CARLETON UNIVERSITY

SCHOOL OF
MATHEMATICS AND STATISTICS

HONOURS PROJECT

TITLE: Dynamic Programming and Its Applications

AUTHOR: Qian Li

SUPERVISOR: Minyi Huang

DATE: January 7, 2019

# Honours Project

## Dynamic Programming Problem and Its Application

# Contents

## Abstract

In this paper, I will introduce a dynamic programming problem and its applications based on the Markov decision process. I will introduce deterministic dynamic programming first and give an example of a factory production problem to show how the deterministic dynamic programming works. In the second part, I will introduce the stochastic dynamic programming. Stochastic dynamic programming is a technique for modelling and solving problems of decision making under uncertainty. I will divide stochastic dynamic programming into two parts. One is the stochastic finite horizon; another is the stochastic infinite horizon. Finally, I will give the real world example about oil storage program to show how dynamic programming works.

## Acknowledgment

First and foremost, I would like to show my deepest gratitude to my supervisor, Dr. Minyi Huang, a respectable and resourceful professor, who has provided me with valuable guidance in every stage of the writing of this thesis. Without his enlightening instruction, impressive kindness and patience, I could not have completed my project. Thanks to Professor Gennady for taking the time to look at my paper and taught me knowledge about Markov decision progress. I am also greatly indebted to all my professors who have taught me in the past four years. They help me to develop the fundamental and essential academic competence to complete this thesis.

# 1 Introduction

R.Bellman and others proposed dynamic programming in the early 1950s. Based on the characteristics of multi-stage decision-making problems, they put forward the optimization principle to solve such problems and successfully solved many practical problems. In 1960, R.A. Howard combined dynamic programming with the Markov decision process to further extend the application of this method. The optimization principle of dynamic programming is: "As the optimal strategy of the whole process, it has the property that regardless of the past state and decision, the remaining decisions must follow the optimal strategy for the state formed by the previous decision. "[1]. When we use dynamic programming methods to solve multi-stage decision problems, we must first convert the actual problems into dynamic programming models. Dynamic System has the form[1]:

$$x_{k+1} = f_k\left(x_k, u_k, w_k\right), k = 0, 1, 2, \cdots, N-1$$

At time $k \in T$, where T is the discrete time interval $[0, T], 0, 1, \cdots, N$, the state space is $S_k$.

$x_k \in S_k$ is the state of the system which contained the past information before time k and related to future optimization. The initial state is $x_0 = S_0$.

$u_k$ is control of decision variable depending on $x_k$ at time k. In real life problem, $u_k$ is often restricted to some range. The range is called the action space, which is $u_k \in A_k(x_k)$.

$w_k$ is a random variable. It acts as disturbance, $\mathbb{N}$ is the time horizon.

$f_k$ is the function describing the system will transit from $x_k$ at time k to $f_k(x_k, u_k, w_k)$ at time k+1 under action $u_(k)$. This transition will incur a cost at time k is $c_k(x_k, u_k, w_k)$ The total cost will be $\sum_{k=0}^{N-1} c_k(x_k, u_k, w_k) + c_N(x_N)$ , where $c_N(x_N)$ is the terminal cost which is the cost incurred at the end of the progress.

Therefore, the optimization of the expected cost is $E\{\sum_{k=0}^{N-1} c_k(x_k, u_k, w_k) + c_N(x_N)\}$

# 2 Deterministic Dynamic Programming

## 2.1 Introduction Of A Decision Problem

Now, consider a decision problem on the discrete time interval $T = [0, N] = 0, 1, \ldots, N$. At time $k \in T$, the state space is $S_k$. $x_k \in S_k$ is the state of the system. The initial state is $x_0 = s \in S_0$ . Given the state $x_k$, $u_k$ is an action depends on $x_k$ at time k and $u_k \in A_k(x_k)$ , $f_k(x_k, u_k)$ is the function describing the system will transit from $x_k$ at time k to a state at time $k+1$ under action $u_k$. This transition will incur a cost at time k is $c_k(x_k, u_k)$.

The cost function for this decision problem is

$$J(x_0, u_0, ..., u_{N-1}) = \sum_{k=0}^{N-1} c_k(x_k, u_k) + c_N(x_N)$$

Where $c_N(x_N)$ is the terminal cost, $J_k$ is called cost-to-go at time k.
When $x_k = x$, we start from unknown state x, the next state depends on the action we take next time. To get the optimal state, we need to find the optimization from time k to $N-1$. Make sure every step we do is right or is the best action, the action is $x_i$.

To find the optimal decision. Suppose $u^* = (u_0^*, ...., u_{N-1}^*)$ is optimal sequences of actions from time k to time $N-1$, $x_k$ is the state at time k, where $k \in (0, N-1)$, the optimal action of $u^*$ on $[k, N-1]$ minimizes the optimization problem

$$J_k(x_k, u_k, ..., u_{N-1}) = \sum_{k=i}^{N-1} c_k(x_k, u_k) + c_N(x_N)$$

Where $x_k$ is an initial state, $c_N(x_N)$ is the terminal cost, $J_k$ is called cost-to-go at time $k$ and the initial $J = J_0$.

To find the optimal u. We need to know the value function for this question (section 2.2).

## 2.2 Value Function

The value function is a function of the state that estimates how good a given action perform in a given state. In deterministic models, it starts from time 0. The value function $V_k(x)$ gives the long-term value of state $k(k = 1, 2, \dots)$. The state value function $V(k)$ of an MDP is the expected return starting from state k.

### 2.2.1 Shortest Paths

Define there is an action to ensure a transition from $i \in S_k$ to $j \in S_{k+1}$, and $a_{ij}^k$ denotes the cost of transition at stage k from i to j. Then with an artificial terminal node t, if $i \in S_N$, $a_{it}^N = c_N(i)$. When there is no action to ensure a transition from i to j at stage k, $a_{ij}^k = \infty$ .

The cost function will be

$$J_k(i) = \min_{j \in S_{k+1}} [a_{ij}^k + J_{k+1}(j)], i \in S_k, k = 0, 1, \dots, N-1$$

When $k = 0$, it has the optimal cost $J_0$.

The most important thing in value function is the action. To get dynamic programming, we need this function can break down into a collection of simpler subproblems. Here, we introduce one method called segmentation.

Denote the value function is

$$V_k(i) = \inf_{u_{k,N-1}} J_k(i, u_k, ..., u_{N-1})$$

where $u_{k,N-1} = (u_k, ..., u_{N-1})$ on (k, i)

Then the optimal problem will become

$$V_k(i) = \min_{j \in S_{k+1}} [a_{ij}^k + V_{k+1}(j)], i \in S_k, k = 0, 1, \dots, N-1$$

The terminal cost is $V_N(i) = c_N(i)$ for $i \in S_k$.

## 2.3 Dynamic Programming

The value function of the decision problem divided into two parts: immediate cost $\sum_{k=i}^{N-1} c_k(x, u_k)$ and the cost of successor state $V_{k+1}(x_{k+1})$

$$V_k(x) = \min_{u_k \in A_k(x_k)} [c_k(x, u_k) + V_{k+1}(x_{k+1})]$$

Where $c_k(x, u_k)$ is the cost function at stage k, $x_{k+1}$ is the evolution of the state $x_{k+1}$=$f_k(x, u_k)$, $k = 0, 1, \ldots, N-1$, $f_k$ is the function describe the system will transit from state $x$ at time k under action $u_k$, then $V_k(x) = [c_k(x, u_k) + V_{k+1}(x_{k+1})]$

We can get the idea from the above equation, to prove the value function satisfies the dynamic programming. We can rewrite the value function for the decision problem.

We know that the optimal decision function is

$$J_k(x, u_k, ..., u_{N-1}) = \sum_{k=i}^{N-1} c_k(x, u_k) + c_N(x_N) \quad [1]$$

We get the value function from 2.2 is

$$V_k(x) = \inf_{u_{k,N-1}} J_k(x, u_k, ..., u_{N-1})$$

on the interval $(k, x)$, and $u_{k,N-1} = (u_k, ..., u_{N-1})$, Where $x \in S_k$, $k = 0, 1, \ldots, N-1$ and the terminal condition is $V_N(x) = c_N(x)$ for $x \in S_N$.

To find the optimal u is called dynamic programming, we minimize the function.

$$\sum_{k=i}^{N-1} c_k(x, u_k) + c_N(x_N)$$

So, we would arrange the function.

