Solving Control Problems with Linear State Dynamics – a Practical User Guide

Juri Hinz and Jeremy Yee
School of Mathematics
University of Technology Sydney
Sydney, Australia
Email: Juri.Hinz@uts.edu.au, Jeremy.Yee@uts.edu.au

***Abstract*—In industrial applications, practitioners usually face a considerable complexity when optimizing operating strategies under uncertainty. Typical real-world problems arising in practice are notoriously challenging from a computational viewpoint, requiring solutions to Markov Decision problems in high dimensions. In this work, we address a novel approach to obtain an approximate solution to a certain class of problems, whose state process follows a controlled linear dynamics. Our techniques is illustrated by an implementation within the statistical language R, which we discuss by solving a typical problem arising in practice.**

***Index Terms*—*Approximate dynamic programming, convex switching systems, Markov decision processes, optimal switching*

I. Introduction

Optimizing a decision policy within an industrial framework usually leads to a problem of sequential decision-making under uncertainty. This class of questions is addressed under the framework of *discrete-time stochastic control* and in most cases can be formulated under the umbrella of *Markov Decision Process* (see [1], [2], [4], and [9]). Closed form solutions to such problems are exceptions and usually an exact solution is out of reach and is not of primary importance in applications. For these reasons, approximate numerical solutions are targeted almost always in practice. Although a vast variety of computational methods have been developed in this area, the complexity of typical real-world questions usually goes beyond what is computationally feasible. To serve this need, the theory of *approximate dynamic programming* (see [8]) aims at providing a generalized view on theoretical insights, working solutions, and well-performing heuristics.

The accumulation of numerical inaccurateness is the main difficulty in the stepwise calculation of approximate solutions via backward induction (see [3]). Due to the interleaved application of numerical integration, the calculation of each value function relies on one which was obtained in the previous step. This concatenation causes a deviation from the true value functions, inevitably progressing with the number of time steps. This difficulty becomes severe for a generic real-world applications, since the underlying state variables in practice usually must be modeled in terms of high-dimensional controlled Markov processes. These issues cause a variety of problems, frequently referred to as the *curse of dimensionality*.

This research was partially supported under Australian Research Council's Discovery Projects funding scheme (project number: DP130103315).

In what follows, we focus on a specific type of Markov decision problems which frequently appear in practice. For this problem class, we present an efficient algorithmic solution. Furthermore, we introduce and discuss a corresponding solution diagnostics method, which is designed for quality assessment. Our diagnostics yields an estimation of the distance between a given approximate solution and the optimal one. Based on this, an a-posteriori justification of the approximate solution can be obtained, if its distance-to-optimality satisfies user-defined precision prerequisites. Otherwise, further solution attempts may be needed.

II. Markov decision theory

Let us review the classical finite-horizon Markov decision theory following [1]. On a finite time horizon $0, \dots, T$, consider a random dynamics whose state x evolves in E and is controlled by actions a from a finite action set A. For each $a \in A$, we assume that $K_t^a(x, dx')$ is a stochastic transition kernel on E. A mapping $\pi_t : E \mapsto A$ which describes the action that the controller takes at time t is called a *decision rule*. A sequence of decision rules $\pi = (\pi_t)_{t=0}^{T-1}$ is called a *policy*. For each initial point $x_0 \in E$ and each policy $\pi = (\pi_t)_{t=0}^{T-1}$, there exists a probability measure $\mathbb{P}^{x_0,\pi}$ and a stochastic process $(X_t)_{t=0}^T$ such that $\mathbb{P}^{x_0,\pi}(X_0 = x_0) = 1$ and

$$\mathbb{P}^{x_0,\pi}(X_{t+1} \in B \,|\, X_0, \dots, X_t) = K_t^{\pi_t(X_t)}(X_t, B) \qquad (1)$$

holds for each $B \subset E$ at all times $t = 0, \dots, T-1$. That is, given that system is in state X_t at time t, the action $a = \pi_t(X_t)$ is used to pick the transition probability $K_t^{a=\pi_t(X_t)}(X_t, \cdot)$ which randomly drives the system from X_t to X_{t+1} with the distribution $K_t^{\pi_t(X_t)}(X_t, \cdot)$. Let us use $\mathcal{K}_t^a$ to denote the one-step transition operator associated with the transition kernel K_t^a when the action $a \in A$ is chosen. In other words, for each action $a \in A$ the operator $\mathcal{K}_t^a$ acts on functions v by

