EM Algorithm and Stochastic Control

Steven Kou[1], Xianhua Peng[2], Xingbo Xu[3]

[1]Risk Management Institute and Dept. of Mathematics
National University of Singapore
[2]Department of Mathematics
HKUST
[3]Goldman Sachs

Outline

Literature Review

- EM algorithm: Started with Dempster, Laird, and Rubin (1977), and thousands of papers after that. Google Citation over 45,000.
- Previous Monte Carlo methods for stochastic control, dynamic programming and BSDE: Kharroubi, Langren, and Pham (2013a, b), Zhang(2004), Crisan, Manolarakis, and Touzi (2010), Bouchard and Touzi (2004), etc.

Our Contribution

- We propose a *Control-EM (C-EM) Algorithm* for stochastic control problems. The implementation of C-EM can be achieved by using Monte Carlo simulation and the Stochastic Approximation (SA) algorithm (or other optimization algorithm, e.g. cross-entropy method).
- If the goal is do a static search for an optimal parameter, then the algorithm becomes the traditional EM algorithm.

Our Contribution

- We propose a *Control-EM (C-EM) Algorithm* for stochastic control problems. The implementation of C-EM can be achieved by using Monte Carlo simulation and the Stochastic Approximation (SA) algorithm (or other optimization algorithm, e.g. cross-entropy method).
- If the goal is do a static search for an optimal parameter, then the algorithm becomes the traditional EM algorithm.

1. We can deal with general stochastic processes, i.e., more than diffusions or Levy processes.
2. Similar to the EM, we show the monotonicity of performance improvement over each iterations, which leads to the convergence results.
3. Unlike many existing algorithms in Approximate Dynamics Programming (ADP) and reinforcement learning literature, we focus on finite-horizon problems, where the optimal policy is not necessarily stationary.

The Main Difficulty

- Traditionally, many stochastic control problems can be solved by the dynamic programming (Bellman equation).
- The main difficulty: Simulation is about going forward in time while dynamic programming is going backward.
- Our new algorithm does not rely on dynamic programming, and the algorithm goes iteratively forward and backward, and does regression on basis functions.

Comparing with Other Approximate Dynamic Programming

- Approximate Dynamic Programming:
  1. Many papers on value function improvement: Longstaff and Schwartz (2001), Tsitsiklis ad Van Roy (2001), Broadie and Glasserman (1997, 2004).
  2. Fewer results on policy improvement: Bertsekas (1999) and etc.
- Value function improvement approaches may not lead to the improvement of overall performance (Bartlett and Baxter, 2001); we also have a counterexample showing that, even starting with the optimal policy, the value function iteration may lead to suboptimal policy.
- On the contrary, our algorithms lead to increasing performance in each iteration, due to the monotonicity.

Comparing with Other Approximate Dynamic Programming

- Policy improvement approaches, e.g. in the book by Kushner and Dupuis (2013), aim at first approximating the underlying process using a Markov chain, and then computing the optimal policy analytically backwards on the Markov chain.
- The computing can be extensive as the Markov chain can be high dimension.
- In contrast, we only use simulation to compute the optimal policy, and thus we can solve high dimensional problems.

Introduction to EM

- The Expectation-Maximization (EM) Algorithm is an iterative method in statistics for finding MLE with missing data (see, e.g., Dempster et al. 1977).
- A typical maximum likelihood estimation problem can be formulated as
$$\max_{\theta\in\Theta} \int u(s,\theta) f(s,\theta) ds, \tag{1}$$
where u is some likelihood function; $f(s,\theta)$ is the probability density function of a random variable or random vector s (related to missing data); θ is the parameter to be estimated.
- The EM algorithm starts from initial guess θ_0 and iterates as below
$$\theta_{n+1} = \arg\max_{\theta\in\Theta} \int u(s,\theta) f(s,\theta_n) ds, n \geq 0. \tag{2}$$

Introduction to EM

- It can be broken down into two steps. First, in **Expectation** step (E-step), the expectation is estimated using θ_n obtained from previous iteration, i.e, the integral in (2). Then in the **Maximization** step (M-step), optimization is used to get an updated θ_{n+1}.
- The EM algorithm allows for very general distribution assumption for f; it also has monotonicity in each iteration, which leads to good convergence properties, e.g. Wu (1983).

1-Period Stochastic Control Problem Setup

- 1-Period problem:

$$\begin{aligned} \max_{c\in\Gamma} \quad & E\left[u(s,c)\right] \\ s.t. \quad & s = \psi(c,z). \end{aligned}$$

where

- $c \in \mathbb{R}^{n_c}$ is control policy
- u is the utility function
- z is the random source
- $s \in \mathbb{R}^{n_s}$ is state which is driven by random source z and policy c
- Γ is the policy space
- $E\left[u(s,c)\right] = \int u(\psi(c,z),c)f(z)dz$, where f is the density of z.

EM Algorithm for the One Period Stochastic Control Problem

- **EM Algorithm**: An iterative method for finding control policies

$$c_{n+1} = \arg\max_{c \in \Gamma} \int u(\psi(c_n, z), c) f(z) dz = \arg\max_{c \in \Gamma} \int u(s_n, c) f(z) dz$$

- It iteratively updates parameters (policies) using previous result c_n.
- **E-Step:** estimate $\int u(\psi(c_n, z), c) f(z) dz$ by using the previous c_n to get the state variable $s_n = \psi(c_n, z)$.
- **M-Step:** do optimization to get the updated result c_{n+1}.
- Monotonicity convergence results.
- It allows very general distributions of z and s.

