Universiteit Utrecht

Bachelor thesis

Continuous-time Markov Decision Processes

Julius Linssen
4002830

supervised by
Karma Dajani

June 16, 2016

Abstract

Markov decision processes provide us with a mathematical framework for decision making. These models are now widely used in many fields, such as robotics, economics and ecology. In this thesis we will be looking at the finite-horizon case in discrete time as well as continuous time. In both cases we will start by formulating the problem, and proving that this formulation is valid. Afterwards we will provide algorithms to solve these processes, such as the dynamic programming algorithm. At the end we will solve a Continuous-time Markov decision problem in multiple ways, in order to analyze the different solutions.

Contents

Chapter 1

Introduction

Decisions are an important part of life, every day we make many of them. These decisions do not only result in a reward, but also have impact on decisions you can make in the future. By ignoring the influence today's decisions have on the decisions of the future, and present and future rewards, we may not achieve a good overall performance. For example, when you have to run ten miles, it is not a good idea to run as fast as you can from the start. Markov decision processes provide a mathematical framework that takes these aspects of decision making into account.

Markov decision processes have been studied since the 1950s and are now used in many areas, including robotics, economics, communications engineering and ecology. In 1957, Richard Bellman introduced what is now known as the Bellman-equation. This layed the groundwork for the solutions to the MDP's known as dynamic programming and reinforcement learning.

In this thesis we will describe the discrete-time and continuous-time Markov decision processes and provide ways of solving them both. The main focus lies on the continuous-time MDP, but we will start with the discrete case. The discrete case is solved with the dynamic programming algorithm. After having solved the discrete case, we will formulate the continuous-time case. The formulation of this case and justification of this formulation is of considerable difficulty and will take up most of chapter 4. In chapter 5 is concerned with the proof that the optimal policy for the continuous-time MDP is piecewise constant. After this proof is completed we describe the algorithm that solves the problem, but that algorithm comes with a problem. We therefore also intorduce the discrete approximation that uses the Discrete-time MDP solution to solve the Continuous-time MDP. To finish up, in chapter 6 we will look at an example where both solutions are used.

Chapter 2

Discrete Time Markov Decision Process

The model for decision making goes as follows: At a specified point in time, a decision maker observes the system state. Based on this information the decision maker chooses an action. This action has two results. Firstly the decision maker receives an immediate reward and the system evolves to a new state at the next point in time according to a probability distribution determined by the action chosen. At this point in time the decision maker faces the same situation. However, this time the decision maker might be in a new system state that presents him with new actions.

The key ingredients of this model are the following five elements:

1. A set of decision epochs
2. A set of system states
3. A set of possible actions
4. A set of immediate rewards/costs, depending on the current state and chosen action
5. A set of transition probabilities, also depending on state and action

We will discuss these elements in more detail below

2.1 The Decision Epochs

We call the points of time where decisions are made the decision epochs. The set of decision epochs will be referred to as T. In the discrete markov decision processes, decisions are made at all discrete time units. In general, the set of decision epochs can either be finite or infinite, but in this thesis we will only concern ourselves with the finite case. In this case, which is called a *finite horizon* problem, we define T $\equiv \{1, 2, \ldots, N\}$.

2.2 The System States and Action Sets

At every decision epoch we assume the decision maker is in some kind of state. We denote the set of possible system states by $S = \{1, \ldots, M\}$. At some decision epoch, the decision maker observes he is in state $s \in S$, and he chooses an action a, that he is allowed to make in state s. We denote the set of allowable decisions in state s by A_s and the set of total decisions by A, where $A = \cup_{s \in S}\ A_s$. The sets S and A_s we will look at, are arbitrary finite sets. In general however, S and A_s can get far more complicated than this.

2.3 The Rewards and Transition Probabilities

When the decision maker is in state s and chooses an action $a \in A_s$ at time t, there are two consequences,

1. the decision maker receives a reward $r_t(s, a)$
2. the system state at the next epoch is determined by the probability distribution $p_t(\cdot|s, a)$

$r_t(s, a)$ may be regarded as income in which case it is positive, or regarded as cost in which case it is negative. The reward can take multiple forms, we only require its (expected) value to be known before choosing an action, and that it is not affected by future actions. In finite horizon Markov decision processes there is no decision made at decision epoch N. Therefore we denote the reward at time N as $r_N(s)$ which we will call the *scrap value*

We have now gone through all the ingredients that make up a Markov decision process. From now on, when we talk about a *Markov decision process*, we refer to the collection of the objects

$$\{T, S, A_s, p_t(\cdot|s, a), r_t(s, a)\}$$

We use the "Markov "qualifier to indicate that the transition probabilities and the rewards are not influenced by the past. What is left for us is a strategy to deal with these MDP's: the policy.

2.4 The Policy

We now know what a discrete Markov decision process looks like: You select an action at each point in time based on the state you are in, and then you receive a reward and transit into a new state until we arrive at the end. Let $D = A_1 \times A_2 \times \cdots \times A_M$. The action you choose is described by the *policy* $\pi(t) : T \to D$. This is a function which tells you what action you must take for each state at time t. We will use the notation $\pi_s(t)$ for the action prescribed by the policy, for state s at time t and we denote the set of all possible policies as Π. We will also require the policy to be Markovian. That is, the policy may only depend on the current system state and the current action taken.

The evolution of the Markov decision process from state to state depends on the policy. The return of the process is the sum of rewards which also depends on the implemented policy. Our goal throughout this thesis will be to determine the policy which maximizes the expected return of the entire process.

Chapter 3

Solving discrete-time Markov decision processes

3.1 The expected total reward

We said we will be maximizing the expected total reward, so it is time to define what that means. We denote the expected total reward of a Markov decision process starting in state s, using a policy π as follows:

$$v^{\pi}(s) \equiv E^{\pi}[\sum_{t=1}^{N-1} r_t(s_t, \pi_{s_t}(t)) + r_N(s_N)]$$

We quickly note that this expression is just the sum of all the acquired rewards of time 1 up to time $N-1$, and lastly the scrap value. As we mentioned earlier, the goal is to find a policy that maximizes this total reward. That is, we are looking for a policy π^* such that

$$v^{\pi^*}(s) \geq v^{\pi}(s), \quad s \in S \tag{3.1}$$

for all $\pi \in \Pi$. The maximal total reward is often referred to as $v^*(s) \equiv \max_{\pi\in\Pi} \ v^{\pi}(s)$.

3.2 The utility functions

To calculate the total expected reward we will be using utility functions $u_t^{\pi} : S \to R$. If we are in state s_t at time t we define u_t^{π} as follows:

$$u_t^{\pi}(s_t) \equiv E^{\pi}[\sum_{i=t}^{N-1} r_i(s_i, \pi_{s_i}) + r_N(s_N)] \tag{3.2}$$

where we let $u_N^{\pi} \equiv r_N(s_N)$. The difference between $v^{\pi}(s)$ and $u_t^{\pi}(s_t)$ is of course that $u_t^{\pi}(s_t)$ denotes the expected reward from epoch t and onwards whereas $v^{\pi}(s)$ includes all the rewards received throughout the entire process.

We will now show how we calculate $v^\pi(s)$ by calculating $u_1^\pi(s)$. This is called the *finite-horizon policy evaluation algorithm.* For this algorithm, we assume a policy $\pi \in \Pi$ to be given.

The finite-horizon policy evaluation algorithm

1. Set $t = N$ and $u_N^\pi(s_N) = r_N(s_N)$ for all $s_N \in S$.
2. If $t = 1$ stop, otherwise go to step 3.
3. Substitute $t-1$ for t and compute $u_t^\pi(s_t)$ for each $s_t \in S$ by

$$u_t^\pi(s_t) = r_t(s_t, \pi_{s_t}(t)) + \sum_{j \in S} p_t(j|s_t, \pi_{s_t}(t)) u_{t+1}^\pi(j) \tag{3.3}$$

4. Return to 2

Some motivation for calculating these utility functions backwards will follow. The expected reward of policy π over epochs t through N is equal to the immediate reward at epoch t by selecting $\pi_s(t)$ plus the expected total reward over periods $t+1$ through N. These are all known when we start calculating u_t^π, so summing over all possible states $j \in S$ for s_{t+1} gives the desired expectation in terms of u_{t+1}^π. This idea is what stands at the basis of dynamic programming, since it reduces the problem of computing expected total rewards to $N-1$ one-period calculations. What is left for us to do, is to show that this algorithm does indeed give us the u_t^π as they were defined. The proof is by induction on t.

Theorem 1. *Let $\pi \in \Pi$ be given and suppose u_t^π has been generated by the finite-horizon policy evaluation algorithm. Then, for all $t \leq N$, (3.2) holds and $v^\pi(s) = u_1^\pi(s)$ for all $s \in S$.*

Proof. By the way we defined $u_N^\pi(s)$, the theorem holds for $t = N$. Suppose now that the (3.2) holds for $t+1, t+2, \ldots, N$. Then by using (3.3) and the induction hypothesis:

$$u_t^\pi(s_t) = r_t(s_t, \pi_{s_t}(t)) + \sum_{j \in S} p_t(j|s_t, \pi_{s_t}(t)) u_{t+1}(j)$$

$$= r_t(s_t, \pi_{s_t}(t)) + \sum_{j \in S} p_t(j|s_t, \pi_{s_t}(t)) E^\pi\{\sum_{i=t+1}^{N-1} r_i(s_i, \pi_{s_i}) + r_N(s_N)|s_{t+1} = j\}$$

$$= r_t(s_t, \pi_{s_t}(t)) + E^\pi[E^\pi\{\sum_{i=t+1}^{N-1} r_i(s_i, \pi_{s_i}) + r_N(s_N)|s_{t+1} = j\}|s_t]$$

$$= r_t(s_t, \pi_{s_t}(t)) + E^\pi[\sum_{i=t+1}^{N-1} r_i(s_i, \pi_{s_i}) + r_N(s_N)|s_t]$$

Since s_t and π_{s_t} are both known at time t we can include the first term in the expectation giving us the desired result. □

So this algorithm works, but we have not yet discussed why we are using it. The way $v^{\pi}(s)$ is defined, it can be calculated rigorously by computing the expectation. This will require us to calculate the chance of every possible combinations of states and actions. Assuming there are M states, this requires us to do M^N multiplications. Using the algorithm, we are required to do M calculations for the expectation in step 3, we have to do that for M states at each point, and there are $N-1$ points where we have to do this calculation. This gives us only $(N-1)*M^2$ calculations. So the algorithm turns a calculation that is exponential in time, into a calculation that is linear in time. That is what makes dynamic programming such a powerful tool.

3.3 Optimality equations

Now we know how to evaluate a policy, it is time to see how we can find the optimal policy. In this section we will introduce the optimality equations and show that these equations are sufficient to achieve optimality. This will be done by optimizig the utility functions, and we define the optimal utility functions as:

$$u_t^*(s_t) \equiv \max_{\pi \in \Pi} u_t^{\pi}(s_t)$$

This denotes the maximum over all policies of the expected reward from decision epoch t onward, given that the state at time t is s_t. We will prove that we can find these optimal utility functions by using the optimality equations which are as follows:

$$u_t(s_t) = \max_{a \in A_{s_t}} r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}(j)$$

for $t = 1, \ldots, N-1$ and $s_t \in S$. For $t = N$ we add the boundary condition

$$u_N(s_N) = r_N(s_N)$$

for $s_N \in S$. These optimality equations are fundamental in Markov decision theory and will show up again in the continuous case. Two of the most important properties of these optimality equations are as follows:

- Solutions to the optimality equations are optimal returns from decision epoch t onward for all t.
- The optimality equations provides a method for determining whether a policy is optimal. If the expected total reward from period t onward satisfies this system of equations for $t = 1, \ldots, N$ it is optimal.

