```
function [V] = rr_state_bellman(alpha,beta,gamma,Rsearch,Rwait)
% RR_STATE_BELLMAN - Seeks an iterative solution to the optimal Bellman equations for the
state-value function.
%
% Inputs:
%
% State Transition Probabilities:
%
% alpha = the probability that after searching with a "high" energy level we end with a
"high"
% beta  = the probability that after searching with a "low" energy level we end with a
"low"
% gamma = the learning rate
%
% Recieved Rewards (Rsearch > Rwait):
%
% Rsearch = the reward given to searching
% Rwait   = the reward given to waiting    (Rwait <= Rsearch)
%
% See ePage 226-227 for the Bellman equation for the optimal state-value function V^*(s)
%
% 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.
%
%-----

if( nargin==0 )
  % State Transition Probabilities:
  %
  % the probability that after searching with a "high" energy level we end with a "high"
  alpha   = 0.8;
  % the probability that after searching with a "low" energy level we end with a "low"
  beta    = 0.2;
  % the learning rate:
  gamma   = 0.9;

  % Recieved Rewards (Rsearch > Rwait):
  %
  Rsearch = 2.0; % <- the reward given to searching
  Rwait   = 1.0; % <- the reward given to waiting
end

% Initialize our state-value function (high,low):
V = [ 0 0 ];

% some parameters for convergence:
%
MAX_N_ITERS = 100; iterCnt = 0;
CONV_TOL    = 1e-3; delta = 1e10;

while( (delta > CONV_TOL) && (iterCnt <= MAX_N_ITERS) )
  delta = 0;
  % update states in the order high then low:
```

```
  v = V(1); % save the old "high" state ...
  V(1)  = max( [ Rsearch + gamma*( alpha*V(1) + (1-alpha)*V(2) ), Rwait + gamma*V(1) ] );
  delta = max( [ delta, abs( v-V(1) ) ] );
  v = V(2); % save the old "low" state ...
  V(2)  = max( [ beta*Rsearch - 3*(1-beta) + gamma*( (1-beta)*V(1) + beta*V(2) ), Rwait +
gamma*V(2), gamma*V(1) ] );
  delta = max( [ delta, abs( v-V(2) ) ] );

  %V, delta
  iterCnt=iterCnt+1;
end
```