Multi-Step Problem Setup

- For $t \geq 1$, we assume that $c_t = c(t, s_t, \theta_t), t \geq 1$, where $c(\cdot)$ is a function and $\theta_t = (\theta_{1,t}, \theta_{2,t}, \ldots, \theta_{d,t})^\top \in \mathbb{R}^d$ is the vector of parameters for the tth period.
- For example, one may assume that

$$c_t := \sum_{i=1}^{d} \theta_{i,t}\phi_{i,t}(s_t), t \geq 1, \tag{3}$$

  where $\{\phi_{i,t} : \mathbb{R}^{n_s} \to \mathbb{R}^{n_c}, i = 1, \ldots, d\}$ is the set of basis functions for the tth period.
- Path dependence can be accommodated by including auxiliary variables in s_t.
- The state evolution equation

$$s_{t+1} = \psi_{t+1}(s_t, c_t, z_{t+1}), \tag{4}$$

  where $\psi_{t+1}(\cdot)$ is the state evolution function and $z_{t+1} \in \mathbb{R}^{n_z}$ is the random vector denoting the random shock in the $(t+1)$th period.

Multi-Step Problem Setup

- At time 0, the decision maker wishes to choose the optimal control $c_0 \in \mathbb{R}^{n_c}$ and the sequence of control parameters $\theta_1, \ldots, \theta_{T-1}$, which determines the sequence of controls $c_1, \ldots, c_{T-1}$,
- To maximize the expectation of his or her utility

$$
\begin{aligned}
\max_{c_0,\theta_1,\ldots,\theta_{T-1}} \quad & E_0\left[\sum_{t=0}^{T-1} u_{t+1}(s_{t+1}, s_t, c_t)\middle| c_0, \theta_1, \ldots, \theta_{T-1}\right] && (5)\\
s.t. \quad & c_t = c(t, s_t, \theta_t), t = 0, 1, \ldots, T-1, && (6)\\
& s_{t+1} = \psi_{t+1}(s_t, c_t, z_{t+1}), t = 0, 1, \ldots, T-1, && (7)
\end{aligned}
$$

where $u_{t+1}(\cdot)$ is the utility function of the decision maker in the $(t+1)$th period; noting that utility function in the first period can include the utility at period 0.

Multi-Step Problem Setup

- A control problem that is more general than the problem (5) is

$$\max_{c_0,\theta_1,\ldots,\theta_{T-1}} \quad E_0\left[u(s_0, c_0, s_1, c_1, \ldots, s_{T-1}, c_{T-1}, s_T)\big|c_0, \theta_1, \ldots, \theta_{T-1}\right] \tag{8}$$

$$s.t. \qquad c_t = c(t, s_t, \theta_t), t = 0, 1, \ldots, T-1, \tag{9}$$

$$s_{t+1} = \psi_{t+1}(s_t, c_t, z_{t+1}), t = 0, 1, \ldots, T-1, \tag{10}$$

  where $u(s_0, c_0, s_1, c_1, \ldots, s_{T-1}, c_{T-1}, s_T)$ is a general utility function that may not be time separable as the one in (5).
- For simplicity of exposition, we will present our C-EM algorithm for the problem (5); however, the C-EM algorithm also applies to the general problem (8).

The Algorithm

1. Initialize $k = 1$ and $x^0 = (c_0^0, \theta_1^0, \theta_2^0, \ldots, \theta_{T-1}^0)$.
2. Iterate k until some stopping criteria are met. In the kth iteration, update $x^{k-1} = (c_0^{k-1}, \theta_1^{k-1}, \theta_2^{k-1}, \ldots, \theta_{T-1}^{k-1})$ to $x^k = (c_0^k, \theta_1^k, \theta_2^k, \ldots, \theta_{T-1}^k)$ by moving backwards from $t = T-1$ to $t = 0$ as follows:

(a) At time $T-1$, update θ_{T-1}^{k-1} to be θ_{T-1}^k such that

$$
\begin{aligned}
& E_0\left[u_T(s_T, s_{T-1}, c_{T-1}) \middle| c_0^{k-1}, \theta_1^{k-1}, \ldots, \theta_{T-2}^{k-1}, \theta_{T-1}^k\right] \\
\geq & E_0\left[u_T(s_T, s_{T-1}, c_{T-1}) \middle| c_0^{k-1}, \theta_1^{k-1}, \ldots, \theta_{T-2}^{k-1}, \theta_{T-1}^{k-1}\right]. \qquad (11)
\end{aligned}
$$

Such an θ_{T-1}^k can be set as a suboptimal (optimal) solution to the problem

$$
\max_{\theta_{T-1} \in \mathbb{R}^d} E_0\left[u_T(s_T, s_{T-1}, c_{T-1}) \middle| c_0^{k-1}, \theta_1^{k-1}, \ldots, \theta_{T-2}^{k-1}, \theta_{T-1}\right]. \qquad (12)
$$

The Algorithm, cont'd

(b) Move backward from $t = T-2$ to $t = 1$. At each time t, update θ_t^{k-1} to be θ_t^k such that

$$
\begin{aligned}
& E_0 \left[\sum_{j=t}^{T-1} u_{j+1}(s_{j+1}, s_j, c_j) \middle| c_0^{k-1}, \theta_1^{k-1}, \ldots, \theta_{t-1}^{k-1}, \theta_t^k, \theta_{t+1}^k, \ldots, \theta_{T-1}^k \right] \\
\geq & E_0 \left[\sum_{j=t}^{T-1} u_{j+1}(s_{j+1}, s_j, c_j) \middle| c_0^{k-1}, \theta_1^{k-1}, \ldots, \theta_{t-1}^{k-1}, \theta_t^{k-1}, \theta_{t+1}^k, \ldots, \theta_{T-1}^k \right].
\end{aligned}
\tag{13}
$$

Such an θ_t^k can be set as a suboptimal (optimal) solution to the problem

$$
\max_{\theta_t \in \mathbb{R}^d} E_0 \left[\sum_{j=t}^{T-1} u_{j+1}(s_{j+1}, s_j, c_j) \middle| c_0^{k-1}, \theta_1^{k-1}, \ldots, \theta_{t-1}^{k-1}, \theta_t, \theta_{t+1}^k, \ldots, \theta_{T-1}^k \right]
\tag{14}
$$

(c) At time 0, update c_0^{k-1} to be c_0^k such that

$$E_0\left[\sum_{j=0}^{T-1} u_{j+1}(s_{j+1}, s_j, c_j)\middle| c_0^k, \theta_1^k, \ldots, \theta_{T-1}^k\right] \geq E_0\left[\sum_{j=0}^{T-1} u_{j+1}(s_{j+1}, s_j, c_j)\right| \tag{15}$$