$$(\mathcal{K}_t^a v)(x) = \int_E v(x') K_t^a(x, dx') \qquad x \in E, \qquad (2)$$

whenever the above integrals are well-defined. Now, let us turn to the definition of the control costs. For each time t, we are given the *t-step reward function* $r_t : E \times A \mapsto \mathbb{R}$, where $r_t(x, a)$ represents the reward for applying an action $a \in A$ when the state of the system is $x \in E$ at time t. At the end

of the time horizon, at time T, it is assumed that no action can be taken. Here, if the system is in a state x, a *scrap value* $r_T(x)$, which is described by a pre-specified *scrap function* $r_T : E \to \mathbb{R}$, is collected. Given an initial point x_0, the goal is to maximize the expected finite-horizon total reward, in other words to find the argument $\pi^* = (\pi_t^*)_{t=0}^{T-1}$ such that

$$\pi^* = \text{argmax}_\pi \mathbb{E}^{x_0,\pi} \left(\sum_{t=0}^{T-1} r_t(X_t, \pi_t(X_t)) + r_T(X_T) \right), \tag{3}$$

where A is the set of all policies, and $\mathbb{E}^{x_0,\pi}$ denotes the expectation over the controlled Markov chain defined by (1). The maximization (3) is well-defined under diverse additional assumptions (see [1], p. 199).

The calculation of the optimal policy is addressed in the following setting. For $t = 0, \ldots, T-1$, introduce the *Bellman operator*

$$\mathcal{T}_t v(x) = \sup_{a \in A} \left(r_t(x,a) + \mathcal{K}_t^a v(x) \right), \quad x \in E \tag{4}$$

which acts on each measurable function $v : E \to \mathbb{R}$ where the integrals $\mathcal{K}_t^a v$ for all $a \in A$ exist. Further, consider the *Bellman recursion*

$$v_T^* = r_T, \quad v_t^* = \mathcal{T}_t v_{t+1}^* \qquad \text{for } t = T-1, \ldots, 0. \tag{5}$$

Under appropriate assumptions, there exists a recursive solution $(v_t^*)_{t=0}^T$ to the Bellman recursion, which gives the so-called *value functions* and determines an optimal policy π^* via

$$\pi_t^*(x) = \text{argmax}_{a \in A} \left(r_t(x,a) + \mathcal{K}_t^a v_{t+1}^*(x) \right), \quad x \in E$$

for all $t = 0, \ldots, T-1$.

Consider now a Markov decision model whose state evolution consists of one discrete and one continuous component. To be more specific, we assume that the state space $E = P \times \mathbb{R}^d$ is the product of a finite space P and the Euclidean space $\mathbb{R}^d$. We suppose that the discrete component $p \in P$ is driven by a finite number of actions $a \in A$ in terms stochastic matrices

$$(\alpha_{p,p'}^a)_{p,p' \in P}$$

where $\alpha_{p,p'} \in [0,1]$ stands for the transition probability from $p \in P$ to $p' \in P$ if the action $a \in A$ was taken. Furthermore, we assume that the continuous state component evolves as an uncontrolled Markov process $(Z_t)_{t=0}^T$ on $\mathbb{R}^d$ whose evolution is driven by random linear transformations

$$Z_{t+1} = W_{t+1} Z_t$$

with pre-specified independent and integrable disturbance matrices $(W_t)_{t=1}^T$. Finally, let us assume that the reward functions

$$r_t(p, \cdot, a), \quad t = 0, \ldots, T-1, \quad p \in P, \quad a \in A$$

and scrap functions

$$r_T(p, \cdot), \quad p \in P$$

are convex and globally Lipschitz continuous in the continuous component $z \in \mathbb{R}^d$ of the state variable (p, z). In this setting, the transition operators are given by

$$\mathcal{K}_t^a v(p,z) = \sum_{p' \in P} \alpha_{p,p'}^a \mathbb{E}(v(p', W_{t+1} z)), \; z \in \mathbb{R}^d \tag{6}$$

for $t = 0, \ldots, T-1$, and $a \in A$. Such Markov decision problems are referred to as *convex switching systems* in what follows.