The first property is simply an interesting property of the optimal policy. It does not only optimize the MDP from period 1 to N, but also from every decision point t onwards, no matter what state occurs at time t. The latter one speaks for itself, this gives us a way to determine whether a policy is optimal, which is much better than calculating the total expected reward over all $\pi \in \Pi$.

We will now state and prove the theorem which shows that utility functions satisfying the optimality equations are the optimal utility functions.

Theorem 2. *Suppose u_t is a solution to the optimality equations for $t = 1, \ldots, N$. Then*

1. *$u_t(s_t) = u_t^*(s_t)$ for all $s_t \in S$, $t = 1, \ldots, N$ and*
2. *$u_1(s_1) = v^*(s_1)$ for all $s_1 \in S$.*

Proof. The proof is in two parts. We will first prove that $u_t(s_t) \geq u_t^*(s_t)$ for all $s_t \in S$ and $t = 1, \ldots, N$. For period N we have $u_N(s_N) = r_N(s_N) = u_N^\pi(s_N)$ for all $s_N \in S$ and $\pi \in \Pi$. Therefore $u_N(s_N) = u_N^*(s_N)$ for all $s_N \in S$. Now assume that $u_t(s_t) \geq u_t^*(s_t)$ for all $s_t \in S$ for $t = n+1, \ldots, N$. Let π' be an arbitrary policy in Π. For $t = n$, the optimality equation becomes

$$u_n(s_n) = \max_{a \in A_{s_n}} \{r_n(s_n, a) + \sum_{j \in S} p_n(j|s_n, a) u_{n+1}(j)\}$$

By the induction hypothesis we have:

$$u_n(s_n) = \max_{a \in A_{s_n}} \{r_n(s_n, a) + \sum_{j \in S} p_n(j|s_n, a) u_{n+1}^*(j)\}$$

$$\geq \max_{a \in A_{s_n}} \{r_n(s_n, a) + \sum_{j \in S} p_n(j|s_n, a) u_{n+1}^{\pi'}(j)\}$$

$$\geq r_n(s_n, \pi'_{s_n}(n)) + \sum_{j \in S} p_n(j|s_n, \pi'_{s_n}(n)) u_{n+1}^{\pi'}(j)$$

$$= u_n^{\pi'}(s_n)$$

Since π' is arbitrary,

$$u_n(s_n) \geq u_n^\pi(s_n) \quad \text{for all } \pi \in \Pi$$

Thus $u_n(s_n \geq u_n^*(s_n))$ and the induction hypothesis holds.

Now we establish that for any $\epsilon > 0$, there exists a $\pi' \in \Pi$ for which

$$u_n^{\pi'}(s_n) + (N-n)\epsilon \geq u_n(s_n)$$

for all $s_n \in S$ and $n = 1, \ldots, N$. To do this, construct a policy π' by choosing $\pi'_n(s_n)$ to satisfy

$$r_n(s_n, \pi'_{s_n}(n)) + \sum_{j \in S} p_n(j|s_n, \pi'_{s_n}(n)) u_{n+1}^*(j) + \epsilon \geq u_n(s_n)$$

which is possible by the way $u_n(s_n)$ is defined. Once again we establish the result by induction. Since $u_N^{\pi'}(s_n) = u_N(s_N)$, the induction hypothesis holds for $t = N$. Assume that $u_t^{\pi'}(s_t) + (N-t)\epsilon \geq u_t(s_t)$ for $t = n+1, \ldots, N$. It then follows that

$$u_n^{\pi'}(s_n) = r_n(s_n, \pi'_{s_n}(n)) + \sum_{j \in S} p_n(j|s_n, \pi'_{s_n}(n)) u_{n+1}^{\pi'}(j)$$

$$\geq r_n(s_n, \pi'_{s_n}(n)) + \sum_{j \in S} p_n(j|s_n, \pi'_{s_n}(n)) u_{n+1}(j) - (N-n-1)\epsilon$$

$$\geq u_n(s_n) - (N-n)\epsilon.$$

Thus the induction hypothesis is satisfied for $n = 1, \ldots, N$. Therefore for any $\epsilon > 0$, there exists a $\pi' \in \Pi$ for which

$$u_n^*(s_n) + (N-n)\epsilon \geq u_n^{\pi'}(s_n) + (N-n)\epsilon \geq u_n(s_n) \geq u_n^*(s_n)$$

and from this part 1 follows. Part 2 follows by definitions of $u_1^*(s)$ and $v^*(s)$

□

Part 1 of this theorem shows that solutions of the optimality equation are the optimal utility functions from period t onward, and result 2 means that the solution to the equation with $n = 1$ is the optimal expected return from decision epoch 1 onward.

We will now state the theorem that shows how to use the optimality equations to find optimal policies, and to verify that a policy is optimal.

Theorem 3. *Suppose $u_t^*, t = 1, \ldots, N$ are solutions of the optimality equations, and that policy $\pi^* \in \Pi$ satisfies*

$$\pi_{s_t}^*(t) = \underset{a \in A_{s_t}}{\operatorname{argmax}}[r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j)] \tag{3.4}$$

for $t = 1, \ldots, N-1$. Then

***a.** For each $t = 1, \ldots, N$,*

$$u_t^{\pi^*}(s_t) = u_t^*(s_t), s_t \in S.$$

***b.** π^* is an optimal policy, and*

$$v^{\pi^*}(s) = v^*(s), s \in S$$

Proof. We once again establish part (a), part (b) follows from part (a), Theorem 1 and Theorem 2. We proof part (a) by induction. For $t = N$ we clearly have

$$u_N^{\pi^*}(s_N) = u_N * (s_N), s_N \in S_N$$

.

Assume the result holds for $t = n+1, \ldots, N$. Then, for $s_n \in S$ by the optimality equations we have

$$u_n^*(s_n) = \max_{a \in S_n}[r_n(s_n, a) + \sum_{j \in S} p_n(j|s_n, a) u_{n+1}^*(j)]$$

$$= r_n(s_n, \pi_{s_n}(n)) + \sum_{j \in S} p_n(j|s_n, \pi_{s_n}(n)) u_{n+1}^{\pi^*}(j)$$

$$= u_n^{\pi^*}$$

Here the second equality follows from how we defined π^* and thus the induction hypothesis holds and the theorem follows. □

This theorem tells us that an optimal policy can be found by first solving the optimality equations, and then choosing a decision rule which selects any action which attains the maximum the right hand side of (3.4) for $t = 1, \ldots, N$. With this theorem we have laid the groundwork for the main result of this section.

3.4 The Backward Induction Algorithm

Backward induction provides us the method for solving finite-horizon discrete-time MPD's. The terms "backward induction"and "dynamic programming" are synonymous. This subsection presents the backward induction algorithm and shows how to use it to find optimal policies and utility functions.

The Backward Induction Algorithm

1. Set $t = N$ and

$$u_N^*(s_N) = r_N(s_N)\text{for all} s_N \in S,$$

2. Substitute $t-1$ for t and compute $u_t^*(s_t)$ for each $s_t \in S$ by

$$u_t^*(s_t) = \max_{a \in A_{s_t}} [r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j)]$$

   Set

$$\pi_{s_t}^*(t) = \underset{a \in A_{s_t}}{\text{argmax}} [r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j)]$$

3. If $t = 1$, stop. Otherwise return to step 2.

Note that we compute the utility functions by using the optimality equations. Using this algorithm and theorem 2.3.2, we achieve the following result

Corollary 4. *Suppose $u_t^*, t = 1, \ldots, N$ and $\pi_{s_t}^*(t) t = 1, \ldots, N-1$ are generated by the Backward Induction Algorithm, then*

a. *for $t = 1, \ldots, N$ and $s_t \in S$*

$$u_t^*(s_t) = \max_{\pi \in \Pi} u_t^\pi(s_t), s_t \in S$$

b. *$\pi^* \in \Pi$ is optimal and satisfies*

$$v^{\pi^*}(s) = sup_{\pi \in \Pi} v^\pi(s), s \in S$$

and

$$u_t^{\pi^*}(s_t) = u_t^*(s_t), s_t \in S$$

for $t = 1, \ldots, N$.

This corrolary looks slightly complicated, but it is simply a formal statement of the following properties of the backward induction algorithm.

- For $t = 1, \dots, N-1$ it finds sets $\pi^*_{s_t}(t)$ which maximize the utility functions.
- It evaluates any policy which selects an action $\pi_{s_t}(t)$ at time t for each $s_t \in S$ for all $t = 1, \dots, N-1$.
- It computes the expected total reward for the entire decision-making horizon, and from each period to the end of the horizon for any optimal policy.

Which are key properties when it comes to optimization: we need the best solutions at every point of time for every state we can be in. Now we have proved that we achieve these goals in the Discrete case of the Markov decision process, we can start looking at the Continuous-time case.

Chapter 4

continuous time Markov decision processes

The Continuous-time Markov Decision process is formulated in a way analogous to the Discrete-Time case. However, the formulation of the continuous case and justification of this formulation is a lot more difficult this time and will take up the biggest part of this chapter. We will try to build up the theory analogous to chapter 2.

4.1 problem formulation

This time we there are no discrete decision epochs. Instead, the system operates from time zero to time T where $T < \infty$ and decision are made at every point in time. Once again we consider a system that may be in one of M states labeled $1, \ldots, M$ at any point in time. This time when the system is in state s, an action a is chosen from the finite set A_s and we receive a return rate $r(s, a)$, this time continuous, depending only on the current state and action taken.

In the discrete case choosing an action also came along with a probability law that determined the state in the next decision epoch. This is the part that changes the most in the continuous case. The evolution of the system state is described by a probability law which depends on a matrix $Q(\pi(t))$ with components $q(j|i, a)$ that can be thought of as the *rate of going from state i to state j choosing action a.* These components satisfy the property that $0 \leq q(j|i, a) < \infty$, $j \neq i$, $0 \leq -q(i|i, a) \leq \infty$ and $\sum_{j \in S} q(j|i, a) = 0$ for all $i \in S$. We will later go into details about the Q-matrix and its properties.

For the policy, let again $D = A_1 \times A_2 \times \cdots \times A_M$, then a policy π is now a function $\pi : [0, T] \to D$. So using a policy π means that if the system is in state s at time t, the action chosen is $\pi_s(t)$, the projection of $\pi(t)$ on the s-component. We require that π is a measurable function, meaning that for any $d \in D$, the set $\{t : \pi(t) = d, 0 \leq t \leq T\}$ is Lebesque measurable. Furthermore, for any $d \in D$ we define $r(d)$ as the $n \times 1$ column vector whose i^{th} element is $r(i, d_i)$ and define $Q(d)$ as the $n \times n$ matrix whose elements q_{ij} are $q(j|i, d_i)$. Lastly, we say a function $g(x)$ is of order x, written symbolicly as $o(x)$, when $\lim_{x \to 0} \frac{g(x)}{x} = 0$.

4.2 The Transition Probabilities

As mentioned earlier, the evolution of the system state depends on the matrix $Q(\pi(t))$ whose components satisfied the following properties:

1. $0 \leq -q_{ii} < \infty$ for all i.
2. $0 \leq q_{ij} \leq \infty$ for all $i \neq j$
3. $\sum_{j \in S} q_{ij} = 0$ for all i.

