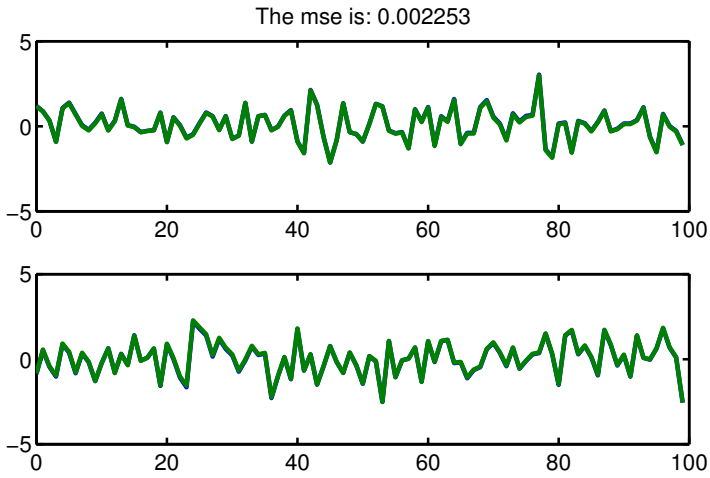
depending on whether the plant had 1, 2 or 3 outputs.

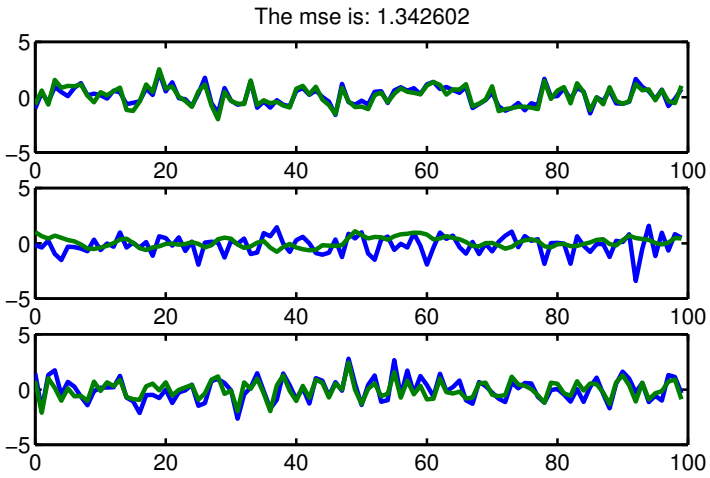
- The reference command to be tracked is generated independently for each output.
- The desired yaw-rate is a first-order Markov process generated by filtering i.i.d. uniform random variables with maximum value 0.03 using a one-pole filter whose pole is at $z = 0.9$.
- The desired bank angle is a first-order Markov process generated by filtering i.i.d. uniform random variables with maximum value 0.12 with a one-pole filter whose pole is at $z = 0.9$.
- The primary disturbance experienced by the dynamics of the airplane are those induced by bursts of wind.
- It is assumed here that the nominal wind values are incorporated into the dynamic model of flight, and that gusts around that nominal value are the disturbances.
- The state of the airplane, x_k, is affected directly by the wind.
- So, the full discrete-time model of the airplane dynamics, with disturbance, is

$$\begin{aligned} \big[x[k+1]\big] &= A_d\left[\big[x[k]\big] + \text{dist}[k]\right] + B_d\big[u[k]\big] \\ \big[y[k]\big] &= C_d\big[x[k]\big]. \end{aligned}$$

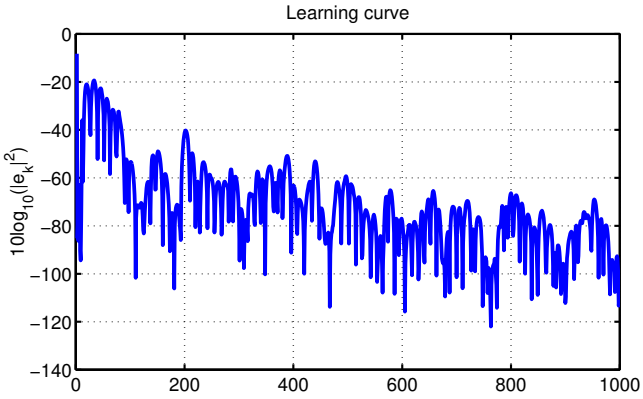
- Tracking results (in the absence of added disturbance) for the three cases simulated are