III. Solution algorithm

Following [5], the first step in obtaining a numerical solution to the backward induction

$$v_T^* = r_T, \qquad v_t^* = \mathcal{T}_t v_{t+1}^*, \qquad t = T-1, \ldots 0$$

is an appropriate discretization of the Bellman operator

$$\mathcal{T}_t v(p,z) = \max_{a \in A} \left(r_t(p,z,a) + \sum_{p' \in P} \alpha_{p,p'}^a \mathbb{E}(v(p', W_{t+1} z)) \right).$$

For this reason, we consider a modified Bellman operator $\mathcal{T}_t^n$ instead of $\mathcal{T}_t$ with the expectation $\mathbb{E}(v(p', W_{t+1} z))$ replaced by its numerical counterpart as

$$\sum_{k=1}^n \nu_{t+1}^n(k) v(p', W_{t+1}(k) z))$$

defined in terms of an appropriate distribution sampling $(W_{t+1}(k))_{k=1}^n$ of each disturbance W_{t+1} with corresponding probability weighting $(\nu_{t+1}^n(k))_{k=1}^n$. In the resulting modified backward induction

$$v_T^{(n)} = r_T, \qquad v_t^{(n)} = \mathcal{T}_t^n v_{t+1}^{(n)}, \qquad t = T-1, \ldots 0 \tag{7}$$

the functions $(v_t^{(n)})_{t=0}^T$ need to be described by algorithmically tractable objects. Since all reward and scrap functions are convex in the second variable, then the modified value functions (7) are also convex. We may then approximate these value functions in terms of piecewise linear and convex functions in the following manner. First, we introduce the so-called sub-gradient envelope $\mathcal{S}_G f$ of a convex function $f : \mathbb{R}^d \to \mathbb{R}$ on a grid $G \subset \mathbb{R}^d$ as

$$\mathcal{S}_G f = \vee_{g \in G} (\nabla_g f)$$

which is a maximum of the sub-gradients $\nabla_g f$ of f on all grid points $g \in G$. Using sub-gradient envelope operator, we define the double-modified Bellman operator as

$$\mathcal{T}_t^{m,n} v(p, \cdot) = \mathcal{S}_{G^m} \mathcal{T}_t^n v(p, \cdot),$$

where the operator $\mathcal{S}_{G^m}$ stands for the sub-gradient envelope on the grid $G^m = \{g^1, \ldots, g^m\}$. The corresponding backward induction

$$v_T^{(m,n)}(p, \cdot) = \mathcal{S}_{G^m} r_T(p, \cdot), \quad p \in P \tag{8}$$

$$v_t^{(m,n)}(p, \cdot) = \mathcal{T}_t^{m,n} v_{t+1}^{(m,n)}(p, \cdot), \qquad t = T-1, \ldots 0 \tag{9}$$

yields the so-called double-modified value functions $(v_t^{n,m})_{t=0}^T$ which enjoy excellent algorithmic properties.

Namely, the functions $(v_t^{(m,n)})_{t=0}^T$ are piece-wise linear and convex, they can be expressed using matrix representations. Note that any piecewise convex function f can be described by a matrix where each of the linear functionals is represented by one of the matrix's rows. To denote this relation, let us agree on the following notation: Given a function f and a matrix F, we write $f \sim F$ whenever $f(z) = \max(Fz)$ holds for all $z \in \mathbb{R}^d$. It turns out that the sub-gradient envelope operation $\mathcal{S}_G$ on a grid G corresponds to a specific row-rearrangement operator in the following sense

$$f \sim F \quad \Rightarrow \quad \mathcal{S}_G f \sim \Upsilon_G[F]$$

where the row-rearrangement operator Υ_G associated with the grid $G = \{g^1, \dots, g^m\} \subset \mathbb{R}^d$ acts on matrix F with d columns as follows:

$$(\Upsilon_G F)_{i,\cdot} = L_{\operatorname{argmax}(Fg^i),\cdot} \qquad \text{for all } i = 1, \dots, m. \quad (10)$$

Similarly, for the maximization holds

$$f_1 \sim F_1, f_2 \sim F_2 \quad \Rightarrow \quad f_1 \vee f_2 \sim F_1 \sqcup F_2$$

where $\sqcup$ stands for binding matrices by rows, which yields a matrix whose rows contain all rows from each participating matrix.

