Ruprecht-Karls-Universität Heidelberg
Fakultät für Mathematik und Informatik
Interdisziplinäres Zentrum für Wissenschaftliches Rechnen

B. Sc. Thesis

Iterative Solution of Markov Decision Processes

Name: Phil Neitzel
Student ID: 4111375
Supervision: Prof. Dr. Roland Herzog
Dr. Karina Koval

Date of Submission: 01.06.2023

Declaration

I hereby confirm that I wrote this work independently and did not use any sources other than those indicated.

Heidelberg, May 28, 2023 . . . . . .

Author name

This bachelor thesis discusses the application of different iterative solvers to the Policy iteration, which is commonly used to optimize the so called Markov Decision Processes (MDPs). In general, MDPs provide us with a mathematical framework used to model decision-making in stochastic environments, which we will be discussing in detail in this thesis. Our focus lies on analysing and discussing the Jacobi, Gauss-Seidel, Richardson and Krylov subspace methods. We are especially interested in understanding which model is most efficient depending on the problem we want to solve and how the different iterators can be interpreted in regards to our model. Meaning, we aim to describe what happens with our MDP in the process of the linear system being solved.

In dieser Bachelorarbeit geht es um verschiedene iterative Löser und ihre Anwendung auf das Policy-Iterationsverfahren, welches ein Standardverfahren ist um so genannte Markow Entscheidungsprozesse (MDPs) zu optimieren. Im allgemeinen bieten MDPs einen mathematischen Rahmen in dem Entscheidungen innerhalb einer stochastischen Umgebung modelliert werden können, welcher in dieser Thesis ausführlich beschrieben und hergeleitet wird. Unser Fokus ist hierbei das Analysieren und Diskutieren der Jacobi-, Gauss-Seidel-, Richardson- und Krylov-Unterraum-Verfahren. Insbesondere interessiert uns, wann man welche Verfahren benutzen sollte und ob wir die verschiedenen Iteratoren in bezug auf die MDPs interpretieren können. Wir versuchen also zu beschreiben, was beim Lösen mit des linearen Gleichungssystem mit unserem Problem passiert.

Contents

1 Introduction 1
- 1.1 Markov Chains . . . 1
- 1.2 The Basic Problem . . . 2
- 1.3 The Dynamic Programming Algorithm . . . 3
- 1.4 Discounted Problems on the Infinite Horizon . . . 4

2 Markov Decision Processes 10
- 2.1 The Model . . . 10
- 2.2 Some Examples for MDPs . . . 11
  - 2.2.1 An Introductory Example . . . 11
  - 2.2.2 Example: Transportation Problem . . . 12
  - 2.2.3 Example: The Studying Problem . . . 14
- 2.3 Policy Iteration . . . 17
- 2.4 Applying Policy Iteration . . . 18
- 2.5 Convergence of Policy Iteration . . . 20

3 Methods for Solving Linear Systems 24
- 3.1 Gauss-Seidel and Jacobi Method . . . 24
- 3.2 Richardson Iteration . . . 27
- 3.3 Krylov Subspace Methods . . . 27
  - 3.3.1 Arnoldi's Method . . . 29
  - 3.3.2 The GMRES Algorithm . . . 31
  - 3.3.3 Krylov Subspace Methods on MDPs . . . 34

4 Conclusion 38

5 Notation 40

1 Introduction

In this thesis, we look at Markov Decision Processes (MDPs) and how to optimize them using Policy Iteration with different solvers for the arising linear system. Our goal is to analyse the different methods and interpret them in regards to our model. MDPs have a wide range of applications such as machine learning, economics and manufacturing. However, we can also apply them to model the decisions we have to make in our everyday life. The general idea of this model is that for any state we are in we know what actions we can take and how much it costs us. Furthermore, we know the probabilities of going to a new state after choosing our action. Of course, in real life we usually do not have access to all this information. There are extensions of MDPs that cover this problem in more detail, like Partially Observable Markov Decision Processes which will not be part of this thesis. Before we introduce Markov Decision Processes we have to discuss the mathematical foundation of the model.

1.1 Markov Chains

We start with the concept of Markov Chains which model the probabilities of moving from one state to another state as presented in the following definition.

Definition 1.1 (Markov Chains from Johannes 2021, Def. 17.01)**.** *Let $\mathbb{T} = \mathbb{N}_0$ (discrete time) or $\mathbb{T} = \mathbb{R}^+$ (continuous time) and S a countable non-empty set (Space of states). A stochastic process $(X_t)_{t\in\mathbb{T}}$ with the Space of states S is called a* Markov Chain *if for all $n \in \mathbb{N}$, all $t_1 < t_2 < \cdots < t_n < t$ in $\mathbb{T}$ and all $s_1, \ldots, s_n, s$ in S with the probability $P(X_{t_1} = s_1, \ldots, X_{t_n} = s_n) \in (0, 1]$ satisfy the Markov Properties:*

$$P(X_t = s | X_{t_1} = s_1, \ldots, X_{t_n} = s_n) = P(X_t = s | X_{t_n} = s_n)$$