For $u_k$ fixed, form $u_{k+1}$ to $u_{n-1}$:

$$V_k(x) = inf[J_k(x, u_k, ..., u_{N-1}]$$

$$V_k(x) \leq c_k(x, u_k) + \sum_{k+1}^{N-1} c_k(x, u_k) + c_N(x_N)$$

$$V_k(x) \leq c_k(x, u_k) + J_{k+1}(f_k(x, u_k), u_{k+1}, ..., u_{N+1})$$

where terminal condition $V_N(x) = c_N(x)$

$$V_k(x) \leq c_k(x, u_k) + V_{k+1}(f_k(x, u_k))$$

For any $u_k$,

$$V_k(x) \leq \min_{u_k \in A_k(x_k)} c_k(x, u_k) + V_{k+1}(f_k(x, u_k))$$

For fixed x and $u_k, ..., u_{N-1}$, $x_k = x$, $x_{k+1}, ...$, Denote $u_{k,j} = (u_k, ..., u_j)$. Then

$$J_k(x, u_{k,N-1}) \geq c_k(x, u_k) + V_{k+1}(f_k(x, u_k))$$

$$J_k(x, u_{k,N-1}) \geq \min_{u_k \in A_k(x_k)} c_k(x, u_k) + V_{k+1}(f_k(x, u_k))$$

$$inf[J_k(x, u_{k,N-1})] \geq \min_{u_k \in A_k(x_k)} c_k(x, u_k) + V_{k+1}(f_k(x, u_k))$$

So the value function satisfies the dynamic programming for the decision problem, the function is,

$$V_k(x) = \min_{u_k \in A_k(x_k)} [c_k(x, u_k) + V_{k+1}(f_k(x, u_k))]$$

Then the value function satisfies the dynamic programming for the decision problem also can write as form

$$V_k(x) = \min_{j \in S_{k+1}} [c_k(x, u_k) + V_{k+1}(x_{k+1})]$$

where $x \in S_k$, $k = 0, 1, ..., N-1$ and terminal condition is $V_N(x) = c_N(x)$ for $x \in S_N$. $c_k(x, u_k)$ is the current step will have the cost, $x_{k+1}$ is the evolution of the state of $f_k(x, u_k)$,$k = 0, 1, ..., N-1$, $V_{k+1}(x_{k+1})$ is from future to past, $x_{k+1}$ is a random variable at state $k+1$ to calculate the best cost in the future.

**Example:**

Suppose a factory has 100 machines, it will be used in four production circles. Every circle has two tasks. By experienced, the x machines will be used in the first task. Then in a production circle, there will have $\frac{1}{3}$ machines will not be used. The left machines $(100 - x)$ will be used in the second task. There will have $\frac{1}{10}$ machines will not be used. If the profit of every machine for the first task is \$5, the profit of every machine for the second task is \$3. How does the factory will arrange the machines to get the maximization profit?

**Solution:** state $k = 1, 2, 3, 4$

$x_k$ is the number of machines can be used at time k.

$u_k$ is the number of machines that were used for the first task during the $kth$ year, then $x_k - u_k$ is the number of machines that be used for the second task during the $kth$ year.

The transition function is $x_{k+1} = (1 - \frac{1}{3})u_k + (1 - \frac{1}{10})(x_k - u_k) = \frac{2}{3}u_k + \frac{9}{10}(x_k – u_k)$.

The profit function is $r_k = 5u_k + 3(x_k - u_k)$

The value function is $V_k = \sum_{k=1}^{n} V_k(x_k, u_k)$

Then we can get the dynamic programming for the decision problem is $V_k(x_k) = \max\{r_k(x,u_k) + V_{k+1}(x_{k+1})\}$

When $k=4$, $V_4(x_4) = max\{5u_4 + 3(x_4 - u_4)\}$

$= max\{3x_4 + u_4\} = 3x_4 + x_4 = 4x_4$,then the optimal $u_4^* = x_4$

When $k=3$, $V_3(x_3) = max\{5u_3 + 3x_3 + 4(\frac{2}{3}u_3 + \frac{9}{10}(x_3 - u_3))\}$

$= max\{\frac{33}{5}x_3 + \frac{27}{5}u_3\} = \frac{33}{5}x_3 + \frac{27}{5}u_3 = 12x_3$, then the optimal $u_3^* = x_3$

When $k=2$, $V_2(x_2) = max\{5u_2 + 3x_2 + 12(\frac{2}{3}u_2 + \frac{9}{10}(x_2 - u_2))\}$

$= max\{\frac{63}{5}x_2 + \frac{17}{5}u_2\} = \frac{63}{5}x_2 + \frac{8}{9} \times 0 = \frac{63}{5}x_2$,then the optimal $u_2^* = 0$

When $k=1$, $V_1(x_1) = max\{5u_1 + 3x_1 + \frac{63}{5}(\frac{2}{3}u_1 + \frac{9}{10}(x_1 - u_1))\}$

$= max\{\frac{717}{50}x_1 + \frac{105}{50}u_1\} = \frac{717}{50}x_1 + \frac{103}{50} \times 0 = \frac{717}{50}x_1$, then the optimal $u_1^* = 0$

Since $x_1 = 100$, then the maximization profit is $V_1(x_1) = \frac{717}{50}x_1 = \frac{717}{50}(100) = \$1431$.

From above, the machines be used for first task is 0 and for the second task is 100 in the first production circle. The machines are used for first task is 0 and for the second task is 100 in the second production circle. The machines are used for first task is 100 and for the second task is 0 in the third production circle. The machines are used for first task is 100 and for the second task is 0 in the fourth production circle.

# 3 Stochastic Finite Horizon

## 3.1 Finite State Markov Chain (MC)[2]

Let $\{X_t, t = 0, 1, 2...\}$ be a random process and $S = \{1, 2, \dots n\}$ such as each $X_t$ takes values from $S$. If $X_t = i$, we say that the process is on the state $i$ at time $t$. For any $t_1 < t_2 < ... < t_k$ and any $i_1, i_2, ..., i_k$ ,we have the probability

$$p(X_{t_k} = i_k | X_{t_{k-1}} = i_{k-1}, X_{t_{k-2}} = i_{k-2}, ..., X_{t_2} = i_2, X_{t_1} = i_1)$$

$$= \frac{p(X_{t_k} = i_k, X_{t_{k-1}} = i_{k-1}, X_{t_{k-2}} = i_{k-2}, ..., X_{t_2} = i_2, X_{t_1} = i_1)}{p(X_{t_{k-1}} = i_{k-1}, X_{t_{k-2}} = i_{k-2}, ..., X_{t_2} = i_2, X_{t_1} = i_1)}$$

$$= p(X_{t_k} = i_k | X_{t_{k-1}} = i_{k-1})$$

Then $X_t, t \in T$ is a Markov chain with state space S.

Suppose that whatever the progress is in state $i$, there is a fixed probability $P_{ij}$ will describe the next state $j$. That is

$$p_{ij} = p(X_{t+1} = j | X_t = i)$$

If from state $i$ to state $j$ need more than one step, for example, $k$ step, we denoted the $p = p_{ij}^k$,which is $p_{ij}^k = p(X_k = j|X_t = i)$. We said it is the $k$-step transition probability from $i$ to $j$.

### 3.2 Markov Decision Progress

For Markov decision process, the most important thing is how to determine the action at a certain state.

In the general setting, a Markov decision process $\{X_t, t \geq 0\}$ may be modeled by

$$(S, \{A_x : x \in S\}, \{Q(\cdot|x, a) : x \in S, a \in A_x\}, x)$$

Where $S$ is the state space of the $MDP$ which is a finite or countable set, $A_x$ is the action space associated with the state x. $A_x$ is a finite set. When $X_t = x$, the action $a_t = a \in A_x$,$X_t = x$, $Q(\cdot|x, a)$ is a probability mass function on $S$ to describe the distribution of $X_{t+1}$.$c$ is the one stage cost function $c(X_t, a_t)$,$0 \leq t \leq N - 1$ and $c_N(X_N)$ is the terminal cost.

The state transition of $X_t$ follows the stochastic kernel $Q$:

$$p(X_{t+1} = j|X_0, a_0, X_1, a_1, ..., X_n = i, a_n = a) = p_{ij}(a)$$

where $a \in A_x$.The distribution $X_{t+1}$ only influenced by $X_n = i, a_n = a$.

For an optimization horizon $[0, N]$, the observed sample path is $(X_0, a_0, X_1, a_1, ..., X_{N-1}, a_{N-1}, X_N)$,the total cost will be $\sum_{t=0}^{N-1} c(X_t, a_t) + c_N(X_N)$, Where $c_N(X_N)$ is the terminal cost and $a_t$ depend on the random sample $\omega$.

Given $X_0 = i$, the expected total cost is

$$J_0(i) = E \sum_{t=0}^{N-1} c(X_t, a_t) + Ec_N(X_N)$$

In the simpler model, a Markov decision process $\{X_t, t \geq 0\}$ may be modeled by

$$(S, A, \{p(a), a \in A\}, c)$$

All the states share the same space $A$, the stochastic kernel replaced by $p(a)$. $p(a)$ determines the transition from $X_t = i \quad to \quad X_{t+1}$.

Let $h_t = (X_0, a_0, X_1, a_1, ..., X_{N-1}, a_{t-1}, X_t)$,$t \geq 0$.When $t = 0, h_0 = X_0$.

Each randomized policy $\pi = \{\pi_t, t \geq 0\}$ is a sequence of stochastic kernels, which is admissible policy $\pi_t(\cdot|h_t)$,$t = 0, 1, ...,$ on $A$, where $h_t$ is t-history of the Markov decision process. The randomized policy $\pi$ gives a probability of taking which action, given $h_t$.

Given the initial state $X_0 = i$, according to $\pi_t$, select the action $a_t$. Let $\Pi$ is the set of all admissible policies. Policy $\pi \in \Pi$, the cost will be

$$J_0(i, \pi) = E \sum_{t=0}^{N-1} c(X_i, a_i) + Ec_N(X_N)$$

The optimal cost is $V_0(i) = \underset{\pi \in \Pi}{inf} J_0(i, \pi)$.