Such a c_0^k can be set as a suboptimal (optimal) solution to the problem

$$\max_{c_0 \in \mathbb{R}^{n_c}} E_0\left[\sum_{j=0}^{T-1} u_{j+1}(s_{j+1}, s_j, c_j)\middle| c_0, \theta_1^k, \ldots, \theta_{T-1}^k\right]. \tag{16}$$

Multi-Step Problem Setup

- In the C-EM algorithm, when we update θ_t^{k-1} to be θ_t^k or updating c_0^{k-1} to c_0^k, if no improvement of the objective function can be found, we simply set $\theta_t^k = \theta_t^{k-1}$ or set $c_0^k = c_0^{k-1}$.
- When we update θ_t^{k-1} to be θ_t^k or updating c_0^{k-1} to c_0^k, we need to evaluate the expectation in (12), (14), or (16), where the expectation is evaluated with all the parameters in other time periods fixed; this corresponds to the E-step in the EM algorithm. And then, the maximization in (12), (14), and (16) corresponds to the M-step in the EM algorithm.
- The C-EM algorithm does not use the dynamic programming principle, and hence it can be applied to stochastic control problem that do not satisfy the dynamic programming principle.

Monotonicity

- Theorem 1: The objective function $U(\cdot)$ defined in (5) monotonically increases in each iteration of the C-EM algorithm, i.e.,

$$U(c_0^k, \theta_1^k, \theta_2^k, \ldots, \theta_{T-1}^k) \geq U(c_0^{k-1}, \theta_1^{k-1}, \theta_2^{k-1}, \ldots, \theta_{T-1}^{k-1}), \forall k. \quad (17)$$

- Proof.

$$\begin{aligned}
& U(c_0^{k-1}, \theta_1^{k-1}, \theta_2^{k-1}, \ldots, \theta_{T-3}^{k-1}, \theta_{T-2}^{k-1}, \theta_{T-1}^{k-1}) \\
\leq\ & U(c_0^{k-1}, \theta_1^{k-1}, \theta_2^{k-1}, \ldots, \theta_{T-3}^{k-1}, \theta_{T-2}^{k-1}, \theta_{T-1}^{k}) \\
\leq\ & U(c_0^{k-1}, \theta_1^{k-1}, \theta_2^{k-1}, \ldots, \theta_{T-3}^{k-1}, \theta_{T-2}^{k}, \theta_{T-1}^{k}) \\
\leq\ & \cdots \\
\leq\ & U(c_0^{k-1}, \theta_1^{k}, \theta_2^{k}, \ldots, \theta_{T-3}^{k}, \theta_{T-2}^{k}, \theta_{T-1}^{k}) \\
\leq\ & U(c_0^{k}, \theta_1^{k}, \theta_2^{k}, \ldots, \theta_{T-3}^{k}, \theta_{T-2}^{k}, \theta_{T-1}^{k}), \quad (18)
\end{aligned}$$

which completes the proof.

Convergences of the Value Function to a Stationary Value

- Let $\{x^k\}_{k\geq 0}$ be the sequence of control parameters generated by the C-EM algorithm. In this subsection, we consider the issue of the convergence of $U(x^k)$ to a stationary value.
- We make the following assumptions on the objective function U:

$$\forall x^0 \text{ such that } U(x^0) > -\infty, \{x \in \mathbb{R}^n : U(x) \geq U(x^0)\} \text{ is compact.} \tag{19}$$

$$U(\cdot) \text{ is continuous and differentiable on } \mathbb{R}^n. \tag{20}$$

- Suppose the objective function $U(\cdot)$ satisfies (19) and (20). Then, we have

$$\{U(x^k)\}_{k\geq 0} \text{ is bounded above for any } x^0 \in \mathbb{R}^n. \tag{21}$$

- Define

$$\mathcal{M} := \text{set of local maxima of } U(\cdot) \text{ on } \mathbb{R}^n, \tag{22}$$

$$\mathcal{S} := \text{set of stationary points of } U(\cdot) \text{ on } \mathbb{R}^n, \tag{23}$$

Convergences of the Value Function to a Stationary Value

Theorem 2. Suppose the objective function U satisfies conditions (19) and (20). Let $\{x^k\}_{k\geq 0}$ be the sequence generated by the C-EM algorithm.

- Suppose that

$$U(x^k) > U(x^{k-1}) \text{ for any } x^{k-1} \notin \mathcal{S} (\text{resp. } x^{k-1} \notin \mathcal{M}). \tag{24}$$

  Then, all the limit points of $\{x^k\}_{k\geq 0}$ are stationary points (resp. local maxima) of U, and $U(x^k)$ converges monotonically to $U^* = U(x^*)$ for some $x^* \in \mathcal{S}$ (resp. $x^* \in \mathcal{M}$).
- Suppose that at each iteration k in the C-EM algorithm, θ_t^k and c_0^k are the optimal solution to the problems (12), (14), and (16) respectively. Then, all the limit points of $\{x^k\}$ are stationary points of U and $U(x^k)$ converges monotonically to $U^* = U(x^*)$ for some $x^* \in \mathcal{S}$.

Convergences of x^k

- Define

$$\mathcal{M}(a) := \{x \in \mathcal{M} : U(x) = a\},$$
$$\mathcal{S}(a) := \{x \in \mathcal{S} : U(x) = a\}.$$

- Under the conditions of Theorem 2, $U(x^k) \to U^*$ and all the limit points of $\{x^k\}$ are in $\mathcal{S}(U^*)$ (resp. $\mathcal{M}(U^*)$). However, this does not automatically imply the convergence of $\{x^k\}_{k \geq 0}$ to a point x^*.
- If $\mathcal{S}(U^*)$ (resp. $\mathcal{M}(U^*)$) consists of a single point x^*, i.e., there cannot be two different stationary points (resp. local maxima) with the same U^*, then $x^k \to x^*$. Hence, we have the following theorem.