Now let $x_t(\omega), 0 \leq t \leq T$, be a sample path of a process which may be in one of M states labeled $1, 2, \ldots M$. If $Prob(x_s(\omega) = i) > 0$ define $p_{ij}(s,t)$ by $p_{ij}(s,t) = Prob(x_t(\omega) = j | x_s(\omega) = i)$ and let $P(s,t)$ be the matrix $(p_{ij}(s,t))$. We want our probability matrix P to be a Markov transition matrix function, which is defined as follows:

Definition 5. *(Doob [3, p. 236]): A Markov transition matrix function is a matrix function $P(\cdot,\cdot)$ satisfying*

1. $P(\cdot,\cdot) \geq 0$

2. $P(\cdot,\cdot)1 = 1$

3. $P(s,u) = P(s,t)P(t,u)$ *where* $0 \leq s < t < u \leq T$

It is convenient here to define $P(s,s) = I$ so that (3) holds for $0 \leq s \leq t \leq u \leq T$ and we will include it in our definition.

As mentioned before it, the elements q_{ij} denote the exponential transition rates of going to state j from state i. This means that for very small t we want to have that approximately $p_{ij} = \delta_{ij} + q_{ij}t$ which in matrix form becomes $P(t, t+h) = I + Q(\pi(t))h$. We will state this more formally, but before we can do that we need to define what it means for a function to be absolutly continuous

Definition 6. *Let (X,d) be a metric space and let I be an interval in $\mathbb{R}$. A function $f : I \to X$ is absolutely continuous on I if for every positive number ϵ, there is a positive number δ such that whenever a finite sequence of pairwise disjoint sub-intervalt $[x_k, y_k]$ of I satisfies*

$$\sum_k |y_k - x_k| < \delta$$

then

$$\sum_k d(f(y_k), f(x_k)) < \epsilon$$

We can now state our problem formally: Given a Lebesque measurable policy π defined on $[0,T]$ and hence a Lebesque measurable matrix function $Q(\pi(\cdot))$ we want to know if there is an absolutely continuous Markov transition matrix function $P(\cdot,\cdot)$ such that for $t > s$

$$P(s,t) = I + Q(\pi(s))(t-s) + o(t-s) \tag{4.1}$$

is satisfied for almost all $s \in (0,T)$ where $o(t-s)$ is a matrix whose components are of order $(t-s)$ and if this matrix function is unique.

In order to do this, consider the matrix function $P(s,\cdot),\ 0 \leq s \leq T$, defined for any measurable policy π by the solution to the differential equations

$$\frac{d}{dt}P(s,t) = P(s,t)Q(\pi(t)) \tag{4.2}$$

for $s \leq t \leq T$ and the initial condition $P(s,s) = I$. Any absolutely continuous function that satisfies the initial condition and the differential equation almost everywhere is to be seen as a solution. We will prove that the matrix function $P(\cdot,\cdot)$ is the unique absolutely continuous Markov transition matrix function satisfying (5) almost everywhere.

The uniqueness of (4.2) is given to us by Coddington and Levinson [4, p. 74]:

Theorem 7. *The unique solution to (4.2) is given by the fundamental matrix $\Phi(\cdot)$ of the differential equations*

$$\frac{dx}{dt} = xQ(\pi(t))$$

satisfying $\Phi(s) = I$ where x is a row vector.

Royden [5, p.84, problem 5.3] gives us the following result from analysis:

Lemma 8. *If $f(x)$ is a continuous function on $[a,b]$ and any derivative, say*

$$D^+f(x) = \limsup_{h\to 0+} \frac{f(x+h)-f(x)}{h}$$

is nonnegative for $x \in (a,b)$ then $f(b) \geq f(a)$

Lemma 9. *If f is contiuous on $[a,b]$ and $D^+f(t)$ is nonnegative whenever $f(t) < 0$ for $t \in (a,b)$, then $f(a) \geq 0$ implies $f(t) \geq 0$ for $t \in [a,b]$*

Proof. The proof is by contradiction. Let t_1 be a point in (a,b) such that $f(t_1) < 0$. Define t' by $t' = \max\{t : f(t) = 0 \text{ and } t < t_1\}$. This maximum exists by continuity of f and the initial condition $f(a) \geq 0$. Then on the interval $(t', t_1), f(t) < 0$ and hence $D^+f(t) \geq 0$ by hypothesis. Using lemma 2 we must have $f(t_1) \geq f(t') = 0$, a contradiction. □

Lemma 10. *The unique solution $P(s,\cdot)$ to (4.2) has the property that $P(s,t) \geq 0$.*

Proof. For $t \in [s,T]$ let $M(t) = \sum_{m=1}^{n}[p_{im}(s,t)]^-$, where $[p_{im}(s,t)]^- = p_{im}(s,t)$ when $p_{im}(s,t)$ is negative and $[p_{im}(s,t)]^- = 0$ otherwise. We will show that the function $M(\cdot)$ satisfies the hypothesis of lemma 9 and is therefore nonegative. This will prove the functions $p_{im}(s,t)$ are nonnegative for arbitrary i, thus proving the theorem.

From the initial condition we have $M(s) = 0$. The function $M(\cdot)$ is continuous since each function $p_{im}(s,t)$ is continuous, implying the function $[p_{im}(s,t)]^-$ is continuous, and the sum of continuous functions is continuous. The final requirement for $M(\cdot)$ to satisfy the hypothesis of lemma 9, is that if $M(t') < 0$ for some $t' \in (s,T)$, then $\limsup_{\epsilon\to 0+} \frac{M(t'+\epsilon)-M(t')}{\epsilon} \geq 0$.

Let $M(\cdot) < 0$ and let

$$J^- = \{j : p_{im}(s,t') < 0\}$$
$$J^0 = \{j : p_{im}(s,t') = 0\}$$
$$J^+ = \{j : p_{im}(s,t') > 0\}$$

By continuity of the functions $p_{ij}(s,\cdot)$ we can pick a $\delta > 0$ such that if $j \in J^-$ then $p_{ij}(s,t) < 0$ for all $t \in (t', t'+\delta)$ and if $j \in J^+$ then

$$p_{ij}(s,t) > 0$$

for all $t \in (t', t'+\delta)$. Hence for $0 \le \epsilon \le \delta$

$$M(t'+\epsilon) = \sum_{m=1}^{n} [p_{ij}(s,t'+\epsilon)]^-$$

$$= \sum_{j\in J^-} p_{ij}(s,t'+\epsilon) + \sum_{j\in J^0 \text{ and } p_{ij}(s,t'+\epsilon)<0} p_{ij}(s,t'+\epsilon)$$

For convenience, let $J_\epsilon^- = \{j : j \in J^- or j \in J^0 and p_{ij}(s,t'+\epsilon) < 0\}$. Then we can write

$$M(t'+\epsilon) - M(t') = \sum_{j\in J_\epsilon^-} [p_{ij}(s,t'+\epsilon) - p_{ij}(s,t')]$$

$$= \sum_{j\in J_\epsilon^-} \int_{t'}^{t'+\epsilon} \sum_{m=1}^{n} p_{im}(s,t)q(j|m,\pi_m(t))dt$$

$$= \int_{t'}^{t'+\epsilon} \{ \sum_{m\in J^+} \sum_{j\in J_\epsilon^-} p_{im}(s,t)q(j|m,\pi_m(t))$$

$$+ \sum_{m\in J^0} \sum_{j\in J_\epsilon^-} p_{im}(s,t)q(j|m,\pi_m(t))$$

$$+ \sum_{m\in J^-} \sum_{j\in J_\epsilon^-} p_{im}(s,t)q(j|m,\pi_m(t))\}dt$$

The first double summation is positive for all $t \in (t', t'+\epsilon)$ since $m \notin J_\epsilon^-$ implies $\sum_{j\in J_\epsilon^-} q(j|m,\pi_m(t)) \ge 0$ and $m \in J^+$ implies $p_{im}(s,t) \ge 0$. The third double summation is positive for all $t \in (t', t'+\epsilon)$ since $m \in J_\epsilon^-$ implies $\sum_{j\in J_\epsilon^-} q(j|m,\pi_m(t)) \le 0$ and $m \in J^-$ implies $p_{im}(s,t) \le 0$. Hence

$$M(t'+\epsilon) - M(t') \ge \int_{t'}^{t'+\epsilon} \sum_{m\in J^0} \sum_{j\in J_\epsilon^-} p_{im}(s,t)q(j|m,\pi_m(t))dt$$

The $lim_{\epsilon\to 0+} p_{im}(s,t'+\epsilon) = 0$ if $m \in J^0$, and the $q(\cdot|\cdot,\cdot)$ are bounded. Hence the righthand side is an order of ϵ and $M(t'+\epsilon) - M(t') \ge o(\epsilon)$.

Dividing both sides by epsilon and taking the lim sup of both sides as ϵ goes to 0 this gives us

$$\limsup_{\epsilon\to 0+} \frac{M(t'+\epsilon) - M(t')}{\epsilon} \ge 0$$

which proves the lemma. □

At this point it is convenient to establish the following result:

Lemma 11. *The diagonal elements $p_{ii}(t)$ are strictly positive for any finite t.*

Proof. The function $p_{ii}(t)$ is defined as the solution to the differential equations $\frac{d}{dt}p_{ii}(t) = q(i|i,\pi_i(t))p_{ii}(t) + \sum_{j\neq i} q(i|j,\pi_j(t))p_ij(t)$ and the initial condition $p_{ii}(0) = 1$. If we let q_i be a lower bound on $q(i|i,a)$ for $a \in A_i$, and define $p_{ii}^{'}(t)$ by $\frac{d}{dt}p_{ii}^{'} = q_i p_{ii}^{'}(t), p_{ii}^{'}(0) = 1$, then the function $(p_{ii}(t) - p_{ii}^{'}(t))$ satisfies the hypothesis of lemma 9 since $p_{ii}^{'}(t) \ge 0$ and by lemma 10 $p_{ij}(t) \ge 0$. Therefore $p_{ii}(t) \ge p_{ii}^{'}(t) = e^{q_i t} > 0$. □

Lemma 12. *The unique solution $P(s,\cdot)$ to (4.2) has the property that $P(s,t)1 = 1$*

Proof. $P(s,t)1 = P(s,s)1 + \int_s^t P(s,u)Q(\pi(u))1du = P(s,s)1 = 1$ since $Q(\pi(u))1 = 0$ □

Lemma 13. *The unique solution to (4.2) satisfies $P(s,u) = P(s,t)P(t,u)$ where $0 \leq s < t < u \leq T$.*

Proof. The result follows from the fact that both $\Psi(s,\cdot)$ and $\Psi(s,t)\Psi(t,\cdot)$ are solutions to $\frac{dx}{dt} = xQ(\pi(\cdot))$ which equal $\Psi(s,t)$ at time t and are identical by of the solution to this differential equation. □

Theorem 14. *The unique matrix function $P(\cdot,\cdot)$ obtained by solving (4.2) is a Markov transition function.*

Proof. The result follows from lemma 10, 12 and 13. □

What is left for us to prove, is that the unique solution to (4.2) satisfies (4.1) almost everywhere with the following definition from Natanson [6, p. 255]:

Definition 15. *If $\lim_{h\to 0} \frac{1}{h}\int_x^{x+h} |f(t) - f(x)|dt = 0$, the point x is said to be a Lebesque point of the function f(t).*

Lemma 16. *If s is a Lebesque point of each of n^2 functions $q(j|i,\pi_i(\cdot))$ defined on $[0,T]$ where π is an arbitrary measurable policy, then the unique solution to (4.2) satisfies*

$$P(s,t) = I + Q(\pi(s))(t-s) + o(t-s)$$

Proof. From (4.2):

$$P(s,t) = P(s,s) + \int_s^t P(s,u)Q(\pi(u))du$$

$$I + \int_s^t P(s,u)[Q(\pi(u))$$

$$-Q(\pi(s))]du$$

$$+\int_s^t P(s,u)Q(\pi(s))du$$

Since s is a Lebesque point of the matrix function $Q(\pi(\cdot))$ and $P(s,\cdot)$ is bounded, the first integral is of order $(t-s)$. The matrix function $P(s,\cdot)$ is continuous so that $P(s,u) = I + K(u)$ where $\lim_{u\to s+} K(u) = 0$. Therefore the second integral equals $Q(\pi(s))(t-s) + o(t-s)$ which proves the lemma. □

From Natanson [6, p.255, theorem 5] we have the following result:

Lemma 17. *If f is integrable on $[a,b]$ then almost every point of $[a,b]$ is a Lebesque is a Lebesque point of f.*

We now combine lemma 16 and 17.

