```
function [pol_pi] = rt_pol_mod(stateseen,Q, pol_pi,
maxNPii,maxNPjj,maxNVii,maxNVjj,maxNAii,maxNAjj)
% MCESTQ - updates the policy based on the estimated action value function for the
racetrack example
%
% Note: to encourage exploration we only consider epsilon-soft policies
%
% Written by:
% --
% John L. Weatherwax                2007-12-07
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

% the epsilon in our epsilon-soft policy: (eps=0 => greedy; eps=1 => random)
%eps = 0.03;
eps = 0.1;
%eps = 0.5;
%eps = 0.85;

% accumulate these values used in computing statistics on this action value function
Q^{\pi}:
% Note 1) in this state transition there is no difference between first occurance and
each occurance ...
for si=1:size(stateseen,1),
  pii=stateseen(si,1); pjj=stateseen(si,2); vii=stateseen(si,3); vjj=stateseen(si,4); %
vii/vjj \in [0,5]
  staInd = sub2ind( [maxNPii,maxNPjj,maxNVii,maxNVjj], pii,pjj,vii+1,vjj+1 );
  posChoices         = velState2PosActions([vii,vjj],maxNVii,maxNVjj,maxNAii,maxNAjj );
  findPosChoices     = find(posChoices);
  numChoices         = length(findPosChoices);
  [dum,greedyChoice] = max( Q(staInd,findPosChoices) );
  nonGreedyChoices   = setdiff( findPosChoices, findPosChoices(greedyChoice) );
  % perform an eps-soft on-policy MC update:
  pol_pi(staInd,findPosChoices(greedyChoice)) = 1 - eps + eps/numChoices;
  pol_pi(staInd,nonGreedyChoices)             = eps/numChoices;
end
```