A Numerical Toy Problem Solved by Dynamic Programming

P. Carpentier, J.-P. Chancelier, M. De Lara and V. Leclère

February 22, 2017

Contents

Abstract

In this practical work, you will solve a toy problem using the dynamic programming algorithm.

1 Formulating a stochastic optimization problem

1.1 Problem statement

We consider a control system given by the dynamics

$$\underbrace{x(t+1)}_{\text{future state}} = \underbrace{x(t)}_{\text{state}} + \underbrace{u(t)}_{\text{control}} + \underbrace{w(t+1)}_{\text{random perturbation}} , \tag{1}$$

where

- time $t \in \mathbb{T} = \{t_i, t_i + 1, \ldots, t_f - 1\}$ is discrete; t denotes the beginning of the period $[t, t+1[$; we call t_i the *initial time* and t_f the *horizon*;

- $x(t)$ is the value of the state at the beginning of period $[t, t+1[$, belonging to a *finite* set $\mathbb{X} = \{1, 2, \ldots, x^\sharp\}$, where $x^\sharp$ is an integer;
- $u(t)$ is the *control* during $[t, t+1[$, decided at the beginning of the time period $[t, t+1[$, and belonging to the *finite* set $\mathbb{U} = \{1, 0, -1\}$;
- $w(t+1)$ is a *random perturbation* occuring during the time period $[t, t+1[$, belonging to the *finite* set $\mathbb{W} = \{1, -1\}$; we assume that
  - the random variables $w(t_i + 1), \ldots, w(t_f)$ are independent;
  - each random variable $w(t)$ follows a uniform probability distribution on the set $\mathbb{W}$

$$\mathbb{P}_{w(t)}\{-1\} = \mathbb{P}_{w(t)}\{1\} = \frac{1}{2} \, . \tag{2}$$

The manager is supposed to make a decision $u(t)$ at each $t \in \mathbb{T}$, at the beginning of the period $[t, t+1[$, *before knowing* the random perturbation $w(t+1)$: such a setup is called *decision-hazard* setting. As the random perturbation $w(t+1)$ comes *after* the decision $u(t)$, it might be that the future state $x(t+1)$ in (1) goes outside the state set $\mathbb{X}$. Therefore, at the boundaries 1 and $x^\sharp$ of the state set $\mathbb{X}$, we have to limit the set of possible controls to ensure that the future state will remain in the prescribed bounds.

Question 1

(a) ***[3]*** *Show that is is possible to find, for each possible state $x \in \mathbb{X}$, a subset $\mathbb{B}(x)$ of $\mathbb{U}$ which will ensure that the next state — as computed by the system dynamics (1) with $x(t) = x$ — will remain in the set $\mathbb{X}$, whatever the random perturbation $w(t+1)$ and the decision $u(t) \in \mathbb{B}(x)$.*

(b) ***[2]*** *Write a* `Scicoslab` *function which returns the subset $\mathbb{B}(x)$ of $\mathbb{U}$ (as a row vector), given the state x as entry.*

Let $x_{\text{ref}} \in \mathbb{X}$ be a given reference state to be achieved at horizon time t_f. The aim of the manager is to minimize the final *cost*

$$K\big(x(t_f)\big) = \Big(x(t_f) - x_{\text{ref}}\Big)^2 \, . \tag{3}$$

The problem to solve is now

$$\min_{u(\cdot), x(\cdot)} \mathbb{E}\Big(\Big(x(t_f) - x_{\text{ref}}\Big)^2\Big) \, . \tag{4}$$

1.2 Numerical data

We will assume that

- $t_i = 1$ and $t_f = 10$;
- $x^\sharp = 9$ and $x_{\text{ref}} = 5$.

