HOMEWORK SOLUTIONS #2

1. (a) Using

$$\begin{aligned}
\mathbf{f} &= [f_0, f_1, \ldots, f_L]^t, \\
\mathbf{e}_\delta &= [0, \ldots, 0, 1, 0, \ldots, 0]^t \text{ (where the 1 is in the } \delta^{th} \text{ position)}, \\
\mathbf{v}(n) &= [v(n), v(n-1), \ldots, v(n-K-L)]^t, \\
\mathbf{B} &= \begin{bmatrix} b_0 & & \\ \vdots & \ddots & \\ b_K & & b_0 \\ & \ddots & \vdots \\ & & b_K \end{bmatrix},
\end{aligned}$$

we see that the impulse response of $B^*(z)F^*(z)$ equals $(\mathbf{Bf})^*$ expressed as a column vector, or $(\mathbf{Bf})^H = \mathbf{f}^H\mathbf{B}^H$ expressed as a row vector. Thus

$$y(n) = \mathbf{f}^H\mathbf{B}^H\mathbf{v}(n).$$

Writing $v(n-\delta) = \mathbf{e}_\delta^t\mathbf{v}(n)$, we have

$$\begin{aligned}
e(n) &= \mathbf{f}^H\mathbf{B}^H\mathbf{v}(n) - \mathbf{e}_\delta^t\mathbf{v}(n) \\
&= (\mathbf{f}^H\mathbf{B}^H - \mathbf{e}_\delta^t)\mathbf{v}(n) \\
&= (\mathbf{Bf} - \mathbf{e}_\delta)^H\mathbf{v}(n)
\end{aligned}$$

since $\mathbf{e}_\delta$ is real-valued.

(b) As for the mean-squared error, we have

$$\begin{aligned}
\mathrm{E}\{|e(n)|^2\} &= \mathrm{E}\{e(n)e^H(n)\} \\
&= \mathrm{E}\{(\mathbf{Bf} - \mathbf{e}_\delta)^H\mathbf{v}(n)\mathbf{v}^H(n)(\mathbf{Bf} - \mathbf{e}_\delta)\} \\
&= (\mathbf{Bf} - \mathbf{e}_\delta)^H \underbrace{\mathrm{E}\{\mathbf{v}(n)\mathbf{v}^H(n)\}}_{\sigma_v^2\mathbf{I}}(\mathbf{Bf} - \mathbf{e}_\delta) \\
&= (\mathbf{Bf} - \mathbf{e}_\delta)^H(\mathbf{Bf} - \mathbf{e}_\delta)\sigma_v^2 \\
&= \|\mathbf{Bf} - \mathbf{e}_\delta\|^2\sigma_v^2.
\end{aligned}$$

(c) To achieve $\mathrm{E}\{|e(n)|^2\} = 0$, the previous equation implies that we need $\mathbf{Bf} = \mathbf{e}_\delta$. In other words,

$$\begin{bmatrix} b_0 & & \\ \vdots & \ddots & \\ b_K & & b_0 \\ & \ddots & \vdots \\ & & b_K \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_L \end{bmatrix} = \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}. \tag{1}$$

To solve the first equation in the matrix system, it is easily seen that we need $f_0 = 0$. For the second equation, since $f_0 = 0$, we need $f_1 = 0$, and so on. We could repeat this procedure from the bottom to find that we need $f_L = 0$, $f_{L-1} = 0$, and so on. In total, these equations imply $\mathbf{f} = \mathbf{0}$, which prevents satisfaction of the middle equation $[b_K, \ldots, b_1, b_0][f_0, f_1, \ldots, f_L]^t = 1$! Thus, we cannot achieve zero error; we cannot equalize the system.

It should be noted that our problem has more equations $(L + K + 1)$ than unknowns $(L + 1)$ since $K > 0$. This implies that we cannot solve $\mathbf{Bf} = \mathbf{x}$ for *generic* $\mathbf{x}$, although there will be solutions for special $\mathbf{x}$ (specifically, those in the column span of $\mathbf{B}$). For certain $\mathbf{B}$, then, $\mathbf{Bf} = \mathbf{e}_\delta$ will have a solution.