Convergences of x^k

Theorem 3. Let $\{x^k\}_{k\geq 0}$ be an instance of a C-EM algorithm satisfying the conditions of Theorem 2, and let U^* be the limit of $\{U(x^k)\}_{k\geq 0}$.

- If $\mathcal{S}(U^*) = \{x^*\}$ (resp. $\mathcal{M}(U^*) = \{x^*\}$), then $x^k \to x^*$.
- If $\|x^{k+1} - x^k\| \to 0$ as $k \to \infty$, then, all the limit points of x^k are in a connected and compact subset of $\mathcal{S}(U^*)$ (resp. $\mathcal{M}(U^*)$). In particular, if $\mathcal{S}(U^*)$ (resp. $\mathcal{M}(U^*)$) is discrete, i.e., its only connected components are singletons, then x^k converges to some x^* in $\mathcal{S}(U^*)$ (resp. $\mathcal{M}(U^*)$).

Implementation of C-EM by Simulation and Stochastic Approximation(SA)

- The stochastic optimization problem at each time t is solved by using Stochastic Approximation.
- Stochastic Approximation (SA) is a simulation-based iterative algorithm for stochastic optimization (Robbins and Monro (1951), Kiefer and Wolfowitz (1952), Broadie, et al. (2011)).
- $\arg\min_x E[G(x,\xi)]$ or finding the root of $0 = E[G(x,\xi)]$.
- Other algorithms, e.g., Cross Entropy approach (Rubinstein and Kroese (2004)), can be alternatives.

Application 1: A Simple Stochastic Growth Model

We consider a simple stochastic growth problem as follows

$$\max_{c_t} \; E_0\left[\sum_{t=0}^{2} u_{t+1}(s_{t+1}, s_t, c_t)\right] = E_0\left[\sum_{t=0}^{2} \log c_t + \log s_3\right]$$

$$s.t. \;\; s_{t+1} = \left(s_t - \frac{s_t}{1+\exp(c_t)}\right)\exp(a + bz_{t+1}), t = 0, 1, 2 \qquad (25)$$

$$s_0 = 1$$

$$c_t \in \mathbb{R}, t = 0, 1, 2$$

where a is a constant, $b > 0$ is the volatility term, and $z_{t+1} \overset{d}{\sim} N(0, 1)$ are i.i.d. normally distributed random noise.

- At the t-th time period, the amount $\frac{s_t}{1+\exp(c_t)}$ is consumed from capital s_t,
- The remaining capital grows at rate $\exp(a + bz_{t+1})$.
- All wealth will be consumed in the end (at time $t = 3$).

The problem can be solve analytically with the following optimal controls and value functions

$$
\begin{aligned}
c_t^* &= \log(3-t),\;\; t=0,1,2. \\
V_0(s_0) &= 6a - 4\log 4 + 4\log s_0 \\
V_1(s_1) &= 3a - 3\log 3 + 3\log s_1 \\
V_2(s_2) &= a - 2\log 2 + 2\log s_2 .
\end{aligned} \tag{26}
$$

Application 1: A Simple Stochastic Growth Model

- To test our algorithm numerically, we choose $a = -0.1$ and $b = 0.2$.
- We use $N = 10^4$ sample paths and $m = 2000$ loops for the SA algorithm.

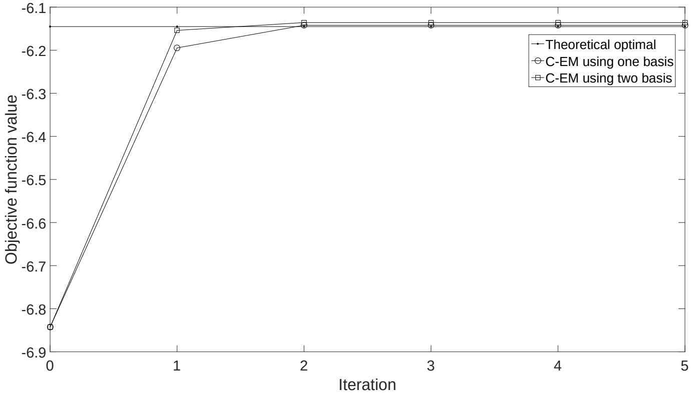
We consider two specification of basis functions. In the first specification, we use only one basis function

$$\phi_1(s) = s$$
$$c_t = \theta_{1,t}\phi_1(s_t).$$

In the second specification, we use two basis functions

$$\phi_1(s) = 1, \phi_2(s) = s,$$
$$c_t = \theta_{1,t}\phi_1(s_t) + \theta_{2,t}\phi_2(s_t).$$

The theoretical optimal control c_t^* lie in the space linearly spanned by the basis in the second specification but not in the first one. In the C-EM algorithm, we choose initial values of c_0 and θ_t to be $c_0^0 = 0, \theta_t^0 = 0, \forall t$.



Each iteration takes around 3 minutes. The theoretical optimal objective function value is -6.1452. The optimal objective function obtained by the C-EM algorithm is -6.1421 (7.4659e-03) with only one basis function and is -6.1358 (7.4755e-03) with two basis functions.

Application 2a: Single-Product Dynamic Pricing of Inventories

- A single-product dynamic pricing inventory problem in Gallego and Van Ryzin (1994). It is a finite-horizon problem with one state and one control.
- Suppose revenue within a short period $(t, t + \Delta t)$ is given by $r = p(\lambda_t)\Delta N^\lambda$, where λ_t is the sale intensity at time t, N^λ is a Poisson counting process with intensity λ_t, $p(\lambda_t)$ is the price at time t, and ΔN^λ is the number of arriving customers in the time interval $(t, t + \Delta t)$.

Application 2a: Single-Product Dynamic Pricing of Inventories

- The continuous-time problem is formulated as follows

$$
\begin{aligned}
V(n^c, T) = \sup_{p} \quad & E_0 \left[\int_0^T p_s dN_s^\lambda \right] \\
s.t. \quad & V(n^c, 0) = V(0, T) = 0, \text{for any } n^c \text{ and any } T \\
& N_T^\lambda \leq n^c, \\
& p_s = -\frac{1}{\alpha} \log \frac{\lambda_s}{a}, \text{ for } s \leq T,
\end{aligned}
\tag{27}
$$

where n^c is the total capacity at the beginning $t = 0$ and T is the time-to-maturity. The price is assumed to follow a parametric function depending on the sale intensity λ.

