```
function [V] = iter_poly_gw_not_inplace()
% ITER_POLY_GW - Performs iterative policy evaluation on the state-value function for the
grid world example.
%
% Iterate Bellman equation:
%
% V(s) <- \sum_a \pi(s,a) \sum_{s'} P_{s,s'}^a (R_{s,s'}^a + \gamma V(s'))
%
% Iterations are not performed in place (i.e. we have two arrays and copy
% between them)
%
% where the policy is uniform random steps in either direction.
%
% See ePage 253 in the Sutton book.
%
% Written by:
% --
% John L. Weatherwax                2007-12-03
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

%gamma = 0.9;
gamma = 1;    % <- take this is an undiscounted task

sideL  = 4;
nGrids = sideL^2;

% An array to hold the values of the state-value function
% (the elements 1 and 16 are place holders i.e. not used):
Vp = zeros(sideL);
Vc = zeros(sideL);

% some parameters for convergence:
%
MAX_N_ITERS = 1000;  iterCnt = 0;
CONV_TOL    = 1e-4;  delta = 1e10;

% a uniform policy:
pol_pi = 0.25;

while( (delta > CONV_TOL) && (iterCnt <= MAX_N_ITERS) )
  delta = 0;
  % update states in the order one indexes matrices
  % states (1,1) and (4,4) are terminal states
  for ii=1:sideL,
    for jj=1:sideL,
      if( (ii==1 && jj==1) || (ii==sideL && jj==sideL) ) continue; end

      v     = Vp(ii,jj);
      v_tmp = 0.0;
      % loop over each possible action {up,down,right,left}:
      %
      % action = UP
      if( ii==1 )               % s is ON the top row ... this action does not change
```

```
our position
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii,jj) );
      elseif( ii==2 && jj==1 )   % s is NOT on the top row but will step into a terminal
state (reward is zero)
        %v_tmp = v_tmp + pol_pi*(  0 + gamma*Vp(ii-1,jj) );
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii-1,jj) );
      else                       % s is NOT on the top row ... this action moves us up
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii-1,jj) );
      end

      % action = DOWN
      if( ii==sideL )                   % s is ON the bottom row ... this action does
not change our position
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii,jj) );
      elseif( ii==sideL-1 && jj==sideL ) % s is NOT on the bottom row but will step into
a terminal state (reward is zero)
        %v_tmp = v_tmp + pol_pi*(  0 + gamma*Vp(ii+1,jj) );
        v_tmp = v_tmp + pol_pi*(  -1 + gamma*Vp(ii+1,jj) );
      else                              % s is NOT on the bottom row ... this action
moves us down
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii+1,jj) );
      end

      % action = RIGHT
      if( jj==sideL )                    % s is ON the right most column ... this action
does not change our position
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii,jj) );
      elseif( jj==sideL-1 && ii==sideL )  % s is NOT on the right most column but will
step into a terminal position (reward is zero)
        %v_tmp = v_tmp + pol_pi*(  0 + gamma*Vp(ii,jj+1) );
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii,jj+1) );
      else                              % s is NOT on the right most column ... this
action moves us right
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii,jj+1) );
      end

      % action = LEFT
      if( jj==1 )                         % s is ON the left most column ... this
action does not change our position
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii,jj) );
      elseif( jj==2 && ii==1 )            % s is NOT on the left most column but this
action will move us into a termial position (reward is zero)
        %v_tmp = v_tmp + pol_pi*(  0 + gamma*Vp(ii,jj-1) );
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii,jj-1) );
      else                              % s is NOT on the left most column ... this
action moves us left
        v_tmp = v_tmp + pol_pi*( -1 + gamma*Vp(ii,jj-1) );
      end

      % update Vc(ii,jj):
      Vc(ii,jj) = v_tmp;

      delta = max( [ delta, abs( v-Vc(ii,jj) ) ] );
    end % jj loop
  end % ii loop
  % overwrite previous with current:
  Vp = Vc;

  iterCnt=iterCnt+1;
  % lets print the iterations if desired:
```

```
  if( 0 && mod(iterCnt,1)==0 )
    fprintf( 'iterCnt (k)=%5d; delta=%10.5f\n', iterCnt, delta );
    %disp( fix(Vc*10)/10 ); % <- just display ONE decimal
    disp( round(Vc*10)/10 ); % <- just display ONE decimal
    %pause
  end
end % while loop

V = Vc;
```