NUMERICAL SOLUTION OF ZERO-SUM STOCHASTIC GAMES AND RELEVANT STOCHSTIC CONTROL SOFTWARE PACKAGE

A Thesis

Presented in Partial Fulfillment of the Requirements for

the Degree Master of Science in the

Graduate School of the Ohio State University

By

Ziyi Sun, B.S.

*****

The Ohio State University

2006

Master's Examination Committee:

Dr. José B. Cruz, Jr., Adviser

Dr. Randolph L. Moses

Approved by

[illegible]

Adviser

Electrical Engineering Graduate Program

ABSTRACT

Tremendous efforts have been devoted to stochastic games and control problems because of their wide applications in practice. A traditional approach for solving stochastic problems is to theoretically analyze the differential equations, which describe the dynamics of the underlying systems. However, the uncertainty that the environment builds into the system makes solving for analytical solutions impractical. Based on the deficiency of standard methods, in this thesis, we apply the Markov Chain Approximation method, developed by Kushner, to numerically solve for the value function and optimal strategies of control problems, and extended it to solve zero-sum stochastic games. In addition, we develop a stochastic control software package to assist in the computation of solution for stochastic problems, as well as Markov Decision Processes. Numerical examples in MDPs are given to demonstrate the use of our package. A one-to-one stochastic Pursuit-Evasion game is provided as an application of the Markov Chain Approximation method. The results have been extended to multiple stochastic PE game by using a decentralized approach.

To my family

ACKNOWLEDGMENTS

I gratefully acknowledge the constant and invaluable academic support received from my advisor, Dr. Jose B. Cruz, Jr. throughout my two-year graduate study at the Ohio State University. Thank you for leading me to the world of game theory; thank you for guiding me and giving me constructive suggestions on my research work. Above all, thank you for preparing me for future challenges.

I also want to thank Dr. Randolph L. Moses, at the Ohio State University for being in my committee and for his valuable suggestions on my thesis writing. My sincere thanks also go to Dongxu, Li for his stimulating discussions on this thesis, to Xiaohuan Tan, for her earnest help with my study, and to my teammates Mo Wei, Dan Shen and Xu Wang for their kind assistance.

I am also deeply indebted to all my classmates in the control group at OSU and friends from Chinese association for their friendship and encouragement

Last but not least, I want to express my gratitude to my dearest parents, for their love, patience, guidance and support throughout all these years. I couldn't have completed my studies without them. I miss them so much and wish them all the best with my deepest heart.

VITA

May 21, 1983	Born – Nanchang, China.
June 2004	B.S. Peking University, Beijing, China

FIELDS OF STUDY

Major Field: Electrical Engineering.
Studies in: Controls

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Stochastic Games

Game theory is a methodology for the study of competitive interaction among players with the relationship ranging between conflict and cooperation. Dynamic games are used to model competitive processes evolving over time. Stochastic games are dynamic games with stochastic transitions, where stochastic transitions are used to model or to formalize inherent uncertainty. They enjoy rich and mature mathematical theories [2], [3] and have a wide range of applications including economics, military, population and evolutionary biology, queuing theory and performance evaluation.

Stochastic games concentrate on decision situations where at different time points the players have to make a choice. The joint choices of all the players together have two applications. First, each player receives some reward, or cost when this reward is negative. Second, the underlying dynamic system moves on along its trajectory. However, the system here plays a role in the sense that the transition is the outcome of a random experiment, which might be dependent on the choices the players made. The dynamic system of a stochastic game is defined in term of a state space and the transitions are defined as moves from one state to another. In any of the states, players have their own action sets, which might be state dependent. When the system arrives at a state, each of the players has to choose an action from his available action set. In the course of a stochastic game being played, each of the players is rewarded by a series of immediate payoffs at different decision moments.

Stochastic games were first defined by L.S. Shapley [1], who studied simple zero-sum stochastic games with probabilistic moves and real pay-off. He proved that the value functions exist for such games and both players have stationary optimal strategies. Shapley also discovered an algorithm by which it is possible to find both the value of the game and the optimal strategies. The algorithm is nearly identical to value iteration for Markov Decision Processes. The other algorithm for solving the simple stochastic games, which is an extension of policy iteration, was introduced by Pollatsched and Avi-Itzhak [11]. Later, stochastic games, which differ from Shapley games in that they can be infinite, have been studied. They are called stochastic games with limiting mean pay-off. In 1981, J. F. Mertens and A. Neyman proved the existence of the value of such a game and of stationary optimal strategies under a hypothesis of ergodicity of the Markov chain [4]. These results have been generalized to cases where restrictions on the number of states and elementary strategies have been removed and to the case of other forms of pay-off [5, 6]. Recent results in zero-sum stochastic games include studying in zero-sum stochastic games with borel state spaces [31] and zero-sum ergodic stochastic games [32, 33].

We will be concerned here with the computation of zero-sum stochastic games. For a deterministic game, one can apply dynamic programming techniques to obtain a solution by computing the appropriate value function. However, this method is only tractable for very simple games. For reasonably complex games such as chess, one must typically apply heuristic techniques and receding horizon approximations in order to reduce the computational complexity. For stochastic games, the value function is defined over the space of all possible probability distributions over the state space. Thus, the problem is much more computationally intensive. Since the dynamics of stochastic games are always described by stochastic differential equations (SDE), it was tempting to try to solve for or approximate the various cost functions and optimal strategies by dealing directly with the appropriate differential equation and numerically approximating their solution. However, a basic impediment is that the differential equations often have only a theoretical meaning which could be impracticable, and standard methods of numerical analysis might not be usable to prove convergence of the numerical methods. For many problems of interest, one cannot even write down a differential equation explicitly. During the recent past, more analytical methods, such as applying reinforcement learning to stochastic games [34, 35], have been studied but even then it seems that many important classes of problems are still not covered. In 1977, Kushner suggested the method of Markov Chain Approximation to approach the numerical solution of stochastic control problems [7]. This method applies a Markov chain approximation to continuous time, continuous state stochastic control problems by renormalizing finite different forms as proper Markov chain transition probabilities. An important advantage of this method is that the Markov chain approximation facilitates convergence proofs for the numerical methods in terms of probabilistic arguments. This method is further advanced by Kushner [8], and by Kushner and Dupuis [9], with the special attention to convergence proofs. This method has been shown to be robust and to converge under broad conditions, and it is a good algorithm for solving the numerical problems, if the dimension is not too high.

1.2 Pursuit-Evasion Games and Motivation

Pursuit-Evasion (PE) games vary in complexity ranging from pursuers capturing evaders, or following moving target, to avoiding stationary targets and navigating. The games have received serious attention from game theorists for decades because of their importance in tactical air combat and other military applications [10].

The most commonly used techniques adapted to deal with PE games include the classical calculus of optimal control technique, dynamic programming, reinforced machine learning and game theory. Traditional game theory has focused on games with discrete moves. However, in the 1950s, Isaacs [13] utilized game theory in modeling and analyzing pursuit-evasion situations such as aerial combat, where moves unfold continuously over time, and control system can vary continuously in the strategies they implement. Isaacs had two basic insights. First, Pursuit-Evasion games do require game theory rather than simple optimality theory. Second, the continuous nature of Pursuit-Evasion games can be modeled using differential equations that specify how state conditions change incrementally as a function of players' strategies and previous state, such that the moves become continuous trajectories through a state-space. Afterward, differential game theory has developed as a tool for analyzing the structure and complexity of PE games. Differential PE games are defined by a set of controls (the actions of each player), a set of dynamics (mapping from the control variables onto the state variables of the game), and a set of termination conditions (state conditions that determine when successful capture or evasion happens). For example, Berkovitz [12] analyzed a classic case, in which a pursuer and an evader move with constant speed in a plane and control the direction of their velocity vector.

Though differential theory may prove useful in analyzing PE games, generally speaking, PE games are characterized by various dimensions of complexity: the number of players (from the two-player case to the multi-player case), the number of moves (from discrete action space to continuous action space), the continuity of the strategy space (from discrete state space to continuous state space), the structure of value function (from zero-sum games to non zero-sum games) and the information structure (from games with perfect information to games with imperfect information). These difficulties indicate that differential PE games are complicated to analyze even under the best circumstances, and that the introduction of realistic complexity makes most of them numerically unfeasible.

To avoid these complexities, differential game theory usually assumes that the PE game is one of perfect information between two players with fixed and pre-determined roles, deterministic dynamics, constant speeds and a zero-sum payoff structure. The case of two-player zero-sum PE games has been solved by computing the value function of the game by means of the Isaacs equation [11]. Mathematically proficient researchers can relax one or two of these assumptions at a time to obtain results for special and simplified cases. Li and Cruz proposed a hierarchical approach and a rigorous approach for solving multiple players PE games [14]. Hepanha, Kim, and Sastry introduced a probabilistic framework of PE game simulation where a swarm of autonomous pursuers are chasing an evader and the objective is to come up with a policy that will maximize the probability of finding the evader in finite time [15]. However, relaxing all the assumptions at once makes the game hopelessly complex. A recent complexity-theoretic analysis of differential PE games by Reif and Tate [16] illustrates the difficulties of designing a control system for robots and autonomous vehicles playing such games.

Therefore, although differential game theory provides a framework for describing the important features of PE games, and a set of normative results concerning optimal strategies in simple cases, it cannot generally provide optimal strategies for practically PE problems, even though the solution can be shown theoretically to exist and to be unique. It motivates us to apply the Markov Chain Approximation method to stochastic PE games, to numerically compute the value functions and optimal strategies.

1.3 Thesis Organization

After this introductory chapter, this thesis is divided into 5 more chapters. In Chapter 2, we formulate stochastic control problem and stochastic games in SDE. We provide Itô's Formula and the differential operator, which gives a theoretical approach to solve the SDE. Then we describe the basic idea of Markov chain approximation method and derive the formula for optimal control problems. We extend it to two-player zero-sum stochastic differential games, in which it is proved that the value exists and the numerical methods converges to this value.

Chapter 3 focuses on the software package which we construct to solve general MDPs problems. We provide algorithms for solving various classes of MDPs, including value iteration for discrete state space models, and Bellman Equation collocation methods for continuous state space models.

In Chapter 4, we introduce the stochastic control software package which we develop to solve general MDPs, stochastic control and zero-sum stochastic games. Description of the functions is provided and the use of this package is explained in details.

In Chapter 5, first we illustrate the use of our stochastic control software package by examples in the field of economics. Then we give a model of a one-to-one PE game and a model of multiple players PE game. We apply the results of Chapter 2 and functions in our software packages to these models. Numerical solutions are computed and discussed.

In the last chapter, we summarize the results obtained in the previous chapters and give some suggestions on future research directions.

1.4 Thesis Contributions

Firstly, we develop a method for the computation of zero-sum stochastic games. We extend the Markov Chain Approximation method, which is initially developed to approximate stochastic control problems by Markov chains, to zero-sum stochastic games. We apply the method to zero-sum stochastic Pursuit-Evasion games, and the simulation results demonstrate the success of the application.

Secondly, we develop a stochastic software package, including functions written as MATLAB files. The package is capable of solving general Markov Decision Processes problems, for both discrete and continuous state space. We use this package to numerically solve MDPs problems more conveniently and systematically.

CHAPTER 2

MARKOV CHAIN APPROXIMATION FOR ZERO-SUM STOCHASTIC DIFFERENTIAL GAMES

In this chapter, we first formulate two-player zero-sum stochastic Pursuit-Evasion games. We provide necessary mathematic background used to formulate and analyze stochastic problems theoretically. Then we introduce the Markov chain approximation method which numerically computes the value function and optimal strategies of stochastic game problems. The method is first derived to solve stochastic optimal control problems and then extended to solve two-player zero-sum stochastic games.

2.1 Problem Formulation

We consider a two-player, zero-sum stochastic Pursuit-Evasion game described by

$$dx(t) = f(x(t), u(t), v(t), t)dt + \sigma(x(t))dw(t) \tag{2-1}$$

where w stands for standard wiener process, and $u(\cdot)$ and $v(\cdot)$ are the control of pursuer and evader respectively . The objective function of the game can be formulated as

$$J(x) = E_x\left(\int_0^\tau G(x,u,v)dt + Q(x(\tau))\right) \tag{2-2}$$

The value of the objective function is the expected value on the space of all random processes x starting at $x(0)$. In a stochastic zero-sum PE game, pursuer (evader) applies the control of $u(\cdot)(v(\cdot))$ to minimize (maximize) the objective function. If the value function exists, the value of the game is defined as

$$V(t, x(t)) = \min_u \max_v \{J(x(t), t, u, v)\} \tag{2-3}$$

In the rest of the thesis, we assume that each player has access to perfect state information. The existence of value functions for general two-player, zero-sum stochastic differential games has been proved by W. H. Fleming and P. E. Souganidis [17]. Since a closed-form solution to the optimal strategies is not available, our objective is to numerically solve the problem and obtain the optimal strategies for both of the pursuer and the evader.

2.2 Stochastic Control Problems

2.2.1 Stochastic Differential Equation

One of the most important models for stochastic systems is the stochastic differential equation (SDE) of the form

$$x(t) = x(0) + \int_0^t f(x(s))ds + \int_0^t \sigma(x(s))dw(s) \tag{2-4}$$

where $x(\cdot)$ is an $\mathbb{R}^n$-valued process with continuous sample paths, $w(\cdot)$ is an $\mathbb{R}^k$-valued process which serves as a "driving noise", $f(\cdot)$ and $\sigma(\cdot)$ are bounded measurable functions mapping $\mathbb{R}^n$ into $\mathbb{R}^n$ and into the space of real $n \times k$ matrices, respectively. The above equation can also be written in the symbolic form

$$dx(t) = f(x(t))dt + \sigma(x(t))dw(t) \tag{2-5}$$

We will consider stochastic integrals with respect to the Wiener process. The classic probabilistic tool used to analyze the solution to (2-4) is the theory of stochastic differential equations due to K. Itô [18]. The resulting stochastic integral and related theory of SDE provide a very convenient family of models that are Markovian and possess continuous sample paths.

Let $x(\cdot)$ be the solution to the above SDE. The link between the process $x(\cdot)$ and certain second order partial differential equations is provided by Itô's formula and the differential operator [18]. Let $A(x) = \sigma(x)\sigma(x)^T = \{a_{ij}(x)\}$, $i, j = 1, \ldots, n$, and for any twice continuously differentiable real value function g on $\mathbb{R}^n$, define the differential operator as

$$(\Gamma g)(x) = g_x^{\,T}(x)f(x) + \frac{1}{2}tr[g_{xx}(x)A(x)]. \tag{2-6}$$

where

$$g_x(x) = \left[\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2}, \ldots, \frac{\partial g}{\partial x_n}\right]^T \tag{2-7}$$

$$g_{xx}(x) = \left\{g_{x_i x_j}(x)\right\} = \left\{\frac{\partial g^2}{\partial x_i \partial x_j}\right\}, \quad i, j = 1, \ldots, n \tag{2-8}$$

$$tr[g_{xx}(x)A(x)] = \sum_{ij} g_{x_i x_j}(x)a_{ij}(x) \tag{2-9}$$

Then Itô's formula states that

$$g(x(t)) = g(x(0)) + \int_0^t (\Gamma g)(x(s))ds + \int_0^t g_x^{\,T}(x(s))\sigma(x(s))dw(s) \tag{2-10}$$

2.2.2 Formulation of Stochastic Control Problem

A stochastic control problem is generally described in the form of SDE

$$dx(t) = f(x,u)dt + \sigma(x)dw \tag{2-11}$$

where $x(\cdot), f(\cdot), \sigma(\cdot)$ are defined as above, and $u(\cdot)$ is the control that takes values in a compact set U. It is a pure Markov control if the control can be written as a function of the current state and time. In the remainder of this thesis, we will consider pure Markov controls that depend only on the state.