For a Markov Chain $(X_t)_{t\in\mathbb{T}}$, $t_1 \leq t_2$ from $\mathbb{T}$. $p_{ij}(t_1, t_2)$ with $i, j \in S$ describes the probability of going from one state i at time t_1 to a state j at t_2 as $p_{ij}(t_1, t_2) = P(X_{t_1} = j | X_{t_2} = i)$. $P(t_1, t_2) = \big(p_{ij}(t_1, t_2)\big)_{i,j\in S}$ is called the probability matrix*. The probability matrix is called* time-homogeneous *if*

$$P(t_1, t_2) = P(0, t_2 - t_1) =: P(t_2 - t_1),$$

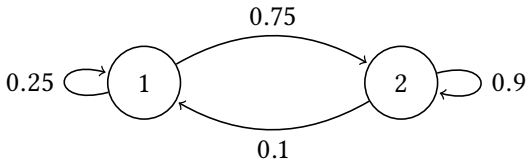
for all $t_1 \leq t_2$ in $\mathbb{T}$.

In this thesis we always assume $\mathbb{T} = \mathbb{N}_0$ and furthermore that our probability matrix is time-homogeneous, so we denote $P := P(1)$. So the Markov Properties mean nothing else

than going from state i to state j in one time step is independent of previous states before i. For example, using the following probability matrix

$$P = \begin{pmatrix} 0.25 & 0.75 \\ 0.1 & 0.9 \end{pmatrix},$$

the Markov chain can be modelled with this diagram



1.2 The Basic Problem

In this part, we introduce a discrete-time dynamic system to model decision-making mathematically as described in Bertsekas 2005. First, we gather everything this model needs, starting with a space of states S_k at a time k. This space consists of all the states x_k one could be in at this point of time. Of course, we have to know what decisions can be made at any time, and we denote this space as C_k with elements u_k referred to as actions or controls. Depending on the current state it might not be possible to choose from all actions, so we usually write $u_k \in U_k(x_k) \subset C_k$. Lastly, our model is generally assumed to come with some uncertainty. Since, as we know, making a decision does not always lead to the wanted result as there is always a chance for some random disturbance making it impossible to be sure of our next state. This disturbance w_k is characterized by the probability $P(w_k|x_k, u_k)$ as the current state and chosen actions can still influence the disturbances. The space of all disturbances at time k is denoted as D_k. In total, we get a function f_k determining the next state

$$x_{k+1} = f_k(x_k, u_k, w_k), \ \ k = 0, 1, \ldots, N-1. \tag{1.1}$$

Of course, taking an action or being in a certain state might have some cost ($g > 0$) or benefit ($g < 0$) to it. This so-called *cost function* is denoted as $g_k(x_k, u_k, w_k)$ for a time k. Furthermore, g_k is considered to be additive. Meaning the cost for a model running for a time of N is

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k),$$

where $g_N(x_N)$ is a *terminal cost* added at the end of the process. However, as the cost cannot be determined with certainty due to the random disturbances, we have to calculate the expected cost instead

$$\mathbb{E}\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right].$$

Additionally, one usually decides on an action depending on their current state. We call those *policies* $\mu_k(x_k)$ and if they fulfil $u_k = \mu_k(x_k) \in U_k(x_k)$ for all $x_k \in S_k$ they are considered *admissible*. The class of policies is then written as

$$\pi = \{\mu_0, \dots, \mu_{N-1}\}.$$

In conclusion, we get an expected cost function for a class of admissible policies depending on an initial state x_0 and denote it as

$$J_\pi(x_0) = \mathbb{E}\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\right].$$

Since the goal is to find an optimal policy π^* minimizing the function $J_\pi(x_0)$ over the set of all admissible policies Π we write

$$J^*(x_0) = J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0) \tag{1.2}$$

and call this the *optimal cost function* for an initial state x_0. We write J^* as the optimal cost function for any initial state.

1.3 The Dynamic Programming Algorithm

To get an idea on how to optimize $J_\pi(x_0)$ we use the *Principle of Optimality*. Again the motivation and algorithm are taken from Bertsekas 2005. First, we consider an optimal class of admissible policies $\pi^* = \{\mu_0^*, \mu_1^*, \dots, \mu_{N-1}^*\}$ for a model of time length N. The expected cost for starting at time $i < N$ is then calculated by

$$\mathbb{E}\left[g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\right].$$

Of course, $\{\mu_i^*, \dots, \mu_{N-1}^*\}$ is the optimal policy for this subproblem, because if there were a better policy we would choose this policy for our main problem as well. This, however, would be a contradiction to π^* being optimal. So we can see the optimal solution of Equation (1.2) can be determined bit by bit by finding the optimal policy for the subproblem of smaller time length and then using this result to calculate the optimal policy for the subproblem with larger time length until we get the result for the original problem. Of course, we can formulate this so-called *Dynamic Programming Algorithm (DP Algorithm)* mathematically.