### 3.3 Value Function

To find the minimized cost $J_0$. Let the initial time $k \in [0, N]$. The policy $\pi = \{\pi_k, ..., \pi_N - 1\}$ as a sequence of stochastic kernels: $\pi_t(\cdot|h_{k,t}), t \in [k, N-1]$ where $h_{k,t} = (X_k, a_k, ..., X_{t-1}, a_{t-1}, X_t)$,and let $\Pi_k$ is the set of all admissible policies. Suppose $X_k = i$ and $\pi \in \Pi_k$, the cost-to-go function will be

$$J_k(i, \pi) = E[\sum_{i=k}^{N-1} c(X_k, a_k) + c_N(X_N)|X_k = i]$$

The value function is $V_k(i) = \underset{\pi \in \Pi_k}{inf} J_k(i, \pi)$. The optimal cost is at $k = 0$, which is $V_0(i) = \underset{\pi \in \Pi}{inf} J_0(i, \pi)$.

So, the value function satisfies the dynamic programming for the cost function, the function is,

$$V_k(i) = \min_{a \in A} [c(i, a) + EV_{k+1}(X_{k+1})|X_k = i]$$

$$V_k(i) = \min_{a \in A} [c(i, a) + \sum_{j \in S} p_{ij}(a) V_{k+1}(j)][2]$$

Where $a$ is an action $\in A$, $X_{k+1} = Q(\cdot|x, a)$, $p_{ij}(a)$ is the probability of in given state $i$ to reach state $j$ is observed at stage $n$ and action a is taken while $V_k(i)$ is the value function in state i at stage $k$. $V_N(i)$=$c_N(i)$ is the terminal condition.

**Example:**

Suppose $X_t$ has three states $\{1, 2, 3\}$, the action space at each state is $A = \{a', a'', a'''\}$ , with probability $p(a'), p(a''), p(a''')$. Consider $T$= $[0, 3]$. We need to use a policy $(\pi_0, \pi_1, \pi_2)$ to choose $a_0, a_1, a_2$. The stochastic kernel $\pi_0$ should depend on $X_0$, $\pi_1$ should depend on $X_1$, and $\pi_2$ should depend on $(X_0, a_0, X_1, a_1, X_2)$.

The cost is given by

$$J_0 = E[c(X_0, a_0) + c_1(X_1, a_1) + c_2(X_2, a_2) + c_3(X_3)|\pi]$$

To find the optimal policy,

Step 1 : Determine $[c_3(1), c_3(2), c_3(3)]$ as the terminal condition of the value function at $t = 3$.

Step 2 : At $X_2 = 1$, determine $a_2(1)$, which is the action at stage 2 given the state 1 at that stage, such that

$$a_2(1) = \underset{a}{argmin} \quad c(1, a) + \sum_j p_{1j}(a)V_3(j)]$$

At $X_2 = 2$, determine $a_2(2)$, which is the action at stage 2 given the state 2 at that stage, such that

$$a_2(2) = \underset{a}{argmin} \quad c(2, a) + \sum_j p_{2j}(a)V_3(j)]$$

At $X_2 = 3$, determine $a_2(3)$, which is the action at stage 2 given the state 3 at that stage, such that

$$a_2(3) = \underset{a}{argmin} \quad c(3, a) + \sum_j p_{3j}(a)V_3(j)]$$

In the end, obtain a mapping $f_2$ at stage 2 from $S$ to $A$, such that

$$[f_2(1), f_2(2), f_2(3)] = [a_2(1), a_2(2), a_2(3)]$$

Step 3 : Replacing step 2 by replacing $X_2$ by $X_1$. We obtain the mapping $f_1$ from $S$ to $A$, such that

$$[f_1(1), f_1(2), f_1(3)] = [a_1(1), a_1(2), a_1(3)]$$

Step 4 : Replacing step 3 by replacing $X_1$ by $X_0$. We obtain the mapping $f_0$ from $S$ to $A$, such that

$$[f_0(1), f_0(2), f_0(3)] = [a_0(1), a_0(2), a_0(3)]$$

Step 5 : Construct the stochastic kernels based on $f_2, f_1, f_0$. At stage $k \in [0, 2]$, if $X_k = i \in S$,$\pi_k(a_k(i)) = 1$, i.e the action $a_k(i)$ is selected with probability 1. Denote

$\pi = \{\pi_0, \pi_1, \pi_2\}$. Then $\pi$ is the optimal policy.

**Example:**

Consider a problem where there are three states and two actions $S = 1, 2, 3$, $A \in \{a^1, a^2\}$ If you are in $s1, s2, s3$

Problem data:

transition cost

$$c = \begin{bmatrix} 2.5 & 1.5 \\ 1 & 3 \\ 2 & 1.5 \end{bmatrix}$$

$$P(a^1) = \begin{bmatrix} 0.3 & 0.5 & 0.2 \\ 0.1 & 0.8 & 0.1 \\ 0.6 & 0.3 & 0.1 \end{bmatrix}$$

$$P(a^2) = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.5 & 0.3 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$

We set the terminal cost.

$$V(3) = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}$$

Then we do the backward introduction

From the output we get

```
V =

    6.6800    5.6000    4.0000
    5.5800    4.3000    3.0000
    6.9600    5.8000    5.0000
```

The optimal policy for time $N = 2$ and $N = 1$ is

$$\begin{bmatrix} a2 & a2 \\ a1 & a1 \\ a1 & a2 \end{bmatrix}$$

# 4 Stochastic Infinite Horizon

## 4.1 Value Function With Discount Cost

For the Markov Decision Process specified by

$$(X, \{A_x : x \in X, Q(\cdot|x, a) : x \in X, a \in A_x\}, c)$$

Let $h_t = (X_0, a_0, X_1, a_1, ..., X_{t-1}, a_{t-1}, X_t)$,$t \geq 0$ be a t-history. let $\Pi$ is the set of all admissible policies. A randomized policy $\pi = \{\pi_t, t \geq 0\}$ is a sequence of stochastic kernels $\pi_t(\cdot|h_k), t = 0, 1, \ldots$ on $A_x$.

Let the initial time $k \in [0, N]$. The policy $\pi = \{\pi_k, \ldots, \pi_N - 1\}$ as a sequence of stochastic kernels: $\pi_t = \pi_t(\cdot|h_{k,t})$,$t \in [k, N-1]$ where $h_{k,t} = (X_k, a_k, \ldots, X_{t-1}, a_{t-1}, X_t)$, and let $\Pi_k$ is the set of all admissible policies. Let $\alpha$ be a discount factor which is a fixed constant. Suppose $X_k = x$ and $\pi \in \Pi_k$, the cost-to-go function will be

$$J_k(x, \pi) = E[\sum_{i=k}^{N-1} \alpha^t c(X_t, a_t) + \alpha^N c_N(X_N)|X_k = x, \pi][5]$$

Where value function is $V_N(i)$=$c_N(i)$ is the terminal condition.

## 4.2 Markov Decision Process With Infinite Horizon

We assume countably infinite state and finite action space.

In the simpler model, a Markov decision process $\{X_t, t \geq 0\}$ may be modeled by

$$(S, A, \{p(a), a \in A\}, c)$$

In general setting, a Markov decision process $\{X_t, t \geq 0\}$ may be modeled by

$$(S, \{A_x : x \in S\}, \{Q(\cdot|x, a) : x \in S, a \in A_x\}, x)$$

Let $h_t = (X_0, a_0, X_1, a_1, ..., X_{t-1}, a_{t-1}, X_t)$,$t \geq 0$ be a t-history.

A randomized policy is the form $\pi = \{\pi_0, \pi_1, \pi_2 \dots\}$ is an infinite sequence of stochastic kernels on A. Let $\Pi$ be the set of all policies.

A randomize policy $\pi \in \Pi$ will be a randomized Markov policy if $\pi_t(\cdot|h_k) = \varphi_t(\cdot|h_t)$ for any t-history $h_t$ , where $\{\varphi_t\}$ is a sequence of stochastic kernels.

A policy $\pi$ is called a pure or deterministic policy if, for each $h_t$, $\pi_t(\cdot|h_k)$ concentrates on a single action $a(h_t)$ depends on $h_t$.

A policy $\pi$ is called a deterministic Markov policy if there is a sequence $\{f_k\}$ of functions from $S$ to $A$ such that $\pi_t(\cdot|h_k)$ concentrates at $\{f_k\}$.

The cost function will be

$$J_0(i, \pi) = E[\sum_{t=0}^{\infty} \alpha^t c(X_t, a_t)|X_0 = i, \pi]$$

where $\alpha \in (0, 1)$ is a discount factor, $c(X_t, a_t)$ is a one stage cost.

The value cost is $V(i) = \inf_{\pi \in \Pi_k} J_0(x)$. The value function with simpler model satisfies the Dynamic Programming function

$$V(i) = \min_{a \in A} [c(i, a) + \alpha \sum_{j=1}^{n} p_{ij}(a)V(j)]$$

The value function with general setting satisfies the Dynamic Programming function

$$V(i) = \min_{a \in A_i} [c(i, a) + \alpha \sum_{j=1}^{n} p_{ij}(a)V(j)][4]$$

### 4.3 Value Iteration

Let $V_0 \in B(S)$, given $V_k$. The new function will be

$$V_{k+1}(i) = \min_{a \in A_i} \{c(i, a) + \alpha \sum_{j \in S} p_{ij}(a)V_k(j)\}$$

Where $c(i, a)$ is the one stage cost is bounded, k is an iteration counter, $V_k$ and $V_{k+1}$ is bounded.