Theorem 18. *Let π be any Lebesque measurable policy defined on $[0,T]$. Then for almost all s*

$$P(s,t) = I + Q(\pi(s))(t-s) + o(t-s)$$

where $P(s,\cdot)$ is the unique solution to (4.2).

We now come to the final theorem of this section:

Theorem 19. *The solution to the differential equations (4.2) is the unique absolutely continuous matrix function whish satisfies (4.1) almost everywhere.*

Proof. Theorems 14 and 18 and the initial condition of (4.2) prove that the solution to (4.2) is an absolutely continuous Markov transition matrix function which satisfies (4.1) almost everywhere. We will show uniqueness by showing that any absolutely continuous Markov transition matrix function, $P^{'}(\cdot,\cdot)$, which satisfies (4.1) almost everywhere is the unique solution to (4.2). Since $P^{'}(\cdot,\cdot)$ is a Markov transition matrix function we have

$$P^{'}(s,t+\Delta t) = P^{'}(s,t)P^{'}(t,t+\Delta t)$$

.
By hypothesis $P^{'}(\cdot,\cdot)$ satisfies (4.1) almost everywhere so that

$$P^{'}(s,t+\Delta t) = P'(s,t)(I + Q(\pi(t))(\Delta t) + o(\Delta t))$$

for almost all t. Hence

$$lim_{\Delta t \to 0+} \frac{P^{'}(s,t+\Delta t) - P^{'}(s,t)}{\Delta t} = P^{'}(s,t)Q(\pi(t))$$

For almost all t. The initial condition of (4.2) is satisfied because $P^{'}(\cdot,\cdot)$ must satisfy $P'(s,s) = I$ which completes the proof. □

This result assures us that the state of the system we consider when a measurable policy π is chosen, is described by a Markov process whose transition probabilities are given by the unique absolutely continuous matrix function that satisfies (4.1) almost everywhere. There is more to be said on this subject before we can truly make this conclusion, but for this I refer you to Miller's paper [2, chapter 2.3, p. 12]

4.3 Piecewise constant policy

We now know that our system is well-defined, we will briefly consider what problems we may run into when examining the optimal solution. In the discrete case we had a finite amount of decision epochs in which we had to select an action. This time however, we are dealing with an interval, which means our policy needs to select an action for uncountable infinite points in time. It is therefore not unthinkable that the optimal solution selects different actions at infinite or even uncountable infinite points in time. This is, at the least, unprefferable and we will now introduce a class of policies besides the one of measurable policies: The piecewise constant policies.

Definition 20. *A policy $\pi(t), 0 \leq t \leq T$, is piecewise constant if for any finite $t', t' \leq T$, the interval $[0,t']$ can be divided into a finite number of intervals $(0,t_1),(t_1,t_2),\ldots,(t_{m-1},t')$ such that $\pi(t)$ is constant on $(t_j,t_{j+1}), 0 \leq j \leq m-1$.*

The ambiguity at the endpoints, called switching points, is resolved by saying that $\pi(t)$ is continuous on the left, thus $\pi(t_j)$ takes the value of π on $(t_{j-1},t_j$ for $j = 1,\ldots,m$, and $\pi(0)$ is arbitrary.

A special case of the piecewise constant policy is the stationary policy where $\pi(t) = d$ for all t. This policy (d,T) will be abbreviated to d.

Chapter 5

Solving continuous-time Markov decision processes

We now arrive at the main part of this thesis, the solution to the continuous-time Markov decision process. This section consists of three parts. Firstly, we will look under which conditions we achieve optimality. In the second part we will prove that there is a piecewise constant policy that satisfies these conditions and thus is optimal. Lastly we will look at a few problems that arrise when computing the optimal policy. This gives reason to compute a solution that is not optimal, but easily computable and can get indefinetly close to the actual optimal solution. This is called the discrete approximation and this is among the mostly used algorithms for solving the CTMDP.

5.1 conditions for optimality

A quick recap: In the finite-horizon problem we seek to maximize the vector

$$v(T,\pi) = \int_0^T P(t)r(\pi(t))dt \tag{5.1}$$

where the matrix function $P(\cdot)$ is the unique solution to

$$\frac{d}{dt}P(t) = P(t)Q(\pi(t)) \text{ for } 0 \leq t \leq T, P(0) = I \tag{5.2}$$

Here we have supressed the s in the notation for the transition matrix. As we saw in the discrete-time case, we will use utility functions to show that we achieve optimality, this time labeled $\psi(t), 0 \leq t \leq T$. It is not obvious from the start that these functions play the same role as the utility functions of the discrete-time case, but we will go more into that later. For now, we will simply use them in our condition for π to be optimal.

Theorem 21. *A necessary and sifficient condition for a measurable policy $\pi(\cdot)$ to be optimal is that for almost all $t \in [0,T]$,*

$$r(d) + Q(d)\psi(d) \tag{5.3}$$

is maximized over the set D by $\pi(t)$ where the column vector $\psi(t)$ is the unique absolutely continuous solution to

$$-\frac{d}{dt}\psi(t) = r(\pi(t)) + Q(\pi(t))\psi(t) \quad \textit{where} \quad \psi(T) = 0, 0 \leq t \leq T. \tag{5.4}$$

Proof. Let π' be any measurable policy. In the following we distinguish between P,Q, and r for π and π' by writing P, Q, r, and P', Q', and r' respectively. We will now establish the equation

$$v(T,\pi) - v(T,\pi') = \int_0^T P'[r + Q\psi - r' - Q'\psi]dt. \tag{5.5}$$

To see this we note first that since $P(0) = P'(0) = I$ and $\psi(T) = 0$ we have

$$[P(T) - P'(T)]\psi(T) - [P(0) - P'(0)]\psi(0) = 0$$

.
Since $P'(\cdot)$,$P(\cdot)$ and $\psi(\cdot)$ are absolutely continuous the function $[(P(\cdot){-}P'(\cdot))\psi(\cdot)]$ is absolutely continuous [7, p. 203, theorem 25] and therefore this function equals the integral of its derivative. Hence

$$0 = \int_0^T \frac{d}{dt}[(P-P')\psi]dt = \int_0^T [\frac{d}{dt}(P-P')\psi + (P-P')\frac{d}{dt}\psi]dt$$

.
By substituting (5.2) and (5.4) into this formula we get

$$\int_0^t [(PQ - P'Q')\psi]dt + (P-P')(-(r+Q\psi))]dt$$

$$= \int_0^T [(P'-P)r + P'(Q-Q')\psi]dt.$$

Using this fact it follows that

$$v(\pi,T) - v(\pi',T) = \int_0^T [Pr - P'r']dt = \int_0^T P'[r + Q\psi - r' - Q'\psi]$$

which proves (5.5). Now if π maximizes (5.3) almost everywhere the integrand of (5.5) is nonnegative a.e. so that π is optimal. The necessity also follows from (5.5). Let π' be a policy which maximizes (5.3) everywhere and assume that π does not. Then π' isn't necessarily measurable since for example if two elements of D both maximize (5.3) over the same set of positive measure then each element might be chosen on a nonmeasurable subset.
In order to exhibit a measurable policy π' maximizing (5.3) everywhere it is convenient to enumerate the elements of D as $d(1),\ldots,d(N)$. For a fixed $d \in D, (r(d) + Q(d)\psi(t))$ is a continuous and hence measurable function of time so that the sets $T_i' = \{t : d(i)$ maximizes $(r(d) + Q(d)\psi(t))$ over $d \in D\}$ are measurable. If we define the mutually exclusive sets T_i by $T_i = T_i' \ \cup_{j=1}^{i-1} T_i'$ then the policy π' defined by $\pi'(t) = \{f(i) : t \in T_i\}$ is both measurable and maximizes (5.3) everywhere. For this policy π' the integrand of (5.5) is nonpositive and strictly negative on a set of positive measure since $P' \geq 0$ and by lemma 9 the diagonal elements are strictly positive. It is a well known result of analysis that if a nonpositive function on a set is negative on a subset of positive measure the the integral over the set is negative. Therefore the return using π' is higher than the return from using π which establishes the necessity and completes the proof. □

The condition (5.3) be maximized a.e. is equivalent to

$$-\frac{d}{dt}\psi = \max_{d\in D}(r(d) + Q(d)\psi(t)), \text{ where } \psi(T) = 0. \tag{5.6}$$

This equation is called the Bellman equation.

We mentioned before that we will show that the utility functions $\psi(t)$ play the same role as our old utility functions u_t that denoted the expected total reward from decision epoch t and onwards. Using some results in Coddington and Levinson [4] we will now show that $\psi(t)$ is the expected reward from time t onwards as well.

Theorem 22. *For any measurable policy π,*

$$\psi(t) = \int_t^T P(t,s)r(\pi(s))ds. \tag{5.7}$$

Proof. From Coddington and Levington, of [4, theorem 3.1] the solution to (5.4) is

$$\psi(t) = \Phi(T,t)\Phi^{-1}(T,s)(-r(\pi(s)))ds \tag{5.8}$$

where $\Phi(T,\cdot)$ is the fundamental matrix $\Phi(\cdot)$ of the differential equations $\frac{dx}{dt} = -Q(\pi(\cdot))x$ with $\Phi(T) = I$. This is the adjoint system of the differential equations $\frac{dx}{dt} = Q(\pi(\cdot))^*x$ whose solution [4, p.70] is Φ^{*-1}.
From theorem (7) we have the result that the fundamental matrix solution to $\frac{dx^*}{dt} = x^*Q(\pi(t))$ is $P(\cdot,\cdot)$ so that $\Phi^{-1}(t,\cdot) = \Phi^{**-1}(T,\cdot) = P(T,\cdot)$. Substituting this result and changing the limits of integration gives us

$$\psi(t) = P(t,T)\int_t^T P(T,s)r(\pi(s))ds$$
$$= \int_t^T P(t,s)r(\pi(s))ds$$

.

□

From this theorem we have the interpretation of $\psi_i(t)$ as the expected return that will be obtained on the interval $[t,T]$ when policy π is used and the system is in state i at time t.