Proposition 1.2 (The DP Algorithm from Bertsekas 2005, Prop. 1.3.1)**.** *For every initial state x_0, the optimal cost $J^*(x_0)$ of the basic problem is equal to $J_0(x_0)$, given by the last step of the following algorithm, which proceeds backward in time from period $N-1$ to period 0:*

$$J_N(x_N) = g_N(x_N) \tag{1.3}$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \mathbb{E}\left[g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))\right], \quad k = 0, 1, \ldots, N-1. \tag{1.4}$$

Where the expectation is taken with respect to the probability distribution of w_k, which depends on x_k and u_k. Furthermore, if $u_k^* = \mu_k^*(x_k)$ minimizes the right side of Equation (1.4) for each x_k and k, the policy $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is optimal.

1.4 Discounted Problems on the Infinite Horizon

Now we have a general idea for the DP Algorithm on the very basic problem. However, in this thesis, we mainly focus on infinite horizon problems. So for this section we consult Bertsekas 2012 and start by introducing $\alpha > 0$, which is called the *discount factor* if $\alpha < 1$. If $\alpha = 1$ we will refer to the problem as *undiscounted*. The intuitive idea behind a discount factor is that costs or benefits are usually worth less when lying further in the future. For example, being asked whether one would like to receive €10 now or €11 in a year, a lot of people would take the €10 now because €11 is being discounted by having to wait a whole year to get only a single Euro more.
Now looking at an admissible policy $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ and a terminal cost of $\alpha^N J(x_N)$ where $J : X \to \mathbb{R}$ can be generally interpreted as the expected cost function for everything coming after the first N states. As we will see throughout this thesis, this function is usually calculated rather than chosen. However, for the moment we assume to have knowledge over J.
Adding the costs for the first $N-1$ states to the terminal cost leads to the total expected cost

$$\mathbb{E}\left[\alpha^N J(x_N) + \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k)\right]. \tag{1.5}$$

As we are referring to an infinite horizon problem, from now on we denote the state x_k as an element of the space X, the control u_k as an element of U and the random disturbances w_k of W. The minimum of Equation (1.5) over all admissible policies π can be determined by applying the DP algorithm to this problem as follows

$$J_{N-k}(x) = \min_{u \in U(x)} \mathbb{E}\left[\alpha^{N-k} g(x, u, w) + J_{N-k+1}(f(x, u, w))\right],$$

with the initial condition

$$J_N(x) = \alpha^N J(x).$$

Where $J_{N-k}(x)$ denotes the optimal cost of the last k stages from state x. So $J_0(x)$ is the optimal cost function for the overall problem regarding the initial state x.
Even though this is a finite horizon problem with our interpretation of J we can view it as a simplified version of an infinite horizon problem. Of course, if we can simplify this infinte-horizon problem to any N-stage problem the most intuitive decision is to choose a one-stage problem with $N = 1$. Here the terminal cost is αJ and the single stage cost g.

Meaning, we calculate the expected cost of the initial state and the discounted expected future costs. To simplify notation, we denote the optimal solution of this one-stage problem as

$$(TJ)(x) = \min_{u \in U(x)} \mathbb{E}\left[g(x,u,w) + \alpha J(f(x,u,w))\right], \ x \in X. \tag{1.6}$$

Definition 1.3 (Stationary Policy from Bertsekas 2005, Sec. 7.1)**.** *A stationary policy is an admissible policy of the form $\pi = \{\mu, \mu, \dots\}$, and its corresponding cost function is denoted by $J_\mu(i)$. For brevity, we refer to $\{\mu, \mu, \dots\}$ as the* stationary policy *μ. We say that μ^* is optimal if*

$$J_{\mu^*}(i) = J^*(i) = \min_\pi J_\pi(i).$$

Using this definition, we denote for any function $J : X \to \mathbb{R}$ and any stationary policy μ

$$(T_\mu J)(x) = \mathbb{E}\left[g(x,\mu(x),w) + \alpha J(f(x,\mu(x),w))\right], \ x \in X, \tag{1.7}$$

which can be interpreted analogously to the above. Next we denote by $T^k J$ the composition of the mapping T with itself k times. Meaning, for all k we write

$$(T^k J)(x) = \left(T(T^{k-1}J)\right)(x) = \min_{u_k \in U_k(x_k)} \mathbb{E}\left[\alpha^N J(x_N) + \sum_{k=0}^{N-1} \alpha^k g(x_k, u_k, w_k)\right], \ x \in X.$$

Thus, $T^k J$ is the function obtained by applying the mapping T to the function $T^{k-1}J$. For convenience, we also write

$$(T^0 J)(x) = J(x), \ x \in X.$$

Similarly, $T_\mu^k J$ is defined as

$$(T_\mu^k J)(x) = T_\mu(T_\mu^{k-1}J)(x) = \mathbb{E}\left[\alpha^N J(x_N) + \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k)\right], \ x \in X \tag{1.8}$$

and

$$(T_\mu^0 J)(x) = J(x), \ x \in X.$$

One can immediately see $(T^k J)(x)$ is the optimal cost function for the k-stage problem and initial state x. While $(T_\mu J)(x)$ is the cost function for the k-stage problem with a certain stationary policy μ.

