```
function [] = exercise_2_7(nB,nA,nP,sigmaReward,sigmaRW)
%
% A nonstationarity problem where the true action values start off equal and
% then follow individual random walks.  We compare two methods
%
% 1) estimating the action value using simple averages AND
% 2) estimating the action value using a constant geometric factor $\alpha=0.1$.
%
% Inputs:
%   nB: the number of bandits
%   nA: the number of arms
%   nP: the number of plays (times we will pull a arm)
%   sigmaReward: the standard deviation of the return from each of the arms
%   sigmaRW: the standard deviation of the random walk
%
% Written by:
% --
% John L. Weatherwax                2007-11-13
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

%close all;
%clc;
%clear;

if( nargin<1 ) % the number of bandits:
  nB = 2000;
end
if( nargin<2 ) % the number of arms:
  nA = 10;
end
if( nargin<3 ) % the number of plays (times we will pull a arm):
  nP = 2000;
end
if( nargin<4 ) % the standard deviation of the return from each of the arms:
  sigmaReward = 1.0;
end

%randn('seed',0);

%-----
% 1) An epsion greedy (eps=0.1) algorithm:
% 2) An alpha=0.1 update algorithm
%-----
tEps  = 0.1;
alpha = 0.1;

avgReward    = zeros(2,nP);
perOptAction = zeros(2,nP);

allRewards_Eps      = zeros(nB,nP);
pickedMaxAction_Eps = zeros(nB,nP);
allRewards_Alp      = zeros(nB,nP);
pickedMaxAction_Alp = zeros(nB,nP);
```

```
for bi=1:nB, % pick a bandit
  % generate the TRUE reward Q^{\star}:
  %qStarMeans = mvnrnd( zeros(nB,nA), eye(nA) );
  qStarMeans = ones(1,nA);
  qT_Eps = zeros(1,nA); % <- initialize value function to zero (no knowledge)
  qN_Eps = zeros(1,nA); % <- keep track of the number draws on each arm
  qT_Alp = zeros(1,nA);

  for pi=1:nP, % make a play ... one for each algorithm

    if( rand(1) <= tEps ) % pick a RANDOM arm ... explore:
      [dum,arm_Eps] = histc(rand(1),linspace(0,1+eps,nA+1));
      arm_Alp = arm_Eps;
    else                  % pick the GREEDY arm:
      [dum,arm_Eps] = max( qT_Eps );
      [dum,arm_Alp] = max( qT_Alp );
    end
    clear dum;

    % determine if the arm selected is the best possible:
    [dum,bestArm] = max( qStarMeans );
    if( arm_Eps==bestArm ) pickedMaxAction_Eps(bi,pi) = 1; end
    if( arm_Alp==bestArm ) pickedMaxAction_Alp(bi,pi) = 1; end
    % get the reward from drawing on that arm:
    reward_Eps = qStarMeans(arm_Eps) + sigmaReward*randn(1);
    reward_Alp = qStarMeans(arm_Alp) + sigmaReward*randn(1);
    allRewards_Eps(bi,pi) = reward_Eps;
    allRewards_Alp(bi,pi) = reward_Alp;
    % update qN,qT incrementally:
    qT_Eps(arm_Eps) = qT_Eps(arm_Eps) + ( reward_Eps-qT_Eps(arm_Eps)
)/(qN_Eps(arm_Eps)+1);
    qN_Eps(arm_Eps) = qN_Eps(arm_Eps) + 1;
    qT_Alp(arm_Alp) = qT_Alp(arm_Alp) + alpha*( reward_Alp-qT_Alp(arm_Alp) );

    [ qT_Eps; qT_Alp; qStarMeans ];

    % To be nonstationary: qStarMeans follows a random walk:
    %
    qStarMeans = qStarMeans + sigmaRW*randn(size(qStarMeans));
    %qStarMeans = qStarMeans + (1e-4)*pi;
  end
end

avgRew          = mean(allRewards_Eps,1);
avgReward(1,:) = avgRew(:).';
avgRew          = mean(allRewards_Alp,1);
avgReward(2,:) = avgRew(:).';

percentOptAction   = mean(pickedMaxAction_Eps,1);
perOptAction(1,:) = percentOptAction(:).';
percentOptAction   = mean(pickedMaxAction_Alp,1);
perOptAction(2,:) = percentOptAction(:).';

% produce the average rewards plot:
%
figure; hold on;
all_hnds = plot( 1:nP, avgReward );
legend( all_hnds, { 'eps:0.1', 'fixed step' }, 'Location', 'SouthEast' );
axis tight; grid on;
xlabel( 'plays' ); ylabel( 'Average Reward' );
```

```
title( [ 'sigmaRW=',num2str(sigmaRW) ] );

% produce the percent optimal action plot:
%
figure; hold on;
all_hnds = plot( 1:nP, perOptAction );
legend( all_hnds, { 'eps:0.1', 'fixed step' }, 'Location', 'SouthEast' );
axis( [ 0, nP, 0, 1 ] ); axis tight; grid on;
xlabel( 'plays' ); ylabel( '% Optimal Action' );
title( [ 'sigmaRW=',num2str(sigmaRW) ] );
```