10.3

6880 Homework 9 Zeyu Liu

10.3

Solution:

$$\because \Phi(n) = \delta \underline{\underline{I}} + \underline{u}(n)\underline{u}^H(n)$$

$$A = B^{-1} + CD^{-1}C^H$$

$$\left.\begin{array}{l} \Phi(n) = A, \quad B^{-1} = \delta \underline{\underline{I}} \\ C = \underline{u}(n) \quad D^{-1} = \underline{\underline{I}} = 1 = D \end{array}\right\}$$

Using matrix inversion Lemma:

$$A^{-1} = B - BC(D + C^H BC)^{-1} C^H B$$

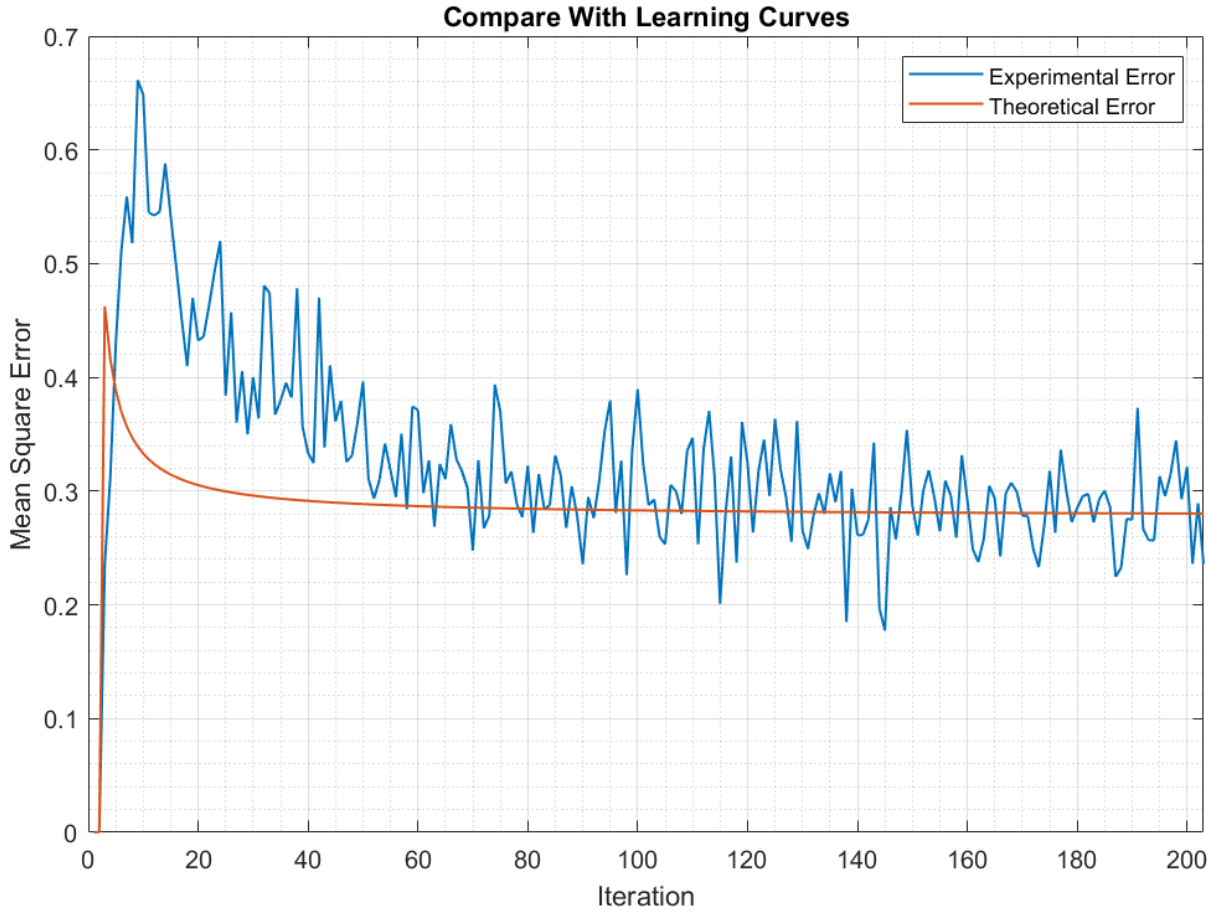
$$\therefore P(n) = A^{-1} = \Phi^{-1}(n) = \frac{1}{\delta}\underline{\underline{I}} - \frac{1}{\delta}\underline{\underline{I}}\,\underline{u}(n)\left(\underline{\underline{I}} + \underline{u}^H(n)\frac{1}{\delta}\underline{\underline{I}}\,\underline{u}(n)\right)^{-1}\underline{u}^H(n)\frac{1}{\delta}\underline{\underline{I}}$$