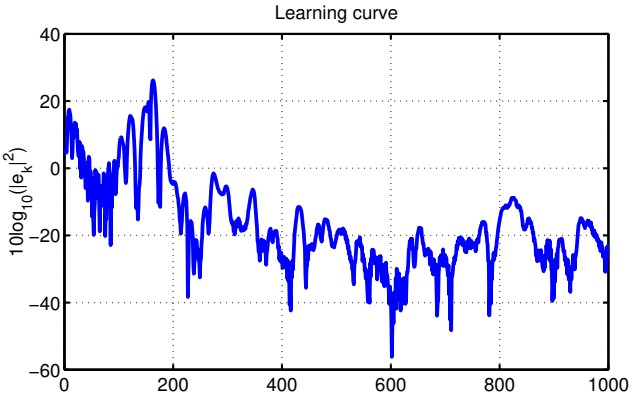


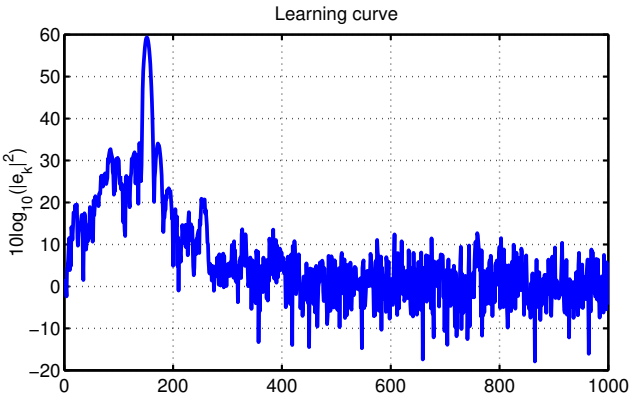




- Tracking is essentially perfect when there are more plant inputs than outputs, and is also very good when there are an equal number of plant inputs and outputs.
- When there are more plant outputs than inputs, it becomes impossible to get perfect tracking in general, and the controller adapts to find the solution which minimizes the mean-square output error.
- Learning curves for the three cases are below. We see convergence in all cases within 200–400 iterations.







7.25: MIMO example: With disturbance

Defining the input disturbance

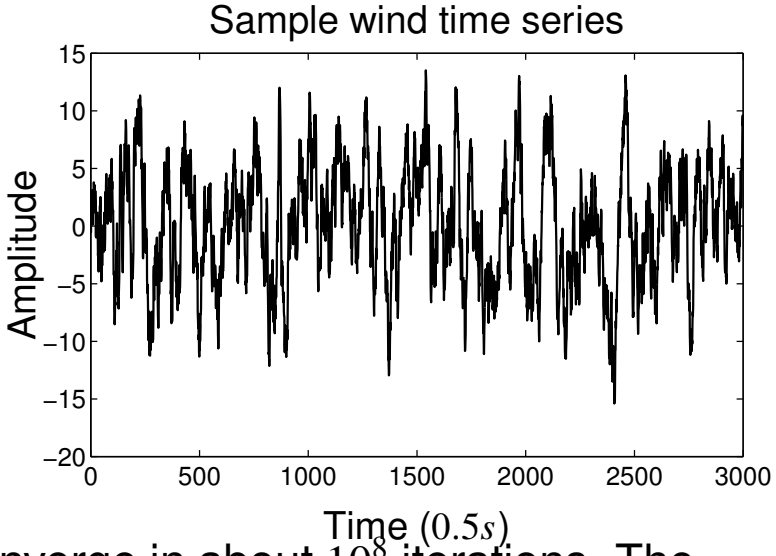
- It is assumed that the wind gusts occur as planar fronts and thus do not affect the yaw-rate, roll-rate or bank-angle directly.
- Instead, the sideslip angle is directly affected by the wind, and the other state-variables are affected indirectly through the dynamical relationship between themselves and the sideslip angle.
- If we model the wind in the lateral direction, then the sideslip angle is perturbed by

$$\Delta\big[\beta[k]\big] = \text{atan}\left(\frac{\text{wind speed}}{\text{airplane speed}}\right).$$