We define a map $T$ from $B(S)$ to $B(S)$, the function will become Bellman operator [3]

$$Th(i) = \min_{a \in A_i} \{c(i,a) + \alpha \sum_{j \in S} p_{ij}(a) h(j)\}$$

Where a map $T$ from $B(S)$ to $B(S)$ is called a contraction if there exists $0 \leq c < 1$ such that

$$|Ty - Tv| \leq c|y - v|$$

So for any $i \in S$, we have

$$Tv(i) = \min_{a \in A_i} \{c(i,a) + \alpha \sum_{j \in S} p_{ij}(a) v(j)\}$$

$$Ty(i) = \min_{a \in A_i} \{c(i,a) + \alpha \sum_{j \in S} p_{ij}(a) y(j)\}$$

Then,

$$|Ty - Tv| \leq \alpha|y - v|$$

where $\alpha \in [0, 1)$. The value function by uniqueness will become

$$V(i) = \min_{a \in A_i} \{c(i,a) + \alpha \sum_{j \in S} p_{ij}(a) V(j)\}$$

### 4.4 Policy From The DP Equation

A randomize policy $\pi = \pi_t, t \geq 0$ will be a randomized stationary policy if $\pi_t(\cdot|h_k) = \varphi_t(\cdot|h_t)$ for any t-history $h_t$ , where $\varphi_t$ is a sequence of stochastic kernels.
A policy $\pi = \pi_t, t \geq 0$ is called a deterministic stationary policy if there exists a stochastic kernel $\varphi$ such that $\pi_t(\cdot|h_k)$ for any t-history $h_t$, where $\varphi$ concentrates at a single point $f(X_t) \in A_{X_t}$.

### 4.5 Policy Iteration

#### 4.5.1 Algorithm

Let $\pi_0$ be a arbitrary stationary policy. The value function $V_k$ of policy $\pi_k$. Choose $\pi_{k+1} = \{g_{k+1}, g_{k+1}, ...\}$ such that

$$g_{k+1}(i) = \underset{a \in A_i}{argmin} \quad \{c(i,a) + \alpha \sum_{j \in S} p_{ij}(a) V_k(j)\}$$

### 4.5.2 Policy Evaluation

For any stationary deterministic policy $\pi = \{g, g, g \dots\}$, its value function [4]

$$V^{\pi}(i) = E[\sum_{t=0}^{\infty} \alpha^t c(X_t, a_t) | X_0 = i]$$

is the unique solution of

$$V^{\pi}(i) = \{c(i, g(i) + \alpha \sum_{j \in S} p_{ij}(a) V^{\pi}(j)\}$$

The value function $V_k$ generated by the policy iteration algorithm such as $V_{k+1} \leq V_k$

**Example:**

Consider a problem where there are three states and two actions $S$=$1, 2, 3$, $A \in \{a^1, a^2\}$. if you are in $s1, s2, s3$

Problem data:

transition cost

$$c = \begin{bmatrix} 2.5 & 1.5 \\ 1 & 3 \\ 2 & 1.5 \end{bmatrix}$$

discount = 0.9;

transition probabilities:

$$P(a^1) = \begin{bmatrix} 0.3 & 0.5 & 0.2 \\ 0.1 & 0.8 & 0.1 \\ 0.6 & 0.3 & 0.1 \end{bmatrix}$$

$$P(a^2) = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.5 & 0.3 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$

Assume initial stationary policy is $a1, a2, a1$, from the output, we get

```
V =
```

```
 25.955139169646827213000506162643
 26.378867983206419012276455760002
 25.426450007774747064104303717613
```

Then we do policy evaluation, get the minimizing controls are $a2, a1, a2$, and get

```
V =

 24.774257502721161472436506301165
 24.617127973876563373778481036425
 24.650404291711947735166177153587
```

Then we do policy improvement to find the minimization:

```
 V =

 12.617851622874795793904922902584
 11.738794435857726057292893528938
  12.791731066460592955599116012454
```

Hence we have $V(1) = 12.61785, V(2) = 11.73879, V(3) = 12.79173$, which implies the optimal action is $a2, a1, a2$.

# 5 Oil Storage Problem

## 5.1 Introduction

Recently people have more interest in applying advanced computational optimization techniques to improve the performance of oil storage, and many optimization algorithms have been implemented to maximize storage performance. Oil facilities are used to store oil to ensure their oil supply and to be able to meet the increase in price or any disruption in the transportation system. In the real world, many companies use storage facilities to optimize their profit during the price changes. In most optimization storage problem, the maximum profit is subjected to constraints. So storage facilities will be used to ensure that the purchase and sale of oil do not be influenced during a specific time. Refer to Zheng, Louis, Benjamin and Khalid (2002), we could formulate the oil storage problem as a dynamic optimal control problem. Thus, we could compute globally optimal control via dynamic programming (DP)[1]. Also, serval authors have discussed the gas storage facilities using Dynamic Programming to manage the underground cavern through buying and selling gas to earning a profit by the maximum capacity of the facility [9] [10] [11] [13] [14] [15]. The value of an oil storage facility can be regarded as the maximum expected profits that the operator of the facility could obtain by optimally operating the facility, that is, "buying low" and "selling high" which is characterized as a stochastic control problem [13].

In this section, I will use stochastic dynamic programming to formulate the storage asset valuation problem. Assume the spot price of oil follows a linear regression model. For the similar gas problem as a regime-switching model refers to [20]. For this section, I will treat the oil storage problem as Markov decision problem to estimate when to buy and inject oil and when to sell and withdrawal oil. An alternative approach is to discretize the price process and treat the resulting discrete-time problem as a Markov Decision Process refers to[11] [12] [16] [19] [21]. For a further discussion of the price model see [7]. My model specifies how to get maximization of profits depends on operational constraints such as the dynamics of oil prices and the oil sufficient capacity. I derive maximum profit value by dynamically buy and inject or withdrawal oil from storage to provide demand quantity in the contract when the price of market conditions changes.

### 5.2 State Variable

We use the following notation for the natural oil storage problem:

$P_t$: the current spot price per unit

$P_d$: the fixed demand price per unit of oil in the contract

$d_0$: the fixed demand quantity of oil in the contract

$L_t$: current amount of oil in the storage. We assume that I can be any value within the domain $[0, L_{max}]$

$a_t$: the actual volume injection and withdrawal at time t

$N$: length of the contract

### 5.3 The Oil Storage Model

Based on N. Bäuerle (2014), assume that the spot price dynamics is

$$P_{t+1} = \alpha(c - P_t) + \xi_t$$

Where $\alpha$ is a discount factor, which is fixed constant, $c$ is a constant, $\xi_t$ is the volatility which follows the uniform distribution.

Using the notations of Warin (2012) and Patrick, Francesco, and Ismail (2013), we consider an oil storage facility with technical constraints on the volume of stored oil, 0 and $L_{max}$ i.e the volume of stored oil $L$ should verify $0 \leq L_t \leq L_{max}$. In this section, $L_t$ can be zero.

At time t and starting from a volume $L_t$, the user has the probability to make two actions, $a_{inj}$ = inject gas, $a_{with}$ = withdrawal gas. Let the actual volume injection or withdrawal at time t given by $a_t$. We assumed inject gas as positive, $a_t > 0$, withdrawal gas as negative $a_t < 0$. Note that it is impossible that withdrawal when the storage is empty and injection when storage is full.

If the user follows a strategy $a$, the feasible region is

$$A_t = \{a_{inj} : [0, d_0 + L_{max} - L_t], a_{with} : [0, L_t]\}$$

So, The storage volume is

$$L_{t+1} = L_t + a_t$$

When an independent random sequence describes the oil inflow, the state variable is the storage capacity at the beginning of each period of the storage. The storage can be divided into n parts from the minimum storage level of 0 to the maximum storage level of $L_{max}$, corresponding to t + 1 oil storage level, each period is the same, the state variable can take this t + 1 discrete values. However, in dynamic programming, when a certain stage state is given, the development after this stage is not affected by the state of the previous stage ( Markov process). That is to say, the current state is a complete summary of history, and the history of the process can only influence its future development through the current state. Therefore, the calculated storage oil level at the end of the period is not subject to this t + 1 state. The model state space is the variable range of the storage's effective storage oil level. So, The application of dynamic programming optimization principle in storage scheduling under stochastic conditions can be summarized as follows: first, the optimal decision of the storage in any period only depend on the state of the storage at the beginning of the period. This optimal decision should make the period, and the sum of the expected values of the profit in the future will be optimal. Second, The optimal decision of the storage at any period has nothing to do with the previous scheduling process.

Refer to Nicole and Viola (2014), we could formulate oil storage valuation problem as a Markov Decision Process(MDP).

Let $\{(P_t, L_t), t = 0, 1, 2...\}$ be a random process and $S \geq 0$ such as each $(P_t, L_t)$ takes values from $S$. If $(P_t, L_t) = (i, s)$, we say that the progress is on the state $(i, s)$ at time $t$. For any $t_1 < t_2 < ... < t_k$ and any $(i_1, s_1), (i_2, s_2), ..., (i_k, s_k)$ ,we have the probability

$$p((P_{t_k} = i_k, L_{t_k} = s_k)|(P_{t_{k-1}} = i_{k-1}, L_{t_{k-1}} = s_{k-1}), ..., (P_{t_2} = i_2, L_{t_2} = s_2), (P_{t_1} = i_1, L_{t_1} = s_1))$$

$$= \frac{p((P_{t_k} = i_k, L_{t_k} = s_k), (P_{t_{k-1}} = i_{k-1}, L_{t_{k-1}} = s_{k-1}), ..., (P_{t_2} = i_2, L_{t_2} = s_2), (P_{t_1} = i_1, L_{t_1} = s_1))}{p((P_{t_{k-1}} = i_{k-1}, L_{t_{k-1}} = s_{k-1}), ..., (P_{t_2} = i_2, L_{t_2} = s_2), (P_{t_1} = i_1, L_{t_1} = s_1))}$$

$$= p((P_{t_k} = i_k, L_{t_k} = s_k)|(P_{t_{k-1}} = i_{k-1}, L_{t_{k-1}} = s_{k-1}))$$

Then $(P_t, L_t), t \in T$ is a Markov chain with state space S.