Lemma 1.4 (Monotonicity Lemma from Bertsekas 2012, Lem. 1.1.1)**.** *For any function $J : X \to \mathbb{R}$ and $J' : X \to \mathbb{R}$, such that for all $x \in X$, $J(x) \leq J'(x)$, and any stationary policy $\mu : X \to U$, we have*

$$(T^k J)(x) \leq (T^k J')(x)$$
$$(T_\mu^k J)(x) \leq (T_\mu^k J')(x)$$

for all $x \in X$, and $k = 1, 2, \dots$

Proof. We will prove this using induction. For $k = 1$ it means:

$$\begin{aligned}(TJ)(x) &= \min_{u \in U(x)} \mathbb{E}\left[g(x,u,w) + \alpha J(f(x,u,w))\right] \\ &\overset{J(x) \leq J'(x)}{\leq} \min_{u \in U(x)} \mathbb{E}\left[g(x,u,w) + \alpha J'(f(x,u,w))\right] = (TJ')(x)\end{aligned}$$

Now we assume that $(T^k J)(x) \leq (T^k J')(x) (*)$ is satisfied and show $(T^{k+1} J)(x) \leq (T^{k+1} J')(x)$

$$\begin{aligned}(T^{k+1}J)(x) &= (T(T^k J))(x) \\ &= \min_{u \in U(x)} \mathbb{E}\left[g(x,u,w) + \alpha (T^k J)(f(x,u,w))\right] \\ &\overset{(*)}{\leq} \min_{u \in U(x)} \mathbb{E}\left[g(x,u,w) + \alpha (T^k J')(f(x,u,w))\right] \\ &= (T^{k+1}J')(x),\end{aligned}$$

which implies $(T^k J)(x) \leq (T^k J')(x)\ \forall x \in X, k = 1, 2, \ldots$. The proof works similarly for $(T_\mu^k J)(x) \leq (T_\mu^k J')(x)$. □

Assumption 1.5 (Assumption D (Discounted Cost - Bounded Cost per Stage) from Bertsekas 2012, Sec. 1.2)**.** *The cost per stage g satisfies for all $(x, u, w) \in X \times U \times W$*

$$|g(x,u,w)| \leq M,$$

where M is some scalar. Furthermore, $0 < \alpha < 1$.

Since we are mainly considering finite sets X, U and W in this thesis, this assumption always holds true and is no real restriction, but it needed to be mentioned in order to understand the argumentation of the following propositions.

Proposition 1.6 (Bertsekas 2012, Prop. 1.2.2)**.** *For every stationary policy μ, the associated cost function satisfies for all $x \in X$*

$$J_\mu(x) = \lim_{N \to \infty} (T_\mu^N J)(x). \tag{1.9}$$

Proof. For every positive integer N, initial state $x_0 \in X$ and stationary policy μ, we break down the cost $J_\mu(x_0)$ into the portions incurred over the first N stages and over the remaining

stages.

$$\begin{aligned}
J_\mu(x_0) &= \lim_{K\to\infty} \mathbb{E}\left[\sum_{k=0}^{K} \alpha^k g(x_k, \mu(x_k), w_k)\right] \\
&= \mathbb{E}\left[\sum_{k=0}^{N-1} \alpha^k g(x_k, \mu(x_k), w_k)\right] + \lim_{K\to\infty} \mathbb{E}\left[\sum_{k=N}^{K} \alpha^k g(x_k, \mu(x_k), w_k)\right] \\
&\leq \mathbb{E}\left[\sum_{k=0}^{N-1} \alpha^k g(x_k, \mu(x_k), w_k)\right] + \left|\lim_{K\to\infty} \mathbb{E}\left[\sum_{k=N}^{K} \alpha^k g(x_k, \mu(x_k), w_k)\right]\right| \\
&\leq \mathbb{E}\left[\sum_{k=0}^{N-1} \alpha^k g(x_k, \mu(x_k), w_k)\right] + \sum_{k=N}^{\infty} \alpha^k M \qquad \text{(Ass. D)} \\
&= \mathbb{E}\left[\sum_{k=0}^{N-1} \alpha^k g(x_k, \mu(x_k), w_k)\right] + \frac{\alpha^N M}{1-\alpha}
\end{aligned}$$

Using this inequality it follows that

$$\begin{aligned}
& J_\mu(x_0) - \frac{\alpha^N M}{1-\alpha} - \alpha^N \max_{x\in X} |J(x)| \\
\leq\ & \mathbb{E}\left[\alpha^N J(x_N) + \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu(x_k), w_k)\right] \\
\leq\ & J_\mu(x_0) + \frac{\alpha^N M}{1-\alpha} + \alpha^N \max_{x\in X} |J(x)|.
\end{aligned}$$