- The model for generating a wind-speed time series is based on data presented in the literature.
- The power spectral density of wind velocity was

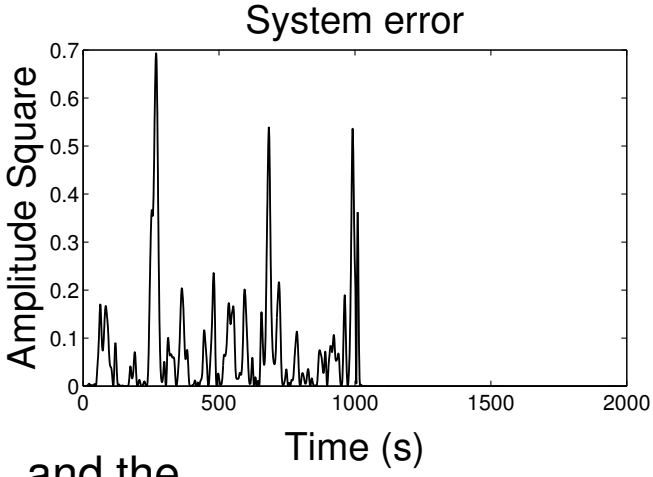
$$\Phi(f) = \frac{3950}{1 + (20\pi f)^2}.$$

- An FIR filter was designed to produce this power spectral density given an input stream of i.i.d. uniform random numbers with maximum magnitude 1.
- A sample wind time-sequence is shown below to the right.



- The maximum absolute wind speed is in the neighborhood of 20 feet per second, so the maximum perturbation to β_k is around 0.03 radians.
- Note: LMS-based methods converge in about 10^8 iterations. The present method converges in about 200–400 iterations!

Disturbance canceling



- Disturbance canceling was tested for the MIMO system, and results are plotted to the right.
- The figure shows the squared-norm of the system error plotted versus iteration.
- Disturbance was present at all times, and the (trained) disturbance canceler was turned "on" at time 1000.
- As can be seen, the disturbance canceler removes essentially all of the disturbance, resulting in near-perfect tracking even in the presence of disturbance.

Where from here?

- Congratulations for completing the material of this course.

- I look forward to hearing how you apply it in practice.
- There are a number of areas of future study you might be interested in:
  - ECE5520: Multivariable Control Systems I
  - ECE5530: Multivariable Control Systems II
  - ECE5550: Applied Kalman Filtering
  - ECE5560: System Identification
  - ECE5570: Optimization Methods for Systems and Control
  - ECE5580: Multivariable Control Systems Analysis and Design in the Frequency Domain
  - ECE5590: Model Predictive Control

