Cost-effective Recovery of an Endangered Species: The Red-cockaded Woodpecker
Ryan M. Finseth and Jon M. Conrad
Land Economics 90:4, November 2014

```
%RCW Stochastic Dynamic Program

% Assign parameter values and initial conditions
alpha = .1;            % carrying capacity decay rate
m = 0;                 % mortality
s = .25;               % translocation success prob
delta = .05;           % discount rate
rho = 1/(1+delta);     % discount factor
c2 = 800;              % cost of constructing one artificial cavity
c1 = 3000;             % cost of translocating one breeding pair
r = .13;               % intrinsic growth rate
K0 = 30;               % carrying capacity at t = 0
MAXK = 50;             % max carrying capacity
N0 = 20;               % number of breeding pairs at t = 0
T = 10;                % time horizon
X1MAX = 6;             % max number of translocations per year
X2MAX = 10;            % max number of artificial cavities constructed
NTstar = 42;           % population target
X2MIN = 0;             % min number of artificial cavities constructed
X1MIN = 0;             % min number of translocations per year
ep = [.75,1,1.25];     % matrix of epsilon values
ep_prob = [.25,.5,.25]; % probability of each epsilon value
R = 5000;              % final function 'reward' value
Q = 40000;             % final function 'penalty' value

%Calculate final function
COST = zeros(MAXK+1,MAXK+1,T+1);
OPTX1 = zeros(MAXK+1,MAXK+1,T);
OPTX2 = zeros(MAXK+1,MAXK+1,T);
CHECK = zeros(MAXK+1,MAXK+1,T+1);
n = 0;
k = 0;
while n <= MAXK
   while (k >= n && k <= MAXK)
      if n > NTstar
         COST(k+1,n+1,T+1) = (rho^T)*R*(NTstar-n);
      elseif n == NTstar
         COST(k+1,n+1,T+1) = 0;
      else
         COST(k+1,n+1,T+1) = (rho^T)*Q*(NTstar-n);
      end
      k = k+1;
   end;
   n = n+1;
```

```
    k = n;
end;

% Begin DP at the time t = T-1
t = T;
while t >= 1
    X1 = X1MAX;
    X2 = X2MAX;
    n = 0;
    k = 0;
    while n <= MAXK
        while (k >= n && k <= MAXK)
            % Generate state transition probabilities for all ep(j)
            num = length(ep);
            for j = 1:num
                if k == 0
                    NT2(j) = ep(j)*(s*X1);
                else
                    NT2(j) = ep(j)*((s*X1)+n+(r*n)-(m*n)-((r*n*n)/k));
                end;
                KT2(j) = ((1-alpha)* k) + X2;
                NT2_floor(j) = floor(NT2(j));
                NT2_ceil(j) = ceil(NT2(j));
                KT2_floor(j) = floor(KT2(j));
                KT2_ceil(j) = ceil(KT2(j));
                K1_cprob(j) = KT2(j) - KT2_floor(j);
                K1_fprob(j) = KT2_ceil(j) - KT2(j);
                N1_cprob(j) = NT2(j) - NT2_floor(j);
                N1_fprob(j) = NT2_ceil(j) - NT2(j);
                if KT2_ceil(j) == KT2_floor(j)
                    K1_fprob(j) = .5;
                    K1_cprob(j) = .5;
                end;
                if NT2_ceil(j) == NT2_floor(j)
                    N1_fprob(j) = .5;
                    N1_cprob(j) = .5;
                end;
                if KT2_ceil(j) > MAXK
                    KT2_ceil(j) = MAXK;
                end;
                if KT2_floor(j) > MAXK
                    KT2_floor(j) = MAXK;
                end;
                NT2_floor(j) = min(NT2_floor(j),KT2_floor(j));
                NT2_ceil(j) = min(NT2_ceil(j),KT2_floor(j));
```

```
end;
% Compare expected costs of feasible recovery strategies
if (n + X1 <= k + X2)
   CHECK(k+1,n+1,t) = CHECK(k+1,n+1,t)+1;
   % Check if cost exists for state space combination
   if CHECK(k+1,n+1,t) == 1
      for j = 1:num
         excost(j) = ep_prob(j)*(((K1_fprob(j)*N1_fprob(j))*...
            COST(KT2_floor(j)+1,NT2_floor(j)+1,t+1))+...
            ((K1_cprob(j)*N1_fprob(j))*COST(KT2_ceil(j)+1,...
            NT2_floor(j)+1,t+1))+((K1_fprob(j)*N1_cprob(j))*...
            COST(KT2_floor(j)+1,NT2_ceil(j)+1,t+1))+...
            ((K1_cprob(j)*N1_cprob(j))*COST(KT2_ceil(j)+1,...
            NT2_ceil(j)+1,t+1)));
      end;
      % Assign cost and optimal recovery actions
      COST(k+1,n+1,t) = (rho^(t-1))*((c1*X1)+(c2*X2))+...
         sum(excost);
      OPTX1(k+1,n+1,t) = X1;
      OPTX2(k+1,n+1,t) = X2;
   else
      % Calculate expected cost
      for j = 1:num
         excost(j) = ep_prob(j)*(((K1_fprob(j)*N1_fprob(j))*...
            COST(KT2_floor(j)+1,NT2_floor(j)+1,t+1))+...
            ((K1_cprob(j)*N1_fprob(j))*COST(KT2_ceil(j)+1,...
            NT2_floor(j)+1,t+1))+((K1_fprob(j)*N1_cprob(j))*...
            COST(KT2_floor(j)+1,NT2_ceil(j)+1,t+1))+...
            ((K1_cprob(j)*N1_cprob(j))*COST(KT2_ceil(j)+1,...
            NT2_ceil(j)+1,t+1)));
      end;
      % Check to see if current recovery actions minimize
      if ((rho^(t-1))*((c1*X1)+(c2*X2))+sum(excost)) < ...
            COST(k+1,n+1,t)
         % Assign cost and optimal recovery actions
         COST(k+1,n+1,t) = ((rho^(t-1))*((c1*X1)+(c2*X2))+...
            sum(excost));
         OPTX1(k+1,n+1,t) = X1;
         OPTX2(k+1,n+1,t) = X2;
      end;
   end;
end;
% Iterate through all recovery action combinations
X1 = X1-1;
if X1 < X1MIN
```

```
                X1 = X1MAX;
                X2 = X2-1;
                if X2 < X2MIN
                    k = k+ 1;
                    X1 = X1MAX;
                    X2 = X2MAX;
                end;
            end;
        end;
        % Iterate through all state space combinations
        n = n+1;
        k = max(n,1);
    end;
    % Decrease time step, loop until t < 0
    t = t-1;
end;
```