5.2 Piecewise constant optimal policy

We now arrive at the most mathematically interesting part, the proof that a piecewise constant policy is optimal. The proof is constructive, which will be of help later when we introduce the algorithm for solving the CTMDP. The proof requires quite a lot of buildup, seen in lemma 23 which we will use for lemma 24 and 25 before we come to our conclusion in theorem 26. Before we get to that part, we will start with some results from Coddington and Levinson [4] for linear differential equations with constant coefficients. Define a vector function $v(t)$ by the following differential equations:

$$v(T) = c \tag{5.9}$$

$$\frac{d}{dt}v(t) = r + Qv(t)$$

where

$$v \text{ is an } n \times 1 \text{ vector,}$$
$$Q \text{ is an } n \times n \text{ matrix,}$$
$$r \text{ is an } n \times 1 \text{ vector,}$$
$$\text{and } t \in [0, T].$$

Then

$$v(t) = ce^{Q(T-t)} + rf(Q, T-t) \text{ for } 0 \leq t \leq T \tag{5.10}$$

where

$$f(Q,t) = t + \frac{t^2}{2!}Q + \frac{t^3}{3!}Q^2 + \dots$$

satisfies (5.9) everywhere. Equation (5.10) can be confirmed by direct differentiation.

We observe that $v(t)$ is infinitely differentiable on $(0, T)$. The notation $v^{(j)}(t)$ will be used to represent the j^{th} derivative of the vector $v(t)$. If the left and right hand derivatives are unequal we say $v^{(j)}(t)$ equals the left hand derivative. It will sometimes be convenient to let $v^0(t) = v(t)$. The derivatives of $v^{(j)}(t)$ are

$$-v^{(1)}(t) = r + Qv(t) \text{ and} \tag{5.11}$$

$$v^{(j)}(t) = -Qv^{(j-1)}(t) \text{ for } j = 2, 3, \dots.$$

we can write $v(t)$ in terms of the derivatives at $t = T$ by direct substitution of (5.11) in to (5.10). We have

$$v(t) = v(T) + \sum_{m=1}^{\infty} v^{(m)}(T)\frac{(t-T)^m}{m!}. \tag{5.12}$$

Lemma 23. *Let $v(t)$ and $v'(t)$ be two $n * 1$ vectors defined by the following differential equations and terminal conditions*

$$v(T) = c \qquad v'(T) = c$$

$$-\frac{d}{dt}v(t) = r + Qv(t), -\frac{d}{dt}v'(t) = r' + Q'v(t)$$

where

$$c \textit{ is an } n \times 1 \textit{ vector}$$

$$Q \textit{ and } Q' \textit{ are } n \times n \textit{ matrices}$$

$$r \textit{ and } r' \textit{ are } n \times 1 \textit{ vectors}$$

$$t \in [0, T].$$

Then if $v^{(m)}(T) = v'^{(m)}(T)$ for $m = 1, 2, \ldots, n+1, v^{(i)}(T) = v'^{(i)}(T)$ for all positive integers i and $v(t) = v'(t)$ for $t \in [0, T]$.

Proof. The n-dimensional vectors, $v^{(i)}, i = 1, 2, \ldots, n+1$, must be linearly dependent. This implies that for some integer $j, 2 \leq j \leq n+1$,

$$v^{(j)}(T) = \sum_{k=1}^{j-1} d_k v^{(k)}(T) = v'^{(j)}(T) = \sum_{k=1}^{j-1} d_k v'^{(k)}.$$

We show by induction that for all $i \leq j$

$$v^{(i)}(T) = \sum_{k=i-j+1}^{i-1} d_{k+j-1} v'^{(k)}(T).$$

$$= v'^{(i)}(T) = \sum_{k=i-j+1}^{i-1} d_{k+j-i} v'^{(k)}(T).$$

Our equations hold for $i = j$ from the above equations. Now we assume they hold for $i = j, j+1, \ldots, m-1$, and show they hold for $i = m$.

$$v^{(m)}(T) = -Qv^{(m-1)}(T)$$

$$= -\sum_{k=m-j}^{m-2} d_{k+j-m+1} Qv^{(k)}(T)$$

$$= \sum_{k=m-j+1}^{m-1} d_{k+j-mv^{(k)}}(T).$$

In the same way $v'^{(m)}(T) = \sum_{k=m-j+1}^{m-1} d_{k+j-m} v'^{(k)}(T)$. Since $v^{(i)}(T) = v'^{(i)}(T)$ for $i = 1, 2, \ldots, j-1$, by the hyposthesis of the lemma, and for $i = j, j+1, \ldots m-1$, by the induction hypothesis, $v^{(m)}(T) = v'^{(m)}(T)$. From representation (5.12) this result inplies $v'(t) = v(t)$ for all $t \in [0, T]$.

□

A method for choosing a policy will now be described. It will be shown in theorem 26 that there is an optimal piecewise constant policy for the finite-horizon problem based on this method of choosing a policy.

The intuitive idea of this choice rule is that for each point in time we pick the set of actions which maximizes the first derivative of the function ψ . If there is a tie we break the tie by considering the second derivative etc. Lemma 23 is important because it says we need only consider the first $n+1$ derivatives.

Given a measurable policy π defined on $[0,T]$ and the corresponding function $\psi(\cdot)$ defined by (5.4) we can define the $n+1$ sets,

$$\begin{aligned} D_1(t) &= \{d : d \in D_0(t) = D, \quad \text{d maximizes } \psi^{(1)}(t,d)\}, \\ D_2(t) &= \{d : d \in D_1(t), \quad \text{d maximizes } -\psi^{(2)}(t,d)\}, \\ &\vdots \\ D_{n+1}(t) &= \{d : d \in D_n(t), \quad \text{d maximizes } (-1)^n\psi^{(n+1)}(t,d)\}, \end{aligned}$$

where

$$\begin{gathered} \psi^{(1)}(t,d) = r(d) + Q(d)\psi(t) \\ \psi^{(j)}(t,d) = Q(d)\psi^{(j-1)}(t) \text{ for } 2 \le j \le n+1, \\ \text{and } \psi^{(j-1)}(t,d) = \psi^{(j-1)}(t,d) \text{ for any } d \in D_{j-1}(t). \end{gathered}$$

In order to insure uniqueness for our selection procedure it is necessary to enumerate the finite set D as

$$d(1), d(2), \ldots, d(N). \tag{5.13}$$

we say that d satisfies the selection procedure based on the measurable policy π at time t if d is the element in $D_{n+1}(t)$ with the lowest index according to the enumeration (5.13).

Lemma 24. *Consider the arbitrary measurable policy π defined on the interval $(t',T]$. The vector function ψ is defined on $[t',T]$ (the interval is closed because ψ is continuous). Let $d*$ be the vector of actions picked by the selection procedure based on the policy π at time t' and set $\pi(t)$ equal to $d*$ on some interval $[t'',t'], t'' < t'$. The vector function ψ is also now defined on $[t'',t')$. The $\pi(t)$ satisfies the selective procedure on the entire interval $[t'-\epsilon, t']$ for some ϵ where $0 < \epsilon < t'-t''$.*

Proof. The first half of the proof consists of showing that if $d' \in D \setminus D_{n+1}(t')$, then

$$d' \notin D_1(t) \text{ for any } t \in (t'-\epsilon(d'), t'), 0 < \epsilon(d') < t' - t''$$

and hence is not chosen by the selective procedure on that interval.

Since a constant policy is used for $t'' < t < t'$ the vector function $\psi(\cdot)$is of the form (5.10) where $T = t', c = \psi(t'), Q = Q(d*)$ and $r = r(d*)$ and therefore is infinitely defferentiable.

For any $d \in D, t'' \le t \le t'$, we can write for any l the Taylor's expansion of $(r(d) + Q(d)\psi(t))$

$$r(d) + Q(d)\psi(t) = r(d) + Q(d)\psi(t') + \sum_{k=1}^{l} Q(d)\psi^{(k)}(t')\frac{(t-t')^k}{k!} + Q(d)\psi^{(l+1)}(t_d)\frac{(t-t')^{l+1}}{(l+1)!} \tag{5.14}$$

where $t \le t_d \le t'$. Let l be the largest integer such that $d' \in D_l(t')(0 \le l \le n,$ since $d' \in D \setminus D_{n+1}(t))$.

The value of

$$\{\sum_{k=1}^{l-1} Q(d)\psi^{(k)}(t')\frac{(t-t')^k}{k!} + r(d) + Q(d)\psi(t')(\text{ if } l < 0)\}$$

is equal for d' and $d*$ since both are element of the sets $D_0(t'), D_1(t'), \ldots, D_l(t')$. Since $d' \in D_{l+1}(t')$ and $d* \in D_{l+1}(t')$, the value of

$$\begin{aligned} &r(d) + Q(d)\psi(t') \quad l = 0 \\ &(-1)^l Q(d)\psi^{(l)}(t') \quad l > 0 \end{aligned}$$

is strictly greater in some coordinates for $d = d*$ than $d = d'$ and we let this vector difference be $\delta d'$. The vector function $\psi^{(l)}(\cdot)$ is uniformly bounded in $t, t'' \le t \le t'$, for fixed l so that there is an $\epsilon(d') > 0$ such that $\frac{\delta d'}{(-1)^l}\frac{(t-t')^l}{l!}$ is strictly greater for all $t, t' - \epsilon(d') \le t \le t'$, in some coordinate than

$$\{Q(d')\psi^{(l+1)}(t_{d'}) - Q(d*)\psi^{(l+1)}(t_{d*})\}\frac{(t-t')^{l+1}}{(l+1)!}$$

where $t \le t_{d'}, t_{d*} \le t'$.
This implies using the representation (5.14) that $r(d*) + Q(d*)\psi(t)$ is greater in some coordinate than $r(d') + Q(d')\psi(t)$ for all $t \in (t' - \epsilon(d'), t')$ which proves $d' \notin D_1(t)$ for $t \in (t' - \epsilon(d'), t')$. The ϵ of our lemma is $\min_{d' \in D \ D_{n+1}(t')}\{\epsilon(d')\}$ which is strictly positive since D is finite.

We now consider $d' \in D_{n+1}(t'), d' \ne d*$, and the differential equations

$$\frac{d}{dt}\psi(t) = -Q(d*)\psi(t) - r(d*) \text{ for } t'' \le t \le t' \text{ and}$$

$$\psi'(t') = \psi(t')$$

$$\frac{d}{dt}\psi'(t') = -Q(d')\psi'(t) - r(d'), \text{ for } t'' \le t \le t'.$$

Since both d' and $d* \in D_{n+1}(t')$, the first $n+1$ derivatives of ψ' and ψ are equal at t' and lemma 23 applies so that $\psi'(t) = \psi(t)$ for $t'' \le t \le t'$.

This implies that for all $t \in (t'', t')$, $Q(d*)\psi(t) + r(d*) = \psi^{(1)}(t) = \psi'^{(1)}(t) = Q(d')\psi'(t) + r(d')$, and for $1 \le k \le n, -Q(d*)\psi^{(k)}(t) = \psi^{(k+1)}(t) = -Q(d')\psi'^{(k)}(t) = -Q(d')\psi^{(k)}(t)$. But this means the set $d_{n+1}(t)$ is constant for $t \in (t' - \epsilon, t')$. Therefore $d*$ satisfies the selective procedure based on π over the interval $(t' - \epsilon, t')$ which proves the lemma. □

We now introduce a revised selective procedure in order to establish lemma 25. This revised selective procedure is used only in the proof this lemma and the original procedure will be continued to be called the selective procedure.