7.26: Appendix: Sample code for complete MIMO AIC using RLS

```
% ================================================================
% Simple method for adapting a MIMO linear controller using adaptive
% inverse control. Copyright (c) 2000. Dr. Gregory L. Plett. Please
% distribute with this copyright message.
% ================================================================
clf;
numpin=2;               % number of plant inputs
numpout=1;              % number of plant outputs
numptap=61;             % number of taps in plant model
plantsetup;
reporttype=1;           % 1=tracking progress; 2=impulse responses
report=1000;            % frequency of reports
maxiter=1000;           % how many adaptive iterations. Convergence ~ 200.
dksave=zeros([numpout report]);
yksave=zeros([numpout report]);
eksave=zeros([1 report]);

WmT=[zeros(numpout) eye(numpout)];  % ref-model trans---a unit delay
MTin=zeros([length(WmT(1,:)) 1]);   % Input to M-transpose
Min=zeros([length(WmT(1,:)) 1]);    % Input to M

Pin=zeros([numpin*numptap 1]);      % Input to the true plant
PHin=Pin;                           % Input to Phat to adapt C-transpose
PHTin=zeros([numpout*numptap 1]);   % Input to Phat-transpose to adapt C-
 trans

numctap=71;
CCin=zeros([numpout*numctap 1]);    % Input to C-copy
CTin=zeros([numpin*numctap 1]);     % Input to C-transpose
WcT=zeros([numpout numpin*numctap]);% Initial weights of the C-transpose

lambda=0.9999;                      % the RLS forgetting factor
PCT=inv(0.0001)*eye(length(CTin));  % The initial RLS inverse cov. matrix
 for C

i=1; mse=0;
for iter=1:maxiter,
  % get system desired response
  ik=randn([numpout 1]);        % random input to system
  PHTin=[ik; PHTin(1:end-length(ik),1)];
  xk=Wpht*PHTin;
```

```
  CTin=[xk; CTin(1:end-length(xk),1)];

  MTin=[ik; MTin(1:end-length(ik),1)];% shift in to M-transpose
  dkt=WmT*MTin;                  % desired output

  % update weights in C-transpose via matrix RLS
  pin=CTin'*PCT;
  rn=1/(lambda+pin*CTin);
  kn=rn*pin;
  chin=dkt-WcT*CTin;
  WcT=WcT+chin*kn;
  PCT=(PCT-pin'*kn)/lambda;
  PCT=triu(PCT)+triu(PCT,1)';

  Wc=ftranspose(WcT,numctap);

  % Feedforward system to plot tracking
  nk=randn([numpout 1]);                % random input to system
  CCin=[nk; CCin(1:end-length(nk),1)];
  uk=Wc*CCin;
  Pin=[uk; Pin(1:end-length(uk),1)];
  yk=Wp*Pin;

  yksave(:,i)=yk;
  Min=[nk; Min(1:end-length(nk),1)];
  dk=WmT*Min;
  dksave(:,i)=dk;

  % add plant noise here
  yksave(:,i)=yk;
  eksys=dk-yk; if iter>500, mse=mse+eksys'*eksys; end;
  eksave(i)=eksys'*eksys;

  i=i+1;
  % display impulse-response graphs every 50 iterations
  if (mod(iter,report)==0),
    plotreport;
    i=1;
  end
end
mse/(maxiter-500)
```


Appendix: Plant setup code

```
% ========================================================================
% Sets up the MIMO plant
% Copyright (c) 2000. Dr. Gregory L. Plett. Please distribute
% with this copyright message.
% ========================================================================
A=[0.88763 -0.30806  0.04148    0.019846
   0.20197  0.39735 -0.0046012  0.0024034
  -1.2515   0.5106   0.76171   -0.013941
  -0.33126  0.15104  0.44069    0.99763];
B=[0.48062 -0.0012813
  -1.5809   0.38869
   0.059924 4.839
   0.038972 1.2585];
% Compute impulse-responses assuming two-input, three-output and use
% only the impulse-responses we need.
C=[0 1 0 0; 0 0 0 1; 0 0 1 0];
D=[0 0; 0 0; 0 0];
sysd=ss(A,B,C,D,0.5);   % create state-space discrete-time system, T=0.5
H=impulse(sysd,(numptap-1)/2);% get impulse responses
h11=H(:,1,1)'; h12=H(:,1,2)'; % form individual impulse responses
h21=H(:,2,1)'; h22=H(:,2,2)';
h31=H(:,3,1)'; h32=H(:,3,2)';

% create the impulse-response filter matrix
Wph=zeros([numpout numpin*numptap]);
Wph(1,1:numpin:numpin*numptap-1)=h11;
Wph(1,2:numpin:numpin*numptap)=h12;
if numpout>1,
  Wph(2,1:numpin:numpin*numptap-1)=h21;
  Wph(2,2:numpin:numpin*numptap)=h22;
  if numpout>2,
    Wph(3,1:numpin:numpin*numptap-1)=h31;
    Wph(3,2:numpin:numpin*numptap)=h32;
  end
end
Wp=Wph; % true plant
% create the transpose impulse-response filter
Wpht=ftranspose(Wph,numptap);
```