Suppose that whatever the progress is in the state $(i, s)$, there is a fixed probability $p_{(i,s)(j,r)}$ will describe the next state $(j, r)$. That is

$$p_{(i,s)(j,r)}(t, a_t) = p((P_{t+1} = j, L_{t+1} = r)|(P_t = i, L_t = s))$$

$$= p((\alpha(c - P_t) + \xi_t = j, L_t + a_t = r)|(P_t = i, L_t = s))$$

$$= p(\alpha(c - i) + \xi_t = j, s + a_t = r)$$

In general setting, a Markov decision process $\{(P_t, L_t), t \geq 0\}$ may be modeled by

$$(S, \{A_{(x,l)} : (x, l) \in S\}, \{Q(\cdot|(x, l), a) : x \in S, a \in A_{(x,l)}\}, (x, l))$$

Where $S$ is the state space of the $MDP$ which is a finite or countable set, $A_{(x,l)}$ is the action space associated with the state $(x, l)$. $A_{(x,l)}$ is a finite set. When $P_t = x, L_t = l$, the action $a_t = a \in A_{(x,l)}$,$P_t = x, L_t = l$, $Q(\cdot|(x, l), a)$ is a probability mass function on $S$ to describe the distribution of $(P_{t+1}, L_{t+1})$.$r$ is the one stage reward function $r((P_t, L_t), a_t)$,$0 \leq t \leq N-1$ and $r_N(P_N, L_N)$ is the terminal reward.

The state transition of $(P_t, L_t)$ follows the stochastic kernel $Q$:

$$p((P_{t+1}, L_{t+1}) = (j, r)|(P_0, L_0), a_0, (P_1, L_1), a_1, ..., (P_0, L_0) = (i, s), a_n = a) = p_{(i,s)(j,r)}(a)$$

where $a \in A_{x,l}$.The distribution $(P_{t+1}, L_{t+1})$ only influenced by $(P_n, L_n) = (i, s), a_n = a$.

For an optimization horizon $[0, N]$, the observed sample path is $((P_0, L_0), a_0, (P_1, L_1), a_1, ..., (P_{N-1}, L_{N-1}), a_{N-1}, (P_N, L_N))$,the total reward will be $\sum_{t=0}^{N-1} r((P_t, L_t), a_t) + r_N(P_N, L_N))$, Where $r_N(P_N, L_N)$ is the terminal reward.

Given $(P_0, L_0) = (i, s)$, the expected total reward is

$$J_0(i, s) = E \sum_{t=0}^{N-1} r((P_t, L_t), a_t) + E r_N(P_N, L_N)$$

In the simpler model, a Markov decision process $\{P_t, L_t, t \geq 0\}$ may be modeled by

$$(S, A, \{p(a), a \in A\}, r)$$

All the states share the same space $A$, the stochastic kernel replaced by $p(a)$. $p(a)$ determines the transition form $(P_t, L_t) = (i, s)$ to $(P_{t+1}, L_{t+1})$.
Let $h_t = ((P_0, L_0), a_0, (P_1, L_1), a_1, ..., (P_{N-1}, L_{N-1}), a_{t-1}, (P_t, L_t))$,$t \geq 0$.When $t = 0, h_0 = (P_0, L_0)$.

Each randomized policy $\pi = \{\pi_t, t \geq 0\}$ is a sequence of stochastic kernels, which is admissible policy $\pi_t(\cdot|h_t)$,$t = 0, 1, ...,$ on $A$, where $h_t$ is t-history of the $MDP$. The randomized policy $\pi$ gives a probability of taking which action, given $h_t$.
Given the initial state $(P_t, L_t) = (i, s)$, according to $\pi_t$, select the action $a_t$. Let $\Pi$ is the set of all admissible policies. Policy $\pi \in \Pi$, the reward will be

$$J_0((i, s), \pi) = E \sum_{t=0}^{N-1} r((P_i, L_s), a_i) + E r_N(P_N, L_N)$$

The optimal reward is $V_0((i, s)) = \sup_{\pi \in \Pi} J_0((i, s), \pi)$.

### 5.4 Value Function On Gas Storage Model

To find the maximized reward $J_0$. Let the initial time $k \in [0, N]$. The policy $\pi = \{\pi_k, ..., \pi_N - 1\}$ as a sequence of stochastic kernels: $\pi_t(\cdot|h_{k,t}), t \in [k, N-1]$ where $h_{k,t} = ((P_k, L_k), a_k, ..., (P_{t-1}, L_{t-1}), a_{t-1}, (P_t, L_t))$, and let $\Pi_k$ is the set of all admissible policies. Suppose $(P_k, L_k) = (i, s)$ and $\pi \in \Pi_k$, the reward-to-go function will be

$$J_k((i, s), \pi) = E[\sum_{i=k}^{N-1} r((P_t, L_t), a_t) + r_N((P_N, L_N))|(P_k, L_k) = (i, s)]$$

The value function is $V_k((i, s)) = \sup_{\pi \in \Pi_k} J_k((i, s), \pi)$. The optimal reward is at $k = 0$, which is $V_0((i, s)) = \sup_{\pi \in \Pi} J_0((i, s), \pi)$.
So, the value function satisfies the dynamic programming for the reward function, the function is,

$$\begin{aligned} V_k(i, s) &= \max_{a \in A} [r((i, s), a) + E V_{k+1}(P_{k+1}, L_{k+1})|(P_k, L_k) = (i, s)] \\ &= \max_{a \in A} [r((i, s), a) + E V_{k+1}(\alpha(c - P_k) + \xi_k, L_k + a_k)] \end{aligned}$$

Recall that given $((x, l), a)$, $Q(\cdot|(x, l), a)$is the distribution of $(P_{k+1}, L_{k+1})$. So I have the relation

$$E[V_{k+1}(P_{k+1}, L_{k+1})|(P_k, L_k) = (i, s), a_k = a] = \int_{y \in (P,L)} V_{k+1}(y) Q(dy|(x, l), a)]$$

By the relation $P_{t+1} = \alpha(c - P_t) + \xi_t$ and $L_{t+1} = L_t + a_t$, we further obtain

$$E[V_{k+1}(\alpha(c - P_k) + \xi_k, L_k + a_k, ] = \int_{y\in(P,L)} V_{k+1}(y)Q(dy|(x, l), a)$$

By summarizing the above, the DP equation takes form

$$V_k(i, s) = \sup_{a\in A_{(x,l)}} [r((i, s), a) + EV_{k+1}(\alpha(c - P_k) + \xi_k, L_k + a_k)]$$

$V_N(i, s)$=$r_N(i, s)$ is the terminal condition.

### 5.5 Numerical Problem

The following parameters were chosen from the numerical examples. Assume $\alpha$= 0.3, c= 0.5, $\xi_t$ follows uniform distribution(-0.2, 0.2), more values of these parameters are shown in Appendix.

Then the price model will become

$$P_{t+1} = 0.3(0.5 - P_t) + \xi_t$$

$P_0 = 0.4$, $P_d = 0.43$, $d_0 = 10^6$, $L_{max} = 10^{10}$
The feasible region is

$$A_t = \{a_{inj} : \left[0, 1.0001 \times 10^{10} - L_t\right], a_{with} : [0, L_t]\}$$

The storage volume is

$$L_{t+1} = L_t + a_t$$

Since this question is a continuous problem, we need to discretize our spot price and the amount in the storage to become a discrete random variable problem. First, I will show how to discretize uniform distribution in Matlab. Details see Appendix. I still need to combine price and volume of oil storage and discretize them to get the probability of each transition. Finally, I can use Dynamic Programming to find the optimization of the oil storage problem. Matlab could realize this question.

# 6 Conclusion

In this paper, I will introduce Markov decision process, value function and dynamic programming in general state space and use Matlab to realize result. In the last section, I discuss the theory and method of stochastic dynamic programming to schedule the oil storage to get optimization based on dynamic programming and Markov stochastic decision process theory. According to the literature review, we generalized the oil storage scheduling problem into a dynamic programming problem, and a dynamic programming model for storage optimization scheduling is established. Through dynamic programming, the method of seeking the optimal strategy of multi-stage decision-making problem is finally obtained to obtain the optimal return value of each stage. Since the problem in the last section has a continuous random variable, so I use discretization to transform a continuous random variable (price and volume) to a discrete random variable. Matlab could obtain the numerical results.