It turns out that the double-modified backward induction it can be rewritten in terms of Υ, $\sqcup$ and summations, applied to matrix representatives of the double-modified value functions:

Pre-calculations: Given a grid $G^m = \{g^1, \dots, g^m\}$, implement the row-rearrangement operator $\Upsilon = \Upsilon_{G^m}$ and the row maximization operator $\sqcup_{a \in A}$. Determine a distribution sampling $(W_t(k))_{k=1}^n$ of each disturbance W_t with the corresponding weights $(\nu_t(k))_{k=1}^n$ for $t = 1, \dots, T$. Given reward functions $(r_t)_{t=0}^{T-1}$ and scrap value r_T, determine the matrix representative of their sub-gradient envelopes

$$\mathcal{S}_{G^m} r_t(p, \cdot, a) \sim R_t(p, a), \qquad \mathcal{S}_{G^m} r_T(p, \cdot) \sim R_T(p)$$

for $t = 0, \dots, T-1$, $p \in P$ and $a \in A$. Introduce matrix representatives of each value function

$$v_t^{(m,n)}(p, \cdot) \sim V_t(p) \quad \text{for } t = 0, \dots, T,\ p \in P,$$

which are obtained via the following matrix form of the backward induction:

Initialization: Start with the matrices

$$V_T(p) = R_T(p), \qquad \text{for all } p \in P.$$

Recursion: For $t = T-1, \dots, 0$ calculate for $p \in P$

$$V_t(p) = \sqcup_{a \in A} \big(\Upsilon[R_t(p, a)] + \sum_{p' \in P} \alpha_{p,p'}^a \sum_{k=1}^n \nu_{t+1}(k) \Upsilon[V_{t+1}(p') \cdot W_{t+1}(k)] \big)$$

Having calculated matrix representatives $(V_t)_{t=0}^T$, the approximations $(v_t)_{t=0}^T$, $(v_t^E)_{t=0}^T$ of the value functions and their expectations are obtained as

$$v_t(p, z) = \max(V_t(p) z), \quad (11)$$

$$v_t^E(p, z) = \max(\sum_{k=1}^n \nu_t(k) \Upsilon[V_t(p) \cdot W_t(k)] z) \quad (12)$$

for all $z \in \mathbb{R}^d$, $t = 1, \dots, T$, and $p \in P$. Furthermore, an approximately optimal strategy $(\tilde{\pi}_t)_{t=0}^{T-1}$ is obtained for $t = 0, \dots, T-1$ by

$$\pi_t(p, z) = \operatorname{argmax}_{a \in A} (r_t(p, z, a) + \sum_{p' \in P} \alpha_{p,p'}^a v_{t+1}^E(p', z)), \quad (13)$$

IV. Diagnostics algorithm

Let us now turn to the diagnostics method whose proof is found in [7]. Suppose that a candidate $(\pi_t)_{t=0}^{T-1}$ for approximatively optimal policy is given. To estimate its distance-to-optimality, we address the performance gap $[v_0^\pi(p_0, z_0), v_0^{\pi^*}(p_0, z_0)]$ at a given starting point $z_0 = Z_0$. In what follows, we construct random variables $\underline{v}_0^{\pi,\varphi}(p_0, z_0)$, $\overline{v}_0^{\varphi}(p_0, z_0)$ satisfying

$$\mathbb{E}(\underline{v}_0^{\pi,\varphi}(p_0, z_0)) = v_0^\pi(p_0, z_0) \le \le v_0^{\pi^*}(p_0, z_0) \le \mathbb{E}(\bar{v}_0^{\pi,\varphi}(p_0, z_0)).$$

The calculation of the expectations $\mathbb{E}(\underline{v}_0^{\pi,\varphi}(p_0, z_0))$, $\mathbb{E}(\bar{v}_0^{\pi,\varphi}(p_0, z_0))$ is realized through an efficient recursive Monte-Carlo scheme, which yields approximations to $\mathbb{E}(\underline{v}_0^{\pi,\varphi}(p_0, z_0))$, $\mathbb{E}(\bar{v}_0^{\pi,\varphi}(p_0, z_0))$ along with appropriate confidence intervals.

For a practical application of the bound estimation, we assume that an approximate solution yields a candidate $(\pi_t)_{t=0}^{T-1}$ for an optimal strategy, as in (13) based on an approximations of the value and of the expected value functions as in (11), (12).