The objective function is defined as

$$J(x,u) = E_x^u[\int_0^\tau G(x(s),u(s))ds + Q(x(\tau))] \tag{2-12}$$

where $\tau = \inf\{t : x(t) \in \partial S\}$, S is the terminal set, and $E_u^x(\cdot)$ is the expectation of the expression with respect to u and x. The optimal value of this control problem is defined as

$$V(x) = \inf_{u \in U} J(x,u) \tag{2-13}$$

We now apply a formal dynamic programming argument to derive the partial differential equation (PDE) which is satisfied by the optimal value function $V(\cdot)$. Refer to [20, 21] for more development details. Suppose that $V(\cdot)$ is as smooth as necessary for the following calculations to be valid and there is an optimal control $u^*(\cdot)$ which is pure Markov. Let $\Delta > 0$ and let α be any value in U. Define $\tilde{u}(\cdot)$ to be the composite control policy that uses the feedback control $u^*(\cdot)$ for $t \geq \Delta$ and uses the control identically equal to α for $t < \Delta$. Define the process $\tilde{x}(\cdot)$ to be the process which corresponds to use of the control $\tilde{u}(\cdot)$. Let $\tilde{\tau}$ be the time that terminal set is reached under this control policy. Let $x(\cdot)$ and τ be the solution and escape time under the optimal control $u^*(\cdot)$. By definition, we have

$$V(x) = E_x^{u^*}[\int_0^\tau G(x(s),u^*(x(s)))ds + Q(x(\tau))] \tag{2-14}$$

The optimality of $V(\cdot)$ implies

$$\begin{aligned} V(x) &\le E_x^{\tilde{u}}[\int_0^\tau G(x(s),\tilde{u}(x(s)))ds + Q(x(\tau))] \\ &= E_x^{\tilde{u}}[\int_0^{\tau\wedge\Delta} G(x(s),\alpha)ds + Q(x(\tau))I_{\{\tau<\Delta\}}] \\ &\quad + E_x^{\tilde{u}}[\int_\Delta^\tau G(x(s),\tilde{u}(x(s)))ds + Q(x(\tau))]I_{\{\tau>\Delta\}} \end{aligned} \tag{2-15}$$

By the Markov property, the definition of $\tilde{u}(\cdot)$, and the optimality of $u^*(\cdot)$, the inequality above may be rewritten as

$$V(x) \le E_x^{\tilde{u}}[\int_0^{\tau\wedge\Delta} G(x(s),\alpha)ds + Q(x(\tau))I_{\{\tau<\Delta\}} + V(x(\Delta))I_{\{\tau\ge\Delta\}}] \tag{2-16}$$

Therefore

$$\frac{1}{\Delta}E_x^{\tilde{u}}[V(x(\Delta)) - V(x) + \int_0^\Delta G(x(s),\alpha)ds] \ge \frac{1}{\Delta}E_x^{\tilde{u}}h(\tau,\Delta,\tilde{u})I_{\{\tau<\Delta\}} \tag{2-17}$$

where

$$h(\tau,\Delta,\tilde{u}) = V(x(\Delta)) - \int_{\tau\wedge\Delta}^\Delta G(x(s),\alpha)ds - Q(x(\tau)) \tag{2-18}$$

is bounded uniformly in w and Δ. If we assume the condition $P_x^{\tilde{u}}\{\tau<\Delta\}/\Delta \to 0$ as $\Delta \to 0$, then the right hand side of the inequality (2-15) tends to zero as $\Delta \to 0$. Therefore, combining with (2-6), (2-10) in Itô's theorem and taking this limit yields, for any value of α in U,

$$\Gamma^\alpha V(x) + G(x,\alpha) \ge 0. \tag{2-19}$$

Let us replace α by $u^*(x(s))$ on $[0,\Delta)$, and that $u^*(\cdot)$ is continuous at x. Then that analogue of (2-19) holds with the inequality replaced by equality. We then formally obtain the equation

$$\Gamma^{u^*(x)}V(x) + G(x,u^*(x)) = 0 \tag{2-20}$$

It follows that

$$\begin{cases} \inf\limits_{\alpha \in U} [\Gamma^{\alpha} V(x) + G(x,\alpha)] = 0, & x \in S^0 \\ V(x) = Q(x), & x \in \partial S \end{cases} \tag{2-21}$$

2.3 Markov Chain Approximation Method

2.3.1 Controlled Markov Chain

Markov chain and Markov decision processes (MDPs) are special cases of two-player zero-sum stochastic games. A Markov chain describes the dynamics of the states of a stochastic game where each layer has a single action in each state. Markov decision processes are stochastic games with a single player. A particular MDPs is defined by its state and action sets and by the one-step dynamics of the system. Let U be the set of possible controls with generic variable α. The transition probabilities are written as $p(x'|x,\alpha)$ or $p_{xx'}^{\alpha}$, which means the probability that the next state is x' when the current state is x and action α is taken. The cost is written as $r(x,\alpha)$ which represents the cost received at this time period when the current state is x and action α is taken.. Let $u = (u_0, u_1, \ldots)$ denote the sequence of control actions taken at time $0, 1, \ldots$. We say that u is admissible if the Markov property continues to hold under use of u

$$P\{x_{t+1} = x' | x_i, u_i, i \le t\} = P\{x_{t+1} = x' | x_t, u_t\} = p(x'|x,u) \tag{2-22}$$

If there is a function $u(\cdot)$ such that $u_t = u(x_t)$, then we refer to the control as a feedback or pure Markov policy.

Classified by the form of objective functions, there are two main types of controlled Markov chain models.

(1) Discounted cost

Suppose that $\gamma > 0$. For an admissible control sequence u, the objective function is defined in the form

$$J(x,u) = E_x^u \sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \tag{2-23}$$

Let $V(x)$ denote the infimum of the objective function $J(x,u)$ over all admissible control sequences and defined as

$$V(x) = \inf_{u \in U} E_x^u \sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \tag{2-24}$$

The value function $V(x)$ is finite for each x due to the discounting and satisfies the dynamic programming

$$V(x) = \min_{\alpha \in U} [\gamma E_x^\alpha V(x') + r(x,\alpha)] \tag{2-25}$$

(2) Control to a terminal Set

When the discounted factor is absent, let S be the terminal set and ∂S denote the boundary of S. A controlled process stops once the state reaches S. The objective function is defined as

$$J(x,u) = E_x^u \sum_{t=0}^{T-1} r(x_t, u_t) + E_x^u q(x_T) \tag{2-26}$$

Let us define the optimal value function $V(x) = \inf_{u \in U} J(x,u)$, where the infimum is over all admissible control sequences. The dynamic programming equation for the objective function is

$$V(x) = \begin{cases} \inf_{\alpha \in U}[E_x^\alpha V(x') + r(x,\alpha)], & x \in S - \partial S \\ q(x), & x \in \partial S \end{cases} \tag{2-27}$$

2.3.2 Markov Chain Approximation Method

For standard control problems, we apply Itô's formula to derive the partial differential equations for the optimal cost. These partial differential equations are generally known as Bellman equations (PDE) or dynamic programming equations. Their forms are very suggestive of good numerical methods. However, the equations themselves have only a theoretical meaning, and little is known concerning the existence and uniqueness of the solution. Also, the operator in Itô's formula is often difficult to calculate. Based on these obstacles, we introduce the Markov chain approximation (MCA) method developed by Kushner. This MCA method can be applied to a very broad class of stochastic and deterministic control problems.

The MCA method suggests that we approximate the original problem with a Markov chain on a finite state space, which is a discretization of the original state space of the underlying problem, and associated objective function for which the desired computation can be carried out. The approximation Markov chain is chosen such that certain local properties of the approximation chain are similar to those of the original controlled process. Then the objective function for the Markov chain model which approximates the objective function for the original model is constructed. This procedure can be used almost automatically, in that there are standard methods to construct the chains and objective functions [8].

And since there are many methods to solve the Markov chain problems, the computation for the chosen Markov chain is accessible at the level of application.

The approximating chain is characterized by a parameter, which is analogous to a finite difference interval or to a finite element size in classical numerical analysis. When the parameter goes to zero, the local properties of the chain are more and more similar to those of the original process. It can be proved that the sequence of optimal value functions for the sequence of approximating chains converges to that for the original process as the approximation parameter goes to zero under broad conditions. We omit the proof here because this thesis's emphasis lies on the application. Refer to [8, 9, 27] for details.

The specific method of MCA to be discussed and used later in the thesis is called the finite difference method. We use it to construct the locally consistent approximating Markov chains. It turns out that when a carefully chosen finite difference approximation is applied to the differential operator in Itô's formula, the coefficients of the resulting discrete equation can serve as the desired transition probabilities and interpolation interval.

We work with the controlled process $x(\cdot)$ satisfying (2-11) with objective function defined in (2-12). The optimal value function is defined as (2-13).

Define the matrix

$$A(x) = \sigma(x)\sigma(x)^T = \{a_{ij}(x)\}, \quad i, j = 1, \ldots, n.$$

The differential operator Γ^{α} of (2-24) is defined as:

$$\Gamma^{\alpha} = \sum_{i=1}^{n} f_i(x,\alpha)\frac{\partial}{\partial x_i} + \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}(x)\frac{\partial^2}{\partial x_i \partial x_j} \tag{2-28}$$

By (2-21), we have the following partial differential equation

$$\inf_{u\in U}\left[\Gamma^{u(x)}V(x,u)+G(x,u(x))\right]=0 \tag{2-29}$$

Let e_i denote the unit vector in the i-th coordinate direction and let $\mathbb{R}_h^n$ denote the uniform h-grid on $\mathbb{R}^n$; i.e. $\mathbb{R}_h^n = \{x : x = h\sum_i e_i m_i : m_i = 0, \pm 1, \pm 2, ...\}$. Here, we use a uniform grid to demonstrate the construction process. However, such a uniform grid is not necessary. Let $S_h = \mathbb{R}_h^n$ denote the state space of the approximating Markov chain. We use $V^h(x,u)$ to denote the finite difference approximation to (2-13) and substitute suitable finite difference approximations for the derivatives at each state x. As we mentioned, the coefficients in the resulting finite difference equation can serve as the transition probabilities and interpolation interval automatically.

The Diagonal Case

First we apply the method on the case where $a_{ij}(x)=0$ for $i \neq j$.

For the first derivatives, we use one sided approximation

$$f_{x_i}(x) \to \begin{Bmatrix} [f(x+e_i h)-f(x)]/h & \text{if } f_i(x,\alpha)\geq 0 \\ [f(x)-f(x-e_i h)]/h & \text{if } f_i(x,\alpha)<0 \end{Bmatrix} \tag{2-30}$$

For the second derivative, we use the standard approximation

$$f_{x_i x_i}(x) \to \frac{f(x+e_i h)+f(x-e_i h)-2f(x)}{h^2} \tag{2-31}$$

Define the positive and negative parts of a real number by

$$a^{+} = \max[a,0], a^{-} = \max[-a,0] \tag{2-32}$$

Substitute (2-30), (2-31) and (2-32) into (2-29), we have

$$\min_{u\in U}\{\sum_{i=1}^{r} f_i^{+}(x,\alpha)[V(x+e_i h)-V(x)]/h+\sum_{i=1}^{r} f_i^{-}(x,\alpha)[V(x-e_i h)-V(x)]/h$$
$$+\sum_{i=1}^{r} a_{ii}[V(x+e_i h)+V(x-e_i h)-2V(x)]\}+G(x)=0 \qquad (2\text{-}33)$$

We collect terms and divide the terms by the coefficient of $V^h(x,u)$. Hence, we obtain the finite difference equation

$$V^h(x,u)=\min_{u\in U}\left[\sum_y p^h(y\mid x,u(x))V^h(y,u)+G(x,u(x))\Delta t^h(x,u(x))\right] \qquad (2\text{-}34)$$

where

$$p^h(x\pm e_i h\mid x,\alpha)=\frac{a_{ii}(x)/2+hf_i^{\pm}(x,\alpha)}{Q^h(x,\alpha)}$$
$$\Delta t^h(x,\alpha)=\frac{h^2}{Q^h(x,\alpha)} \qquad (2\text{-}35)$$
$$Q^h(x,\alpha)=\sum_i[a_{ii}(x)+h\mid f_i(x,\alpha)\mid]$$

We assume that $\Delta t^h(x,\alpha)=O(h)$. For y not taking on the listed values, we define $p^h(y\mid x,\alpha)=0$. The constructed $p^h(y\mid x,\alpha)$ are nonnegative, and for each x and α, they sum (over y) to 1. Thus, they can be considered as the transition probabilities for a controlled Markov chain.

The dynamic programming equation for the optimal value function is

$$V^h(x)=\min_{\alpha\in U}\left[\sum_y p^h(y\mid x,\alpha)V^h(y)+G(x,\alpha)\Delta t^h(x)\right] \qquad (2\text{-}36)$$

in which, $p^y(y\mid x,\alpha)$ serves as the transition probabilities. Thus, the above equation is in a standard form of Markov chain, and any conventional method to solve Markov Decision Process can be used to solve this problem.

The General Case

Now consider the case where the off-diagonal terms $a_{ij}(x),\ i \neq j$, are not all zero. For the first and second derivates, the formulas are the same as the previous. For the mixed derivatives, where $i \neq j$, we use the standard finite difference approximations: For $a_{ij}(x) \geq 0$, use

$$\begin{aligned} f_{x_i x_j}(x) \to & [2f(x) + f(x+e_i h + e_j h) + f(x - e_i h - e_j h)]/2h^2 \\ & -[f(x+e_i h) + f(x-e_i h) + f(x+e_j h) \\ & + f(x-e_j h)/2h^2 \end{aligned} \tag{2-37}$$

If $a_{ij} < 0$, use

$$\begin{aligned} f_{x_i x_j}(x) \to & -[2f(x) + f(x+e_i h + e_j h) + f(x - e_i h - e_j h)]/2h^2 \\ & -[f(x+e_i h) + f(x-e_i h) + f(x+e_j h) \\ & + f(x-e_j h)/2h^2 \end{aligned} \tag{2-38}$$

We assume that $a_{ii}(x) - \sum_{j:j \neq i} | a_{ij}(x) | \geq 0$, for all i, x. Substituting (2-30), (2-31), (2-32), (2-37) and (2-38) into (2-27), collecting terms and dividing the terms by the coefficient of $J^h(x,u)$, we get the finite difference equation as follows.

$$V^h(x,u) = \min_{u \in U} \left[\sum_y p^h(y \mid x, u(x)) V^h(y,u) + G(x,u(x)) \Delta t^h(x,u(x)) \right] \tag{2-39}$$

where

$$\begin{aligned} & p^h(x \pm e_i h \mid x, \alpha) = [a_{ii}(x)/2 - \sum_{j:j \neq i} | a_{ij}(x) | /2 + h f_i^{\pm}(x,\alpha)]/Q^h(x,\alpha) \\ & p^h(x + e_i h + e_j h \mid x, \alpha) = p^h(x, x - e_i h - e_j h \mid \alpha) = a_{ij}^{+}(x)/2Q^h(x,\alpha) \\ & p^h(x - e_i h + e_j h \mid x, \alpha) = p^h(x, x + e_i h - e_j h \mid \alpha) = a_{ij}^{-}(x)/2Q^h(x,\alpha) \\ & \Delta t^h(x,\alpha) = \frac{h^2}{Q^h(x,\alpha)} \\ & Q^h(x,\alpha) = \sum_i [a_{ii}(x) + h \mid f_i(x,\alpha) \mid] \end{aligned} \tag{2-40}$$

For y not taking on the listed values, define $p^h(y \mid x, \alpha) = 0$. The different choices for the finite difference approximation to the mixed second order derivatives are made in order to guarantee that the coefficients of the off-diagonal terms

$$p^h(x + e_i h \pm e_j h \mid x, \alpha), p^h(x - e_i h \pm e_j h \mid x, \alpha), \quad i \neq j,$$

are nonnegative. Also these choices guarantee that the coefficients sum to 1, so that they can be considered to be transition probabilities for an approximating Markov chain.