# Appendices

## A Matlab Code For Example 2

```
clear all
%Consider a problem where there are three states and two actions S={1,2,3}, A = {a1,a2}
%if you are in s1,s2,s3
% Problem data
c = [2.5 1.5;1 3;2 1.5];
P = cell(3,3);
P{1} = [0.3 0.5 0.2;0.1 0.8 0.1;0.6 0.3 0.1];
P{2} = [0.1 0.4 0.5;0.5 0.3 0.2;0.2 0.2 0.6];
% Initialization
V=zeros(3,3);
V(:,3)=[4;3;5]
V3=[4;3;5];
% backward introduction
s1=0;
s2=0;
s3=0;
s4=0;
s5=0;
s6=0;
s7=0;
s8=0;
s9=0;
s10=0;
s11=0;
s12=0;
for j=1:3
    s1=s1+(P{1}(1,j)*V3(j));
```

```
    s2=s2+(P{2}(1,j)*V3(j));
    s3=s3+(P{1}(2,j)*V3(j));
    s4=s4+(P{2}(2,j)*V3(j));
    s5=s5+(P{1}(3,j)*V3(j));
    s6=s6+(P{2}(3,j)*V3(j));
end
V2=[c(1,1)+s1,c(1,2)+s2;c(2,1)+s3,c(2,2)+s4;c(3,1)+s5,c(3,2)+s6];
[minValues,minIndices]=min(V2,[],2)
V2=minValues;
V(:,2)=[min(c(1,1)+s1,c(1,2)+s2);min(c(2,1)+s3,c(2,2)+s4);min(c(3,1)+s5,c(3,2)+s6)]
for j=1:3
    s7=s7+(P{1}(1,j)*V2(j));
    s8=s8+(P{2}(1,j)*V2(j));
    s9=s9+(P{1}(2,j)*V2(j));
    s10=s10+(P{2}(2,j)*V2(j));
    s11=s11+(P{1}(3,j)*V2(j));
    s12=s12+(P{2}(3,j)*V2(j));
end
V1=[c(1,1)+s7,c(1,2)+s8;c(2,1)+s9,c(2,2)+s10;c(3,1)+s11,c(3,2)+s12];
[minValues,minIndices]=min(V1,[],2)
V1=minValues;
V(:,1)=[min(c(1,1)+s7,c(1,2)+s8);min(c(2,1)+s9,c(2,2)+s10);min(c(3,1)+s11,c(3,2)+s12)]

V =

     0     0     4
     0     0     3
     0     0     5


minValues =
```

```
    5.6000
    4.3000
    5.8000


minIndices =

     2
     1
     1


V =

         0    5.6000    4.0000
         0    4.3000    3.0000
         0    5.8000    5.0000


minValues =

    6.6800
    5.5800
    6.9600


minIndices =

     2
     1
     2
```

```
V =

    6.6800    5.6000    4.0000
    5.5800    4.3000    3.0000
    6.9600    5.8000    5.0000
```

## B Matlab Code For Example 3

```
clear all
%Consider a problem where there are three states and two actions S={1,2,3}, A = {a1,a2}
%if you are in s1,s2,s3
% Problem data
c = [2.5 1.5;1 3;2 1.5];
P = cell(3,3);
discount = 0.9;
%transition probabilities
P{1} = [0.3 0.5 0.2;0.1 0.8 0.1;0.6 0.3 0.1];
P{2} = [0.1 0.4 0.5;0.5 0.3 0.2;0.2 0.2 0.6];
% Initialization
V=0;
%assume initial policy is a1,a2,a1
%policy evaluation
syms V1 V2 V3
equations=[c(1,1)+0.9*(P{1}(1,1)*V1+P{1}(1,2)*V2+P{1}(1,3)*V3)==V1 , c(2,2)+0.9*(P{2}(2,
s = solve(equations, [V1 V2 V3]);
V1=s.V1;
V2=s.V2;
```

```
V3=s.V3;
V1=vpa(V1,6);
V2=vpa(V2,6);
V3=vpa(V3,6);
V=[V1;V2;V3]
%policy Improvement
s1=0;
s2=0;
s3=0;
s4=0;
s5=0;
s6=0;
s7=0;
s8=0;
s9=0;
s10=0;
s11=0;
s12=0;
for j=1:3
    s1=s1+0.9*(P{1}(1,j)*V(j));
    s2=s2+0.9*(P{2}(1,j)*V(j));
    s3=s3+0.9*(P{1}(2,j)*V(j));
    s4=s4+0.9*(P{2}(2,j)*V(j));
    s5=s5+0.9*(P{1}(3,j)*V(j));
    s6=s6+0.9*(P{2}(3,j)*V(j));
end
V=[c(1,1)+s1,c(1,2)+s2;c(2,1)+s3,c(2,2)+s4;c(3,1)+s5,c(3,2)+s6];
[minValues,minIndices]=min(V,[],2)%actions for optimal policy
V=[min(c(1,1)+s1,c(1,2)+s2);min(c(2,1)+s3,c(2,2)+s4);min(c(3,1)+s5,c(3,2)+s6)]
%policy evaluation
%we know the optimal policy is a2,a1,a2
```

```
syms V1 V2 V3
equations=[c(1,2)+0.9*(P{2}(1,1)*V1+P{2}(1,2)*V2+P{2}(1,3)*V3)==V1 , c(2,1)+0.9*(P{1}(2,
s = solve(equations, [V1 V2 V3]);
V1=s.V1;
V2=s.V2;
V3=s.V3;
V1=vpa(V1,6);
V2=vpa(V2,6);
V3=vpa(V3,6);
V=[V1;V2;V3]
%policy improvement
for j=1:3
    s7=s7+0.9*(P{1}(1,j)*V(j));
    s8=s8+0.9*(P{2}(1,j)*V(j));
    s9=s9+0.9*(P{1}(2,j)*V(j));
    s10=s10+0.9*(P{2}(2,j)*V(j));
    s11=s11+0.9*(P{1}(3,j)*V(j));
    s12=s12+0.9*(P{2}(3,j)*V(j));
end
V=[c(1,1)+s7,c(1,2)+s8;c(2,1)+s9,c(2,2)+s10;c(3,1)+s11,c(3,2)+s12];
[minValues,minIndices]=min(V,[],2)
V=[min(c(1,1)+s7,c(1,2)+s8);min(c(2,1)+s9,c(2,2)+s10);min(c(3,1)+s11,c(3,2)+s12)]

V =


 25.955139169646827213000506162643
 26.378867983206419012276455760002
 25.426450007774747064104303717613



minValues =
```

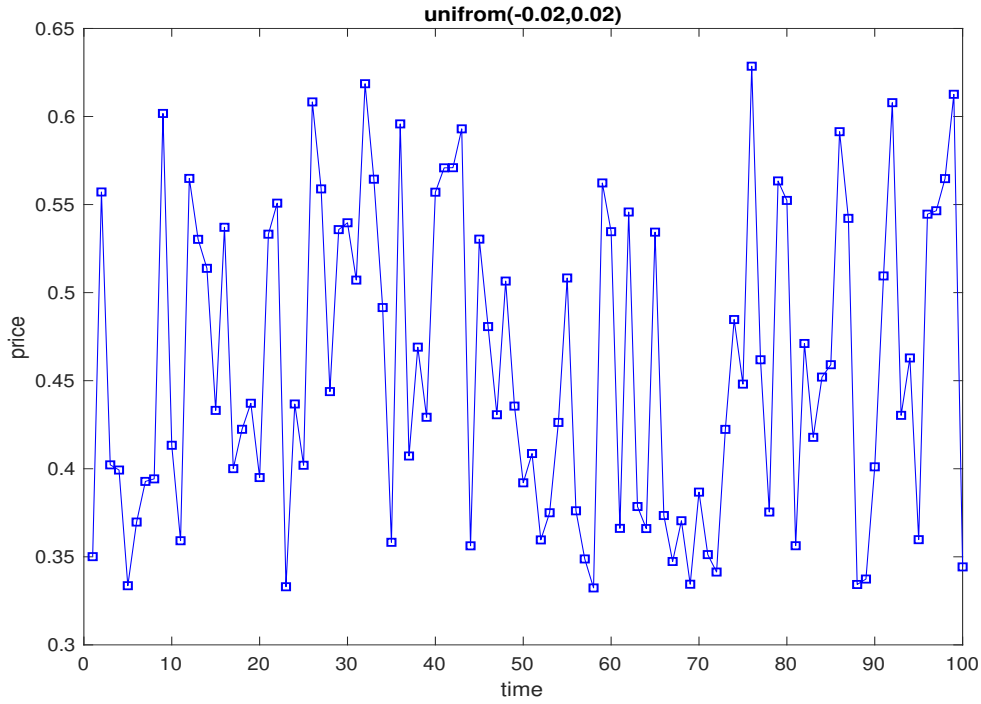
```
 24.774257502721161472436506301165
 24.617127973876563373778481036425
 24.650404291711947735166177153587


minIndices =

     2
     1
     2


V =

 24.774257502721161472436506301165
 24.617127973876563373778481036425
 24.650404291711947735166177153587


V =

 12.617851622874695749487727880478
 11.738794435857698772451840341091
 12.791731066460670263040810823441


minValues =

 12.617851622874795793904922902584
 11.738794435857726057292893528938
  12.791731066460592955599116012454
```