(d) With the trivial channel $B^*(z) = z^{-\delta}$, the choice $F^*(z) = 1$ (i.e., $\mathbf{f} = [1, 0, 0, \ldots]^t$) equalizes with delay δ. This can be seen by plugging the values $b_\delta = 1$ and $b_k\big|_{k \neq \delta} = 0$ into (1) and noting that the first column of $\mathbf{B}$ equals $\mathbf{e}_\delta$.

2. (a) Denoting the output of $F^*(z)$ by $y_1(n)$ and the output of $G^*(z)$ by $y_2(n)$, we use the methods and notation from the previous problem to write

$$\begin{aligned}
y_1(n) &= \mathbf{f}^H \mathbf{B}^H \mathbf{v}(n) \\
y_2(n) &= \mathbf{g}^H \mathbf{C}^H \mathbf{v}(n) \\
y(n) &= (\mathbf{f}^H \mathbf{B}^H + \mathbf{g}^H \mathbf{C}^H) \mathbf{v}(n) \\
&= (\mathbf{Bf} + \mathbf{Cg})^H \mathbf{v}(n) \\
&= \left(\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \right)^H \mathbf{v}(n) \\
e(n) &= y(n) - v(n - \delta) \\
&= y(n) - \mathbf{e}_\delta^t \mathbf{v}(n) \\
&= \left(\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} - \mathbf{e}_\delta \right)^H \mathbf{v}(n)
\end{aligned}$$

(b) As for the mean-squared error, we have

$$\mathrm{E}\{|e(n)|^2\} = \left\| \begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} - \mathbf{e}_\delta \right\|^2 \sigma_v^2.$$

To achieve $\mathrm{E}\{|e(n)|^2\} = 0$ for delay δ we need

$$\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} = \mathbf{e}_\delta. \tag{2}$$

For solutions to (2) to exist for *any* $\delta \in \{0, L + K\}$, $\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix}$ must have at least $L + K + 1$ linearly independent columns. Note that $\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix}$ has $L + K + 1$ rows and $2(L + 1)$ columns

$$\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix} = \begin{bmatrix} b_0 & & & c_0 & & \\ \vdots & \ddots & & \vdots & \ddots & \\ b_K & & b_0 & c_K & & c_0 \\ & \ddots & \vdots & & \ddots & \vdots \\ & & b_K & & & c_K \end{bmatrix}.$$

A necessary condition, then, is that we have *enough* columns. This can be stated as:

$$\begin{aligned}
2(L + 1) &\geq L + K + 1 \\
\Leftrightarrow \quad L &\geq K - 1,
\end{aligned}$$

In communication terminology, the equalizer length must be greater or equal to the channel length minus one. But adequate equalizer length is not sufficient; consider, e.g., the case where $B^*(z) = C^*(z)$: there would be at most $L+1$ linearly independent columns! Thus, a necessary and sufficient condition is that $\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix}$ must be *full row rank* (i.e., the rank equals the number of rows, which in this case is $L+K+1$).

(c) Solving for the solution to (2), we must keep in mind that $\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix}$ may have more columns than rows. While the straighforward matrix inverse doesn't exist for non-square matrices, the psuedo-inverse does:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix}^H \left(\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix}^H \right)^{-1}}_{\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix}^+} \mathbf{e}_\delta$$

Note that $\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix}^H$ is full-rank and square, so the inverse is well defined. It is easily verified that these coefficients solve (2).

3. AR process design using Yule-Walker method:

(a) Parameter design:

M	$a_0, a_1, \ldots$	σ_v^2
2	-1.7625, 0.9503	8.8919e-03
4	-3.5707, 5.1243, -3.4867, 0.9511	8.1109e-05
5	-4.4752, 8.4404, -8.3603, 4.3470, -0.9511	7.7452e-06

(c) Empirically estimated autocorrelation compared to desired autocorrelations:

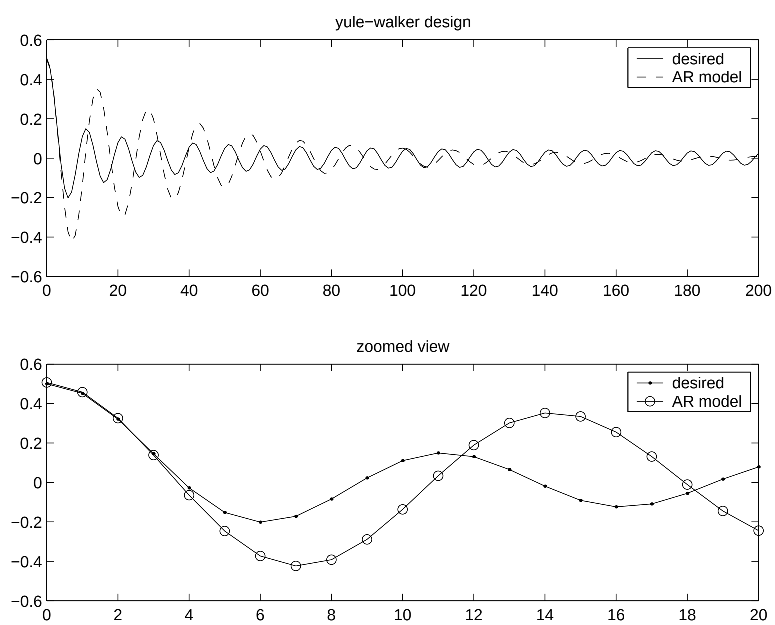

Figure 1: Yule-Walker design for $M = 2$.

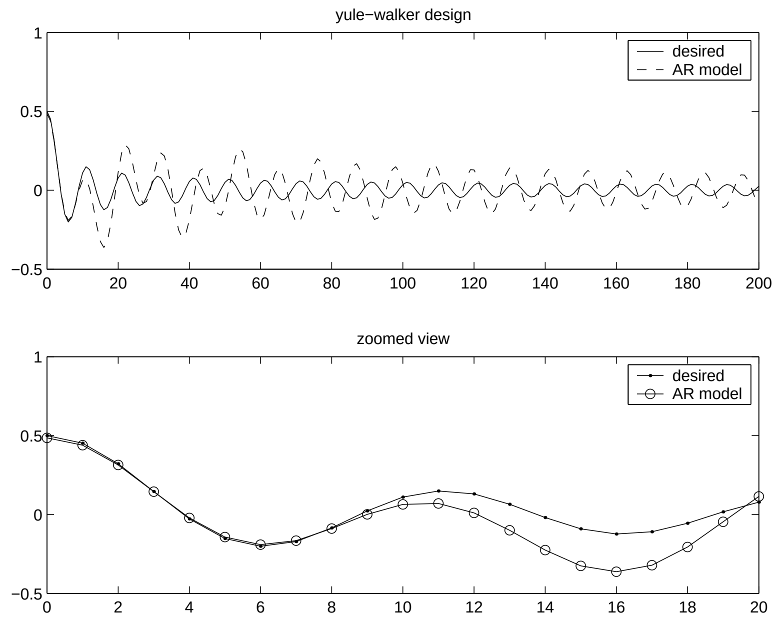

Figure 2: Yule-Walker design for $M = 4$.

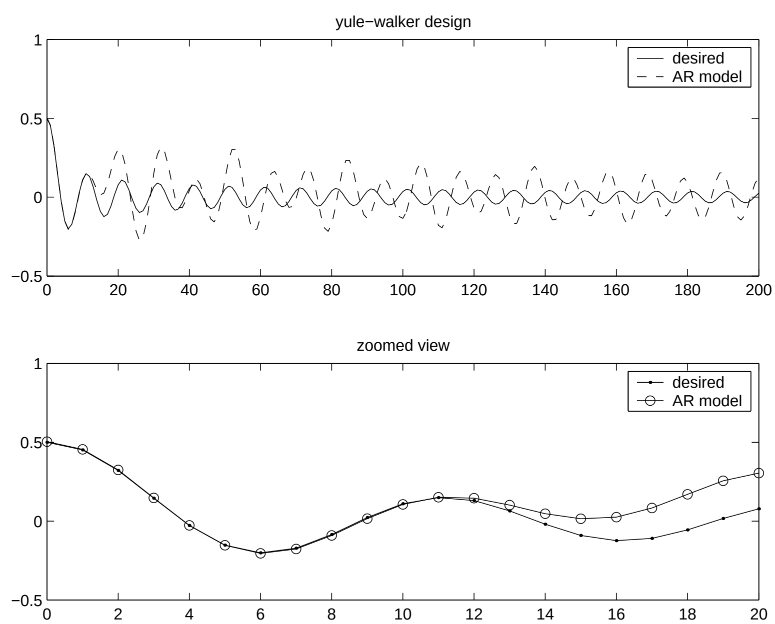

Figure 3: Yule-Walker design for $M = 5$.