The dynamic programming equation for the optimal value function is

$$V^h(x) = \min_{\alpha \in U} \left[\sum_y p^h(y \mid x, \alpha) V^h(y) + G(x, \alpha) \Delta t^h(x) \right] \tag{2-41}$$

2.3.3 Extend to Two-Player Zero-Sum Stochastic Games

The approximating procedure is similar to the control problem.

For the dynamics of the game and the objective function described by (2-1) and (2-2), define the matrix

$$A(x) = \sigma(x)\sigma(x)^T = \{a_{ij}(x)\}, \quad i, j = 1, \ldots, n.$$

The differential operator $\Gamma^{\alpha,\beta}$ of (1) is defined as follows.

$$\Gamma^{\alpha,\beta} = \sum_{i=1}^{n} f_i(x, \alpha, \beta) \frac{\partial}{\partial x_i} + \frac{1}{2} \sum_{i,j=1}^{n} a_{ij}(x) \frac{\partial^2}{\partial x_i \partial x_j} \tag{2-42}$$

By Itô's formula, we have the following partial differential equation

$$\min_{u \in U} \max_{v \in V} \left[\Gamma^{u(x)v(x)} \overline{V}(x, u, v) + G(x, u(x), v(x)) \right] = 0 \tag{2-43}$$

Let e_i denote the unit vector in the i-th coordinate direction and let $\mathbb{R}_h^r$ denote the uniform h-grid on $\mathbb{R}^n$, i.e., $\mathbb{R}_h^n = \{x : x = h\sum_i e_i m_i : m_i = 0, \pm 1, \pm 2, \ldots\}$. We use $S_h = \mathbb{R}_h^n$ as the state space of the approximating Markov chain.

We only present for the general case here. By same construction process, with the assumption that $a_{ii}(x) - \sum_{j:j\neq i} |a_{ij}(x)| \geq 0$, for all i, x, we get the finite difference equation as follows

$$\overline{V}^h(x,u,v) = \min_{u\in U} \max_{v\in V} \left[\sum_y p^h(y \mid x, u(x), v(x)) \overline{V}^h(x,u,v) + G(x,u(x),v(x)) \Delta t^h(x,u(x),v(x)) \right] \tag{2-44}$$

where

$$\begin{aligned} &p^h(x \pm e_i h \mid x, \alpha, \beta) = [a_{ii}(x)/2 - \sum_{j:j\neq i} |a_{ij}(x)|/2 + hf_i^{\pm}(x,\alpha,\beta)] / Q^h(x,\alpha,\beta) \\ &p^h(x + e_i h + e_j h \mid x, \alpha, \beta) = p^h(x, x - e_i h - e_j h \mid \alpha, \beta) = a_{ij}^+(x)/2Q^h(x,\alpha,\beta) \\ &p^h(x - e_i h + e_j h \mid x, \alpha, \beta) = p^h(x, x + e_i h - e_j h \mid \alpha, \beta) = a_{ij}^-(x)/2Q^h(x,\alpha,\beta) \\ &\Delta t^h(x,\alpha,\beta) = \frac{h^2}{Q^h(x,\alpha,\beta)} \\ &Q^h(x,\alpha,\beta) = \sum_i [a_{ii}(x) + h \mid f_i(x,\alpha,\beta) \mid] \end{aligned} \tag{2-45}$$

For y not taking on the listed values, define $p^h(y \mid x, \alpha, \beta) = 0$.

Define $\underline{V}(x)$

$$\underline{V}(x) = \max_{v\in V} \min_{u\in U} [\sum_y p^h(y \mid x, u(x), v(x)) \underline{V}(y) + G(x,u(x),v(x)) \Delta t^h(x,u(x),v(x))] \tag{2-46}$$

If $\overline{V}(x) = \underline{V}(x)$, then $V(x) = \overline{V}(x) = \underline{V}(x)$ is called value function.

CHAPTER 3

COMPUTATION OF MARKOV DECISION PROCESSES AND A RELEVANT SOFTWARE PACKAGE

In this chapter, we provide the algorithms that we use in our software package for solving Markov decision processes with different structures of state space and action space. The different cases of MDPs that we discuss include: finite MDPs with finite horizon, finite MDPs with infinite horizon, MDPs with continuous state space. We also extend the algorithm for finite MDPs to the algorithm for two-player zero-sum games.

3.1 Discrete Time, Discrete State Space MDPs

3.1.1 Models

A Markov decision process is called a finite MDPs if its state space and action set are finite. A particular finite MDPs is defined by its state and action sets and by the one-step dynamics of the environment. Define U the set of possible actions with variable α. Let $\{x_t\}$ be the corresponding Markov chain defined on a finite state space X. In every period t, the player observes the state of a dynamic system x_t, takes an action u_t, and earns a reward (or cost if the reward is negative) $r(x_t, u_t)$ that depends on both the state of the system and the action taken. The player's objective is to seek a control strategy $\{\mu_t\}$ that given any state, the player take an action $u_t = \mu_t(x_t)$ which maximizes (minimizes) the objective function.

A discrete finite MDPs may be either deterministic or stochastic. In the stochastic case, given any state and action, x and α, the probability of each possible next state, x', is

$$p_{xx'}^{\alpha} = P\{x_{t+1} = x' \mid x_t = x, u_t = \alpha\} \tag{3-1}$$

These quantities are called transition probabilities. Similarly, given any state and action, x and α, the expected value of the next reward is

$$r(x,\alpha) = E\{r_{t+1} \mid x_t = x, u_t = \alpha\} \tag{3-2}$$

In the deterministic case, the next period's state is known with certainty once the current period's state and action are known, a state transition function f, instead the transition probability, can be used to explicitly gives the state transitions

$$x_{t+1} = f(x_t, u_t) \tag{3-3}$$

Let us consider the objective function with discounted cost for example. The objective function is in the following form

$$J_t = E_x^u \sum_{i=t}^{T} \gamma^i r(x_i, u_i) \tag{3-4}$$

where γ is a per-period discount factor. The value function is the maximization of the objective function and can be defined as

$$V_t = \max_u E_x^u \sum_{i=t}^{T} \gamma^i r(x_i, u_i) \tag{3-5}$$

A finite MDPs problem may have an infinite horizon ($T = \infty$) or a finite horizon ($T < \infty$).

The classic tool to analyze discrete finite MDPs is the dynamic programming methods developed by Richard Bellman [22]. The method is based on the principle of optimality. The principle of optimality can be expressed in the form of the Bellman equation, which implies that the value function $V_t(x)$ must satisfy

$$V_t(x) = \max_{\alpha \in U}\{r(x,\alpha) + \gamma \sum_{x' \in S} p_{xx'}^{\alpha} V_{t+1}(x')\} \tag{3-6}$$

In the case of infinite horizon, the value function $V(x)$ must satisfy

$$V(x) = \max_{\alpha \in U}\{r(x,\alpha) + \gamma \sum_{x' \in S} p_{xx'}^{\alpha} V(x')\} \tag{3-7}$$

3.1.2 Algorithm

1. Backward Recursion

For the finite MDPs with finite horizon, we can use backward recursion to compute the optimal value and policy functions V_t and u_t. The algorithm is described as follows.

```
Initialize the reward function $r$, transition probabilities $p$, discount factor $\gamma$, terminal
period $T$, terminal value function $V_{T+1}$
  Set $t \leftarrow T$
  Repeat
      For each $x \in X$
          $V_t(x) \leftarrow \max_{\alpha \in U}(r(x,\alpha) + \sum_{x'} \gamma p_{xx'}^{\alpha} V_{t+1}(x'))$
          $u_t(x) \leftarrow \arg\max_{\alpha \in U}(r(x,\alpha) + \sum_{x'} \gamma p_{xx'}^{\alpha} V_{t+1}(x'))$
          $t \leftarrow t-1$
  Until $t = 0$
```

However, it should be noticed that it may be possible to have more than one sequence of optimal policies due to the ties occurring when performing the maximization embedded in the Bellman equation.

2. Value Iteration

For the finite MDPs with infinite horizon case, one of the algorithms is called Value iteration. We consider the Bellman equation in (3-7) as a fixed –point equation. We can compute the optimal value and policy function V_t and u_t using standard function iteration methods. The algorithm is described as follows.

```
Initialize the reward function $r$, transition probabilities $p$, discount factor $\gamma$, convergence
tolerance $\theta$. Initialize $V$ arbitrarily, e.g., $V(x)=0$, for all $x \in X$
    Repeat
        For each $x \in X$
            $v\_old \leftarrow V(x)$
            $V(x) \leftarrow \max_{\alpha \in U} \sum_{x'} (r(x,\alpha) + \gamma p_{xx'}^{\alpha} V(x'))$
        $\Delta \leftarrow norm(v\_old - V)$
    Until $\Delta < \theta$
    Output a policy $u$, such that
        $u(x) \leftarrow \arg\max_{\alpha \in U} (r(x,\alpha) + \sum_{x'} \gamma p_{xx'}^{\alpha} V(x'))$
```

3.2 Discrete time, Continuous State Space MDPs

3.2.1 Model

In this section, we discuss the discrete time, continuous state space MDPs problems. The most distinct property of the discrete time, continuous state space

MDPs is that the state space X includes continuous state variables whose ranges are intervals of the real line. Define U the set of possible actions with variable α. Let $\{x_t\}$ be the corresponding Markov chain defined on a finite state space X. In every period t, the player observes the state of a dynamic system x_t, takes an action u_t, and earns a reward (or cost if the reward is negative) $r(x_t, u_t)$ that depends on both the state of the system and the action taken. The player's objective is to seek a control strategy $\{\mu_t\}$ that given any state, the player take the action $u_t = \mu_t(x_t)$ which maximizes (minimizes) the objective function.

A continuous MDPs may also be deterministic or stochastic. In the case of stochastic, the dynamic of the system can be described as

$$x_{t+1} = f(x_t, u_t, w_{t+1}) \tag{3-8}$$

where w_t serves as random noise. In the case of deterministic, w_t is absent. The general objective function with discounted cost is in the following form

$$J_t = E_w \sum_{i=t}^{T} \gamma^i r(x_i, u_i) \tag{3-9}$$

where γ is a per-period discount factor. The value function is the maximization of the objective function and can be defined as

$$V_t = \max_u E_w \sum_{i=t}^{T} \gamma^i r(x_i, u_i) \tag{3-10}$$

A continuous state space MDPs problem may have an infinite horizon ($T = \infty$) or a finite horizon ($T < \infty$).

Like the finite MDPs, the discrete time, continuous state space MDPs can be analyzed using dynamic programming methods, which can be expressed in the form of Bellman equation. By using the principle of optimality, it implies that the value function $V_t(x)$ must satisfy

$$V_t(x) = \max_{u \in U}\{r(x,u) + \gamma E_w V_{t+1}(f(x,u,w))\} \tag{3-11}$$

In the case of infinite horizon, the value function $V(x)$ must satisfy

$$V(x) = \max_{u \in U}\{r(x,u) + \gamma E_w V(f(x,u,w))\} \tag{3-12}$$

3.2.2 Algorithm

For the continuous state MDPs problems, we cannot calculate the value for each state one by one or express the value in a vector form. Hence, the algorithms for the finite MDPs problems are no longer suitable here. Instead, we introduce Bellman equation collocation method [23]. The basic idea of the collocation method is to approximate the value function by a linear combination of n known basic functions. Therefore, we can use these basic functions to calculate the value at any state.

Consider the Bellman equation (3-12) for an infinite horizon, discrete time, continuous state MDPs problem. To compute the approximation to the Bellman equation, we first write the value function approximation as a linear combination of n known basic functions $\phi_1, \phi_2, \ldots, \phi_n$ on X with corresponding unknown coefficient $c_1, c_2, \ldots, c_n$

$$V(x) \approx \sum_{j=1}^{n} c_j \phi_j(x) \tag{3-13}$$

Second, we compute the basis function coefficients $c_1, c_2, \ldots, c_n$ such that the approximating value function satisfies the Bellman equation at n collocation nodes

$x_1, x_2, ..., x_n$. Specifically, to compute the coefficients $c_1, c_2, ..., c_n$, we need to solve the set of equations

$$\sum_{j=1}^{n} c_j \phi_j(x_k) = \max_{u \in U}\{r(x_i,u) + \gamma E_w \sum_{j=1}^{n} c_j \phi_j(f(x_i,u,w))\}, i = 1,2,...,n \qquad (3\text{-}14)$$

We can express the above n equations in vector form as the collocation equation

$$\Phi c = v(c) \qquad (3\text{-}15)$$

where Φ is a $n \times n$ collocation matrix, whose ij th element is the j th basic function evaluated at the k th collocation node

$$\Phi_{ij} = \phi_j(x_i) \qquad (3\text{-}16)$$

And v is a $\mathbb{R}^n$ to $\mathbb{R}^n$ collocation function where

$$v_i(c) = \max_{u \in U}\{r(x_i,u) + \gamma E_w \sum_{j=1}^{n} c_j \phi_j(f(x_i,u,w))\} \qquad (3\text{-}17)$$

3.2.2.a Choice of Basis Function and Collocation Nodes

There are many basis-node schemes available to implement the collocation method. We introduce two typical choices for basic function here: Chebyshev Polynomials and Piecewise Polynomial Splines.

1. Chebyshev Polynomials

First we define $z = 2(x-a)/(b-a) - 1$to normalize the interval of approximation $[a,b]$ to the interval $[-1,1]$. Then the Chebyshev polynomials are defined recursively as follows.

$$\begin{aligned}
&T_0(z) = 1 \\
&T_1(z) = z \\
&T_2(z) = 2z^2 - 1 \\
&\vdots \\
&T_j(z) = 2zT_{j-1}(z) - T_{j-2}(z)
\end{aligned} \tag{3-18}$$

Both of numerical analysis theory and empirical experience suggest that polynomial approximation over a bounded interval $[a,b]$ should be constructed by interpolating the underlying function at Chebyshev nodes:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{n-i+0.5}{n}\pi), \forall i = 1,2,...,n \tag{3-19}$$

Chebyshev polynomials are an excellent basis function set for constructing polynomials that interpolate function values at the Chebyshev nodes. Chevychev basis polynomials with Chebyshev interpolation nodes yield an extremely well-conditioned interpolation equation that can be solved efficiently.