Application 2a: Single-Product Dynamic Pricing of Inventories

- In this problem, the state variable is the residual capacity $R_s = n^c - N_s^\lambda$ and the control is λ_s, which determines the sale price p_s and the dynamics of future arrivals N_{s+}. The capacity constraint is automatically taken care of because of the continuous setting.
- When $\alpha = 1$, the optimal solution given in Gallego and Van Ryzin (1994).

Application 2a: Single-Product Dynamic Pricing of Inventories

- We discretize the whole time horizon $[0, T]$ into n_T equal periods, denoted as $t_0 = 0, \ldots, t_{n_T} = T$. We choose c_{t_i} as the control and formulate the discrete problem as follows

$$\max_{c_{t_i}} \quad E_0 \left[\sum_{i=0}^{n_T - 1} p(\lambda_{t_i})(N^c_{t_{i+1}} - N^c_{t_i}) \right]$$

$$s.t. \quad N^\lambda_{t_{i+1}} - N^\lambda_{t_i} \sim \text{Poisson}(\lambda_{t_i} \Delta t), i = 0, 1, \ldots, n_T - 1$$

$$N^c_{t_{i+1}} - N^c_{t_i} = \min(n^c - N^c_{t_i}, N^\lambda_{t_{i+1}} - N^\lambda_{t_i}), i = 0, 1, \ldots, n_T - 1 \tag{28}$$

$$p(\lambda_{t_i}) = -\frac{1}{\alpha} \log \frac{\lambda_{t_i}}{a}, i = 0, 1, \ldots, n_T - 1$$

$$\lambda_{t_i} = \frac{a}{1 + \exp(c_{t_i})}, i = 0, 1, \ldots, n_T - 1$$

$$c_{t_i} \in \mathbb{R}, i = 0, 1, \ldots, n_T - 1.$$

Single-Product Dynamic Pricing of Inventories

	$n^c = 20$			$n^c = 10$			$n^c = 5$		
	Theoretical		C-EM	Theoretical		C-EM	Theoretical		C-EM
	continuous	discrete		continuous	discrete		continuous	discrete	
Mean	7.3576	7.3494	7.3777	7.2231	7.2207	7.2237	6.000	5.8964	5.9419
Stderr	N/A	0.0271	0.0270	N/A	0.0257	0.0260	N/A	0.0205	0.0204

We discretize the time horizon $[0,1]$ into $n_T = 4$ equal periods. We use $N = 10,000$ sample paths in the C-EM algorithm, and we use 1000 iteration in the SA algorithm. We specifies the control c_t as the linear combination of three basis functions:

$$c_t = \theta_1^t \phi_1(R_t) + \theta_2^t \phi_2(R_t) + \theta_3^t \phi_3(R_t),$$
$$\phi_i(R) = R^i, i = 0, 1, 2.$$

We choose initial values of c_0 and θ_t to be $c_0^0 = 0, \theta_t^0 = 0, \forall t.$

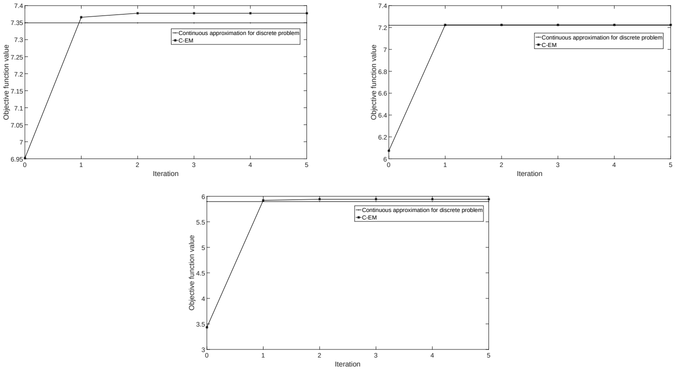


Figure: $n_c = 20$, $n_c = 10$, $n_c = 5$. The C-EM converges in about 2 iterations. Each iteration takes about 3 minutes.

Application 2b: Multi-Product Dynamic Pricing of Inventories

- Consider an airline network sales problem with n_r legs, n_i itineraries.
- Example: A network with three nodes $\{1, 2, 3\}$, two legs $\{1 \to 2, 2 \to 3\}$, and three itineraries $\{1 \to 2, 2 \to 3, 1 \to 2 \to 3\}$.
- Prices of itineraries $p \in \mathbb{R}^{n_i}$, customer arrival rate $\lambda \in \mathbb{R}^{n_i}$. Initial capacity $n^c \in \mathbb{R}^{n_r}$.
- $A = [a_{ij}] \in \mathbb{R}^{n_i \times n_r}$ defines whether flight leg j is a part of itinerary i using $a_{ij} \in \{0, 1\}$.
- $$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$
- The problem is proposed (Gallego and Van Ryzin (1997))

Application 2b: Multi-Product Dynamic Pricing of Inventories

- Continuous version

$$\sup_{\lambda} E\left[\int_0^T p_s^\top dN_s^\lambda\right],$$

$$s.t.\ \int_0^T A^\top dN_s^\lambda \leq n^c,\ \ p^j(\lambda) = (\epsilon_{0,j}^{-1}\log\frac{\lambda_0^j}{\lambda^j}+1)p_0^j, \text{for } j=1,\ldots,n_i.$$

$$V(n,0)=V(0,t)=0,\ \forall n\in\mathbb{N}^{n_r},\forall t>0.$$

- Difficult to solve the HJB equation

Application 2b: Multi-Product Dynamic Pricing of Inventories

- We focus on a discrete-time setting of the problem.