So by definition of $(T_\mu^N J)(x_0)$ it follows immediately

$$\begin{aligned}
& J_\mu(x_0) - \frac{\alpha^N M}{1-\alpha} - \alpha^N \max_{x\in X} |J(x)| \\
\leq\ & (T_\mu^N J)(x_0) \\
\leq\ & J_\mu(x_0) + \frac{\alpha^N M}{1-\alpha} + \alpha^N \max_{x\in X} |J(x)|
\end{aligned}$$

and by taking the limit $N \to \infty$, we receive the desired result

$$\begin{aligned}
& J_\mu(x_0) \leq \lim_{N\to\infty} (T_\mu^N J(x_0) \leq J_\mu(x_0) \\
\implies\ & J_\mu(x_0) = \lim_{N\to\infty} (T_\mu^N J)(x_0).
\end{aligned}$$

□

Proposition 1.7 (Convergence of the DP Algorithm from Bertsekas 2012, Prop. 1.2.2)**.** *For any bounded function $J : X \to \mathbb{R}$, we have for all $x \in X$,*

$$J^*(x) = \lim_{N\to\infty} (T^N J)(x).$$

Proof. The proof is similar to the one from Proposition 1.6 with the main difference being we now have $\pi = \{\mu_0, \mu_1, \dots\}$ instead of a stationary policy. This means μ is also dependent on k. In general, Proposition 1.6 is a special case of this proposition. □

Proposition 1.8 (Bertsekas 2012, Prop. 1.2.4)**.** *For every stationary policy μ, the associated cost function satisfies for all $x \in X$,*

$$J_\mu(x) = \mathbb{E}\left[g(x, \mu(x), w) + \alpha J_\mu(f(x, \mu(x), w))\right]$$

or, equivalently

$$J_\mu = T_\mu J_\mu.$$

Furthermore, J_μ is the unique solution of this equation within the class of bounded functions. Moreover, for any bounded function J with $J \geq T_\mu J$ (or $J \leq T_\mu J$), we have $J \geq J_\mu$ (or $J \leq J_\mu$), respectively.

Proof. We can immediately show $J_\mu = T_\mu J_\mu$ because of Proposition 1.6

$$\begin{aligned}
(T_\mu J_\mu)(x) &= \mathbb{E}\left[g(x, \mu(x), w) + \alpha J_\mu(f(x, \mu(x), w))\right] \\
&= \mathbb{E}\left[g(x, \mu(x), w) + \alpha\left(\lim_{N\to\infty} (T_\mu^N J)(f(x, \mu(x), w))\right)\right] \\
&= \lim_{N\to\infty} \mathbb{E}\left[g(x, \mu(x), w) + \alpha (T_\mu^N J)(f(x, \mu(x), w))\right] \\
&= \lim_{N\to\infty} (T_\mu^{N+1} J)(x) \\
&= \lim_{N\to\infty} (T_\mu^N J)(x) \\
&= J_\mu(x).
\end{aligned}$$

To show uniqueness, we assume some bounded function J satisfies $J = T_\mu J$ then $J = \lim_{N\to\infty} T_\mu^N J$ by Proposition 1.6. So we get $J = J_\mu$ which proves uniqueness of the solution. Now what is left to show is $J \geq T_\mu J \implies J \geq J_\mu$ (analogue for $J \leq T_\mu J \implies J \leq J_\mu$).
So let $J \geq T_\mu J$.
Using the Lemma of Monotonicity Lemma 1.4 it follows

$$J \geq T_\mu J \geq T_\mu^2 J \geq \cdots \geq T_\mu^k J \geq T_\mu^{k+1} J \geq \dots$$

and by taking the limit $k \to \infty$ it follows

$$J \geq \lim_{k\to\infty} T_\mu^k J = J_\mu.$$

□

Proposition 1.9 (Bellamn's Equation from Bertsekas 2012, Prop. 1.2.3)**.** *The optimal cost function J^* satisfies for all $x \in X$,*

$$J^*(x) = \min_{u \in U(x)} \mathbb{E}\left[g(x,u,w) + \alpha J^*(f(x,u,w))\right] \tag{1.10}$$

or equivalently,

$$J^* = TJ^*.$$

Furthermore, J^* is the unique solution of this equation within the class of bounded functions. Moreover, for any bounded function J with $J \geq TJ$ (or $J \leq TJ$), we have $J \geq J^*$ (or $J \leq J^*$, respectively)

Proof. Again this Proposition is proven with the same argumentation as in the proof of Proposition 1.8 □