2 Writing the stochastic dynamic programming equation

2.1 Stochastic dynamic programming equation

Since the perturbations $w(t_i+1), \dots, w(t_f)$ are supposed to be independent random variables, Dynamic Programming applies and the equation associated with the problem of *minimizing the expected cost* (9) writes

$$\begin{aligned} V(t_f, x) &= K\big(x\big) \ , \\ V\big(t, x\big) &= \min_{u \in \mathbb{B}(x)} \mathbb{E}\Big(V\big(t+1, x+u+w(t+1)\big)\Big) \ , \end{aligned} \tag{5}$$

for time t varying from $t_f - 1$ to t_i. The expectation $\mathbb{E}$ is taken with respect to the marginal probability distribution $\mathbb{P}_{w(t+1)}$ given by (2).

2.2 Pseudo code for the stochastic dynamic programming equation

Writing the code to compute the Bellman function V in (5) (and the optimal control) follows a standard scheme, that we describe now. In the simple example of §1.1, we directly store the Bellman values in a 2-indices matrix $V = (V(t,x))_{t \in [t_i, t_f], x \in \mathbb{X}}$.

```
// Initialization
Tf = ZZZZ ;
Nx = ZZZZ ;
state = [1:Nx] ;
// Initialization of the matrix used to store Bellman values
V = ones(Tf,Nx) * %inf;
// Compute Bellman function at final time
V(Tf,:) = ZZZZ
// make a time backward loop
for t = Tf-1:-1:Ti
  // make a loop on possible states
  for x = ZZZZ
    // make a loop on admissible controls (for state x) to obtain the best possible
    // expected cost starting from state x in cost_x
    cost_x = %inf
    for u = ZZZZ
      // make a loop on the random perturbation to compute the expected
      // cost cost_x_u
      for w = ZZZZ
        // for a given perturbation compute the cost associated to
        // control u and state x using the instantaneous cost and the
        // Bellman function at time t+1
```

```
        cost_x_u_w = (cost_t(x,u) + V(t+1,ZZZZ))*p(w)
      end
      // update cost_x_u with cost_x_u_w
      // compare the current cost_x_u to cost_x and
      // update cost_x if cost_x_u is better than the current cost_x
      //
    end
    // store cost_x in V(t,x)
  end
end
```

Question 2

(a) ***[3]*** *Use the previous pseudo code pattern to program the computation of the Bellman equation* (5) *associated with the minimization problem described in §1.1.*

(b) ***[3]*** *Draw and comment on the obtained Bellman values.*

Question 3 ***[3]*** *Modify the previous code in order to also keep track of the optimal control function. As for the Bellman value function, you will use a 2-indices matrix* $U = (U(t,x))_{t\in[t_i,t_f],x\in\mathbb{X}}$ *to store the optimal control at time* t *when the current state is* x*. Do you find the optimal policy intuitive?*

3 Computing and evaluating an optimal policy

3.1 Computing an optimal policy

Note that, if we know the Bellman function V, we can use Equation (5) to compute the optimal control $u^\sharp(t,x)$ to be used for a given state x at time t by

$$u^\sharp(t,x) \in \arg\min_{u\in\mathbb{B}(x)} \mathbb{E}\Big(V\big(t+1, x+u+w(t+1)\big)\Big) \tag{6}$$

This way of doing is used when Bellman functions are computed on grids, but that the system dynamics produce states outside the grid. In that case, we use interpolation to compute Bellman values and we obtain better controls by recomputing them from Equation (5), than by using stored controls associated with states limited to the grid.

Question 4 *Write a function which returns the optimal control knowing current time* t *and state* x*. This function will use the previouly computed matrix* V *filled with Bellman optimal values.*

3.2 Evaluation of a given policy

An *admissible policy* $\gamma : \mathbb{T} \times \mathbb{X} \to \mathbb{U}$ assigns an admissible control, that is, $\gamma(t,x) \in \mathbb{B}(x)$. The optimal control obtained by dynamic programming is an admissible policy.