Report plotting code

```
% ========================================================================
% Plots reports for the simulations
% Copyright (c) 2000. Dr. Gregory L. Plett. Please distribute
% with this copyright message.
% ========================================================================
if reporttype==1, % tracking
  themse=sum(sum((dksave-yksave).^2))/report;
  themse=sprintf('The mse is: %f',themse);
  subplot(numpout,1,1);
  plot(0:report-1,dksave(1,:),0:report-1,yksave(1,:));
  title(themse);
  if numpout>1,
    subplot(numpout,1,2);
    plot(0:report-1,dksave(2,:),0:report-1,yksave(2,:));
    if numpout>2,
      subplot(numpout,1,3);
      plot(0:report-1,dksave(3,:),0:report-1,yksave(3,:));
    end
  end
elseif reporttype==2, % impulse responses
  wctranspose=ftranspose(Wc,numctap);
  c11=wctranspose(1,1:2:end-1); c21=wctranspose(1,2:2:end);
  I11=conv(h11,c11)+conv(h12,c21);
  if numpout>1,
    c12=wctranspose(2,1:2:end-1); c22=wctranspose(2,2:2:end);
    I12=conv(h11,c12)+conv(h12,c22);
    I21=conv(h21,c11)+conv(h22,c21);
    I22=conv(h21,c12)+conv(h22,c22);
    if numpout>2,
      c13=wctranspose(3,1:2:end-1); c23=wctranspose(3,2:2:end);
      I13=conv(h11,c13)+conv(h12,c23);
      I23=conv(h21,c13)+conv(h22,c23);
      I31=conv(h31,c11)+conv(h32,c21);
      I32=conv(h31,c12)+conv(h32,c22);
      I33=conv(h31,c13)+conv(h32,c23);
    end
  end
  subplot(numpout,numpout,1); stem(0:length(I11)-1,I11); title('I11');
  if numpout>1,
    subplot(numpout,numpout,2); stem(0:length(I12)-1,I12); title('I12');
```

```
    subplot(numpout,numpout,numpout+1); stem(0:length(I21)-1,I21); title('
     I21');
    subplot(numpout,numpout,numpout+2); stem(0:length(I22)-1,I22); title('
     I22');
    if numpout>2,
      subplot(numpout,numpout,3); stem(0:length(I13)-1,I13); title('I13');
      subplot(numpout,numpout,6); stem(0:length(I23)-1,I23); title('I23');
      subplot(numpout,numpout,7); stem(0:length(I31)-1,I31); title('I31');
      subplot(numpout,numpout,8); stem(0:length(I32)-1,I32); title('I32');
      subplot(numpout,numpout,9); stem(0:length(I33)-1,I33); title('I33');
    end
  end
elseif reporttype==3, % impulse responses of Wc
  wctranspose=ftranspose(Wc,numctap);
  c11=wctranspose(1,1:2:end-1); c21=wctranspose(1,2:2:end);
  if numpout>1,
    c12=wctranspose(2,1:2:end-1); c22=wctranspose(2,2:2:end);
    if numpout>2,
      c13=wctranspose(3,1:2:end-1); c23=wctranspose(3,2:2:end);
    end
  end
  subplot(2,numpout,1); stem(0:length(c11)-1,c11); title('c11');
  subplot(2,numpout,numpout+1); stem(0:length(c21)-1,c21); title('c21');
  if numpout>1,
    subplot(2,numpout,2); stem(0:length(c12)-1,c12); title('c12');
    subplot(2,numpout,numpout+2); stem(0:length(c22)-1,c22); title('c22');
    if numpout>2,
      subplot(2,numpout,3); stem(0:length(c13)-1,c13); title('c13');
      subplot(2,numpout,numpout+3); stem(0:length(c23)-1,c23);
      title('c23');
    end
  end
elseif reporttype==4, % impulse responses of Wv
  WvT=ftranspose(Wv,numctap);
  c11=WvT(1,1:2:end-1); c21=WvT(1,2:2:end);
  if numpout>1,
    c12=WvT(2,1:2:end-1); c22=WvT(2,2:2:end);
    if numpout>2,
      c13=WvT(3,1:2:end-1); c23=WvT(3,2:2:end);
    end
  end
  subplot(2,numpout,1); stem(0:length(c11)-1,c11); title('v11');
```

```
  subplot(2,numpout,numpout+1); stem(0:length(c21)-1,c21); title('v21');
  if numpout>1,
    subplot(2,numpout,2); stem(0:length(c12)-1,c12); title('v12');
    subplot(2,numpout,numpout+2); stem(0:length(c22)-1,c22); title('v22');
    if numpout>2,
      subplot(2,numpout,3); stem(0:length(c13)-1,c13); title('v13');
      subplot(2,numpout,numpout+3); stem(0:length(c23)-1,c23); title('v23'
       );
    end
  end
end; drawnow
```