$$\begin{aligned}
\max_{c} \quad & E_0\left[\sum_{k=0}^{n_T-1} p(\lambda_{t_k})^\top (N^c_{t_{k+1}} - N^c_{t_k})\right] \\
s.t. \quad & N^{\lambda,j}_{t_{k+1}} - N^{\lambda,j}_{t_k} \sim Poisson(\lambda^j_{t_k}\Delta t), j = 1, \ldots, n_i \\
& N^c_{t_{k+1}} = G(n^c, N^c_{t_k}, N^\lambda_{t_{k+1}} - N^\lambda_{t_k}) \\
& p^j(\lambda^j_{t_k}) = (\epsilon_{0,j}^{-1} \log \frac{\lambda_{0,j}}{\lambda^j_{t_k}} + 1) p_{0,j}, j = 1, \ldots, n_i \\
& \lambda^j_{t_k} = \min(\lambda_{0,j} e^{\epsilon_{0,j}}, \max(c^j_{t_k}, 0)), j = 1, \ldots, n_i, \\
& c^j_{t_k} \in \mathbb{R}, j = 1, \ldots, n_i.
\end{aligned} \tag{29}$$

- The control of the problem is $c_{t_k} = (c^1_{t_k}, \ldots, c^{n_i}_{t_k})^\top$. The state variables of the problem are the residual capacities $R_{t_k} = n^c - AN^c_{t_k}$.

Example: Multi-Product Dynamic Pricing of Inventories

Deterministic benchmarks (by Gallego and Van Ryzin (1997)):

- For its continuous version, the HJB equation does not not analytical solution.
- Gallego and Van Ryzin (1997) gave two heuristic policies, MTS and MTO, by considering their deterministic versions, and showed their asymptotic optimality.
- The deterministic versions are solved via a constrained convex programming.
- MTO allows itineraries to share airline capacity when available, while MTS does not.
- Both MTO and MTS assume stationary policies, while our problem has a much higher dimension.
- Other approaches via the value function approximation: Bertsimas & de Boer (2005), Adelman (2007) and etc.

Example: Multi-Product Dynamic Pricing of Inventories

- Consider $T = 1$, $n_T = 6$, $N = 100$. The total running time is about 5 hours (SA is the bottleneck).
- A network with three nodes $\{1, 2, 3\}$, two legs $\{1 - 2, 2 - 3\}$, and three itineraries $\{1 - 2, 2 - 3, 1 - 2 - 3\}$.
- State variables: $R_{ij} = n^c_{ij} - N_{ij}$ for $(i, j) \in \{(1, 2), (2, 3)\}$.
- Policy: $\lambda = [\lambda_{12}, \lambda_{23}, \lambda_{123}]^\top$.
- Assume linear basis (linear of states)

$$\begin{cases} \phi^1_{12}(R) & = [1, 0, 0]^\top \\ \phi^2_{12}(R) & = [R_{12}, 0, 0]^\top \\ \phi^3_{12}(R) & = [R_{23}, 0, 0]^\top \end{cases} \quad \begin{cases} \phi^1_{23}(R) & = [0, 1, 0]^\top \\ \phi^2_{23}(R) & = [0, R_{12}, 0]^\top \\ \phi^3_{23}(R) & = [0, R_{23}, 0]^\top \end{cases} \quad \begin{cases} \phi^1_{123}(R) & = [0, 0, 1]^\top \\ \phi^2_{123}(R) & = [0, 0, R_{12}]^\top \\ \phi^3_{123}(R) & = [0, 0, R_{23}]^\top \end{cases}$$

- Denote the corresponding coefficients by $\{\theta_{k,l}\}$ for $l \in \{(1, 2), (2, 3), (1, 2, 3)\}$ and $k = 1, 2, 3$.
- The initial policies are set to be equal to the optimal deterministic controls, i.e.,

$$\theta^0_{2,l} = \theta^0_{3,l} = 0, \theta^0_{1,l} = \lambda^{d,*}_l$$

Applicatinn 2b: Multi-Product Dynamic Pricing of Inventories

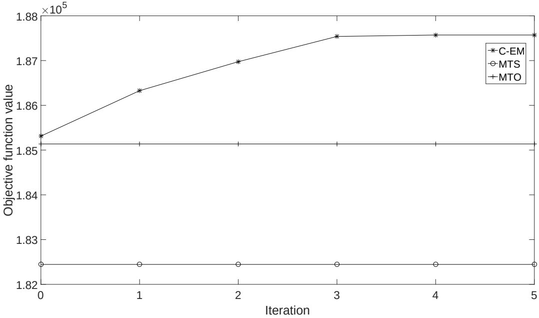


Figure: The algorithm converged after 4 iterations. It uses $N = 10,000$ sample path in the simulation. It takes 1.3 hours for each iteration. The optimal revenue is 1.8757×10^5 (with standard error 52).

Example: Multi-Product Dynamic Pricing of Inventories

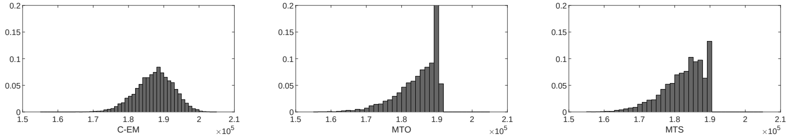


Figure: Histograms of total revenue with optimal control (left), heuristic MTO (middle) and heuristic MTS (right). Based on 10,000 simulation sample paths.

Example: Multi-Product Dynamic Pricing of Inventories

	Total revenue			Revenue at 3rd period			Revenue at 6th period		
	C-EM	MTO	MTS	C-EM	MTO	MTS	C-EM	MTO	MTS
mean	187.57	185.06	182.48	31.82	31.69	30.70	29.45	26.78	24.71
stderr	0.05	0.06	0.06	0.04	0.04	0.04	0.04	0.06	0.06
skewness	-0.30	-1.36	-1.00	0.14	0.14	0.09	-0.38	-0.70	-0.35
kurtosis	3.06	4.74	3.80	2.94	2.96	3.00	3.21	3.83	3.11
1% quantile	174.02	166.56	165.16	22.48	22.52	21.26	19.86	9.15	10.34
5% quantile	178.17	173.16	170.97	25.09	24.99	24.02	22.94	15.69	14.70
95% quantile	195.57	190.57	189.29	39.01	38.81	37.67	35.18	35.21	33.26
99% quantile	198.62	190.89	189.29	41.82	41.85	40.73	37.07	38.48	36.41

Table: Dynamic multi-product pricing inventories : statistics of revenues (in units of 1,000). Stderr indicates the standard error of the mean estimation.