Given a measurable policy π defined on $[0, T]$ and the corresponding functions $\psi(\cdot)$ defined by (5.4) we can define the $n+1$ sets

$$\hat{D}_1(t) = \{d : d \in D, \quad \text{d maximizes } \hat{\psi}^{(1)}(t,d)\},$$
$$\hat{D}_2(t) = \{d : d \in \hat{D}_1(t), \quad \text{d maximizes } -\hat{\psi}^{(2)}(t,d)\},$$
$$\vdots$$
$$\hat{D}_{n+1}(t) = \{d : d \in \hat{D}_n(t), \quad \text{d maximizes } (-1)^n \hat{\psi}^{(n+1)}(t,d)\},$$

where $\hat{\psi}^{(1)}(t,d) = r(d) + Q(d)\psi(t), \hat{\psi}^{(j)}(t,f) = Q(f)\psi^{(j-1)}(t), 2 \leq j \leq n+1$, and $\hat{\psi}^{(j-1)}(t) = \hat{\psi}^{(j-1)}(t,d)$ for any $d \in \hat{D}_{j-1}(t)$. We enumerate the set D as before by (5.13). We say that d satisfies the revised selective procedure based on the measurable policy π at time t is d is the element in $\hat{D}_{n+1}(t)$ with the lowest index according to enumeration (5.13).

Lemma 25. *If the policy π satisfies the selective procedure corresponding to the policy π everywhere on $(t', T], t' < T$, then π must be constant on the interval $(t', t'+\epsilon)$ for some $\epsilon > 0$.*

Proof. We will establish lemma 25 by exhibiting the $d \in D$ such that $\pi(t) = d$ on the interval $(t', t'+\epsilon)$ for some $\epsilon > 0$. Since $\psi(\cdot)$ is continuous it is defined also at t', and we let $d*$ be the unique element of D chosen by the revised selection procedure at time t'. Now consider the vector function $\psi'(\cdot)$ defined by the differential equations

$$\psi'(t') = \psi(t')$$
$$-\frac{d}{dt}\psi'(t) = r(d*) + Q(d*)\psi'(t)$$

for $t' \leq t \leq T$.

We now show $d*$ satisfies both the revised and original selective procedure based on the vector function $\psi'(\cdot)$ on the open interval $(t', t'+\epsilon)$ for some $\epsilon > 0$ (even though d* does not necessarily satisfy the selective procedure (original) at time t'). Using the same argument as in lemma 24, if $d \notin \hat{D}_{n+1}(t')$ then $d \notin \hat{D}_1(t)$ for $t \in (t', t'+\epsilon)$ and some $\epsilon > 0$. Also if $d \in \hat{D}_{n+1}(t')$ then $d \in \hat{D}_{n+1}(t)$ for $t \in (t', t'+\epsilon)$ so that $d*$ satisfies the revised selective procedure on this interval. Also for $t \in (t', t'+\epsilon)$ either $f \notin \hat{D}_1(t)$ or $d \in \hat{D}_{n+1}(t)$ so that $\hat{D}_1(t) = \hat{D}_2(t) = \cdots = \hat{D}_{n+1}(t)$.

We now prove by induction on i that $\hat{D}_i(t) = D_i(t)$ for $1 \leq i \leq n+1$ and $t \in (t', t'+\epsilon)$. $\hat{D}_1(t) = D_1(t)$ since both sets are defined identically. Now we assume $\hat{D}_i(t) = D_i(t)$ for $i = 1, 2, \ldots, l-1$ and show it holds for $i = l$. The relation $\psi^{(1)}(\cdot) = \hat{\psi}^{(1)}(\cdot)$ on $(t', t'+\epsilon)$ implies $\psi^{(l-1)}(\cdot) = \hat{\psi}^{(l-1)}(\cdot)$ on $(t', t'+\epsilon)$. Since $\hat{D}_l(t)\hat{D}_{l-1}(t)$, the value of $\{Q(d)\hat{\psi}^{(l-1)}(t)\}$ must be the same for all $d \in \hat{D}_{l-1}(t)$ which implies the value of $\{(-1)^{l-1}Q(d)\hat{\psi}^{(l-1)}(t)\}$ must be the same for all $d \in \hat{D}_{l-1}(t)$. However sine $D_{l-1}(t) = \hat{D}_{l-1}(t)$ and $\psi^{(l-1)}(t) = \hat{\psi}^{(l-1)}(t)$ which means $D_l(t) = D_{l-1}(t) = \hat{D}_{l-1}(t) = \hat{D}_l(t)$. Therefore we can conclude that $D_{n+1}(t) = \hat{D}_{n+1}(t)$ for $t \in (t', t'+\epsilon)$ and since $d*$ satisfies the revised selective procedure, it satisfies the selective procedure (original) on that interval. We will have shown that $\pi(t) = d*$ on $(t', t'+\epsilon)$ if we can prove that $\psi'(\cdot) = \psi(\cdot)$ on this interval because the selective procedure is unique.

Since $d* \in \hat{D}_1(t)$ for $t \in (t', t'+\epsilon)$, the differential equations defining $\psi'(\cdot)$ can be written

$$\psi'(t') = \psi(t')$$
$$\frac{d}{dt}\psi'(t) = \max_{d \in D}(r(d) + Q(d)\psi'(t))$$

for $t \in (t', t'+\epsilon)$. But by hopothesis π satisfies the original selective procedure on $(t'T)$ so that $\pi(t) \in F_1(t)$ ($F_1(t)$ being based on $\psi(\cdot)$), and $\psi(\cdot)$ also satisfies the above differential equations. The uniqueness of the solution of these differential equations [8, p.321, theorem 1] implies $\psi = \psi'$on $(t', t'+\epsilon)$ and completes the proof. □

Theorem 26. *There is a piecewise constant (from the left) policy π defined on $[0, T]$ which maximizes equation (5.3) everywhere. This policy is optimal.*

Proof. The proof is by construction. Consider the following algorithm which goes through steps (1-5) consecutively.

(1) Initialization; set $t' = T$ and $\psi(T) = 0$.

(2) Use the selective procedure based on $\psi(t')$ to determine $\pi(t')$.

(3) Obtain $\psi(t)$ for $0 \le t \le t'$ by solving the differential equations

$$-\frac{d}{dt}\psi(t) = r(\pi(t')) + Q(\pi(t'))\psi(t)$$

using the previous value of $\psi(t')$ as te terminal condition.

(4) Set $t'' = \inf\{t : \pi(t')$ satisfies the selective procedure on the interval (t, t') based on the vector function $\psi(t)\}$

(5) If $t'' \le 0$ terminate; if $t'' > 0$ go to step (2) with $t' = t''$

Because of condition (24) the policy π satisfies the selective procedure corresponding to π everywhere and hence always lies in the set $D_1(t)$ for all t. This condition is equivalent to maximizing equation (5.3) for all t so that π is optimal where it is defined. It remains to be shown that there are a finite number of switches when this algorithm is used, so that the algorithm goes through equations (21-25) a finite number of times. From equation (24) and lemma 24 we note that the points t'_i, corresponding to the value of t' in the algorithm at the i^{th} iteration, are strictly decreasing. Suppose the algorithm is not finite. Let $t* = \inf\{t_i\}$. Then the policy $\pi(\cdot)$ defined by the algorithm on the half open interval $(t*, T]$ satisfies the hypotheses of lemma 25. Thus there is an $\epsilon > 0$ such that $\pi(\cdot)$ is constant on $(t*, t*+\epsilon)$, which contradicts the fact that infinitely many t_i lie in $(t*, t*+\epsilon)$. Thus the algorithm must terminate in finitely many steps which completes the proof. □

5.3 the discrete approximation

To summarize, we have now shown there exists a piecewise constant optimal policy and have constructed an algorithm in order to find it. You could say that we are done at this point, but there is still one problem for us to solve and that comes with step (24) from the algorithm

$$\text{Set } t'' = \inf\{t : \pi(t') \text{ satisfies the selective procedure}$$

$$\text{on the interval } (t, t') \text{ based on the vector function } \psi(t)\}$$

Finding the infimum of the right hand side of (24) is equivalent to the problem of finding the first zero crossing of a function which is a linear combination of the utility functions of the optimal policies. These utility functions are the solution of linear differential equations and are the sums of exponentials and linear terms. Therefore, in general, it is possible to only converge to t' in equation (24) and we suspect we must be content with infinite algorithms which converge to an optimal policy or finite algorithms which are $\epsilon-$optimal.

Definition 27. *A measurable policy π is ϵ-optimal if $v(T,\pi)+\epsilon 1 \geq v(T,\pi*)$ where 1 is a vector and $\pi*$ is an optimal policy.*

An approach suggested by Howard [9, p.124] for solving continuous-time finite-horizon Markov decision problems is to approximate the continuous problem by the discrete-time variant and solve that problem using dynamic programming.

We implement this suggestion by setting a grid $\Delta = \frac{T}{m}$ over the time horizon as in the continuou-time algorithm approach. For each $d \in D$ we have a return vector $r(d)\Delta$ and a transition matrix $I + Q(d)\Delta$. For this approach to be meaningful Δ must be small enough that $I + Q(d)\Delta$ has only positive elements. The periods are numbered $1, 2, \ldots, m$, where period j occurs during the time interval $((j-1)\Delta, j\Delta)$. Our problem is to find a policy $\{d(1), d(2), \ldots, d(m)\}$ which maximizes the vector

$$v(m) = \sum_{i=1}^{m} P(i) r(f(i)) \tag{5.15}$$

where

$$P(i) = \prod_{j=1}^{i-1} [I + Q(d(j))\Delta] \text{ for } i > 1 \tag{5.16}$$

$$P(1) = I$$

To solve this problem using dynamic programming we let $v_j(k,m)$ be defined as the maximum expected return that can be attained over the periods $k, k+1, \ldots, m$, when the system is in state j at period k. From the priciple of optimality

$$v(m,m) = \max_{d \in D} r(d)\Delta \qquad \text{and} \tag{5.17}$$

$$v(k,m) = \max_{d \in D}(r(d)\Delta + [I + Q(d)\Delta]v(k+1,m)) \text{ for } 1 \leq k < m.$$

The m vectors $\{d(1), d(2), \ldots, d(m)\}$ which maximize $v(1,m), v(2,m), \ldots, v(m,m)$, repectively are the solution to our problem. A continuous policy is obtained from solving the discrete problem by setting $\pi(t,m)$ equal to $d(k)$ if $(k-1)\Delta \leq t < k\Delta$.

The equations (5.17) can be rewritten as

$$v(m+1,m) = 0$$

$$v(k,m) = v^{(1)}(k+1,m)\Delta + v(k+1,m) \text{ where} \tag{5.18}$$

$$v^{(1)}(k+1,m) = \max_{d \in D}[r(d) + Q(d)v(k+1,m)] \text{ and } 1 \leq k \leq m.$$

When the system is actually continuous $v(1,m)$ does not represent the return that will be achieved using policy $\pi'(\cdot,m)$ but only an approximation to it. From equation (5.12) the true value of expected return using policy $\pi'(\cdot,m)$, $v'(1,m)$, is given by

$$v'(m+1,m)=0$$

$$v'(k,m)=\sum_{i=1}^{\infty} v'^{(i)}(k+1)\frac{\Delta^i}{i!}+v'(k+1,m) \tag{5.19}$$

$$\text{where } v'^{(1)}(k+1)=\max_{d\in D}[r(d)+Q(d)v'(k+1,m)]$$

$$\text{and } v'^{(j)}(k+1)=Q(d)v'^{(j-1)}(k+1) \qquad j>1.$$

In equation (5.12) the minus sign before Q cancels with the sign of $(t-T)$.

We will now show that the discrete approximation does indeed approach the optimal policy as the grid size goes to zero. We will obtain this result in theorem (reftheorem 6 of chapter IV). The plan is to prove that $v(j,m)\to v'(j,m)$ uniformly in j as m tends to infinity and then use this result to obtain the final result. Now we first establish the following results.