Proposition 1.10 (Necessary and Sufficient Condition for Optimality from Bertsekas 2012, Prop. 1.2.5)**.** *A stationary policy μ is optimal if and only if $\mu(x)$ attains the minimum in Bellman's equation* (1.10) *for each $x \in X$; i.e.,*

$$TJ^* = T_\mu J^*.$$

Proof. If $TJ^* = T_\mu J^*$, then using the Bellman's equation ($J^* = TJ^*$), we have $J^* = T_\mu J^*$. So by the uniqueness part of Proposition 1.8, we obtain $J^* = J_\mu$; i.e., μ is optimal. Conversely, if the stationary policy μ is optimal, we have $J^* = J_\mu$, which by Proposition 1.8, yields $J^* = T_\mu J^*$. Combining this with Bellman's equation, we obtain $TJ^* = T_\mu J^*$. □

The previous propositions should give a better understanding as to why J can be interpreted as an expected cost function and of course the general understanding of finding the optimal solution to our model. Additionally, they will also become useful for grasping the concept of optimizing Markov Decision processes which we look into in the next chapter.

2 Markov Decision Processes

With the understanding of the general idea of decision-making, and the mathematical derivations gather in the previous chapter, in this chapter we introduce the model for Markov Decision processes (MDPs), which is a special version of the problem discussed in the introduction. Again the model presented in this chapter is taken from Bertsekas 2012.

2.1 The Model

As mentioned in the previous chapter the space of states X, controls U and disturbances W are assumed to be finite sets. Meaning, MDPs will be introduced working with finite-state Markov chains. An important assumption made throughout this thesis is that the process is discounted, so $0 < \alpha < 1$, where α is the discount factor. Furthermore, the states are simply enumerated and denoted as i or x, respectively. So, having an MDP with n states, the space of states is

$$X = \{1, \dots, n\}.$$

The disturbances used in the previous chapter are now generally the state following a made decision. Thus, $w_k = x_{k+1}$ for a time k and hence, the *transition probability* is denoted as going from state i to state j in a single time period, so

$$p_{ij}(u) = P\left(x_{k+1} = j | x_k = i, u_k = u\right), \quad \text{for } i, j, \in X, u \in U(i).$$

Using the same notation we introduced in the last chapter the mappings of the expected cost per stage TJ and $T_\mu J$ can be written as

$$(TJ)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha J(j)), \ i \in X,$$

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}(\mu(i))(g(i,\mu(i),j) + \alpha J(j)), \ i \in X.$$

However, to simplify this equation we assume g to be independent of j. This does not make a significant difference, because if a problem's cost is dependent on j we can always define

$$g(i,u) := \sum_{j=1}^{n} p_{ij}(u) g(i,u,j). \tag{2.1}$$

Meaning, the following mappings are equivalent to the ones above

$$(TJ)(i) = \min_{u \in U(i)} \left(g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J(j) \right), \ i \in X, \tag{2.2}$$

$$(T_\mu J)(i) = g(i, \mu(i)) + \alpha \sum_{j=1}^{n} p_{ij}(\mu(i)) J(j), \ i \in X. \tag{2.3}$$

As the goal is to find an optimal policy for all n states we denote

$$J = \begin{pmatrix} J(1) \\ \vdots \\ J(n) \end{pmatrix}, \qquad TJ = \begin{pmatrix} (TJ)(1) \\ \vdots \\ (TJ)(n) \end{pmatrix}, \qquad T_\mu J = \begin{pmatrix} (T_\mu J)(1) \\ \vdots \\ (T_\mu J)(n) \end{pmatrix}.$$

Furthermore, we can write the probability matrix for a stationary policy μ as

$$P_\mu = \begin{pmatrix} p_{11}(\mu(1)) & \dots & p_{1n}(\mu(1)) \\ \vdots & \ddots & \vdots \\ p_{n1}(\mu(n)) & \dots & p_{nn}(\mu(n)) \end{pmatrix}$$

and the corresponding cost vector

$$g_\mu = \begin{pmatrix} g(1, \mu(1)) \\ \vdots \\ g(n, \mu(n)) \end{pmatrix}.$$

Leading us to the equation as a vector notation

$$T_\mu J = g_\mu + \alpha P_\mu J. \tag{2.4}$$

Using Proposition 1.8 we also know that J_μ is the unique solution for a stationary policy μ meaning

$$J_\mu = T_\mu J_\mu = g_\mu + \alpha P_\mu J_\mu. \tag{2.5}$$

Which we can rewrite as

$$(\mathbb{I} - \alpha P_\mu) J_\mu = g_\mu, \tag{2.6}$$

with $\mathbb{I}$ being the $n \times n$ identity matrix. This gives us a linear system we can solve for J_μ which is important for finding an optimal policy as we will see later in this chapter.

2.2 Some Examples for MDPs

2.2.1 An Introductory Example

First, let us begin with a simple example where we can choose between two actions and move between to different states. Meaning, $X = \{1, 2\}$ and $U = \{a, b\}$. For each action we receive a

probability matrix as follows

$$P_a = \begin{pmatrix} 0.3 & 0.7 \\ 0.3 & 0.7 \end{pmatrix}, \qquad P_b = \begin{pmatrix} 0.6 & 0.4 \\ 0.6 & 0.4 \end{pmatrix},$$

with the corresponding cost vectors being

$$g_a = \begin{pmatrix} 3 \\ 0.5 \end{pmatrix}, \qquad g_b = \begin{pmatrix} 1 \\ 5 \end{pmatrix}.$$

Note, that in this case the transition probability is independent from the previous state. as the probabilities in each row are the same. Of course, this is not the case for most models, but it does simplify the visualisation of the MDP, seen in the following diagram.