2. Piecewise Polynomial Splines

A spline is a special function defined piecewise by polynomials. It has a locally very simple form, yet at the same time be globally flexible and smooth. Splines are very useful for modeling arbitrary functions. An order-k spline consists of a series of k th-order polynomial segments spliced together so as to preserve continuity of derivatives of order $k-1$ or less. Suppose that the interval of approximation is $[a,b]$, the points at which the polynomial pieces are spliced together, $a = v_0 < v_1 < v_2 < ... < v_m = b$, are called the knots of the spline.

The most useful way to apply the spline to numerical work is the B-splines, or basic splines. B-splines for an order-k spline with v knots vector can be defined as using the Cox-de Boor recursion formula

$$B_{j,0}(x)=\begin{cases}1, & \text{if } v_j \le x < v_{j+1} \\ 0, & \text{otherwise}\end{cases} \tag{3-20}$$

$$B_{j,k}(x)=\frac{x-v_j}{v_{j+k}-v_j}B_{j,k-1}(x)+\frac{v_{j+k+1}-x}{v_{j+k+1}-v_{j+1}}B_{j+1,k-1}(x) \tag{3-21}$$

When the knots are equally spaced, we call the B-spline is uniform, otherwise we call it non-uniform.

The differentiation and integral of the spline can be computed as follows

$$\frac{dB_{j,k}(x)}{dx}=\frac{k}{v_{j+k}-v_j}B_{j,k-1}(x)+\frac{k}{v_{j+k+1}-v_{j+1}}B_{j+1,k+1}(x)$$

$$\int_a^x B_{j,k}(s)ds=\sum_{i=j}^{n}\frac{v_{i+k}-v_i}{k}B_{i+1,k+1}(x)$$

3.2.2.b Choice of Algorithm to Solve Collocation Equation

The collocation equation can be solved by any nonlinear equation solution method. We provide two algorithms here.

1. Fixed-point problem

One way is to view the collocation method as a fixed-point problem

$$c=\Phi^{-1}v(c)$$

The algorithm is described as follows.

```
Initialize the reward function $r$, discount factor $\gamma$, basic function $\Phi$, convergence tolerance $\theta$. Initialize the coefficient $c$ arbitrarily
    Repeat
            $c\_old \leftarrow c$
            $c \leftarrow \Phi^{-1}v(c)$
        $\Delta \leftarrow norm(c-c\_old)$
    Until $\Delta < \theta$
```

2. Root finding problem

Another way is to view the collocation equation as a root finding problem

$$\Phi c - v(c) = 0$$

We can apply the Newton's method to solve a root finding problem. The algorithm is described as follows.

```
Initialize the reward function $r$, discount factor $\gamma$, basic function $\Phi$, convergence tolerance $\theta$. Initialize the coefficient $c$ arbitrarily
    Repeat
        $c\_old \leftarrow c$
    Compute the $v'(c)$ is the $n \times n$ Jacobian of the collocation function $v$ at $c$
        For each $i = 1,...,n, j = 1,...,n$
            $v'_{ij}(c) = \frac{\partial v_i}{\partial c_j}(c) = \gamma E_w \phi_j(f(x_i, u, w))$
        end
    $c \leftarrow c - [\Phi - v'(c)]^{-1}[\Phi c - v(c)]$
    $\Delta \leftarrow norm(c - c\_old)$
    Until $\Delta < \theta$
```

3.3 Remark 1

In this section, we discuss the extension of the above results on MDPs to results on two-player zero-sum stochastic games with finite states spaces and actions, so called Markov games.

The model of a finite state space and action space two-player zero-sum game is minor modification to the model of finite MDPs. Define U, V the sets of possible actions for Player 1 and Player 2 respectively. Let $\{x_t\}$ be the corresponding Markov chain defined on a finite state space X. In every period t, the Player 1 (the Player 2) observes the state of a dynamic system x_t, takes an action u_t (v_t), and earns a reward (or cost if the reward is negative) $r(x_t, u_t, v_t)$ that depends on the state of the system and both the action taken by the Player 1 and the Player 2. The Player 1's (the Player 2's) objective is to seek a control strategy μ (ν) that given any state, the Player 1 (the Player 2) takes the action $u_t = \mu(x_t)$ ($v_t = \nu(x_t)$) which minimizes (maximizes) the objective function.

A game may be either deterministic or stochastic. In the stochastic case, given any state and action, the probability of each possible next state, x', is

$$p_{xx'}^{\alpha,\beta} = P\{x_{t+1} = x' \mid x_t = x, u_t = \alpha, v_t = \beta\} \tag{3-22}$$

Similarly, given any state and action, x and α, together with any next state x', the expected value of the next reward is

$$r(x,\alpha,\beta) = E\{r_{t+1} \mid x_t = x, u_t = \alpha, v_t = \beta\} \tag{3-23}$$

The general objective function of the two-player zero-sum game is in the form

$$J = E_x^{u,v} \sum_{t=1}^{\infty} \gamma^t r(x_t, u_t, v_t) \tag{3-24}$$

where γ is a per-period discount factor.

Define

$$\overline{V} = \min_u \max_v E_x^{u,v} \sum_{t=1}^{T} \gamma^t r_t(x,u,v) \tag{3-25}$$

and

$$\underline{V} = \max_{v} \min_{u} E_x^{u,v} \sum_{t=1}^{T} r_t(x,u,v) \tag{3-26}$$

If $\overline{V}(x) = \underline{V}(x)$, then $\overline{V}(x) = \underline{V}(x) = V(x)$ is called the value function of the game.

By using the principle of optimality, it implies that the value function $V(x)$ must satisfy

$$V(x) = \min_{\alpha \in U} \max_{\beta \in V} \{r(x,\alpha,\beta) + \gamma \sum_{x' \in X} p_{xx'}^{\alpha,\beta} V(x')\} \tag{3-27}$$

All the algorithms described above for the infinite horizon finite MDPs problems are suitable for solving Markov games. Let us take the policy iteration method for example. The algorithm is described as follows

```
Initialize the reward function $r$, transition probabilities $p$, discount factor $\gamma$,
convergence tolerance $\theta$. Initialize $V$ arbitrarily, e.g., $V(x) = 0$, for all $x \in X$
  Repeat
    For each $x \in X$
      $v\_old \leftarrow V(x)$
      $V(x) \leftarrow \min_{\alpha \in U} \max_{\beta \in V} (r(x,\alpha,\beta) + \gamma \sum_{x'} p_{xx'}^{\alpha,\beta} V(x'))$
    $\Delta \leftarrow norm(v\_old - V)$
  Until $\Delta < \theta$
  Output the policy $\mu, v$, such that
    $u(x), v(x) \leftarrow \arg\min_{\alpha \in U} \max_{\beta \in V} (r(x,\alpha,\beta) + \gamma \sum_{x'} p_{xx'}^{\alpha,\beta} V(x'))$
```

3.4 Remark 2

Our ultimate purpose in developing a software package is to provide a set of programs for solving two-player zero-sum stochastic games as well as stochastic optimization problems. When we use $\min\max$ to calculate the optimal value function of zero-sum games, in essence it is an extended optimal control problem. Therefore, it is reasonable that the software package also includes functions for solving stochastic control problems as well and that is also a reason that we first describe the algorithms for solving control problems for convenience and then extend them to games.

It is evident that when we use the Markov Chain Approximation method to solve the two-player zero-sum stochastic games or stochastic control problems, the resulting Markov chain is exactly in the form of finite MDPs. Since the method is straightforward and can be used under broad conditions, by using this software package, we can automatically calculate the value function and optimal strategies for stochastic control problems and zero-sum stochastic games, when a suitable Markov chain is chosen and corresponding transition probabilities and reward (cost) is formulated.

CHAPTER 4

STOCHASTIC CONTROL SOFTWARE PACKAGE

In this chapter, we introduce a preliminary stochastic control software package, which is written on the platform of MATLAB. The software package is a collection of functions whose prime purpose is to solve general stochastic games and stochastic optimal control problems. The software package includes routines for the following problems

- Discrete time, discrete state space Markov decision processes with discrete action space
- Discrete time, continuous state space Markov decision processes with discrete action space
- Continuous time, continuous state space stochastic control with discrete action space
- Continuous time, continuous state space two-player zero-sum stochastic games with discrete action space

4.1 Functions for Discrete Time, Discrete State Space MDPs

4.1.1 Function: *finite_Viter*

Description

finite_Viter solves general discrete time, discrete state space MDPs problems with finite horizon described by (3-6)

Syntax

```
[V_opt, act_opt]=finite_Viter(v, f, prob, T, gamma)
```

Input Arguments

v	Array containing the value function for each time and state. In a problem which has M discrete states and T time periods, it is an $M \times (T+1)$ array, where $V(:,T+1)$ is the value of terminal set, $v(:,1:T)$ is the initialization of the value function for each period from $t=1$ to $t=T$
f	Array containing the rewards for each state and action, In a problem which has M discrete states and N discrete actions, it is an $M \times N$ array, where $f(i,j)$ is the reward when the current state is x_i and action u_j is taken
$prob$	Array containing the transition probability for each state and action, In a problem which has M discrete states and N discrete actions, it is an $N \times M \times M$ array, where $prob(i,j,k)$ is the value of $p_{x_k x_j}^{u_i}$
T	Time periods
$gamma$	Discount factor

Output Arguments

V_opt	Array containing the value function for each time and state
act_opt	Array containing the optimal action for each state and period. In a problem which has M discrete states and N discrete actions, it is an $M \times T$ array, where $act\_opt(i,t)$ is the optimal action for state x_i at time t

Algorithm

finite_Viter utilizes the Backward Recursion algorithm described in section 3.1.2.1.

4.1.2 Function: *infinite_Viter*

Description

infinite_Viter solves general discrete time, discrete state space MDPs problems with infinite horizon described by (3-7)

Syntax

[V_opt, act_opt,V_iter]=infinite_Viter(v, f, prob, gamma, tol)

Input Arguments

v	Vector containing the initialization of the value function for each state. In a problem which has M discrete states, it is an $M \times 1$ vector
f	Array containing the rewards for each state and action, In a problem which has M discrete states and N discrete actions, it is an $M \times N$ array, where $f(i,j)$ is the reward when the current state is x_i and action u_j is taken
$prob$	Array containing the transition probability for each state and action, In a problem which has M discrete states and N discrete actions, it is an $N \times M \times M$ array, where $prob(i,j,k)$ is the value of $p_{x_k x_j}^{u_i}$
$gamma$	Discount factor
tol	Criterion for the iteration. Iteration stops when the error between v and the previous v is less than tol

Output Arguments

V_opt	Vector containing the optimal value function for each state after the iteration. In a problem which has M discrete states, it is an $M \times 1$ vector
act_opt	Vector which stores the optimal action for each state. In the problem which has M discrete states, it is an $M \times 1$ vector.
V_iter	Array containing the value function for each state during the iteration. In a problem which has M discrete states and n times of iteration, it is an $M \times n$ array, where $V\_iter(:,i)$ contains the value function for each state during i^{th} iteration

Algorithm

infinite_Viter utilizes the Value Iteration algorithm described in section 3.1.2.2.

4.2 Functions for Discrete Time, Continuous State Space MDPs

4.2.1 Function: *mdp_con_c*

Description

mdp_con_c solves the coefficients c in general discrete time, continuous state space MDPs problems described by (3-12)

Syntax

c_opt=mdp_con_c(c, basic_f, e, w, gamma, tol)

Input Arguments

c	Vector containing the initialization of coefficients for the basic functions. In a problem when user wants to use n basic functions, it is an $n \times 1$ vector
$basic\_f$	Structure containing the basic functions used to approximate the value function. It is defined by function fun_def, in which user can choose the type of basic functions, either 'Chebyshev Polynomials' or 'Polynomial Splines'
e	Vector containing the value of the noise
w	Vector containing the probabilities of the noise
$gamma$	Discount factor
tol	Criterion for the iteration

Output Arguments

c_opt	Vector containing the coefficients of the basic functions after iteration

Algorithm

mdp_con_c utilizes the Collocation Method algorithm described in section 3.2.2.

4.2.2 Function: *mdp_con*

Description

mdp_con solves the approximating value functions and optimal actions in general discrete time, continuous state space MDPs problems given the basic functions and corresponding coefficients.

Syntax

[V_opt, act_opt]=mdp_con(c, basic_f, node, e, w, gamma)

Input Arguments

c	the coefficients gotten by using function mdp_con_c
$basic\_f$	Structure containing the basic functions used to approximate the value function. It should be the same as the one used in function mdp_con_c
node	Vector containing the states where the user wants the value of the approximating value function
e	Vector containing the value of the noise
w	Vector containing the probabilities of the noise
gamma	Discount factor

Output Arguments

V_opt	Vector containing the approximation of value function for corresponding states.
act_opt	Vector containing the optimal actions for corresponding states.

Remark

User needs to write two functions named *Act* and *Reward*, which will be called by function *mdp_con*. The syntax of function *Act* is action=Act(x), in which the input argument x is the current state and the output argument *action* is the action set under the current state. The syntax of function is *reward* is reward=Reward(x, action), in which the input arguments x and *action* are the current state and action respectively and the output argument *reward* is the reward under the current state and action.

4.3 Functions for Stochastic Control Problems

4.3.1 Function: *stocha_min*

Description

stocha_min solves the value function of general stochastic optimal control problems described by (2-11) and (2-12) with dimension 1 or 2.

Syntax

V_opt=stocha_min(A, h, x, v, v_term, u, tol)

Input Arguments

A	Matrix defined as $A(x) = \sigma(x)\sigma(x)^T = \{a_{ij}(x)\}, \quad i,j = 1,\ldots,n$
h	Variable containing the width of the grid used in the construction of desired Markov chain
x	Array containing the states where the value function is approximated by the constructed Markov chain
v	Array containing the initialization of the value function of the Markov chain
v_term	Array containing the value function of the terminal set
u	Action of the player
tol	Criterion for the iteration

Output Arguments

V_opt	Array containing the optimal value function of the approximating Markov Chain

Algorithm

stocha_min utilizes the Markov Chain Approximation Method algorithm described in section 2.3.2.

4.3.2 Function: *opt_act_control*

Description

opt_act_control solves the optimal action for players of general stochastic optimal control problems with dimension 1 or 2, given the value function of the approximating Markov Chain.

Syntax

```
opt_act=opt_act_control(A, h, x, v, v_term, u)
```

Input Arguments

A	Matrix defined as $A(x)=\sigma(x)\sigma(x)^T=\{a_{ij}(x)\},\quad i,j=1,\dots,n$
h	Variable containing the width of the grid used in the construction of desired Markov chain
x	Array containing the states where the value function is approximated by the constructed Markov chain
v	Array containing the value function of the Markov chain
v_term	Array containing the value function of the terminal set
u	Action of the player

Output Arguments

opt_act	Array containing the optimal action for the player at each state of the constructed Markov chain

Remark

User needs to write two functions named *b*1 _ *value* and *b*2 _ *value*, which will be called by function *stocha_min* and *opt_act_control*. The syntax of function *b*1 _ *value* is y=b1_value(x, u1, u2), in which the input argument x is the current state, $u1$ is the action and $u2$ is a virtual variable which will not be used, and the output argument y is calculated by system's first-dimension dynamic equation. In the case of two dimensional problems, the syntax of function *b*2 _ *value* is the same as function *b*1 _ *value*. The input arguments of function *b*2 _ *value* are the same as function *b*1 _ *value*, while the output argument y is calculated by system's second-dimension dynamic equation. In the case of one dimensional problems, the output argument y is set to be 0.