Bound estimation:

1) Chose a path number K and a nesting number $I \in \mathbb{N}$ to obtain for each $k = 1, \dots, K$ and $i = 0, \dots, I$ independent realizations $(w_t^{i,k})_{t=0}^T$ of the random variables $(W_t)_{t=0}^{T=1}$.
2) Define for $k = 1, \dots, K$ the state trajectories $(z_t^k)_{t=0}^T$ recursively

$$z_0^k := z_0, \quad z_{t+1}^k = w_{t+1}^{0,k} z_t^k, \qquad t = 0, \dots, T-1$$

and determine all realizations

$$\varphi_{t+1}^k(p, a) = \frac{1}{I} \sum_{i=1}^I \sum_{p' \in P} \alpha_{p,p'}^a v_{t+1}(p, w_{t+1}^{i,k} z_t^k) - \sum_{p' \in P} \alpha_{p,p'}^a v_{t+1}(p, w_{t+1}^{0,k} z_t^k).$$

for $t = 0, \dots, T-1$, $k = 1, \dots, K$, $p \in P, a \in A$.
3) For each $k = 1, \dots, K$ initialize the recursion at $t = T$ as

$$\bar{v}_T^{\pi,\varphi}(p, z_T^k) = r_T(z_T^k), \quad \underline{v}_T^{\pi,\varphi}(p, z_T^k) = r_T(z_T^k),\ p \in P$$

and continue for $t = T-1, \ldots, 0$ by

$$\bar{v}_t^{\pi,\varphi}(p, z_t^k) = \max_{a \in A} \big[r_t(p, z_t^k, a) + \varphi_{t+1}^k(p, a) + \sum_{p' \in P} \alpha_{p,p'}^a \bar{v}_{t+1}^{\pi,\varphi}(p', z_{t+1}^k) \big], \quad (14)$$

$$\underline{v}_t^{\pi,\varphi}(p, z_t^k) = r_t(p, z_t^k, \pi_t(z_t^k)) + \varphi_{t+1}^k(p, \pi_t(z_t^k)) + \sum_{p' \in P} \alpha_{p,p'}^{\pi_t(p, z_t^k)} \underline{v}_{t+1}^{\pi,\varphi}(p', z_{t+1}^k). \quad (15)$$

Having finished the calculation, store the values for $k = 1, \ldots, K, p \in P$ as

$$\bar{\psi}(p,k) := \bar{v}_0^{\pi,\varphi}(p, z_0^k) \quad \underline{\psi}(p,k) := \underline{v}_0^{\pi,\varphi}(p, z_0^k).$$

4) Calculate the sample means

$$\frac{1}{K} \sum_{k=1}^{K} \bar{\psi}(p,k), \quad \frac{1}{K} \sum_{k=1}^{K} \underline{\psi}(p,k)$$

to estimate the performance gap $[v_0^{\pi^*}(p, z_0), v_0^{\pi^*}(p, z_0)]$ from above and below, possibly using in-sample confidence bounds.

V. Example: A swing option

Let us consider the so-called swing option which a financial contract popular in energy business. In the simplest form, it gives the owner the right to obtain a certain commodity (gas, electricity) at a pre-specified price and volume at a number of exercise times which can be freely chosen by the contract owner, within the lifetime of the contract. Let us consider a specific case of such contract, referred to as a *unit-time refraction period* swing option. In this contract, there is a limit to exercise only one right at any time. Given the discounted commodity price $(S_t)_{t=0}^T$, the so-called fair price of a swing option with N rights is given by the supremum

$$\sup_{0 \le \tau_1 < \cdots < \tau_N \le T} \mathbb{E}\Big[\sum_{n=1}^{N} (S_{\tau_n} - K e^{-\rho \tau_n})^+ \Big]$$

over all stopping times $\tau_1, \ldots, \tau_N$ with values in $\{0, \ldots, T\}$. In order to represent this control problem as a switching system, we use the position set $P = \{1, \ldots, N+1\}$ to describe the number of exercise rights remaining. That is $p \in P$ stands for the situation when there are $p-1$ rights remaining to be exercised. The action set $A = \{1, 2\}$ represents the choice between exercising ($a = 1$) or not exercising ($a = 2$). The control matrices $(\alpha_{p,p'}^a)$ are given for exercise action $a = 1$

$$\alpha_{p,p'}^1 = \begin{cases} 1 & \text{if } p' = 1 \vee (p-1) \\ 0 & \text{else,} \end{cases}$$

and for not-exercise action $a = 2$ as

$$\alpha_{p,p'}^2 = \begin{cases} 1 & \text{if } p' = p \\ 0 & \text{else} \end{cases}$$

for all $p, p' \in P$. In the case of the swing option, the transition between p and p' occurs deterministically, since once the controller decides to exercise the right, the number of rights remaining is diminished by one. The deterministic control of the discrete component is easier to describe in therm of the matrix $(\alpha(p,a))_{p \in P, a \in A}$ where $p' = \alpha(p,a) \in P$ stands for the discrete component which is reached from $p \in P$ by the action $a \in A$. For the case of the swing option this matrix is

$$(\alpha(p,a))_{p \in P, a \in A} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 3 \\ \ldots & \ldots \\ N & N+1 \end{bmatrix}. \quad (16)$$