Example: Mult-Product Dynamic Pricing of Inventories

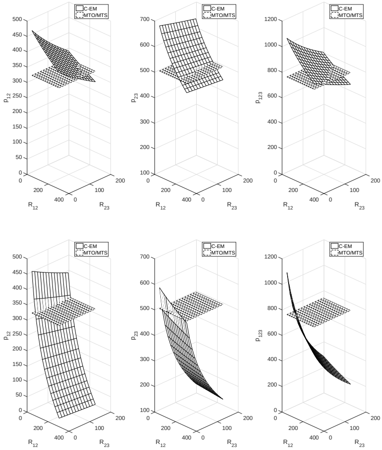


Figure: The airline ticket prices. The first row plots the prices at the beginning of the 3rd period (at time $t_2 = 1/3$), and the second row plots the prices at the beginning of the 6th period (at time $t_5 = 5/6$).

Application 3: Real Business Cycle

- We follow the models in Christiano (1990) to compare the log-linear LQ approximation for real business cycle model as in Kydland and Prescott (1982), Long and Plosser (1983), and Hansen (1985).
- The original infinite horizon problem:

$$\begin{aligned} \max_{g_t} \quad & E_0\left[\sum_{t=0}^{\infty} \beta^t u(k_{t-1}, k_t, x_t)\right] = E_0\left[\sum_{t=0}^{\infty} \beta^t \frac{g_t^{1-\tau}}{1-\tau}\right] \qquad (30) \\ s.t. \quad & x_{t+1} = \rho x_t + \epsilon_{t+1}, t \geq 0 \\ & k_t = \exp(x_t) k_{t-1}^{\gamma} - g_t + (1-\delta) k_{t-1}, t \geq 0 \\ & g_t \in [0, \exp(x_t) k_{t-1}^{\gamma} + (1-\delta) k_{t-1}], t \geq 0 \end{aligned}$$

where (k_{-1}, x_0) is given as the initial state at time $t = 0$; x_t is the technology innovation level at period t, $\exp(x_t)k_{t-1}^{\gamma}$ is the total production at period t; g_t is the consumption at period t; k_t is the end-of-period-t capital, which depends on the depreciation rate of capital δ. The state of the model at time t is $s_t = (k_{t-1}, x_t)$.

Application 3: Real Business Cycle

- Infinite-horizon (IH) version is well studied.
- Log-linear LQ approximates the objective function with linear-quadratic functions. Then it is solved analytically by Linear-Quadratic programming.
- Not suitable for the finite-horizon (FH) problem:
  1. The solution is not stationary for FH problem.
  2. FH problem has a much higher dimension. IH problem implicitly assumes only optimization only for one period.

- We consider the finite horizon version as follows

$$\max_{c_t, 0\le t\le T-1} \quad E_0\left[\sum_{t=0}^{T} \beta^t u(k_{t-1}, k_t, x_t)\right] = E_0\left[\sum_{t=0}^{T} \beta^t \frac{g_t^{1-\tau}}{1-\tau}\right] \tag{31}$$

$$\begin{aligned} s.t.\quad & x_{t+1} = \rho x_t + \epsilon_{t+1}, 0 \le t \le T-1 \\ & k_t = \exp(x_t) k_{t-1}^{\gamma} - g_t + (1-\delta) k_{t-1}, 0 \le t \le T-1 \\ & g_t = \frac{1}{1+\exp(c_t)}\left[\exp(x_t) k_{t-1}^{\gamma} + (1-\delta) k_{t-1}\right], 0 \le t \le T \\ & g_T = \exp(x_T) k_{T-1}^{\gamma} + (1-\delta) k_{T-1}, \qquad (32) \\ & c_t \in \mathbb{R}, 0 \le t \le T-1, \end{aligned}$$

where (32) means that the available capital at period T is all consumed at period T. Hence, the last period utility of problem (31) is given by

$$u_T(s_T, s_{T-1}, c_{T-1}) = \beta^{T-1} \frac{g_{T-1}^{1-\tau}}{1-\tau} + \beta^T \frac{g_T^{1-\tau}}{1-\tau}.$$

Application 3: Real Business Cycle

Suppose the problem parameters are $\beta = 0.98$, $\gamma = 0.33$, $\tau = 0.5$, $\delta = 0.025$, $\rho = 0.95$, and $\epsilon_t \sim N(0, \sigma_e^2)$ with $\sigma_e = 0.1$. The initial state is $s_0 = (k_{-1}, x_0) = (k^*, 0)$. The control c_t is specified as

$$c_t = \sum_{i=1}^{4} \theta_{i,t} \phi_i(k_{t-1}, x_t),$$

where $\{\phi_i\}$ are the basis functions defined as

$$\begin{aligned}
\phi_1(k_{t-1}, x_t) &= 1, \\
\phi_2(k_{t-1}, x_t) &= k_{t-1}, \\
\phi_3(k_{t-1}, x_t) &= \exp(x_t), \\
\phi_4(k_{t-1}, x_t) &= k_{t-1}^{\gamma}.
\end{aligned}$$

In the C-EM algorithm, we initialize $c_0^0 = 0, \theta_t^0 = 0, \forall t$. We simulate $N = 10,000$ sample paths in the C-EM algorithm, and we use 2000 iterations in the SA algorithm.

