% +-----------+ +

% x(k) ---+--->| P(z) |--yp(k)----------------> sum --+---> e(k)

% | +-----------+ ^- |

% | | |

% | \ ys(k) |

% | +-----------+ +-----------+ | |

% +--->| C(z) |--yw(k)-->| S(z) |---+ |

% | +-----------+ +-----------+ |

% | \ |

% | \----------------\ |

% | \ |

% | +-----------+ +-----------+ |

% +--->| Sh(z) |--xs(k)-->| LMS |<-------+

% +-----------+ +-----------+

clear

T=1000; %Simulation Duration

% We do not know P(z) and S(z), So we have to make dummy paths

Pw=[0.01 0.25 0.5 1 0.5 0.25 0.01];

Sw=Pw\*0.25;

x\_iden=randn(1,T); %generating white noise signal to estimate S(z)

% send it to the actuator, and measure it at the sensor position,

y\_iden=filter(Sw, 1, x\_iden);

% Then, start the identification process

Shx=zeros(1,16); % the state of Sh(z)

Shw=zeros(1,16); % the weight of Sh(z)

e\_iden=zeros(1,T); % data buffer for the identification error

%LMS algorithm

% [Shy,Shw]= lms(Shx,y\_iden,x\_iden,Shw,e\_iden,T)

mu=0.1; % learning rate

for k=1:T, % discrete time k

Shx=[x\_iden(k) Shx(1:15)]; % update the state

Shy=sum(Shx.\*Shw); % calculate output of Sh(z)

e\_iden(k)=y\_iden(k)-Shy; % calculate error

Shw=Shw+mu\*e\_iden(k)\*Shx; % adjust the weight

end

% Lets check the result

subplot(2,1,1)

plot([1:T], e\_iden)

title('Before applying Fxlms')

ylabel('Amplitude');

xlabel('Discrete time k');

legend('Identification error');

subplot(2,1,2)

stem(Sw)

hold on

stem(Shw, 'r\*')

ylabel('Amplitude');

xlabel('Numbering of filter tap');

legend('Coefficients of S(z)', 'Coefficients of Sh(z)')

% The second task is the active control

X=randn(1,T);

% measure the arriving noise at the sensor position,

Yd=filter(Pw, 1, X);

% Initiate the system,

Cx=zeros(1,16); % the state of C(z)

Cw=zeros(1,16); % the weight of C(z)

Sx=zeros(size(Sw)); % the dummy state for the secondary path

e\_cont=zeros(1,T); % data buffer for the control error

Xhx=zeros(1,16); % the state of the filtered x(k)

% FxLMS algorithm

% [Cy,Cw]= FxLMS(X,Cx,Cw,Sx,Sw,Shx,Shw,e\_cont,Xhx,T,Yd)

mu=0.1; % learning rate

for k=1:T, % discrete time k

Cx=[X(k) Cx(1:15)]; % update the controller state

Cy=sum(Cx.\*Cw); % calculate the controller output

Sx=[Cy Sx(1:length(Sx)-1)]; % propagate to secondary path

e\_cont(k)=Yd(k)-sum(Sx.\*Sw); % measure the residue

Shx=[X(k) Shx(1:15)]; % update the state of Sh(z)

Xhx=[sum(Shx.\*Shw) Xhx(1:15)]; % calculate the filtered x(k)

Cw=Cw+mu\*e\_cont(k)\*Xhx; % adjust the controller weight

end

% Report the result

figure

subplot(2,1,1)

plot([1:T], e\_cont)

title('After applying FxLMS algorithm for noise control')

ylabel('Amplitude');

xlabel('Discrete time k');

legend('Noise residue')

subplot(2,1,2)

plot([1:T], Yd)

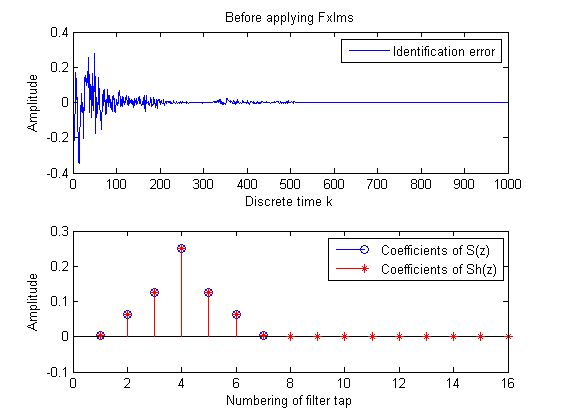
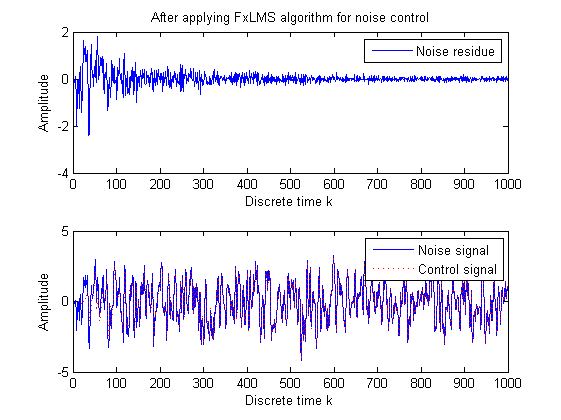
hold on

plot([1:T], Yd-e\_cont, 'r:')

ylabel('Amplitude');

xlabel('Discrete time k');

legend('Noise signal', 'Control signal')

[*Published with MATLAB® R2014a*](http://www.mathworks.com/products/matlab/)