Having modeled the discounted commodity price process as an exponential mean-reverting process with a reversion parameter $\kappa \in [0, 1[$, long run mean $\mu > 0$ and volatility $\sigma > 0$, we obtain the logarithm of the discounted price process as

$$\tilde{Z}_{t+1} = (1-\kappa)(\tilde{Z}_t - \mu) + \mu + \sigma \epsilon_{t+1}, \quad \tilde{Z}_0 = \ln(S_0). \quad (17)$$

A further transformation of the state space is required before linear state dynamics can be achieved. If we introduce an augmentation with 1 via

$$Z_t = \begin{bmatrix} 1 \\ \tilde{Z}_t \end{bmatrix}, \qquad t = 0, \ldots, T.$$

then it becomes possible to represent the evolution as the linear state dynamics

$$Z_{t+1} = W_{t+1} Z_t, \qquad t = 0, \ldots, T-1$$

with independent and identically distributed matrix-valued random variables $(W_t)_{t=1}^T$ given by

$$W_{t+1} = \begin{bmatrix} 1 & 0 \\ \kappa\mu + \sigma\epsilon_{t+1} & (1-\kappa) \end{bmatrix}, \quad t = 0, ..., T-1.$$

The reward and scrap values are given by

$$r_t(p, (z^{(1)}, z^{(2)}), a) = (e^{z^{(2)}} - K e^{-\rho t})^+ \big(p - \alpha(p,a)\big) \quad (18)$$

for $t = 0, \ldots, T-1$ and

$$r_T(p, (z^{(1)}, z^{(2)})) = (e^{z^{(2)}} - K e^{-\rho T})^+ \big(p - \alpha(p,1)\big) \quad (19)$$

respectively for all $p \in P$ and $a \in A$.

VI. R Package

Having described the algorithms in Sections II and III, we now turn to their implementation in our R package *rcss*. The computational effort is done mostly at the C++ level and so our package represents a user friendly but efficient way to utilise these algorithms. We shall demonstrate its usage on the problem described in Section V by first setting our parameters.

```
## Parameters
rho <- 0
kappa <- 0.9
mu <- 0
sigma <- 0.5
K <- 0
n_dec <- 101  ## number of time epochs
N <- 5  ## number of rights
## Grid
n_grid <- 1001
```

```
grid <- cbind(rep(1, n_grid),
              seq(-2, 2, length = n_grid))
## Control matrix
control <- cbind(c(1,1:N), 1:(N+1))
```

Our grid is chosen to be equally spaced points on the interval $[-2, 2]$ and since our position evolves deterministically, we can represent its evolution by the two-dimensional matrix (16) with the interpretation that the (i, j)-th entry representing the next position after apply action j to current position i. We now specify our representation of the reward function using sub-gradients.

```
## Reward subgradient representation
reward <- array(0, dim = c(n_grid, 2,
                           nrow(control), 2, n_dec))
slope <- exp(grid[, 2])
for (t in 1:n_dec) {
    discount <- exp(-rho * (t - 1))
    for (p in 2:n_pos) {
        intercept <- (exp(grid[,2]) - K * discount) -
                     slope * grid[, 2]
        reward[, 1, p, 1, t] <- intercept
        reward[, 2, p, 1, t] <- slope
    }
}
```

The sub-gradients chosen to approximate the reward function are calculated using the gradients at grid points. Finally, an appropriate distribution sampling for the disturbance matrix is computed using quantiles of the standard normal distribution in the code shown below.

```
## Disturbance matrices
n_disturb <- 10000
disturb <- array(0, dim = c(2, 2, n_disturb))
disturb[1, 1,] <- 1
disturb[2, 2,] <- 1 - kappa
quantile <- qnorm(seq(0, 1, length = (n_disturb + 2)
                  )[c(-1, -(n_disturb + 2))])
disturb[2,1,] <- kappa * mu + sigma * quantile
r_index <- matrix(c(2, 1), ncol = 2)
disturb_weight <- rep(1/n_disturb, n_disturb)
```