4. AR process design using Extended Yule-Walker method:

(a) Parameter design:

L	$a_0, a_1, \ldots, a_4$	σ_v^2
10	-4.2455, 7.6262, -7.2003, 3.5637, -0.7393	4.7480e-05
20	-4.0111, 6.8570, -6.1784, 2.9211, -0.5787	1.1132e-04
100	-4.0323, 6.9606, -6.3555, 3.0573, -0.6194	6.4985e-05

(c) Empirically estimated autocorrelation compared to desired autocorrelations:
(Note that increasing L results in a better match to desired $r(k)$ for large k but at the expense of mismatching the values at small k. For example, Fig. 6 shows that $r(0)$ is not well matched when L is large.)

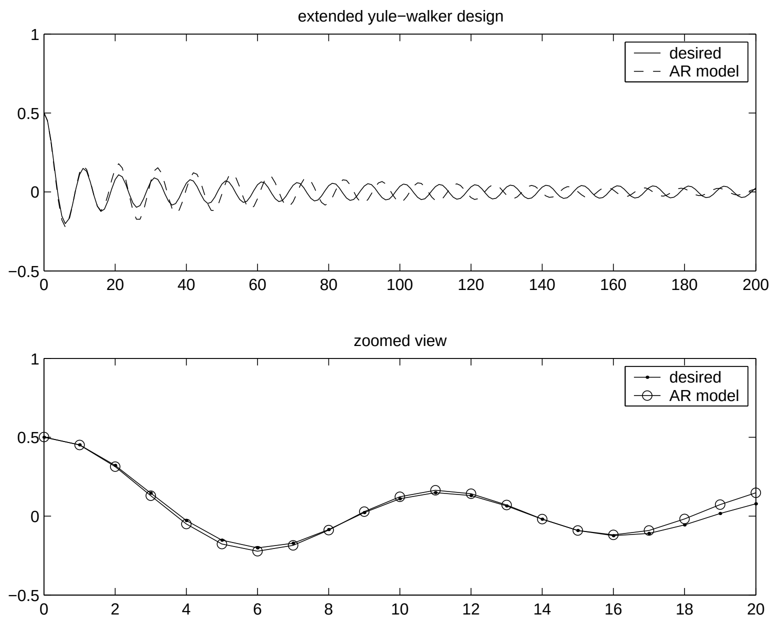

Figure 4: Extended Yule-Walker design for $M = 5$ and $L = 10$.

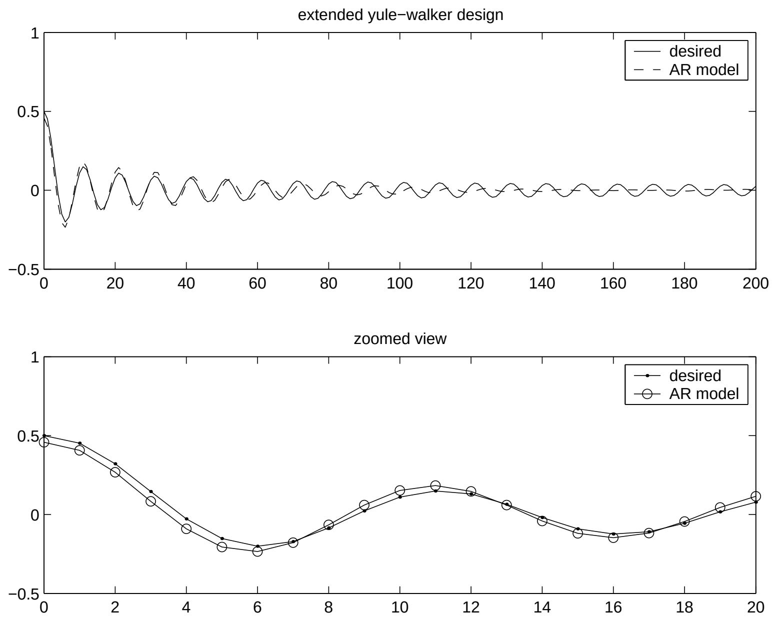

Figure 5: Extended Yule-Walker design for $M = 5$ and $L = 20$.