Given an admissible policy γ and a *(perturbation) scenario* $w(\cdot) = \big(w(t_i+1), \dots, w(t_f)\big)$, we are able to build a state trajectory $x(\cdot) = \big(x(t_i), \dots, x(t_f-1), x(t_f)\big)$ and a control trajectory $u(\cdot) = \big(u(t_i), \dots, u(t_f-1)\big)$ produced by the "closed-loop" dynamics initialized at the initial time t_i by

$$x(t_i) = x_i$$

and propagated from $t = t_i$ up to $t = t_f - 1$ according to

$$\begin{aligned} u(t) &= \gamma\big(t, x(t)\big) \\ x(t+1) &= x(t) + u(t) + w(t+1) \; . \end{aligned} \tag{7}$$

We also obtain the payoff associated with the scenario $w(\cdot)$

$$J^\gamma\big(t_i, x_i, w(\cdot)\big) = K\big(x(t_f)\big) \; , \tag{8}$$

where $x(\cdot)$ and $u(\cdot)$ are given by (7). The *expected payoff* associated with the policy γ is

$$\mathbb{E}\Big(J^\gamma\big(t_i, x_i, w(\cdot)\big)\Big) \; , \tag{9}$$

where the expectation $\mathbb{E}$ is taken with respect to the product probability $\mathbb{P}$, whose marginals are given by (2). The true expected value (9) is difficult to compute,[1] and we evaluate it by the *Monte Carlo* method using N_s scenarios $\big(w^1(\cdot), \dots, w^{N_s}(\cdot)\big)$:

$$\frac{1}{N_s} \sum_{s=1}^{N_s} J^\gamma\big(t_i, x_i, w^s(\cdot)\big) \; . \tag{10}$$

By the law of large numbers, the mean payoff (10) is a "good" approximation of the expected payoff(9) if the number of scenarios is "large enough".

We propose three different functions in order to evaluate the mean payoff associated with a policy given by a macro named `policy`.

The first version uses a Monte Carlo approach. The third argument of the function `N` is the number of requested simulation for Monte Carlo.

```
function [cost]=simulation_mc(x0,policy,N)
  cost = 0;
  for i = 1:N
    x = x0;
```

[1] Note however that this computation is achievable insofar all quantities it involves belong to finite sets.

```
    for t = 1:TF-1
      w = W(grand(1,1,'uin',1,2));
      x = x + policy(t,x) + w;
    end
    cost = cost + (x-xref)^2
  end
  cost = cost/N;
endfunction
```

The second version uses the law of the random perturbations to compute the expectation.

```
function [cost]=simulation_ex(x0,policy)
// Exact computation with the law of W
  Wa=all_w(TF-1);
  cost = 0;
  for i = 1: size(Wa,'r')
    x = x0;
    for t = 1:TF-1
      x = x + policy(t,x) + Wa(i,t);
    end
    cost = cost + (x-xref)^2
  end
  cost = cost/size(Wa,'r');
endfunction

function W=all_w(n)
// generated all the possible (W_1,...,W_(TF-1))
  if n == 1 then W=[-1;1]
  else
    Wn = all_w(n-1);
    W=[-1*ones(size(Wn,'r'),1),Wn;
       +1*ones(size(Wn,'r'),1),Wn];
  end
endfunction;
```

The third version uses dynamic programming.

```
function costs = simulation_dp(policy)
// evaluation by dynamic programming with fixed policy
  Vs = ones(TF,length(X))*%inf;
  // Bellman function at time TF
  Vs(TF,:) = (X-xref).^2;
  // Compute final value functions
  // Loop backward over time:
  for t = (TF-1):-1:1
    for x= 1:10
      // loop on noises
```

```
      EV=0;
      for iw = 1:size(W,"*")
        next_state = x + policy(t,x) + W(iw);
        EV = EV + P(iw)*Vs(t+1,next_state);
      end
      Vs(t,x) = EV;
    end
  end
  costs= Vs(1,:);
endfunction
```

Question 5 *Use the proposed simulation macros to evaluate the mean payoff associated with the optimal policy in* (6)*. Compare the different proposed solutions.*