4.4 Functions for Zero-Sum Stochastic Games

4.4.1 Function: *stocha_minimax*

Description

stocha_minimax solves the value function of general two-player, zero-sum stochastic games described by (2-1) and (2-2) with dimension 1 or 2.

Syntax

```
V_opt=stocha_minimax(A, h, x, v, v_term, u1, u2,tol)
```

Input Arguments

A	Matrix defined as $A(x) = \sigma(x)\sigma(x)^T = \{a_{ij}(x)\}, \quad i, j = 1, \ldots, n$
h	Variable containing the width of the grid used in the construction of desired Markov chain
x	Array containing the states where the value function is approximated by the constructed Markov chain
v	Array containing the initialization of the value function of the Markov chain
v_term	Array containing the value function of the terminal set
$u1$	Action of Player 1
$u2$	Action of Player 2
tol	Criterion for the iteration

Output Arguments

V_opt	Array containing optimal value function of the approximating Markov Chain

Algorithm

stocha_minimax utilizes the Markov Chain Approximation Method algorithm described in section 2.3.3.

4.4.2 Function: *opt_act_game*

Description

opt_act_game solves the optimal action for both of the players of general two-player, zero-sum stochastic games described by (2-1) and (2-2) with dimension 1 or 2, given the value function of the approximating Markov Chain.

Syntax

```
[opt_u1, opt_u2]=opt_act_game(A, h, x, v, v_term, u1, u2)
```

Input Arguments

A	Matrix defined as $A(x)=\sigma(x)\sigma(x)^T=\{a_{ij}(x)\},\quad i,j=1,\dots,n$
h	Variable containing the width of the grid used in the construction of desired Markov chain
x	Array containing the states where the value function is approximated by the constructed Markov chain
v	Array containing the value function of the Markov chain
v_term	Array containing the value function of the terminal set
$u1$	Action of Player 1
$u2$	Action of Player 2

Output Arguments

opt_u1	Array containing the optimal action for Player1 at each state of the constructed Markov chain
opt_u2	Array containing the optimal action for Player2 at each state of the constructed Markov chain

Algorithm

stocha_minimax utilizes the searching method to solve $\min\max$ problems. For each action of Player 1, it searches over each action of Player 2 and identifies the action in correspondence with the maximum value of the value function. Then it identifies the action of Player 1 which is in correspondence with the minimum value among the previous maximum values.

Remark

User needs to write two functions named $b1\_value$ and $b2\_value$, which will be called by function $stocha\_minimax$ and opt_act_game. The syntax of function $b1\_value$ is $\mathrm{y{=}b1\_value(x, u1, u2)}$, in which the input argument x is the current state, $u1$ is the action of Player 1 and $u2$ is the action of Player 2, and the output argument y is calculated by system's first-dimension dynamic equation. In the case of two dimensional problems, the syntax of function $b2\_value$ is the same as function $b1\_value$. The input arguments of function $b2\_value$ are the same as function $b1\_value$, while the output argument y is calculated by system's second-dimension dynamic equation. In the case of one dimensional problems, the output argument y is set to be 0.

CHAPTER 5

APPLICATION TO PURSUIT-EVASION GAMES

In this chapter, we first provide three numerical examples to demonstrate the use of our stochastic control software package. Then we emphasize the application of the Markov Chain Approximation method to a typical zero-sum stochastic game: pursuit-evasion games. We illustrate the validity and practicability of this method by numerically solving a PE game with one pursuer and one evader. An example of PE games with multiple pursuers and evaders is also given in which the decentralized strategy is used to decouple the multiple players into several one-to-one PE games.

5.1 Numerical Examples of Markov Decision Processes and Demonstrations of Stochastic Control Software Package

5.1.1 A Numerical Example of Discrete Time, Discrete State Space MDPs with Finite Horizon

5.1.1.a Problem Description: Mine Extraction [29]

A mine operator wants to decide how much ore to extract from a mine that will be shut down and abandoned after T years of operation. The price of extracted ore is p dollars per ton, and the total cost of extracting a tons of ore in any year, given that the mine contains x tons at the beginning of the year, is $c(x,a)$ dollars. The mine currently contains M tons of ore. The value of the ore is depreciated by discount factor δ. Assuming the amount of ore extracted in any year must be an integer number of tons, we are asked to seek the optimal extraction strategy to maximize profits.

5.1.1.b Model

This problem can be formulated as a discounted, finite MDPs with finite horizon.

State space: $x \in \{0,1,2,...,M\}$

Action space: $a \in \{0,1,...,x\}$.

Discount factor: γ

Reward function: $f(x,a) = pa - c(x,a) = pa - \dfrac{a^2}{1+x}$

Transition probability: $p(x'|x,a) = \begin{cases} 1, & x' = x - a \\ 0, & \text{otherwise} \end{cases}$

5.1.1.c Numerical Simulation

In this mine extraction example, let the current content M to be 200 tons, the operation period T to be 20 years, the price p to be 1 dollar per ton, and the discount factor γ to be 0.9. We use the function *finite_viter* to solve this finite MDPs problem with finite horizon. The value of the input arguments are listed in Table 5.1

v	A $200\times(20+1)$ array. $V(:,21)$ is the value function of the terminal set, which is zero because the operator does not want to see any ore left when the mine is shut down; $v(:,1:20)$ is the initialization of the value function for each period from $t=1$ to $t=20$, we set it to be 0
f	A 200×200 array, where $f(i,j)=j-\frac{j^2}{1+i}$ when $i\geq j$ and $f(i,j)=-\inf$ when $i<j$ because the value function doesn't exist for other states
prob	A $200\times 200\times 200$ array, where $prob(i,j,k)=1$ when $j=k-i$, otherwise $prob(i,j,k)=0$
T	20
gamma	0.9

Table 5.1 Value of Input Arguments in Example: Mine Extraction

MATLAB codes, which define the input arguments and call the function *finite_viter*, are as follows.

```
M=200;

T=20;

value=zeros(M,T+1);

gamma=0.9;
```

```
f=zeros(M,M);
for i=1:M
    for j=1:M
        if i>=j
            f(i,j)=j-j^2/(1+i);
        else
            f(i,j)=-inf;
        end
    end
end


for i=1:M
    for j=1:M
        for k=1:M
            if j==k-i
                prob(i,j,k)=1;
            else
                prob(i,j,k)=0;
            end
        end
    end
end
```

```
[V_opt,act_opt]=finite_Viter(value,f,prob,T,gamma);
```

Figure 5.1 shows the optimal strategy at each state. Figure 5.2 shows the stock of the mine through the operation period by using optimal strategy.

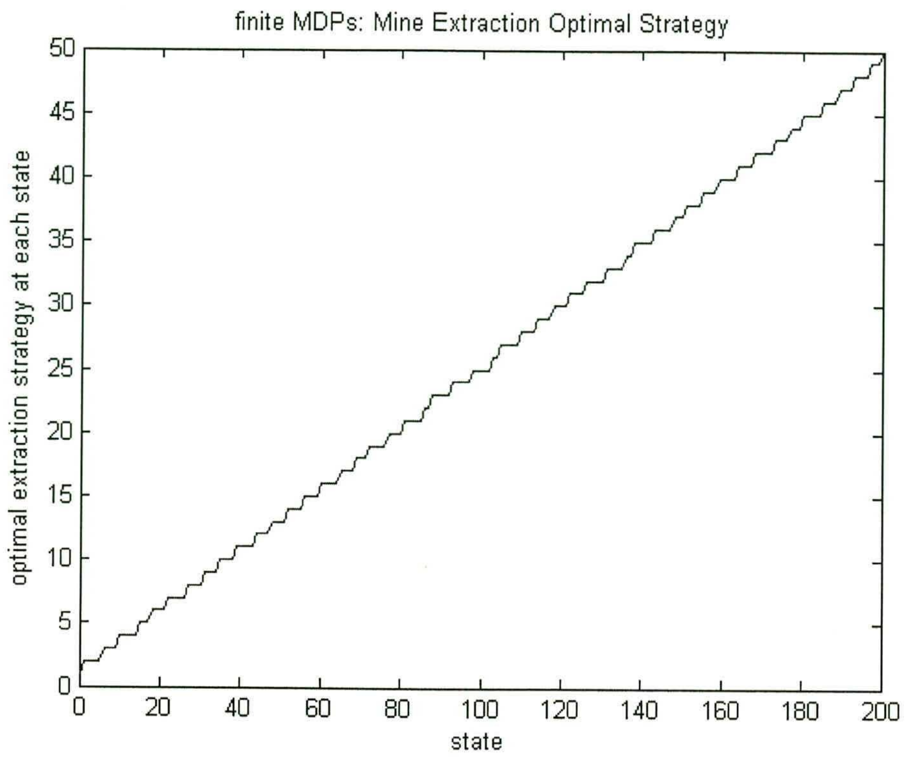

Figure 5.1 Optimal Strategy, Mine Extraction

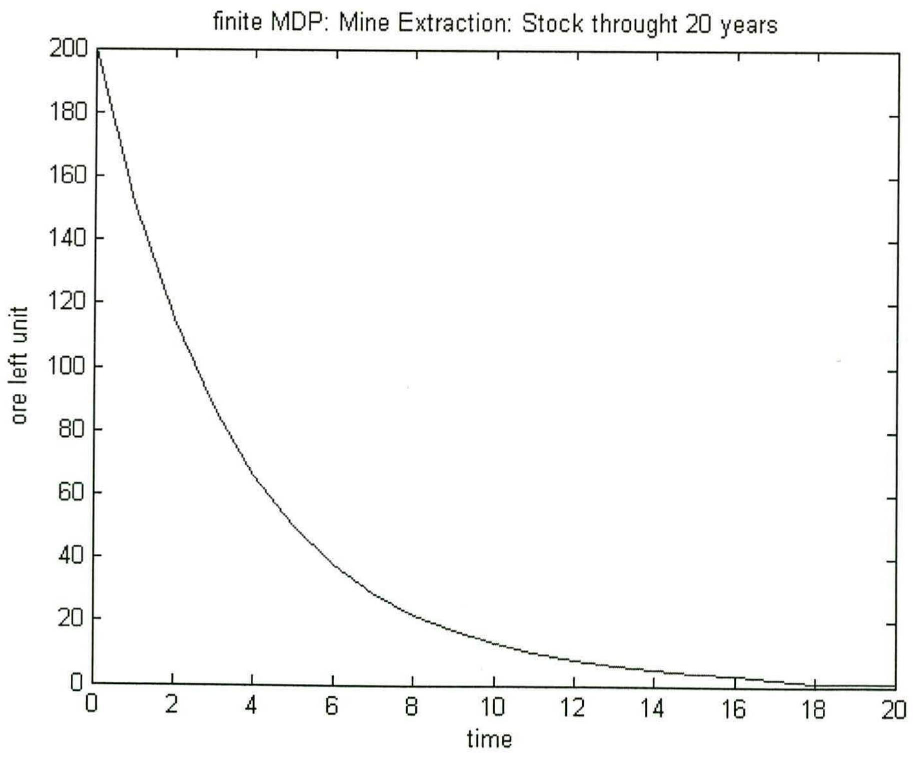

Figure 5.2 Stock Though 15 year, Mine Extraction

5.1.2 Numerical Example for Discrete Time, Discrete State Space MDPs with Infinite Horizon

5.1.2.a Example Description: Gambler's Problem [23]

5.1.2.b Model

At the beginning of each year, a manufacturer must decide whether to continue to operate an aging physical asset or replace it with a new one. An asset that is x years old yields a profit contribution $f(x)$ up to N years, at which point the asset becomes unsafe and must be replaced by law. The cost of a new asset is c. We are asked to seek the replacement policy which maximizes the profit.

This problem can be formulated as an discounted finite MDPs with infinite horizon. State space: $x \in \{1, 2, ..., N\}$

Action space: $a \in \{\text{replace, keep}\}$.

Discount factor: $\gamma = 0.9$

Reward function: $f(x,a) = \begin{cases} -25, & \text{if a=replacement} \\ 50 - 2.5x - 2.5x^2, & \text{if a=keep} \end{cases}$

Transition probability: $p(x' \mid x, a) = \begin{cases} 1, & \text{if a=replacement, x'=1,} \\ 1, & \text{if a=keep, x'=x+1;} \\ 0, & \text{otherwise} \end{cases}$

5.1.2.c Numerical Simulation

We use the function *infinite_viter* to solve this finite MDPs problem with infinite horizon.

In this example, let the maximum operation age of the machine N to be 5 dollars. We use the function *infinite_viter* to solve this finite MDPs problem with infinite horizon. The value of the input arguments is listed in Table 5.2.

v	A 5×1 vector, whose initial value is set to be 0
f	A 2×5 array, where $f(i,j)=-25$ when $i=1$, Otherwise, $f(i,j)=50-2.5j-2.5j^2$
prob	A $2\times5\times5$ array, where $prob(i,j,k)=1$ when $i=1, j=1$, $prob(i,j,k)=1$ when $i=2, j=k+1$,and $prob(i,j,k)=0$ for other situations
gamma	0.9
tol	10^{-8}

Table 5.2 Value of Input Arguments in Example: Asset Replacement

MATLAB codes, which define the input arguments and call the function *finite_viter*, are as follows.

```
n=5;

gamma=0.9;

value=zeros(n,1);

tol=10^(-8);


f=zeros(2,n)
```

```
f(1,:)=50-75;

f(2,:)=(50-2.5*(1:n)-2.5*(1:n).^2)';


prob=zeros(2,n,n);

for i=1:n

    for j=1:n

        if i==j+1

            prob(2,i,j)=1;

        end

    end

end

prob(1,1,:)=1;
```

```
[V_opt,act_opt,V_iter]=infinite_Viter(value,f,prob,gamma,tol);
```

Figure 5.3 shows the value function at each state. Figure 5.3 also shows the iteration procedure of the value function, which is initially set to be 0 for each state and gradually converges to the optimal value function. Table 5.3 shows the optimal strategy at each state. For example, we should replace the machine in its 4th year of operation.