Theorem 28. *Let π be a measurable policy for the finite-horizon problem and $\delta(\cdot)$ be an integrable function defined on $[0,T]$ such that*

$$r(\pi(t))+Q(\pi(t))\psi(t)+\delta(t)1\geq max_{d\in D}\{r(d)+Q(d)\psi(t)\}\ a.e.$$

on $[0,T]$ where the vector function $\psi(\cdot)$ is based on the policy π and is the solution of the differential equations (5.4) and 1 is a vector. The $v(T,\pi)+\int_0^T\delta(t)1dt\geq v(T,\pi*)$ where $\pi*$ is an optimal policy.

Proof. The proof is based on (5.5)

$$v(T,\pi)-v'(T,\pi')=\int_0^T P'[r+Q\psi-r'-Q'\psi]dt$$

If we let $\pi*=\pi'$ and substitute the inequailty of the hypothesis we obtain

$$v(T,\pi)-v(T,\pi*)\leq\int_0^T P*[\delta(t)1]dt=\int_0^T\delta(t)1dt$$

which proves the theorem. □

Lemma 29. *Suppose for $1\leq j\leq m$ that*

$$h(j)-h'(j)=M(j)(h(j+1)-h'(j+1))+\sum_{i=2}^{\infty}N(j)^{i-1}(Q(j)h'(j+1)+g(j))c^i \tag{5.20}$$

where $h(\cdot),h'(\cdot)$ and $g(\cdot)$ are $n\times 1$ vectors, $M(\cdot),N(\cdot)$, and $Q(\cdot)$ are $n\times n$ matrices and c is a scalar. Then if $h(m+1)=h'(m+1)$,

$$h(j)-h'(j)=\sum_{k=j}^{m}(\prod_{l=j}^{k-1}M(l))T(k)$$

where $T(k)=\sum_{i=2}^{\infty}N(j)^{i-1}(Q(j)h'(j+1)+g(j))c^i$and $\prod_{l=j}^{j-1}M(l)=I$.

Proof. The roof is by induction. The lemma holds for $j = m$ by direct substitution. We now assume it holds for $j = s+1, s+2, \ldots, m$, and shows it holds for $j = s$. Using the induction hypothesis equations (5.20) becomes

$$\begin{aligned}
h(s) - h'(s) &= M(s)[\sum_{k=s+1}^{m}(\prod_{l=s+1}^{k-1} M(l))T(k)] + T(s)\\
&= \sum_{k=s+1}^{m}(\prod_{l=s}^{k-1} M(l))T(k) + (\prod_{l=s}^{s-1} M(l))T(s)\\
&= \sum_{k=s}^{m}(\prod_{l=s}^{k-1} M(l))T(k)
\end{aligned}$$

which proves the lemma. □

Lemma 30. *For any $n \times n$ matrix A the matrices $A_m = \sum_{i=0}^{m}\binom{m}{i}(\frac{1}{m}A)^i$, $m = 1, 2, \ldots$ are bounded.*

Proof. Let $a*$ be the largest element of the matrix A. Then it is a simple matter to prove by induction that any element of the matrix A^k is bounded by $(na*)^k$ which implies that each element of the matrix A_m is not greater than $\sum_{i=0}^{m}\binom{m}{i}(\frac{na*}{m})^i$. From the binomial expansion law, for $m > na*$ we have

$$\sum_{i=o}^{m}\binom{m}{i}(\frac{na*}{m})^i(1-\frac{na*}{m})^{m-1} = na*$$

which implies

$$\sum_{i=o}^{m}\binom{m}{i}(\frac{na*}{m})^i \le na*(1+\frac{na*}{m-na*})^m$$

which proves the lemma since $\lim_{m\to\infty}(1+\frac{na*}{m})^m = e^{na*}$.

□

Lemma 31. *The vectors $v(j,m)$, $j = 1, 2, \ldots, m$ defined by equations (5.18) converge uniformly in j to the vectors $v'(j,m), j = 1, 2, \ldots, m$, defined by equations (5.19) as m tends to infinity.*

Proof. From equations (5.18) and (5.19)

$$\begin{aligned}
v(j,m) - v'(j,m) &= v(j+1,m) + [r(d(j)) + Q(d(j))v(j+1,m)]\frac{T}{m} - v'(j+1,m)\\
&- \sum_{i=1}^{\infty}[Q(d(j))]^{i-1}[r(d(j)) + Q(d(j))v'(j+1,m)][\frac{T}{m}]^i, \text{ for } j = 1, 2, \ldots, m,
\end{aligned}$$

where $v'(m+1,m) = v(m+1,m) = 0$ and $d(1), d(2), \ldots, d(m)$, is the policy obtained using the discrete algorithm for periods $1, 2, \ldots, m$.

By letting $M(j) = I + Q(d(j))\frac{T}{m}, Q(j), N(j) = Q(d(j)), h(j) = v(j), h'(j) = v'(j), g(j) = r(d(j))$ and $c = \frac{T}{m}$ we satisfy the hypothesis of lemma 29 and

$$\begin{aligned}
v(j,m) - v'(j,m) = \sum_{l=j}^{m}(\prod_{l=j}^{k-1}(I + Q(d(l))\frac{T}{m})(\sum_{i=2}^{\infty}[Q(d(k))]^{i-1}\\
[Q(d(k))v'(k+1,m) + r(d(k))][\frac{T}{m}]^i))
\end{aligned}$$

We let $Q*$ be a matrix such that $q_{ij}*$ is an upper bound on $|q_{ij}(d)|$ for $d \in D$ and $r*$ be a vector such that r_i* is an upper bound on $|r_i(d)|$ for $d \in D$ and let r be the upper bound on the absolute value of elements $r_i(d)$ for all i and $d \in D$. From the binomial expansion $\prod_{l=j}^{k-1}(I + Q(d(j))\frac{t}{m}) \leq \sum_{i=0}^{k-j} \binom{k-j}{i}(\frac{Q*T}{m})^i$ and theorem 22 implies $v'(k+1, m)$ is bounded by $rT1$ where 1 is a vector. Therefor

$$||v(j,m) - v'(j,m)|| \leq (m+1-j)\frac{T^2}{m^2}||(\sum_{i=0}^{m}\binom{m}{i}(\frac{Q*T}{m})^i)(Q*\sum_{i=2}^{\infty}(\frac{Q*T}{m})^{i-2})(Q*rT1 + r*)||$$

where $||\cdot||$ is the absolute value now defined by $||x|| = \sum_{j=1}^{n}|x_j|$. The expression $\sum_{i=0}^{m}\binom{m}{i}(\frac{Q*T}{m})^i$ converges from lemma 30 and it is well known that $\sum_{i=2}^{\infty}(\frac{Q*T}{m})^{i-2}$ converges for m sufficiently large which proves the lemma.

□

Theorem 32. *Let $\epsilon_m = inf\{\epsilon : \pi'(\cdot, m)$ is $\epsilon - optimal\}$ where $\pi'(\cdot)$ is obtained using the discrete-approximation aproach. Then $lim_{\epsilon_m} = 0$.*

Proof. The proof consists of showing that for each policy $\pi'(\cdot, m)$ there is a function $\delta'_m(\cdot)$ which satisfies the hypothesis of theorem 28 which is bounded by K'_m, and that $K'_m \to 0$ which proves the theorem since T is fixed. The policy $\pi'(\cdot, m)$ obtained from the discrete approximation satisfies

$$r(\pi'(\frac{kT}{m}, m)) + Q(\pi'(\frac{kT}{m}, m))v(\frac{kT}{m}, m) = max_{d\in D}\{r(d) + Q(d)v(\frac{kT}{m}, m)\} \text{ for } k = 1, 2, \ldots, m.$$

This equality implies that for $t \in (0, \frac{1}{m}), k = 1, 2, \ldots, m,$

$$\begin{aligned}
&r(\pi'(\frac{kT}{m}, m)) + Q(\pi'(\frac{kT}{m}, m))v'(\frac{kT}{m} - t, m)\\
&+\{Q(\pi'(\frac{kT}{m}, m))[v'(\frac{kT}{m}, m) - v'(\frac{kT}{m} - t, m)]\\
&+Q(d')[v'(\frac{kT}{m}, m) - v'(\frac{kT}{m}, m)]\}\\
&+\{Q(\pi'(\frac{kT}{m}, m))[v(\frac{kT}{m}) - v'(\frac{kT}{m}, m)]\\
&+Q(d')[v'(\frac{kT}{m}, m) - v(\frac{kT}{m}, m)]\} \geq r(d') + Q(d')v'(\frac{kT}{m} - t, m)
\end{aligned}$$

d' maximizes $r(d) + Q(d)v'(\frac{kT}{m} - t, m)$ over $d \in D$. We shall now prove that the first expression in braces is bounded. Define $Q*$, $r*$ and r as in lemma 31. Then $\frac{d}{dt}\psi(\cdot)$ is bounded by $(r * + Q * rT1)$ so the expression in braces is bounded by $2Q * (r * + Q * rT1)t$. This proves the expression is bounded since t is bounded by $\frac{T}{m}$. The second expression in braces goes to zero as m gets large from lemma 31 which proves the theorem.

□

Chapter 6

Worked out example

At the end we will look at one example so we can see all the theorems come together. We will solve the problem in two ways, with the exact method and the approximate method, so we can see the power and weaknesses of the approximation as well.
We consider a Continuous-Time MDP with $T = 0$ and two system states. In the first state we have two actions:

Action $a_{1,1}$	Action $a_{1,2}$
$r_{a_{1,1}} = 3$	$r_{a_{1,2}} = 10$
$q(1\|1, a_{1,1}) = -2,\ q(2\|1, a_{1,1}) = 2$	$q(1\|1, a_{1,1}) = -10,\ q(2\|1, a_{1,1}) = 10$

and in the second state we only have one action:

$$\begin{gathered}\text{Action } a_{2,1}\\ r_{a_{2,1}} = 0\\ q(1|1, a_{2,1}) = 1,\ q(2|1, a_{2,1}) = -1\end{gathered}$$

We will now analyse this problem for a bit before continuing with the actual algorithm to solve this problem. We can see that there is only one action in state 2. That means that whenever we are in state 2, we know we have a return rate of 0 and a transition rate of 1.
State 2 has two actions, so we must analyze the difference between the two. Action $a_{1,1}$ has a return rate of 3, and the transition rate of going to state two is 2. When selecting this action we stay $\frac{1}{2}$ units of time in state 1 before going to state 2. There we stay 1 unit of time with a payoff of 0. Approximately this gives us a payoff $\frac{1}{2} * 3 + 0 * 1 = 1.5$ in 1.5 units of time.
Action $a_{2,1}$ has a return rate of 10, and the transition rate of going to state two is 10. When selecting this action we stay $\frac{1}{10}$ units of time in state 1 before going to state 2. There we stay 1 unit of time with a payoff of 0. Approximately this gives us a payoff $\frac{1}{10} * 10 + 0 * 1 = 1$ in 1.1 units of time.
This tells us that we expect action $a_{1,1}$ is preferable over action $a_{2,1}$ when we have more than 1.5 units of time left. However, at then end of the process the higher payoff of action $a_{2,1}$ must become more relevant. We can now use the algorithm to find out when this transition occurs. We first look at the exact algorithm.