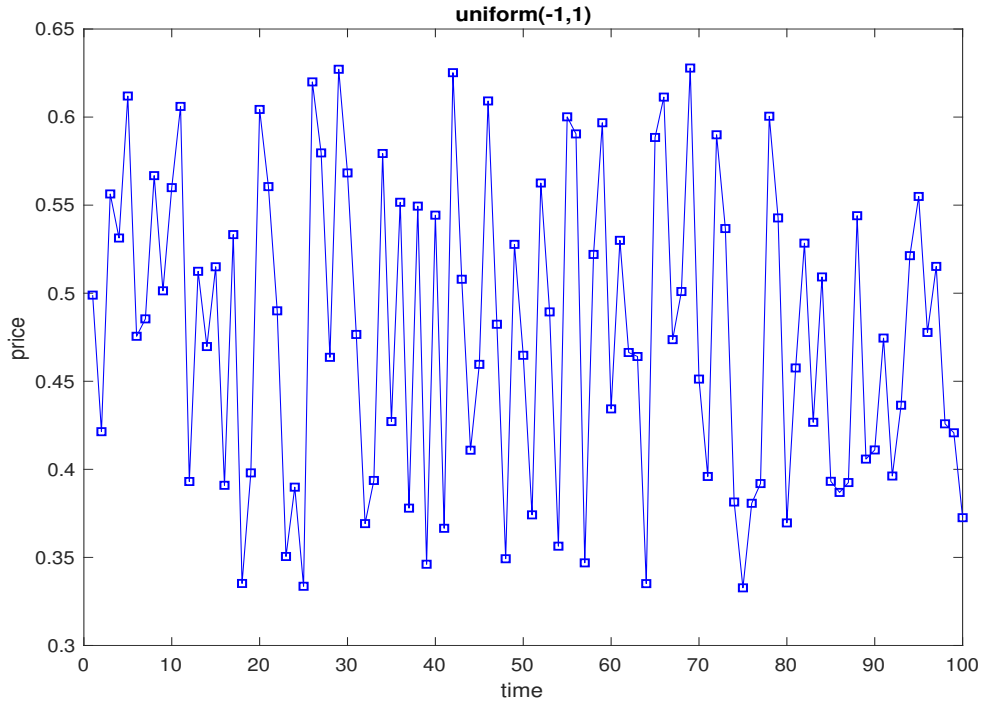
```
minIndices =

     2
     1
     2


V =

 12.617851622874795793904922902584
 11.738794435857726057292893528938
  12.791731066460592955599116012454
```

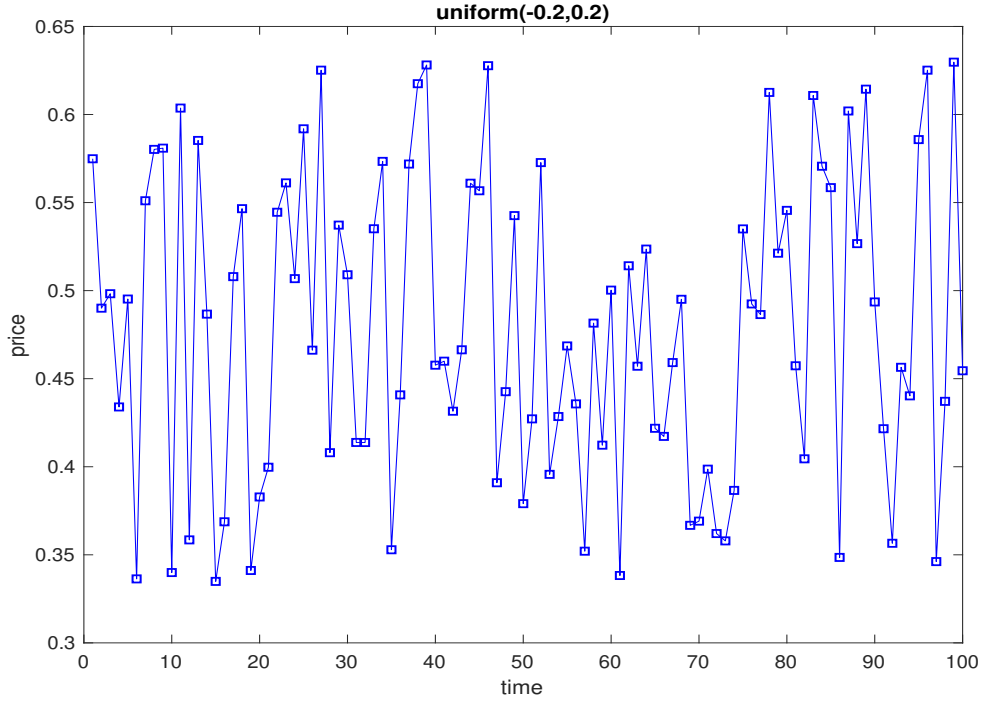
## C Choice Of Parameters For The Price Model

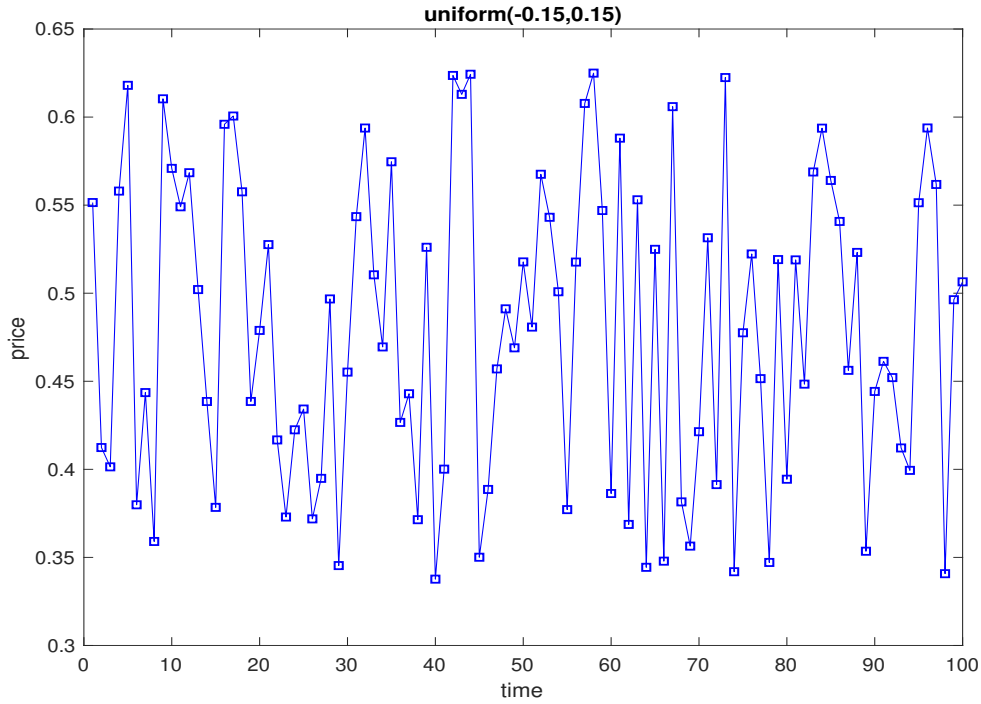
```
r1 = -0.02 + (0.02+0.02)*rand(100,1);
price=0.4;
price1=0.3*(0.5-price)+r1;
r2 = -1 + (1+1)*rand(100,1);
price=0.4;
price2=0.3*(0.5-price)+r2;
r3 = -0.2 + (0.2+0.2)*rand(100,1);
price=0.4;
price3=0.3*(0.5-price)+r3;
r4 = -0.15 + (0.15+0.15)*rand(100,1);
price=0.4;
price4=0.3*(0.5-price)+r4;
r5 = -0.25 + (0.25+0.25)*rand(100,1);
```

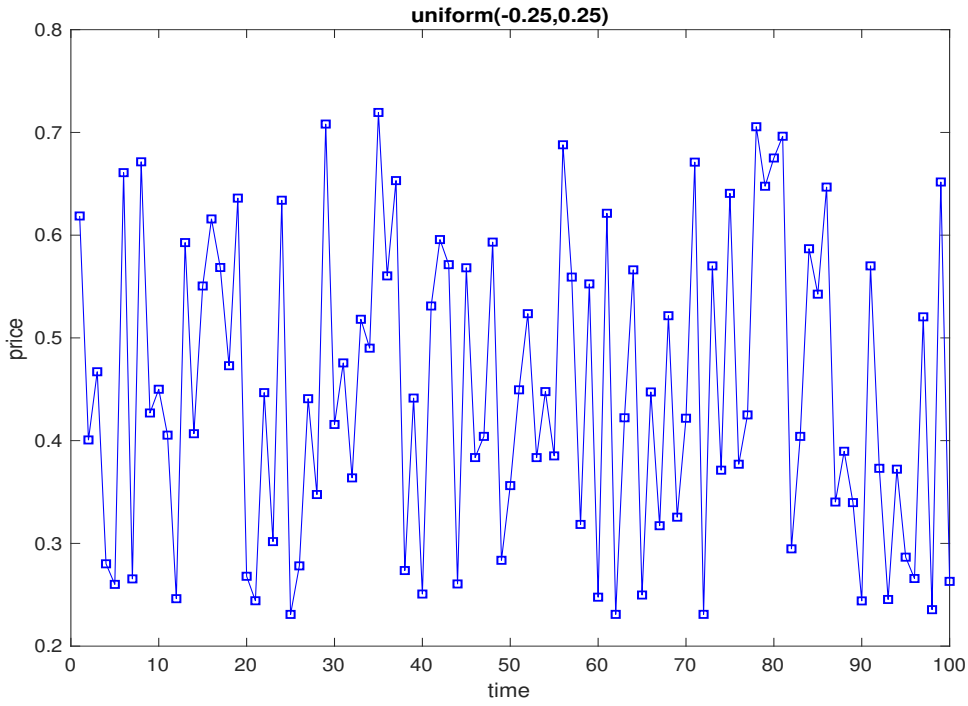
```
price=0.4;
price5=0.3*(0.5-price)+r5;

figure(1)
plot(price1,'b-s');
figure(2)
plot(price2,'b-s');
figure(3)
plot(price3,'b-s');
figure(4)
plot(price4,'b-s');
figure(5)
plot(price5,'b-s');
```


unifrom(-0.02,0.02)
price
time
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0
10
20
30
40
50
60
70
80
90
100


uniform(-1,1)
price
time
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0
10
20
30
40
50
60
70
80
90
100