Example: Real Business Cycle

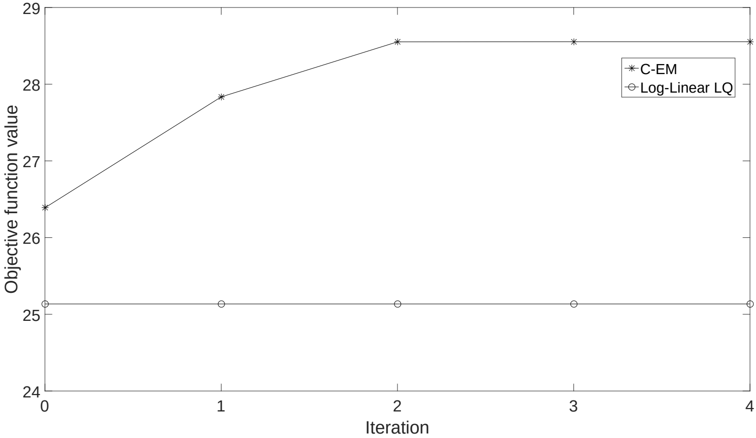
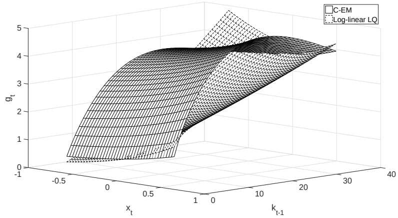


Figure: $T = 6$. The C-EM algorithm converges after 3 iterations. It takes 18 minutes for each iteration.



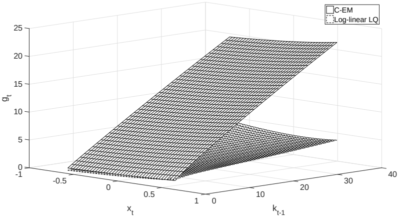


Figure: $T = 6$. The top figure plots g_t for $t = 2$, and the bottom one plots g_t for $t = 5$, which is the second to the last period.

Example: Real Business Cycle

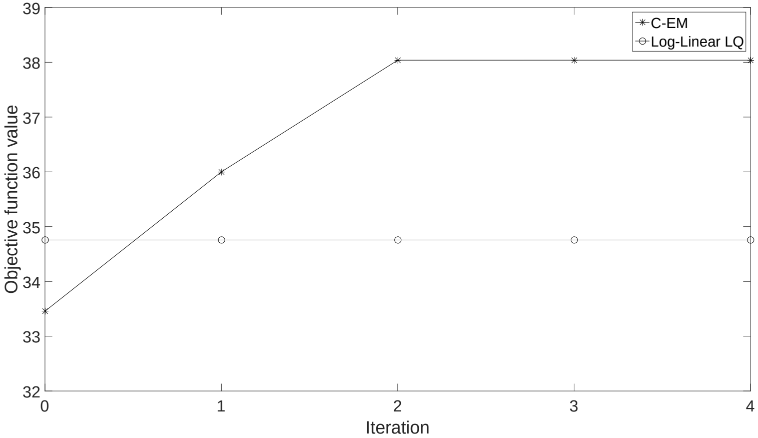


Figure: $T = 10$. The C-EM algorithm converges after 3 iterations. It takes 18 minutes for each iteration.

Example: Real Business Cycle

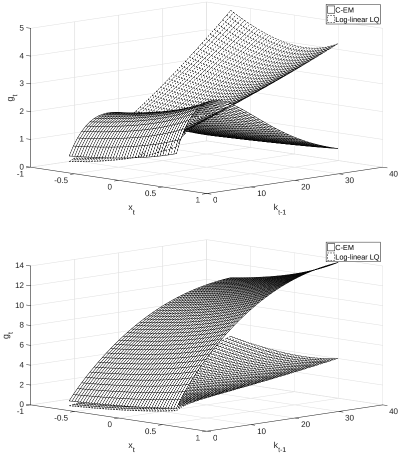


Figure: $T = 10$. The top figure plots g_t for $t = 2$, and the bottom one plots g_t for $t = 9$, which is the second to the last period.

Thank you!

Example: Comparison with Value Function Approximation

- Value Function Improvement: it approximates and keeps tracks of value function. The policy from the approximated function may not necessarily lead to performance improvement.
- Consider the state space is $\{1, 2\}$ and policy space is $\{a_1, a_2\}$. For each policy, the state follows a discrete-time Markov process as follows

$$P(a_1) = \begin{bmatrix} 1/3 & 2/3 \\ 1/3 & 2/3 \end{bmatrix}, \quad P(a_2) = \begin{bmatrix} 2/3 & 1/3 \\ 2/3 & 1/3 \end{bmatrix}.$$

- The objective is

$$\max_c E_{s_1}\left[\sum_{t=1}^{T-1} \alpha^t u(s_t) + \alpha^T u_T(s_T)\right], \text{ for } \alpha \in (0,1) \text{ and } s_1 \text{ given}$$

where $u(1) = 0, \quad u(2) = 1.$

Example: Comparison with Value Function Approximation

- For the infinite-horizon version of the problem, the optimal value function is
$$J(1) = \frac{2\alpha}{3-3\alpha}, \quad J(2) = 1 + \frac{2\alpha}{3-3\alpha}$$
The optimal policy is $c^*(s) = a_1$ for any s.
- Choosing $u_T = J$, the optimal policy for the finite-horizon problem is also $c_t^*(s) = a_1$ for $t = 0, ..., T-1$.
- Suppose we use basis
$$\phi(1) = 2, \quad \phi(2) = 1.$$
- Approximate value function $\hat{J}_t(i) = w_t \phi(i)$ for any $i \in \{1, 2\}$. Where
$$w_t = \arg\min_w E_{s_{t-1}, c_{t-1}} \left[w\phi(s_t) - J_t(s_t) \right]^2.$$

Example: Comparison with Value Function Approximation

- Suppose we use basis

$$\phi(1) = 2, \quad \phi(2) = 1.$$

- Approximate value function $\hat{J}_t(i) = w_t\phi(i)$ for any $i \in \{1, 2\}$. Where

$$w_t = \arg\min_{w} E_{s_{t-1},c_{t-1}} \left[w\phi(s_t) - J_t(s_t)\right]^2. \tag{33}$$

- Initialize with $J_t = J$.
- Start backward from T, (33) gives $w_T^1 = \frac{2+\alpha}{9-9\alpha} > 0$.
- So $\hat{J}_T(1) > \hat{J}_T(2) \Rightarrow \hat{c}_{T-1}^1(s) = a_2 \neq c_{T-1}^*(s)$. Suboptimal!
- If continue with the iteration, the suboptimal policy still cannot be improved over multiple rounds.