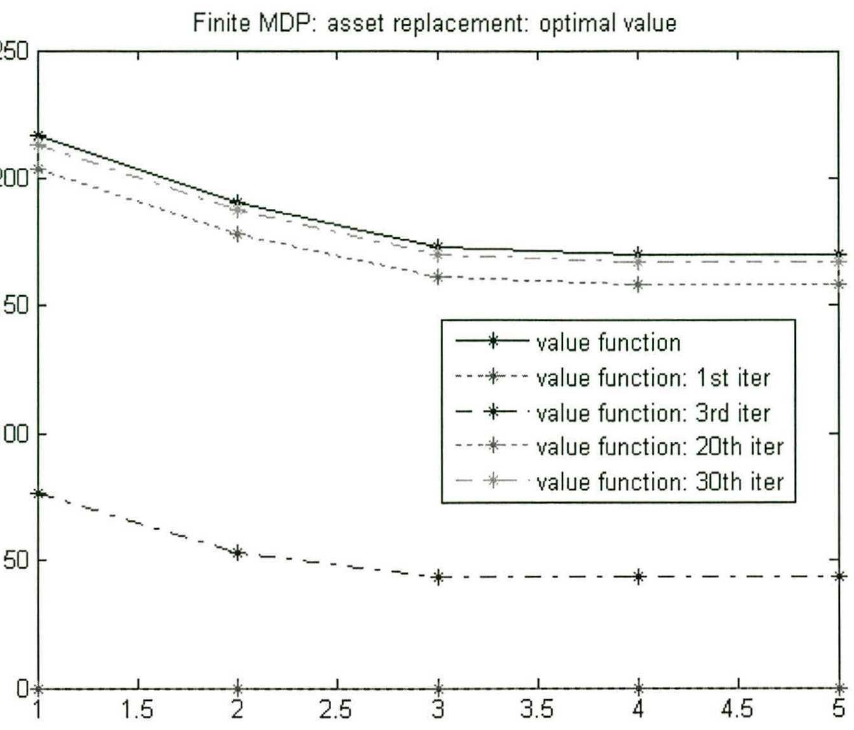

Figure 5.3 Optimal Value during iteration, Asset Replacement

Age of the Machine	Optimal Strategy
1	Keep
2	Keep
3	Keep
4	Replace
5	Replace

Tabel 5.3 Optimal Strategy, Asset Replacement

5.1.3 Numerical Example of Discrete Time, Continuous State Space MDPs

5.1.3.a Example Description: Water Management [30]

Water from a reservoir can be used for either irrigation or recreation. Irrigation during the spring benefits farmers, but reduces the reservoir level during the summer, damaging the interests of recreational users. Specifically, if the reservoir contains x units of water at the beginning of the year and a units are released for irrigation, farmer and recreational user benefits during the year will be $F(a)$ and $U(x-a)$ respectively. Reservoir levels are replenished by random rainfall during the winter. Specifically, it rains k units with probability $p_k, k = 1, 2, ..., K$. The reservoir can hold only M units of water, and excess rainfall flows out without benefit to either farmer or recreational user. The reward is the sum of farmer's and recreational user's benefit. We are asked to seek the irrigating policy to maximize the reward.

5.1.3.b Model

This problem can be formulated as continuous state space MDPs with infinite horizon.

State space: $x \in [2, M]$

Action space: $a \in \{0, 0.05, ..., x\}$

Rainfall: k units with probability $p_k, k = 1, 2, ..., K$

State transition function: $f(x, a, e) = \min(x - a + e, M)$

Farmer's benefit: $F(a) = \alpha_1 a^{\beta_1}$

Recreational user's benefit: $U(x, a) = \alpha_2 (x - a)^{\beta_2}$

Reward function: $f(x, a) = F(a) + U(x, a)$

Bellman equation: $V(x) = \max_{0 \le a \le x}\{R(x,a) + V(\min(x-a+k, M))\}$

5.1.3.c Numerical Simulation

In this example, let the reservoir capacity M to be 7, and the rainfalls are $0.7, 1, 1.3$ with probability $0.2, 0.6, 0.2$ respectively. Let $\alpha_1 = -1, \beta_1 = -1, \alpha_2 = -1, \beta_2 = -2.$ We use the function *mdp_con* to solve this continuous MDPs problem. The value of input arguments of function *mdp_con* is listed in Table 5.3.

The user defined function *action* is written as follows.

```
function [action]=Act(x)

    action=[0:0.05:x];
```

The user defined function *reward* is written as follows.

```
function reward=Reward(x,a)

    a1=-1;

    a2=-1;

    a2=-1;

    b2=-2;

    reward=a1*x^b1+a2*(x-a)^b2;
```

c	We use 10 basic functions to approximate the original value function. Therefore, c is a 10×1 vector containing the coefficients. The initial value is set to be 0.
$basic\_f$	Structure containing the basic functions used to approximate the value function. It is defined by $fun\_def('\text{cheb}',10,2,7)$, which means the basic functions are 10 chebyshev polynomials, and the range of states is [2, 7]
e	$[0.7,1,1.3]$
w	$[0.2,0.6,0.2]$
$gamma$	0.9
tol	10^{-6}

Table 5.4 Value of Input Arguments in Example: Water Managmet

MATLAB codes, which define the input arguments and call the function mdp_con_c and mdp_con, are as follows.

```
n=10;

s1=2;

s2=9;

gamma=0.9;

tol=10^(-6);

fspace=fun_def('cheb',n,s1,s2);
```

```
snodes=fun_node(fspace);

c=zeros(n,1);

e=[0.7;1;1.3];

w=[0.2;0.6;0.2];


c_opt=mdp_con_c(c, basic_f, e, w, gamma, tol)

xplot=(s1:(s1-s2)/20:s2)';

[vplot,actplot]=mdp_con_c(c,basic_f,xplot,e,w,gamma);
```

Figure 5.4 shows the approximation of original value function at each state. Figure 5.5 shows the optimal strategy at each state. For example, when the reservoir contains 7 unit of water, the optimal strategy is to extract 2.1 units of water for irrigation.

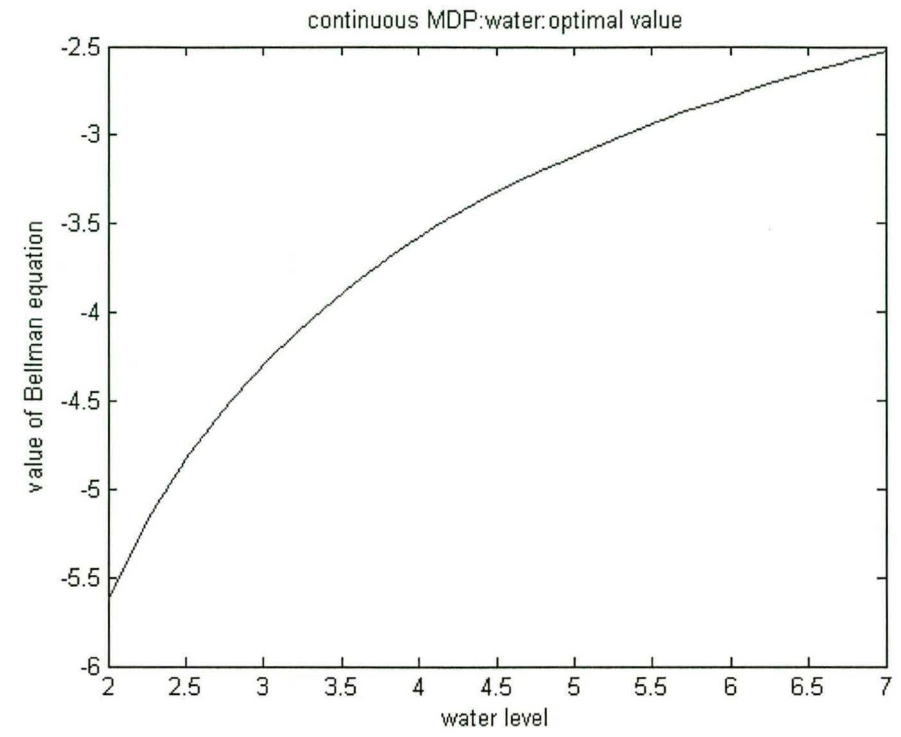

Figure 5.4 Value Function, Water Management

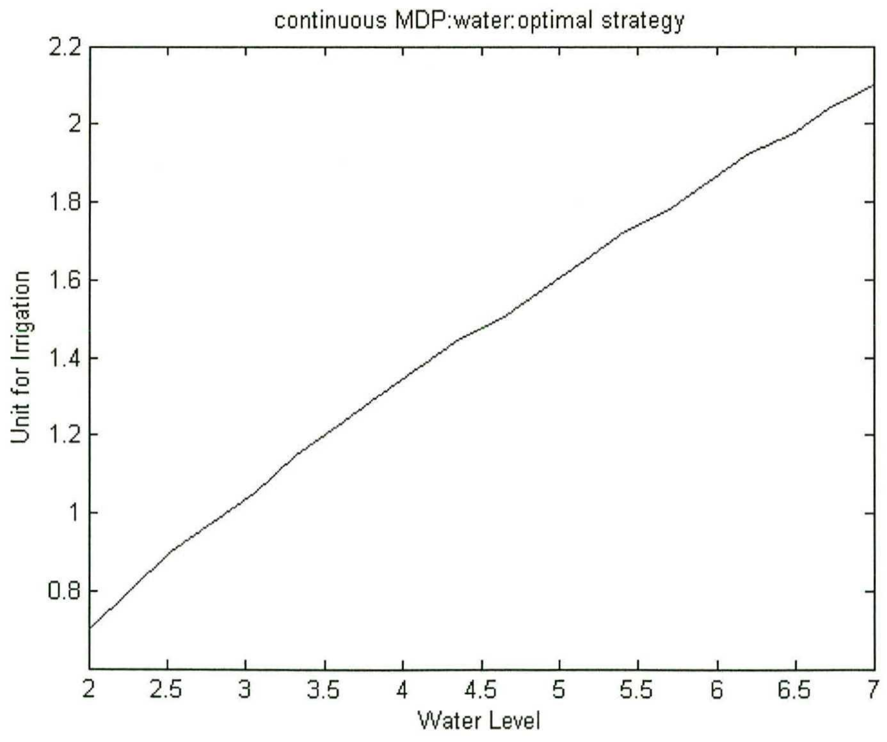

Figure 5.4 Optimal Strategy, Water Management

5.2 Deterministic Pursuit-Evasion Games

We consider a one to one deterministic game with two dimensions. The dynamics of the Pursuer and Evader are as follows:

$$\begin{aligned} dx^p &= \cos\theta dt \\ dy^p &= \sin\theta dt \\ dx^e &= b(t)+1+\sqrt{2} \\ dy^e &= -2 \end{aligned}$$

in which the moving direction $\theta(t)$ is the control of the pursuer and the velocity $b(t)$ is the control of the evader with $-1 \le b(t) \le 1$ for $t \ge 0$. The terminal set S of the game is defined as $\{(x^p, y^p),(x^e, y^e) \mid y^p \ge y^e\}$. The objective function is defined as $J = \int_0^T (b(t)+1+\sqrt{2}-\cos(\theta(t)))dt$. The analytical solution to this problem is given in [36] and the value function and the optimal strategies are given as follows

$$\begin{aligned} &V(x^p, y^p, x^e, y^e) = y^e - y^p \\ &\theta^*(t) = \frac{1}{4}\pi, \\ &b^*(t) = 1 \end{aligned}$$

We apply the Markov Chain Approximation method to this problem. By defining $x_1 = x^p - x^e, x_2 = y^p - y^e$, we get the following new dynamics of the system:

$$\begin{aligned} dx_1 &= (\cos(\theta(t)) - b(t) - 1 - \sqrt{2})dt \\ dx_2 &= (\sin(\theta(t)) + 2)dt \end{aligned}$$

Since it is a deterministic case, the matrix $A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. By substituting the concrete form of $f(x, \alpha, \beta)$, we have

$$V^h(x,u)=\min_{u\in U}\max_{v\in V}\left[\sum_y p^h(y\,|\,x,u(x),v(x))V^h(y,u)+(\cos(u(x))-b(x)-1-\sqrt{2})\Delta t^h(x,u(x),v(x))\right]$$

where

$$p^h((x_1\pm h,x_2)\,|\,(x_1,x_2),\alpha,\beta)=\frac{h(\cos\theta-b-1-\sqrt{2})^{\pm}}{Q^h((x_1,x_2,\theta,b)}$$
$$p^h((x_1,x_2\pm h)\,|\,(x_1,x_2),\alpha,\beta)=\frac{h(\sin\theta+2)^{\pm}}{Q^h((x_1,x_2),\theta,b)}$$
$$\Delta t^h((x_1,x_2),\alpha,\beta)=\frac{h^2}{Q^h((x_1,x_2),\theta,b)}$$
$$Q^h((x_1,x_2),\alpha,\beta)=\sum_i h\,|\cos\theta-b-1-\sqrt{2}\,|+h\,|\sin\theta+2\,|$$

For the numerical simulation, let

$h=1$,

$\theta\in\{0,\frac{1}{16}\pi,\frac{1}{16}\pi,\ldots,\frac{15}{16}\pi\}$,

$b\in\{-1,-0.9,\ldots,1\}$

We use the function *stocha_minimax* to solve this two-player zero-sum MDPs problem. The value of input arguments is listed in Table 5.5.

The user defined function *b1_value* is as follows.

```
function y=b1_value(x,u1,u2)

        y=cos(u1)-(u2+1+sqrt(2));
```

The user defined function *b2_value* is written as follows.

```
function y=b2_value(x,u1,u2)

        y=sin(u1)+2;
```

A	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
h	1
x	A 21×21 array, in which each element stands for a state
v	A 21×21 array containing the initialization of the value function. We set it to be 0.
v_term	We set it to be a reasonably large number 200
$u1$	$\{0,\frac{1}{16}\pi,\frac{1}{16}\pi,...,\frac{15}{16}\pi\}$, actions of Player 1
$u2$	$\{-1,-0.9,...,1\}$, actions of Player 2
tol	10^{-6}

Table 5.5 Value of Input Arguments in Example: One-to-one deterministic PE Game

Figure 5.6 shows the trajectories of the pursuer and the evader according to the numerical solution and analytical solution respectively. Figure 5.7 shows the trajectories of the relative position of the pursuer and the evader according to the numerical solution and analytical solution respectively. The starting point of the pursuer is [0, 0] and it is [2, 1] of the evader. The numerical solution is identical to the analytical solution. It proves the validity of the Markov Chain Approximation method and our software package.