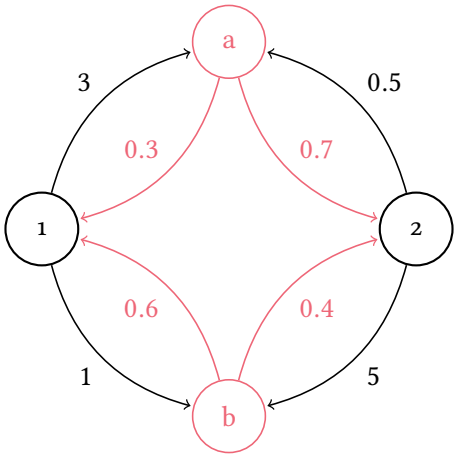


Figure 2.1: The MDP visualised with the states in black and actions in red.

2.2.2 Example: Transportation Problem

In this next example the MDP models the question of how someone should get to work. They could take the bike with a guarantee to be at work in 45 minutes. Another option would be to take the car. However, with this option they don't know the traffic status. We assume there's a chance of low traffic in which case they arrive in only 15 minutes, medium traffic, leading to 30 minutes of travel and high traffic leading to 70 minutes of travel.
Lastly, they might also decide on taking the train, meaning they will have to walk to the train station first which will take them 5 minutes. Of course, arriving at the station there is a chance that the train might be delayed so they might have to wait for it. If the trains does not arrive in the next three minutes, the worker can decide to go home and choose a different use of transport or they could wait another three minutes. When the train arrives it will take them 25 minutes to arrive at work. Of course, the goal of the worker is to get to work as fast as possible, so we need to minimize the amount of minutes he needs to get to work. Mathematically the set of all states X and the set of all actions U can look as follows

$$X = \{\text{Home, Waiting Room, Train, On Bike, Low, Medium, High, Work}\},$$
$$U = \{\text{bike, car, train station, wait, go home, go to work, do the work}\}.$$

Here, the "do the work" action is only to complete the model, as some action needs to be decided on for being in the "at work" state. The probability matrices all look very similar so only one exemplary matrix is written down to get an idea for the overall problem.

$$P_{\text{car}} = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Notice, we wrote a probability of "1" for staying in the same state except if one takes the action "car" when they are in the "Home" state as this is the only state where it is allowed to take the car. Of course, keeping them in the same state is not the only solution to this problem, but we always have to fulfil the properties of a probability matrix with every P_μ. A similar problem occurs if we look at the cost vector. To keep the worker from choosing the car, for example when they are in the waiting room, we simply have to make the cost for doing so extremely high. This way the action is always prevented from being chosen. So for example $g_{\text{go to work}}$ should look something like this

$$g_{\text{go to work}} = \begin{pmatrix} \infty \\ \infty \\ 25 \\ 45 \\ 15 \\ 30 \\ 70 \\ \infty \end{pmatrix}.$$