At time 10 the selective procedure tells us to maximize $r(d)$, which is done by choosing $d = (2, 1)$. $\psi(t)$ is then obtained by solving

$$-\frac{d}{dt}\psi(t) = \begin{pmatrix} 10 \\ 0 \end{pmatrix} + \begin{pmatrix} -10 & 10 \\ 1 & -1 \end{pmatrix} \psi(t)$$

which gives us

$$\begin{pmatrix} \psi_1(t) \\ \psi_2(t) \end{pmatrix} = \begin{pmatrix} -\frac{10}{121e^{110}}(-120e^{110} + 10e^{11t} + 11e^{110}t) \\ \frac{10}{121e^{110}}(109e^{110} + e^{11t} - 11e^{110}t) \end{pmatrix}$$

Now we have to find the place where the best decision is no longer $d = (2, 1)$ but $d = (1, 1)$ by solving

$$3 - 2\psi_1(t) + 2\psi_2(t) = 10 - 10\psi_1(t) + 10\psi_2(t)$$

Which gives us that action $a_{1,1}$ becomes better at $t = 9.70151$. Continuing with the algorithm at this point gives already too complicated results to write out in this section. The reader is encouraged to do this himself, with the help of a computer of course. When doing so, the algorithm will indeed give us that for the next step $d = (1, 1)$ is the best decision for $t \in (-\infty, 9.70151)$ and gives us that $v = \begin{pmatrix} 10.852 \\ 9.852 \end{pmatrix}$.
Now consider the discrete approximations with $m = 200$, $m = 1000$ and $m = 100000$ (see appendix A for the code we use to calculate the outcome of the algorithm). For the first approximation the grid size is 0.05 and this gives us that for $t \in (0, 9.75)$ decision $\pi(t) = (1, 1)$ and that $\pi(t) = (2, 1)$ for $t \in (9.75, 10)$. Giving us a final payoff of $v = \begin{pmatrix} 10.910 \\ 9.910 \end{pmatrix}$
The second approximation has a grid size of 0.01 and this gives us $\pi(t) = (1, 1)$ for $t \in (0, 9.71)$, with a final payoff of $v = \begin{pmatrix} 10.863 \\ 9.863 \end{pmatrix}$
The last approximation has a grid size of 0.0001 and this gives us $\pi(t) = (1, 1)$ for $t \in (0, 9.7016)$, with a final payoff of $v = \begin{pmatrix} 10.852 \\ 9.852 \end{pmatrix}$
We can see that even for large grid sizes "true" value of $t = 9.70151$ lies within the exact last grid the algorithm finds. However, only the last algorithm found final payoffs that were within a range 0.001 of the true final payoffs. It becomes clear that we indeed need a small grid size for good end results, however even with this large grid size the algorithm works very, very fast. Even faster already then the continuous time algorithm, that took considerable time on the second step. It goes to show that the approximation we are using is indeed a very strong one.

Chapter 7

Conclusion

With the conclusion of our example, we arrive at the end of this thesis. In the first part we studied the discrete-time MDP and found that the algorithm not only solved the problem, but did so with very little computations. Afterwards we formulated the continuous-time MDP. We proved that the continuous-time case is optimized by a piecewise constant policy, a proof that was necessary in order to establish an algorithm that solves the problem exactly in finite time. However, the algorithm that calculates the exact optimal policy for the problem is still quite slow. In order to find a faster algorithm we approximated the continuous-time MDP with the discrete-time case. This allowed us to use the very fast dynamic programming algorithm to find solutions for the continuous-time case as well. In the end we showed that this approximation finds answers very close to the optimal answer very, very quickly. Showing once again that the dynamic programming algorithm is an incredibly useful tool.

I hope the reader has enjoyed reading this thesis. The mathematics that is involved in solving these seemingly simple models for decision making, quickly becomes considerably difficult. In my opinion, that is where mathematics becomes the most elegant: when it is used to solve problems that, at a first glance, look easy, but turn out to be very complicated. I therefore enjoyed writing this thesis very much. If the reader enjoyed this subject I encourage him to look furhter into it. There is still a lot to be said on the subject. There are still many more interesting applications of MDP's, such as queueing theory, and there are more, also very interesting algorithms that have been developed to solve them, such as the Q-learning algorithm. There are even cases of the MDP that are still being solved, hopefully providing us with even more ingenious algorithms in the future.

Appendix A

The code of the solver

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;

namespace MDPSolver
{
        class Program
        {
                static void Main(string[] args)
                {
                        Solver solver = new Solver();
                        solver.inlezen();
                        solver.solve();
                        solver.output();
                }
        }

        class Solver
        {
                public double T;
                public int m;
                public int nstates;
                public List¡State> states = new List¡State>();
                public double[,] v;
                public double[,] d;


                public void inlezen()
                {
                        Console.WriteLine("Wat is de tijd T?");
                        T = LeesDoubleIn();

                        Console.WriteLine("Wat is het aantal partities m?");
                        m = LeesIntIn();
```

```
            Console.WriteLine(”Hoeveel system states zijn er?”);
            nstates = LeesIntIn();

            for (int i = 1; i <= nstates; i++)
            {
                states.Add(Maakstate(i));
            }                   }

        public State Maakstate(int i)
        {
            State state = new State();

            Console.WriteLine(”Hoeveel actions heeft state {0}”, i);
            int actions = LeesIntIn();

            for (int j = 1; j <= actions; j++)
            {
                Console.WriteLine(”Wat zijn de return rate en state transistions van
action {0}?”, j);
                Action a = MaakAction();
                state.actions.Add(a);
            }
            return state;
        }

        public Action MaakAction()
        {
            double d;
            double Qcheck = 0;
            string s = Console.ReadLine();
            string[] ss = Regex.Replace(s, @”[.]”, ”,”).Split(’,’);
            Action a;

            try
                {
                d = double.Parse(ss[0]);
                }
            catch
            {
                Console.WriteLine(”De return rate is geen reel getal, probeer opnieuw”);
                a = MaakAction();
                return a;
                }

            try
            {
                for(int i = 1; i < ss.Length; i++)
                {
                    d = double.Parse(ss[i]);
                }
            }
            catch
            {
                Console.WriteLine(”Niet alle transition rates zijn rele getallen, probeer
                opnieuw”);
                a = MaakAction();
                return a;                    }

            for (int i = 1; i < ss.Length ; i++)
            {
                Qcheck += double.Parse(ss[i]);
            }

            if(Qcheck != 0)
            {
                Console.WriteLine(”Deze transition rates zijn niet toegestaan, probeer
opnieuw.”);
                a = MaakAction();
                return a;
                }
                else
            {
                a = new Action(ss, nstates);
                return a;
            }

        }

        public double LeesDoubleIn()
        {
            double d;

            try
            {
                d = double.Parse(Console.ReadLine());
            }
            catch
            {
                Console.WriteLine(”Voer een reel getal in”);
                d = this.LeesDoubleIn();
            }

            return d;
        }

        public int LeesIntIn()
        {
            int i;
            try
            {
                i = int.Parse(Console.ReadLine());
            }
            catch
            {
                Console.WriteLine("Voer een geheel getal in");
                i = this.LeesIntIn();
            }

              return i;
    }

      public void solve()
    {
            v = new double[nstates, m+1];
            d = new double[nstates, m+1];

            for(int i = 0; i < nstates; i++)
            {
                v[i, m] = -100;
                for(int j = 0; j < states[i].actions.Count; j++)
                {
                    if(states[i].actions[j].r * T / m > v[i,m])
                    {
                        v[i, m] = states[i].actions[j].r * T / m;
                        d[i, m] = j;
                    }
                }
            }

              double tussenwaarde;
            double[] transvector = new double[nstates];

              for(int i = 1; i <= m; i++)
            {
                for (int a = 0; a < nstates; a++)
                {
                    v[a, m-i] = -100;
                    for (int b = 0; b < states[a].actions.Count; b++)
                    {
                        for (int j = 0; j < nstates; j++)
                        {
                        if (states[a].actions[b].transrates[j] >= 0)
                        {
                        transvector[j] = states[a].actions[b].transrates[j] * T / m;
                        else
                        {
                        transvector[j] = 1 + states[a].actions[b].transrates[j] * T / m; }
                        }
                        tussenwaarde = 0;

                        for (int j = 0; j < nstates; j++)
                        {
                            tussenwaarde += transvector[j] * v[j, m - i + 1];
                        }

                        tussenwaarde += states[a].actions[b].r * T / m;

                        if (tussenwaarde > v[a, m-i])
                        {
                            v[a, m-i] = tussenwaarde;
                            d[a, m-i] = b;
                        }
                    }
                }
            }
        }

        public void output()
        {
            bool changed = false;
            bool end = false;
            int changedcount = m;
            List<String> outputs = new List<String>();
            for(int i = 1; i <= m; i++)
            {
                for(int j = 0; j < nstates; j++ )
                {
                    if (d[j, m - i] != d[j, m - i + 1])
                    {
                        changed = true;
                    }
                    if(i == m)
                    {
                        end = true;
                    }
                }

                if(changed)
                {
                    int a = m - i + 1;

                    string s = "for " + a.ToString() + " through " +
                    changedcount.ToString() + " use " ;

                    for(int k = 0; k < nstates; k++)
                    {
```

```
                                    s += d[k,m-i + 1];
                                    if(k+1 != nstates)
                                    {
                                        s+= ",";
                                    }
                                }
                                outputs.Add(s);
                                changedcount = m - i;
                                changed = false;
                            }

                            if(end)
                            {
                                int a = 0;
                                string s = "for " + a.ToString() + " through " +
changedcount.ToString() + " use ";
                                for (int k = 0; k < nstates; k++)
                                {
                                    s += d[k, m - i];
                                    if (k + 1 != nstates)
                                    {
                                        s += ",";
                                    }
                                }
                                outputs.Add(s);
                            }
                        }

                        Console.WriteLine();
                        for(int i = 1; i <= outputs.Count;i++)
                        {
                            Console.WriteLine(outputs[outputs.Count - i]);
                        }

                        for (int i = 0; i < nstates ; i++)
                        {
                            Console.WriteLine(v[i, 0].ToString());
                        }
                        Console.ReadLine();
                    }
                }

                class State
                {
                    public List<Action> actions = new List<Action>();
                }

                class Action
                {
```

```
        public double r;
        public List<double> transrates = new List<double>();

        public Action(string[] ss, int nstates)
        {
            r = double.Parse(ss[0]);
            for(int i = 1; i <= nstates; i++ )
            {
                transrates.Add(double.Parse(ss[i]));
            }
        }
    }
}
```

Bibliography

[1] Puterman, M., Markov Decision Processes, Discrete Stochastic Dynamic Programming, John Wiley & Sons, Inc., Hoboken, New Jersey 1st edition, 2005.

[2] Miller, B., Finite State Continuous-Time Markov Decision Processes With Applications To A Class Of Optimization Problems In Queueing Theory, Stanford University Press, Stanford, California, 1967.

[3] Doob,J.L., Stochastic Processes, John Wiley, New York, 1953.

[4] Coddington, E., and Levinson, N., Theory of Ordinary Differential Equations, McGraw-Hill, New York, 1955.

[5] Royden, H. L., Real Analysis, The MacMillan Company, New York, 1963.

[6] Natanson, I. P., Theory of Functions of a Real Variable, translated from the Russian by Leo Boron, F. Ungar, New York, 1955.

[7] Graves, L., The Theory of Functions of Real Variables, McGraw-Hill, New York, 1960.

[8] Bellman, R., Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1957.

[9] Howard, R. A., Dynamic Programming and Markov Processes, John Wiley, New York, 1960.