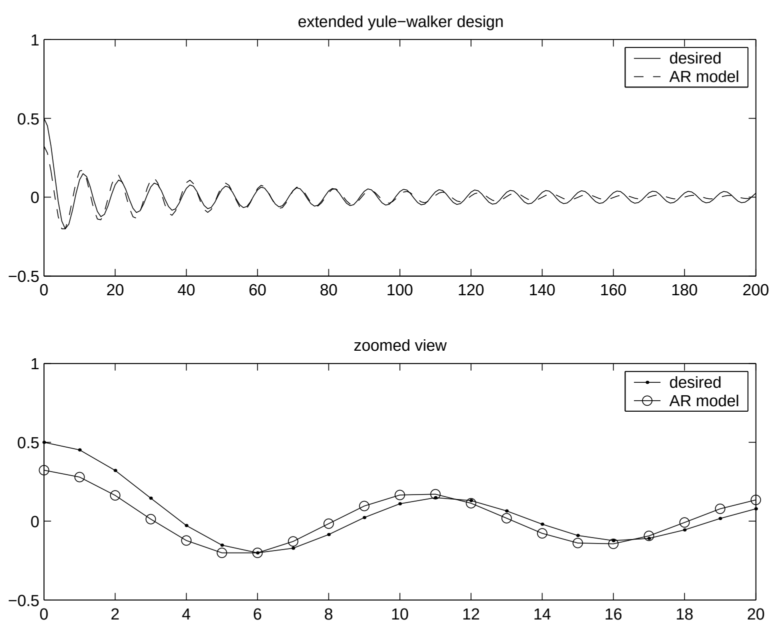

Figure 6: Extended Yule-Walker design for $M = 5$ and $L = 100$.

Matlab Code:

```
% fits a bessel function autocorrelation w/ an AR model

fmbyFs = 0.1; % normalized frequency
P = 1; % power
M = 5; % AR model order
Me = 100;   % extended yule walker fitting order
Mp = 200; % plotting order
N = 10e4;

% desired autocorrelation sequence
rr = P/2*besselj(0,2*pi*fmbyFs*[0:Mp]).';

% standard yule-walker
r_y = rr(2:M+1);
R_y = toeplitz(rr(1:M).',rr(1:M));
a_y = -R_y\r_y;
sig2_v = [1; a_y]'*rr(1:M+1);
v1 = randn(1,N);
u = filter(1,[1; a_y],v1*sqrt(sig2_v));
tmp = xcorr(u,u,Mp,'unbiased'); r_hat = tmp(Mp+1:2*Mp+1);

% plot standard
figure(1)
subplot(211)
h1a=plot([0:Mp],rr,...
     [0:Mp],r_hat,'g--');
legend(h1a,'desired','AR model');
title('yule-walker design');
subplot(212)
h1b=plot([0:20],rr(1:21),'b.-',...
     [0:20],r_hat(1:21),'g-o');
legend(h1b,'desired','AR model');
title('zoomed view');

% extended yule-walker
tmp = toeplitz(rr(1:Me).',rr(1:Me)); R_e = tmp(:,1:M);
r_e = rr(2:Me+1);
a_e = -pinv(R_e)*r_e;
A = [convmtx([1;a_e],M+1),[zeros(1,M); convmtx(flipud([1;conj(a_e)]),M)]];
tmp = A.'\[zeros(M,1);1;zeros(M,1)]; r_e1 = [tmp(M+1:2*M+1);zeros(Me-M,1)];
for k=M+2:Me+1, r_e1(k) = -a_e'*r_e1(k-[1:M]); end;
sig2_ve = r_e1'*rr(1:Me+1)/norm(r_e1)^2;
u_e = filter(1,[1; a_e],v1*sqrt(sig2_ve));
tmp = xcorr(u_e,u_e,Mp,'unbiased'); r_hat_e = tmp(Mp+1:2*Mp+1);

% plot extended
figure(2)
subplot(211)
h2a=plot([0:Mp],rr,...
     [0:Mp],r_hat_e,'g--');
legend(h2a,'desired','AR model');
title('extended yule-walker design');
subplot(212)
h2b=plot([0:20],rr(1:21),'b.-',...
     [0:20],r_hat_e(1:21),'g-o');
legend(h2b,'desired','AR model');
title('zoomed view');
```