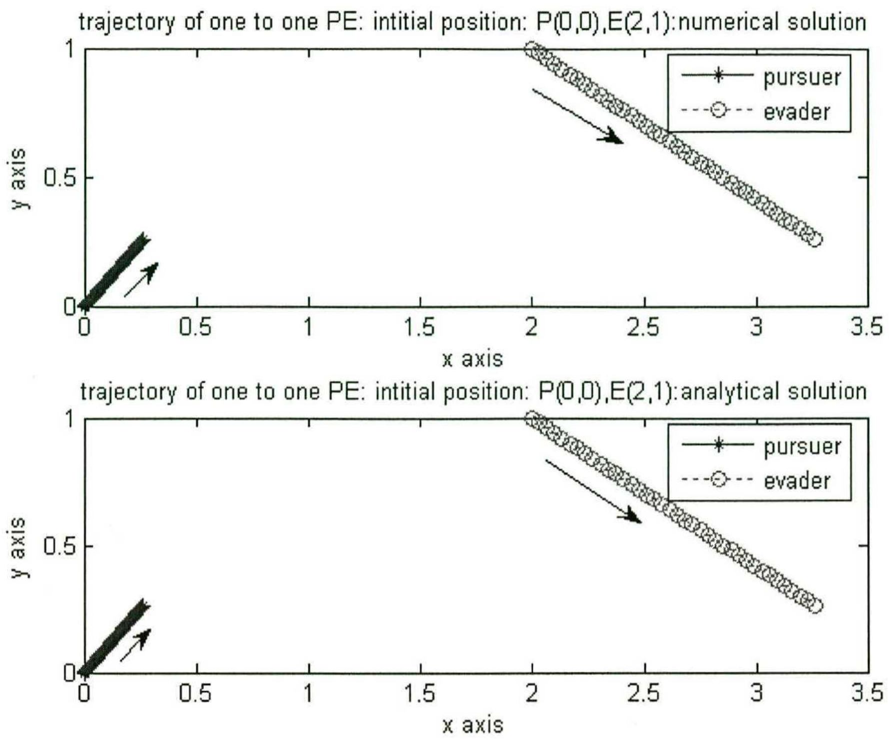

Figure 5.6 Trajectories of the Players: numerical and analytical solution

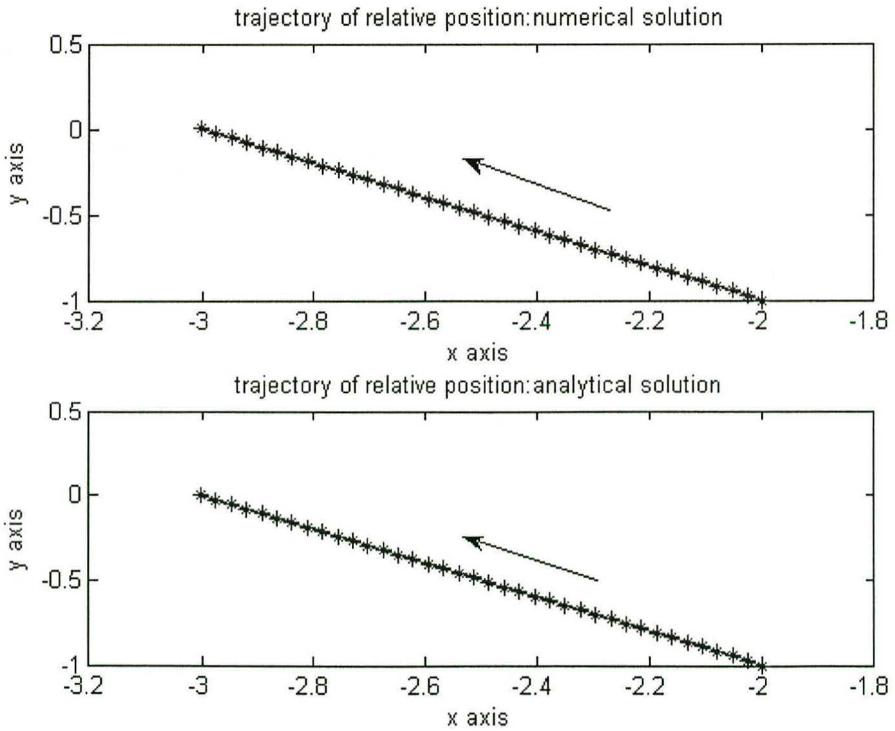

Figure 5.7 Trajectories of the relative position: numerical and analytical solution

5.3 Stochastic Pursuit-Evasion Games

5.3.1 One Pursuer One Evader PE Games

We consider a one to one PE game with two dimensions [14]. The dynamics of the Pursuer and Evader are as follows:

$$\begin{aligned}
dx^p &= v_p \cos(\theta_p(t))dt + \sigma(x^p)dw \\
dy^p &= v_p \sin(\theta_p(t))dt + \sigma(y^p)dw \\
dx^e &= v_e \cos(\theta_e(t))dt + \sigma(x^e)dw \\
dy^e &= v_e \sin \theta_e(t)dt + \sigma(y^e)dw
\end{aligned}$$

in which the moving direction $\theta_i(t)$ is the only control.

The new state variables are represented as

$$\begin{aligned}
x_1 &= x^p - x^e \\
x_2 &= y^p - y^e
\end{aligned}$$

Thus, we have

$$\begin{aligned}
dx_1 &= (v_p \cos(\theta_p(t)) - v_e \cos(\theta_e(t)))dt + \sigma(x_1)dw \\
dx_2 &= (v_p \sin(\theta_p(t)) - v_e \sin(\theta_e(t)))dt + \sigma(x_2)dw
\end{aligned}$$

with $\quad x_{10} = x_0^p - x_0^e, x_{10} = y_0^p - y_0^e$.

The terminal set is defined as $S = \{(x_1, x_2) \mid \|(x_1, x_2)\|_2 \le \varepsilon\}$. The objective of the pursuer is to minimize the capture time. The objective function is $J = \int dt$. Hence, G=1 and Q=0 in the general model.

The analytical solution of is given as follows [36].

$$V(x_1, x_2) = \frac{\sqrt{x_1^2 + x_2^2}}{v_p - v_e} + \frac{\sigma_1^2 + \sigma_2^2}{4(v_p - v_e)^2} \ln\left(x_1^2 + x_2^2\right) + C(\varepsilon)$$

where $\quad C(\varepsilon) = -\dfrac{\varepsilon}{v_p - v_e} - \dfrac{(\sigma_1^2 + \sigma_2^2)\ln(\varepsilon^2)}{4(v_p - v_e)^2}$

We assume $\sigma(x)=\begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix}$ is constant, hence the matrix $A(x)=\sigma(x)\sigma(x)^T=\begin{bmatrix} c^2 & 0 \\ 0 & c^2 \end{bmatrix}$.

We use the uniform h-grid on $\mathbb{R}^2$ and the Markov Chain Approximation method described in Chapter 2 to construct the approximating Markov chain and to numerically approximate the value function. The matrix A is in the general form and satisfies the assumption that $a_{ii}(x)-\sum_{j:j\neq i}|a_{ij}(x)|\geq 0$, for all i. By substituting the concrete form of $f(x,\alpha,\beta)$ into (2-47) and (2-48), we have

$$V(x)=\min_{u\in U}\max_{v\in V}[\sum_y p^h(y\,|\,x,u(x),v(x))V(y)+\Delta t^h(x,u(x),v(x))]$$

where

$$\begin{aligned}
&p^h((x_1+h,x_2)\,|\,(x_1,x_2),\alpha,\beta)=h(v_p\cos\theta_p-v_e\cos\theta_e)^{\pm}/Q^h((x_1,x_2),\theta_p,\theta_e)\\
&p^h((x_1,x_2+h)\,|\,(x_1,x_2),\alpha,\beta)=h(v_p\sin\theta_p-v_e\sin\theta_e)^{\pm}/Q^h((x_1,x_2),\theta_p,\theta_e)\\
&p^h((x_1+h,x_2+h)\,|\,(x_1,x_2),\theta_p,\theta_e)=p^h((x_1-h,x_2-h)\,|\,(x_1,x_2),\theta_p,\theta_e)=c^2/2Q^h((x_1,x_2),\theta_p,\theta_e)\\
&p^h((x_1+h,x_2-h)\,|\,(x_1,x_2),\theta_p,\theta_e)=p^h((x_1-h,x_2+h)\,|\,(x_1,x_2),\theta_p,\theta_e)=0\\
&\Delta t^h((x_1,x_2),\theta_p,\theta_e)=\frac{h^2}{Q^h((x_1,x_2),\theta_p,\theta_e)}\\
&Q^h((x_1,x_2),\theta_p,\theta_e)=2c^2+h|v_p\cos\theta_p-v_e\cos\theta_e|+h|v_p\sin\theta_p-v_e\sin\theta_e|
\end{aligned}$$

For the numerical simulation, let

$v_p=12, v_e=10, h=1, c^2=0.1$,

$\theta_p\in\{0,\frac{1}{16}\pi,\frac{1}{16}\pi,\ldots,\frac{15}{16}\pi\}$,

$\theta_e\in\{0,\frac{1}{16}\pi,\frac{1}{16}\pi,\ldots,\frac{15}{16}\pi\}$

We use the function *stocha_minimax* to solve this two-player zero-sum MDPs problem. The value of input arguments is listed in Table 5.6.

The user defined function $b1\_value$ is as follows.

```
function y=b1_value(x,u1,u2)

          vp=12;

          ve=10;

          y=vp*cos(u1)-ve*cos(u2);
```

The user defined function $b1\_value$ is written as follows.

```
function y=b2_value(x,u1,u2)

          vp=12;

          ve=10;

          y=vp*sin(u1)-ve*sin(u2);
```

A	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$
h	1
x	A 21×21 array, in which each element stands for a state
v	A 21×21 array containing the initialization of the value function. We set it to be 0.
v_term	We set it to be a reasonably large number 200
$u1$	$\{0,\frac{1}{16}\pi,\frac{1}{16}\pi,...,\frac{15}{16}\pi\}$, actions of Player 1
$u2$	$\{0,\frac{1}{16}\pi,\frac{1}{16}\pi,...,\frac{15}{16}\pi\}$, actions of Player 2
tol	10^{-6}

Table 5.6 Value of Input Arguments in Example: One-to-one PE Game

Figure 5.8 shows the numerical solution of the value function at each state. Figure 5.9 shows the analytical solution of the value function at each state. Two figures are in identical shape. The difference between the numerical solution and analytical solution is because we approximate the value function on a discrete state space $\mathbb{R}^h$ instead of on the original continuous state space and the terminal value is not accurate. Figure 5.10 shows the trajectories of the Pursuer and the Evader in an example of one-to-one PE games when the Pursuer's starting position is $[0,0]$ and the Evaders' is $[-9,-4]$. In our case, the objective function is the capture time. However, it should be noticed that the resulting value function does not exactly give us the optimal value at each state because of the noise. Instead, it only gives relative value which is used to calculate the optimal strategy at each state.

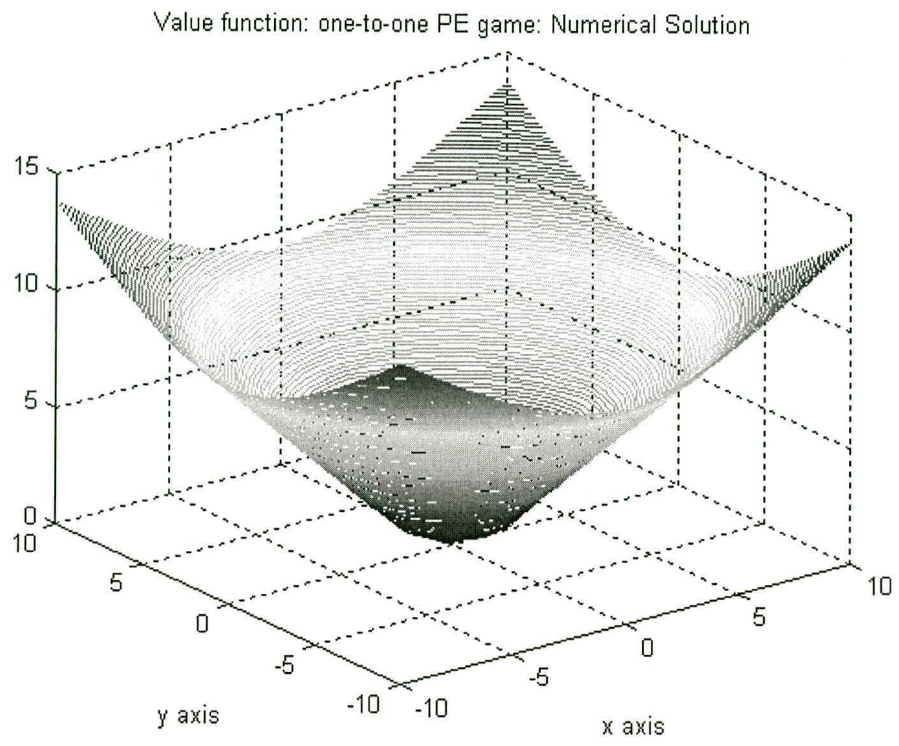

Figure 5.8 Value Function, One to One PE game: Numerical Solution

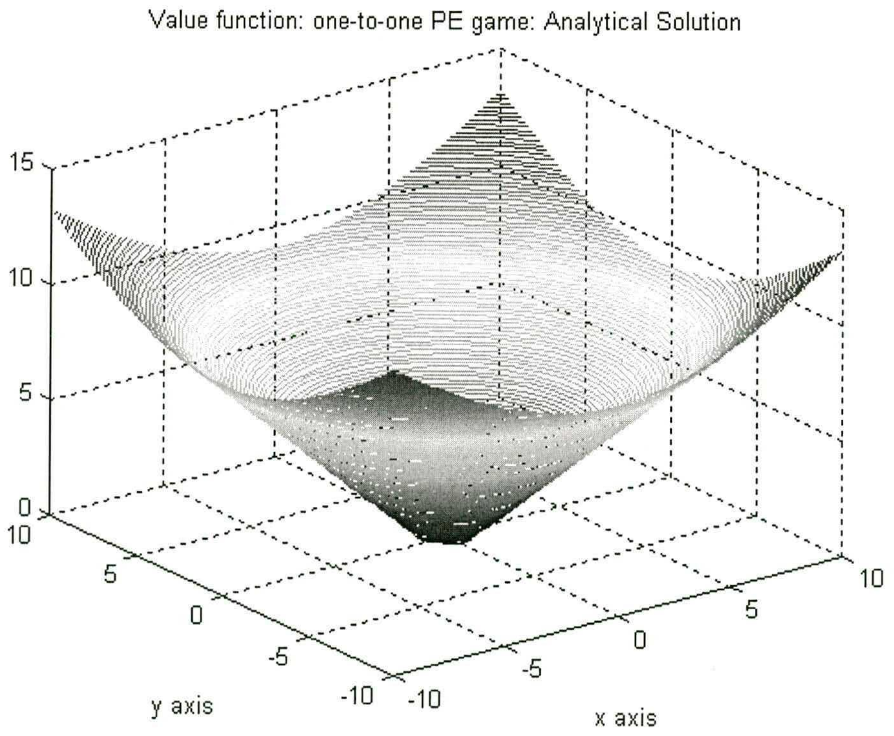

Figure 5.9 Value Function, One to One PE game: Analytical Solution

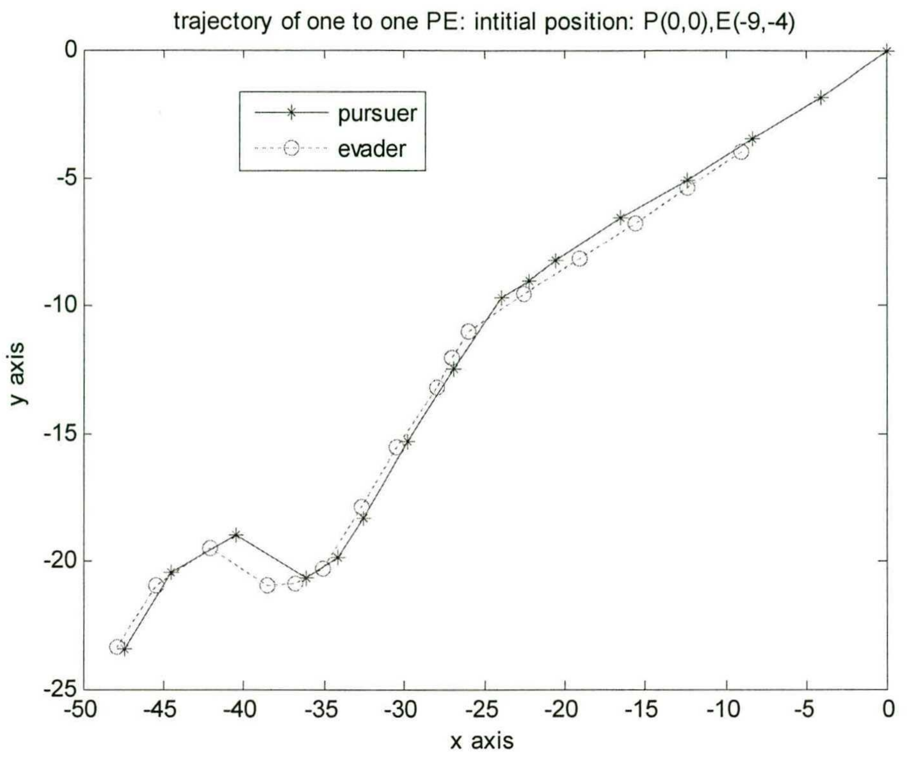

Figure 5.10 Trajectories of Pursuer and Evader, One to One PE

5.3.2 Multiple Players Pursuit-Evasion Games

Let us consider a PE game with N pursuers and M evaders. The dynamics of each pursuer is described as

$$\begin{aligned} dx_i^p &= v_i^p \cos\theta_i^p dt + \sigma(x_i^p)dw \\ dy_i^p &= v_i^p \sin\theta_i^p dt + \sigma(y_i^p)dw, \quad i = 1,2,\ldots,N \end{aligned}$$

Similarly, the dynamics of each evader is described as

$$\begin{aligned} dx_j^e &= v_j^e \cos\theta_j^e dt + \sigma(x_j^e)dw \\ dy_j^e &= v_j^e \sin\theta_j^e dt + \sigma(y_j^e)dw, \quad j = 1,2,\ldots,M \end{aligned}$$

Let $T_j (j = 1,\ldots,M)$ denote the capture time for Evader j. We use the minimum capture time $T = \max_j\{T_j\}$ as the objective function. It is possible to take all the pursuers and evaders as a whole and solve the problem as a high dimensional stochastic game by mechanically applying the Markov Chain Approximation method. However, for a PE game with N pursuers and M evaders played on a plane, the dimension of the state will be at least $2\times(N+M-1)$. If we take the increase of possible action space into consideration, the time that the iteration of the value function takes to converge rises more than exponentially with the increase of number of players [28]. To avoid this dimension curse, we use a decentralized approach. At each time period, we decompose the game into several one-to-one PE games. Each pursuer and evader will determine his optimal strategy according to the given engagement scheme. Let us assume $N \geq M$. The algorithm is carried out as follows.