After setting the above variables, we can now perform the Bellman recursion through the following function in our R package.

```
bellman <- FastBellman(grid, reward, control, disturb,
                       disturb_weight, r_index)
```

The efficiency of this method is due to next-neighbor approximations, whose details are described in [6]. The diagnostic algorithm presented in Section III requires a simulation of sample paths, which can be achieved through methods provided in our package.

```
## Generate and classify paths
n_path <- 1000
path_disturb <- array(0, dim = c(2, 2,
                                 n_dec - 1, n_path))
path_disturb[1, 1,,] <- 1
path_disturb[2, 2,,] <- 1 - kappa
path_disturb[2, 1,,] <- kappa * mu +
                        sigma * rnorm((n_dec - 1) * n_path)
start <- c(1, 0)
path <- Path(start, path_disturb)
path_nn <- Neighbour(matrix(path, ncol = 2),
                     grid, 1, "kdtree", 0, 1)$indices
```

The following code sets a nested simulation of disturbance realizations before computing the martingale increments.

```
## Set subsimulation disturbances
n_subsim <- 1000
subsim <- array(0, dim = c(2, 2, n_subsim,
                           n_path, n_dec - 1))
subsim[1, 1,,,] <- 1
subsim[2, 2,,,] <- 1 - kappa
subsim[2, 1,,,] <- kappa * mu +
                   sigma * rnorm(n_subsim * n_path *
                           (n_dec - 1))
subsim_weight <- rep(1/n_subsim, n_subsim)

## Find martingale increments
mart <- FastMartingale(bellman$value, path, path_nn,
                       subsim, subsim_weight, grid,
                       control = control)
```

To perform the upper and lower bound estimation, we use the reward function, which must be supplied to our method. The code required therefor is the usual function definition in R. This code is then passed into bounds estimation.

```
## Exact reward function
RewardFunc <- function(state, time) {
    output <- array(0, dim =
                    c(nrow(state), nrow(control) * 2))
    discount <- exp(-rho * (time - 1))
    for (i in 2:nrow(control)) {
        output[, 2 * i - 1] <-
            pmax(exp(state[, 2]) - K * discount, 0)
    }
    return(output)
}

## Duality
path_action <- PathPolicy(path, path_nn, control,
                          RewardFunc, bellman$expected,
                          grid)
duality <- Duality(path, control, RewardFunc, mart,
                   path_action)
```

For the initial state $\bar{Z}_0 = 0$, we obtain the following results:

Rights	CSS Value	99% Confidence Interval
1	3.246	[3.255, 3.270]
2	6.076	[6.090, 6.106]
3	8.670	[8.686, 8.703]
4	11.099	[11.18,11.135]
5	13.402	[13.423,13.441]

Thereby, the first column stands for the number of right

remaining, the second column yields the realization of the approximate value function whereas the last column represents an in-sample confidence interval for the value of the exact solution. For more information regarding the usage of the R package, we refer the reader to a detailed documentation of the R package *rcss*.

VII. Conclusion

We discuss an *a posteriori* justification of the numerical approximation to a novel class of algorithms which address stochastic switching problems with linear state dynamics. Using this method, a distance-to-optimality estimation of the approximate solution becomes possible due to sound and reliable diagnostics and quality assessment tool. The authors believe that such combination of efficient numerical schemes with a subsequent diagnostic check can be very helpful in applications and hope that practitioners find our implementation useful and interesting.

References

[1] N. Bäuerle and U. Rieder. *Markov Decision Processes with Applications to Finance*. Springer, Heidelberg, 2011.

[2] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[3] Bender C., Gärtner C., and Schweizer N. Pathwise dynamic programming. *Preprint*, 2015.

[4] E. A. Feinberg and A. Schwartz. *Handbook of Markov Decision Processes*. Kluwer Academic, 2002.

[5] J. Hinz. Optimal stochastic switching under convexity assumptions. *SIAM Journal on Control and Optimization*, 52(1):164–188, 2014.

[6] J. Hinz and N. Yap. Algorithms for optimal control of stochastic switching systems. *Theory of Probability and its Applications*, to appear.

[7] J. Hinz and J. Yee. An algorithmic approach to optimal asset liquidation problems. *Preprint*, 2015.

[8] W. B. Powell. *Approximate dynamic programming: Solving the curses of dimensionality*. Wiley, 2007.

[9] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.