

## 1.2 Wiener Filtering

1. Generate an AR (auto-regressive) process  $x(n)$  by filtering white Gaussian noise with power 1.0 with the following filter:

$$G(z) = \frac{1}{1 - 2z^{-1} + 1.91z^{-2} - 0.91z^{-3} + 0.205z^{-4}}.$$

You can implement this with the **IIR** actor. More interestingly, you can implement it with an **FIR** filter in the feedback loop. Try it both ways, but turn in the latter implementation.

2. Define the “desired” signal to be

$$d(n) = g(n) * x(n) + w(n),$$

where  $w(n)$  is a white Gaussian noise process with power 0.5, uncorrelated with  $x(n)$ , and  $g(n)$  is the impulse response of a filter with the following transfer function:

$$G(z) = 1 + 2z^{-1} + 3z^{-2} + 4z^{-3}.$$

Generate  $d(n)$ .

3. Design a fixed Wiener filter for estimating  $d(n)$  from  $x(n)$ , and implement that filter. Generate the error signal  $e(n) = d(n) - y(n)$ , where  $y(n)$  is the output of the Wiener filter. Find a simple expression for  $e(n)$  in terms of  $x(n)$ ,  $w(n)$ , and  $g(n)$ .
4. Use an adaptive filter to perform the same function as the fixed Wiener filter in part 3. Use the **LMSAdaptive** actor with default initial tap values. Experiment with the `stepSize` parameter to get a reasonable convergence rate while maintaining stability. Compare the error signals for the adaptive system to the error signal for fixed system by comparing their power. How closely does the adaptive filter performance approximate that of the fixed Wiener filter? How does its performance depend on the adaptation step size? How quickly does it converge? How much do its final tap value look like the optimal Wiener filter solution?

**Ptolemy Hints:** The **PowerEstimate** actor (in the signal processing palette) is convenient for estimating power. To view the tap values of the adaptive filter as they adapt, connect the **BarGraph** actor to the `tapValues` output of the **LMSAdaptive** actor.