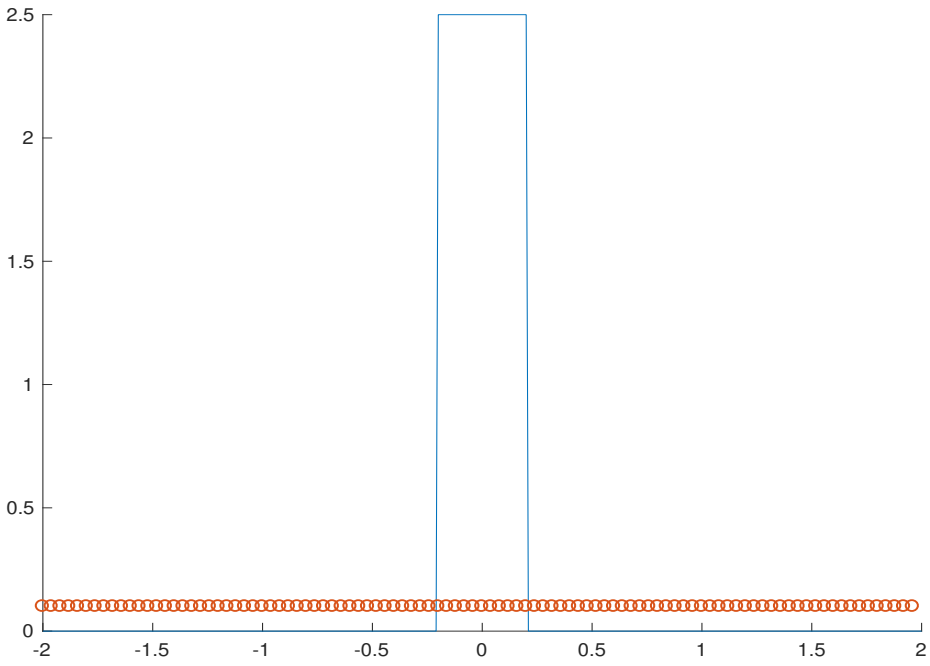

uniform(-0.2,0.2)
price
time
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0
10
20
30
40
50
60
70
80
90
100


uniform(-0.15,0.15)
price
time
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0
10
20
30
40
50
60
70
80
90
100

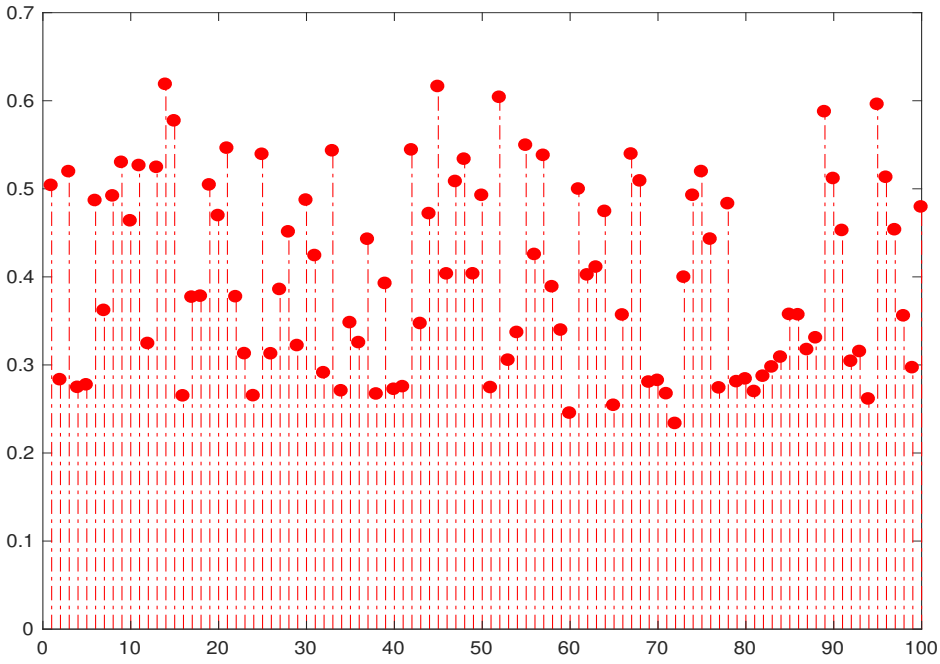



# D Discretize Uniform Distribution

```
x = -2:.01:2;
accuracy = 100;
uniform_ctn_to_discrete_converter(-0.2,0.2,x,accuracy,true);
function discrete_pdf = uniform_ctn_to_discrete_converter(lower, upper, x,accuracy,draw)
pd = makedist('Uniform','lower',lower,'upper',upper);
pd_function = @(x)1/(upper-lower)+0*x;
ctn_pdf=pdf(pd,x);

index_width = floor(numel(x)/accuracy);
halfwidth = (x(numel(x))-x(1))/accuracy/2;
discrete_x = 1:1:accuracy;
discrete_pdf = 1:1:accuracy;
for n = 1:1:accuracy
    x_index = 1+ index_width*(n-1);
    discrete_x(n) = x(x_index);
```




```
    min_value=x(x_index)- halfwidth;
    max_value=x(x_index)+ halfwidth;
    discrete_pdf(n) = integral(pd_function,min_value,max_value);
end
if draw
    figure;
    hold on;
    plot(x,ctn_pdf);
    scatter(discrete_x,discrete_pdf);
    hold off;
end

end
```




## E Discretize Price

```
x = 0:0.01:1;
price=0.4;
r1 = -0.2 + (0.2+0.2)*rand(100,1);
price1=0.3*(0.5-price)+r1;
price1=price+price1;
stem(price1,'fill','r-.');
```

## References

[1] Bertsekas, D.P.(2000) Dynamic Programming And Optimal Control (Vol. I) *Belmont, MA: Althena Scientific*

[2] Ross, S.M.(2014) Introduction to Probability Models(11th ed.) *S.1:Elsevier Academic Press*

[3] Bertsekas, D.P.(2000) Dynamic Programming And Optimal Control (Vol. II) *Belmont, MA: Althena Scientific*

[4] Ross, Sheldon M. (1983) Introduction to stochastic dynamic programming *Academic Press*

[5] Blackwell, D. (1965) Discounted dynamic programming *Ann. Math. Statist. 36,226-235*

[6] Hénaf, P., Laachir, I., Russo, F. (2013) Gas Storage Valuation and Hedging: A Quantification of Model Risk *International Journal of Financial Studies, 6, 27*

[7] Eydeland, A., Wolyniec, K.(2003) Energy and power risk management *New developments in modeling, pricing, and hedging, pp 504*

[8] Chen, Z., Forsyth, P.A. (2010) Implications of a regime-switching model on natural gas storage valuation and optimal operation *Quantitative Finance, 10, 159–176*

[9] Bauerle, N., Riess, V.(2016) Gas storage valuation with regime switching *Energy Systems 7(3), 499–528*

[10] Boogert, A., De Jong, C. (2008) Gas storage valuation using a Monte Carlo method *The Journal of Derivatives pp. 81–98*

[11] Lai, G., Margot, F., Secomandi, N.(2010) An approximate dynamic programming approach to benchmark practice-based heuristics for natural gas storage valuation *Operations Research, 58, 564–582, (2010)*

[12] Boogert A., De Jong, C. (2011) Gas storage valuation using a multi-factor price process *J. Energy Mark.4(4),29-52*

[13] Chen, Z., Forsyth, P.A. (2008) A semi-Lagrangian approach for natural gas storage valuation and optimal operation *SIAM Journal on Scientific Computing 30(1), 339–368*

[14] Thompson, M., Davison, M., Rasmussen, H. (2009) Natural gas storage valuation and optimization: A real options application *Naval Research Logistics (NRL) 56(3), 226–238*

[15] Warin, X. (2012) Gas storage hedging *Numerical Methods in Finance: Bordeaux, June 2010, pp. 421–445. Springer, Berlin, Heidel- berg*

[16] Felix, B., Weber, C. (2012) Gas storage valuation applying numerically constructed recombining trees *European Journal of Operational Research, 216, 178–187*

[17] Maragos, S.(2002) Valuation of the operational flexibility of natural gas storage reservoirs *Real Options and Energy Management Using Options Methodology to Enhance Capital Budgeting Decisions, 431-456*

[18] Kjaer, M., Ronn, E.I.(2008) Valuation of a natural gas storage facility *The journal of Energy Markets(3-22)*

[19] Malyscheff, A.M., Trafalis, T.B. (2017) Natural gas storage valuation via least squares Monte Carlo and support vector regression *Springer-Verlag Berlin Heidelberg*

[20] Bauerle, N., Riess, V.(2014) Gas Storage Valuation with regime switching *arXiv: 1412.1298*

[21] Secomandi, N. (2010) Optimal Commodity Trading with a Capacitated Storage Asset *Management Science, 56, 449-467*

[22] Wen, Z., Durlofsky, L.J., Benjamin, V.R., Aziz, K. (2012) Approximate Dynamic Programming For Optimizing Oil Production