We assume that each pursuer can only chase one evader at each time period. At each time period t, assume that we know the value function of the games between any pursuer and evader $V(p_i, e_j)$. The search for optimal engagement can be formulated as follows.

$$\min J = \min_{\{b_{ij}\}}\{\max_{j} \sum_{i=1}^{N} V(p_i, e_j) b_{ij}\}$$

$$\text{Subject to} \quad \begin{aligned} & b_{ij} \in \{0,1\} \\ & \sum_{i=1}^{N} b_{ij} = 1, \quad j = 1, \ldots, M \\ & \sum_{i=1}^{M} b_{ij} \leq 1, \quad i = 1, \ldots, N \end{aligned}$$

Here, $b_{ij} = 1$ means Pursuer i chases Evader j and $b_{ij} = 0$ means not. The case is much more complicated when $N < M$ because the search space is larger. However, the approach is similar in principle.

In the specific case of deterministic PE games, such as the homicidal chauffeur game described in [19], $V(p_i,e_j)$ can be expressed by an analytical equation and can be calculated precisely. However, in stochastic PE games, as well as in a majority of deterministic games, we are unable to do the same calculation because the exact value of $V(p_i,e_j)$ is unavailable. To solve this problem we first solve the one-to-one stochastic PE game for all the possible combinations of any one pursuer from the multiple pursuers and any one evader from the multiple evaders. For example, we solve the one-to-one stochastic game with Pursuer i and Evader j to obtain the value function. Then we use the resulting value function as an approximation of $V(p_i,e_j)$ and substitute it for the true $V(p_i,e_j)$ in decentralization approach described above.

We take a stochastic PE game with two pursuers and two evaders as an example. The value of parameters used for the simulation is listed in the Table 5.8.

	Pursuer 1	Pursuer 2	Evader 1	Evader 2
(x_0, y_0)	(0,0)	(7,3)	(3,-4)	(5,-2)
$v_p(v_e)$	12	11	10	8
c^2	0.1	0.1	0.1	0.1

Table 5.7 Value of Parameters in Example: Multiply PE Game

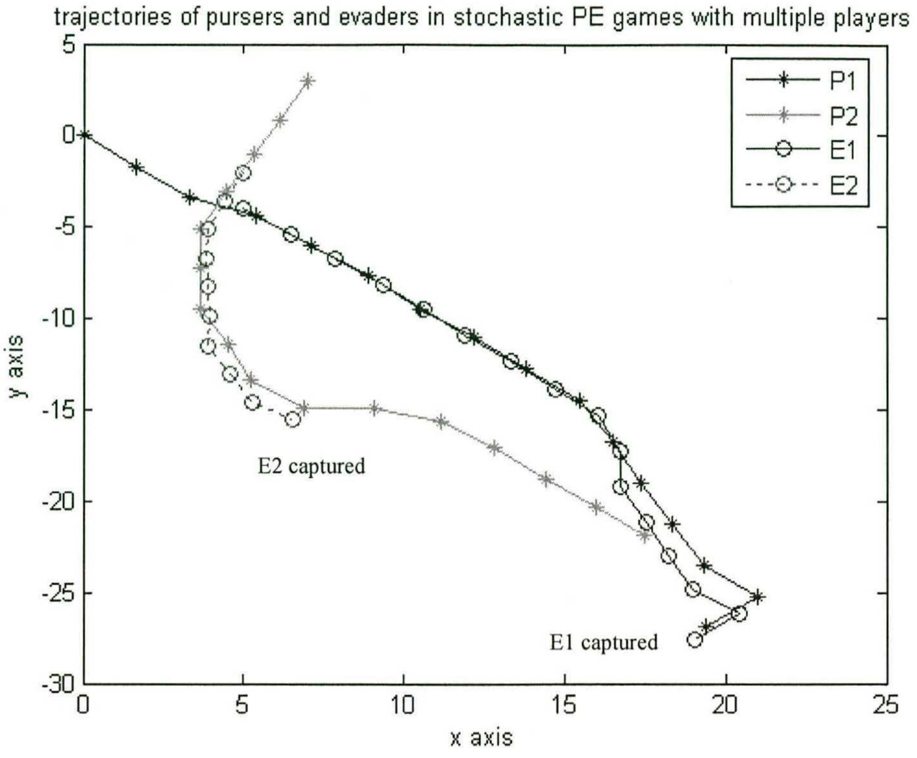

Figure 5.11 Trajectories of Pursuers and Evaders, Multiple PE

The corresponding trajectories are illustrated in the Figure 5.11. Pursuer 1 and Pursuer 2 start at (0, 0) and (7, 3) respectively. Evader 1 and Evader 2 start at (3, -4) and (5, -2) respectively. At first, Pursuer 1 is engaged in chasing Evader 1, and Pursuer 2 is engaged in chasing Evader 2. As the pursuit-evasion game moves on, Evader 2 is first captured by Pursuer 2 when Pursuer 1 is still on the path of chasing Evader 1. After that point, Evader 2 stops at the position where it is captured and Pursuer 2 continues to chase Evader 1. At the end of the game, Evader 1 is captured by Pursuer 1 and stops moving.

CHAPTER 6

CONCLUSIONS AND FURTHER RESEARCH

6.1 Conclusions and Summary of Thesis Contributions

The first contribution of this thesis is that we developed a method for numerically solving zero-sum stochastic Pursuit-Evasion games. It is called the Markov Chain Approximation method, and it selects a suitable Markov chain to approximate the original problem. A significant advantage of this method is that the construction of the Markov chain is straightforward and the resulting Markov chain is in the standard form of finite MDPs problems which can be solved conveniently by existing algorithms.

The second contribution is that we also developed a stochastic control software package, which is capable of solving general Markov Decision Processes problems. The package includes functions for solving both optimal control problems of finite MDPs and MDPs with continuous state space and for solving zero-sum stochastic games with finite state space and action space.

Several numerical examples are given to demonstrate the use of our software package. A one-to-one stochastic Pursuit-Evasion game is discussed step by step to illustrate the application of the Markov approximation method. This is extended to a method for solving a stochastic PE game with multiple players, using a decentralized approach.

6.2 Further Research

The Bellman collocation method for MDPs with continuous space is promising and it can also be used for solving finite MDPs problems when the state space is large, though finite. The idea of the method is similar to the parametric approximation algorithm of Neuro-Dynamic programming [25], which uses Neural networks to approximate the optimal value function and selects the weights of the networks. In this thesis, we only solve one dimensional control problems or games by applying this method. We need to extend the method to higher dimensional problems. Neuro-Dynamic programming should be a useful technique for this. Also, we must consider how to quantify the effects of approximation.

Reinforced machine learning, in which the players learn how to map situations to actions, has also been a useful technique to analyze PE games. The basic idea is simply to capture the most important aspects of the real problem facing a learning agent interacting with its environment to achieve a goal [24]. Methods like TD-learning and Q-learning [24, 26] used in Artificial Intelligence might be a good initial approach to solve stochastic pursuit-evasion games.

The information structure of stochastic PE games we considered in this thesis is the closed-loop perfect state information structure, i.e., the state of the system and the other player's objective function, are known without error. In realistic applications, this is not always the case. The players' observation might be imperfect. One player might not know the value of some parameters in the other player's objective function or even does not know the expression of the other player's objective function. Is the problem still feasible by using the Markov Chain Approximation under these conditions?

The stochastic software package that we developed in this thesis should be extended when considering the extended problems listed above. Further improvement and extension are necessary to refine this software package for a practical toolbox, such as the toolboxes in MATLAB.

BIBLIOGRAPHY

[1] L. S. Shapley, "Stochastic games," in *Proc. of National Academy of Sciences*, vol. 39, pp. 1095–1100, 1953.

[2] J. Filar and K. Vrieze, *Competitive Markov Decision Processes*, New York, NY: Springer-Verlag, 1997.

[3] S. Sorin, *A First Course on Zero-Sum Repeated Games*, Berlin, NY: Springer-Verlag, 2002.

[4] J. F. Mertens and A. Neyman, "Stochastic games have a value," in *Proc. of National Academy of Sciences*, vol. 79, pp. 2145–2146, 1982.

[5] T. Bewley and E. Kohlberg, "The asymptotic theory of stochastic games," *Mathematics of Operations Research*, vol. 1, pp. 197–208, 1976.

[6] L. S. Shapley and T. E. S. Raghavan, *Stochastic Games and Related Topics: In Honor of Professor L. S. Shapley*, Boston, MA: Kluwer Academic Publishers, 1991.

[7] H. J. Kushner, *Probability Methods for Approximations in Stochastic Control and for Elliptic Equations*, New York, NY: Academic Press, 1977.

[8] H. J. Kushner, "Numerical methods for stochastic control in continuous Time," *SIAM Journal on Control and Optimizations*, vol. 28, pp. 999-1048, 1990.

[9] H. J. Kushner and P. G. Dupuis, *Numerical Methods for Stochastic Control in Continuous Time*, Berlin, NY: Springer-Verlag, 1992.

[10]Y. Yavin and M. Pachter, *Pursuit Evasion Differential Games*, Elmsford, NY: Pergamon Press, 1987.

[11] O. J. Vrieze, "Stochastic games with finite state and action spaces," CWI Tracts No. 33, *Center for Mathematics and Computer Science*, Amsterdam, The Netherlands, February, 1987.

[12] L. D. Berkovitz, "Two-person zero-sum differential games: an overview," in *The Theory and Application of Differential Games* (J. D. Grote, eds), pp.12-22, Dordrecht, Holland: D. Reidel Publishing Co., 1975.

[13] R. Isaacs, *Differential Games*, New York, NY: John Wiley and Sons, 1965.

[14] Dongxu, Li and Jose B. Cruz, Jr., "An iterative method for pursuit-evasion games with multiple players," submitted to *the IEEE Trans. on Automatic Control.*

[15] J. P. Hespanha, H. J. Kim, and S. Sastray, "Multiple-agent probabilistic pursuit evasion games," in *Proc. of the IEEE 38th Conference on Decision and Control*, vol. 3 (Phoenix, AZ), pp. 2432-2437, December, 1999.

[16] J. H. Reif and S. R. Tate, "Continuous alternation: the complexity of pursuit in continuous domains," *Algorithmica,* vol. 10, pp. 157-181, 1993.

[17] W. H. Fleming and P. E. Souganidis, "On the existence of value functions of two-player, zero-sum stochastic differential games," *Indiana University mathematics Journal,* vol. 38, pp. 293-314, 1989.

[18] K. Itô, "Stochastic differential equations in a differentiable manifold", *Nagoya Mathematical Journal*, vol. 1, pp. 35-47, 1950.

[19] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, New York, NY: Academic Press, revised ed., 1998.

[20] W. H. Fleming and R. Risher, *Deterministic and Stochastic Optimal Control*, Berlin, NY: Springer-Verlag, 1975.

[21] W. H. Fleming and H. M. Soner, *Controlled Markov Processes and Viscosity Solution*, New York, NY: Springer-Verlag, 1992.

[22] R. E. Bellman, *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.

[23] M. J. Miranda and P. L. Fackler, *Applied Computational Economics and Finance*, Cambridge, MA: MIT Press, 2002.

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA: MIT Press, 1998.

[25]D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Belmont,

MA: Athena Scientific, 1996.

[26] L. P. Kaelbling, M. L. Littman and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[27] G. Yin, S. Miao and Q. Zhang, "Finite difference methods and markov chain approximation to a class of robust control problems," in *Proc. of the IEEE 34th Conference on Decision and Control*, vol. 4 (New Orleans, LA), pp. 3366-3371, December, 1995.

[28] M. Bowling and M. Veloso, "An analysis of stochastic game theory for multi-agent reinforcement learning," Technical report CMU-CS-00-165, Computer Science Department, Carnegie Mellon University, 2000.

[29] N. L. Stokey, R. E. Jucas, Jr. and E. C. Prescott, *Recursive Methods in Economic Dynamics*, Cambridge, MA: Harvard University Press, 1989.

[30] J. Adda and R. W. Cooper, *Dynamic Economics: Quantitative Methods and Applications*, Cambridge, MA: MIT Press, 2003.

[31] O. Hernandez-Lerma and J. B. Lasserre, "Zero-sum stochastic games in borel spaces: average payoff criteria," *SIAM Journal on Control and Optimization*, vol. 39, pp. 1520-1539, 2000.

[32] A. S. Nowak, "Sensitive equilibria for ergodic stochastic games with countable state spaces," *Mathematical Methods of Operations Research*, vol. 50, pp. 65-76, 1999.

[33] A. S. Nowak, "Optimal strategies in a class of zero-sum ergodic stochastic games," *Mathematical Methods of Operations Research*, vol. 50, pp. 399-419, 1999.

[34] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: theoretical framework and an algorithm," in *Proc. Of the 15th International Conference on Machine Learning*, (San Francisco, CA), pp. 242-250, July, 1998.

[35] S. Singh, M. Kearns and Y. Mansour, "Nash convergence of gradient dynamics in general-sum games," in *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, (Stanford University, CA), pp. 541-548, June, 2000.

[36] Dongxu, Li, "Multi-player pursuit-evasion differential games", Ph.D. dissertation, the Ohio State University, Columbus, OH, 2006.