$$= \frac{1}{\delta}\underline{\underline{I}} - \frac{1}{\delta^2}\underline{\underline{I}}\,\underline{u}(n)\left(\frac{\delta}{\underline{\underline{I}}\,(\delta + \underline{u}^H(n)\underline{u}(n))}\right)\underline{u}^H(n)$$

$$= \frac{1}{\delta}\underline{\underline{I}} - \frac{1}{\delta}\frac{\underline{u}(n)\underline{u}^H(n)}{\delta + \underline{u}^H(n)\underline{u}(n)}$$

$$= \frac{1}{\delta}\left(\underline{\underline{I}} - \frac{\underline{u}(n)\underline{u}^H(n)}{\delta + \underline{u}^H(n)\underline{u}(n)}\right)$$

10.9



From the figure, the blue line is RLS learning curve, the red line Is small step-size statistical theory in 10.68.

At the beginning of the iteration, the two line are not fit well, but when convergence steady, they fit well. I also found when choose different delta, when delta is large, the convergence will become slower. You can try revise my code: delta = 20 to delta = 100. The results figure will appear.

```
function function_10_9
% 6880
% Zeyu Liu
% 2/18/2020
% Adaptive filter theory 5 edition

% 10.9(d)(e)
N = 200;
nums = 100;
a1 = 0.1; % AR parameter
a2 = -0.8;
NV = 0.2775; % Noise variance from the 6.17(a),perivous homework5.
SDV = sqrt(NV);
mu = 0.05;
lambda = 0.99;
delta = 20;% delta choose small positive constant for high SNR
           % choose large when low SNR
u = zeros(N+3,1); % input data stream
e = zeros(N+3,1); % error between prediction and results
% e1 = zeros(N+3,1); % error between weight and AR parameter 1
% e2 = zeros(N+3,1); % error between weight and AR parameter 1
```

```
J = zeros(N+3,1);  % experiment error
JJ = zeros(N+3,1); % theoretical error

for k = 1:nums
    P = delta^(-1)*eye(2); % reset the P(0) every Montecarlo
    W = zeros(2,N+3); % reset weights to zero every Montecarlo
    for n = 3:N+3
        u(n) = a1*u(n-1)+a2*u(n-2)+randn(1)*SDV;
        U = [u(n-1);u(n-2)];
        % e1(n) = a1-W(1,n-1);
        % e2(n) = a2-W(2,n-1);
        kappa = lambda^(-1)*P*U/(1+lambda^(-1)*U'*P*U);
        e(n) = u(n)-W(1,n-1)*u(n-1)-W(2,n-1)*u(n-2);
        W(:,n)=W(:,n-1)+kappa*e(n);
        P = lambda^(-1)*P-lambda^(-1)*kappa*U'*P; %update weights per RLS
    end
    J = J+e.^2;
end
J = J/nums; % experiment error
for n = 3:N+3
    JJ(n) = NV+(2/n)*NV; % theoretical error, from the book function(10.68),M=2
end
x = 1:N+3;

%(d)(e)
subplot(1,1,1)
plot(x,J,x,JJ,'linewidth',1);xlim([0 N+3])
legend('Experimental Error','Theoretical Error')
title('Compare With Learning Curves')
xlabel('Iteration')
ylabel('Mean Square Error')
grid on
grid minor
```