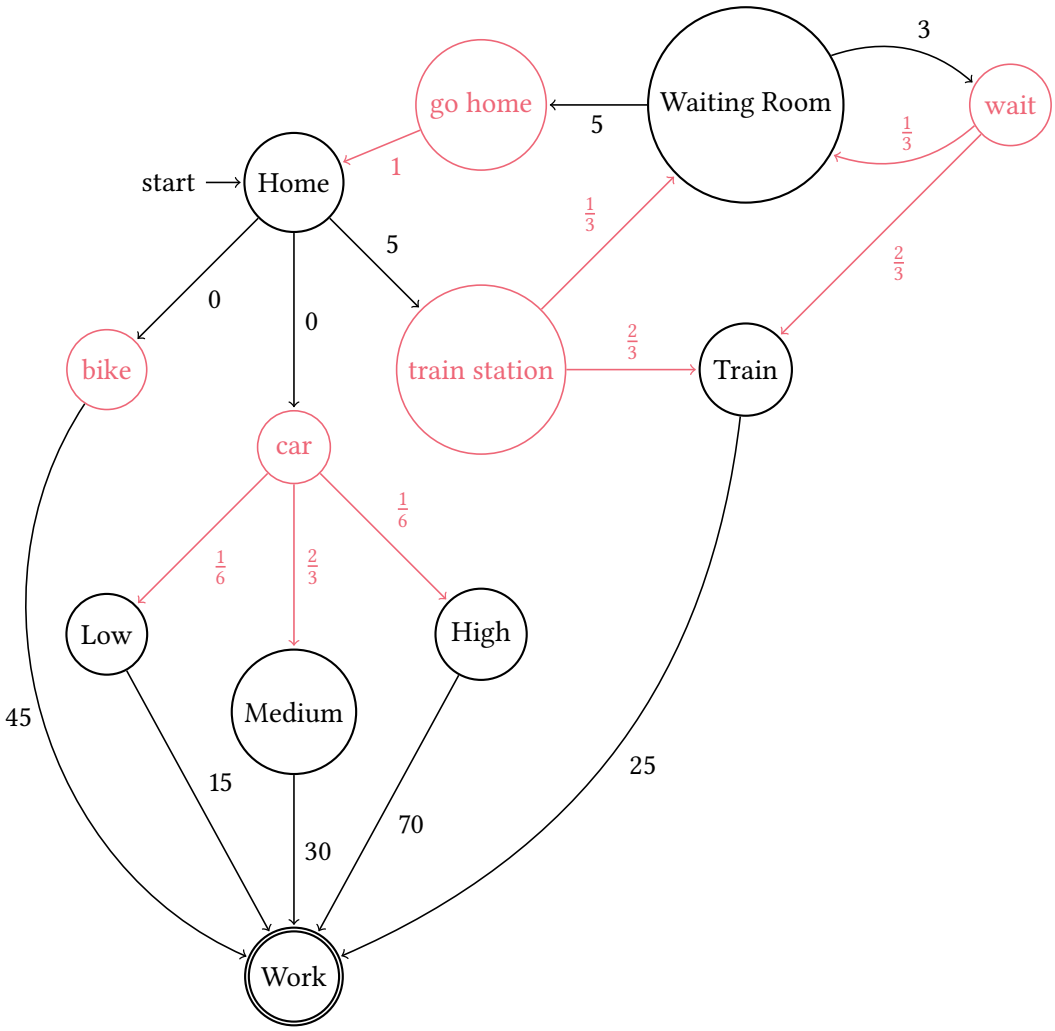


Figure 2.2: The MDP visualised with the different states (black) and actions (red).

To simplify the model and make it more readable there are some missing states and actions that can be let out in the drawing but are still mathematically relevant. For example if the worker reaches the "Train" state they will automatically take the action "go to work" or if they decide to take the bike they will automatically reach the state "On Bike".

2.2.3 Example: The Studying Problem

In this last example we look at a student who needs to decide on the amount of hours they want to study every day for an upcoming exam. They can choose between 0.5, 2 or 4 hours everyday. Their university grants the grades 1 - 5, where 1 is a great result and 5 means they failed the course. Understandably, the main goal of the student is to pass the exam. So we use a point system to decide the costs that occur depending on the action they choose. The general goal is to have as few points as possible and the hours they spend studying always

awards them with points worth the amount of hours they studied. Now the table for the points received for each grade is as follows:

Grade	Points
1	−10
2	−7
3	−4
4	−1
5	10

So the overall cost for an action they choose is determined by the expected grade and the hours they study. This means the cost is not only dependent on the current state but also the state the student will land in. As mentioned in Equation (2.1) we can simply calculate the expected cost and use this as our cost for every state i. For example for $i = 1$ and $u = 4h$, using the probabilities from below, the cost is calculated as follows:

$$\begin{aligned} g(1, 4h) &= p_{11}(4h)g(1, 4h, 1) + p_{12}(4h)g(1, 4h, 2) + p_{13}(4h)g(1, 4h, 3) \\ &= 0.75 \cdot (4 - 10) + 0.2 \cdot (4 - 7) + 0.05 \cdot (4 - 4) \\ &= -5.1 \end{aligned}$$

Another important assumption in this model is that the amount of study needed depends on the previous grade they have obtained in this field, as this is an indicator for how well someone understands the subject before they even start studying. So if, for example, the student had a good grade before, there is a higher likelihood for them to receive a good grade again and vice versa. So to describe this problem mathematically we have

$$X = \{1, 2, 3, 4, 5\}, \qquad U = \{0.5h, 2h, 4h\},$$

with the different transition matrices for each action

$$P_{0.5h} = \begin{pmatrix} 0 & 0.5 & 0.35 & 0.15 & 0 \\ 0 & 0 & 0.5 & 0.4 & 0.1 \\ 0 & 0 & 0.3 & 0.4 & 0.3 \\ 0 & 0 & 0 & 0.35 & 0.65 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$P_{2h} = \begin{pmatrix} 0.35 & 0.55 & 0.1 & 0 & 0 \\ 0.1 & 0.5 & 0.3 & 0.1 & 0 \\ 0 & 0 & 0.6 & 0.35 & 0.05 \\ 0 & 0 & 0.1 & 0.6 & 0.3 \\ 0 & 0 & 0 & 0.5 & 0.5 \end{pmatrix},$$

$$P_{4h} = \begin{pmatrix} 0.75 & 0.2 & 0.05 & 0 & 0 \\ 0.4 & 0.4 & 0.2 & 0 & 0 \\ 0.1 & 0.65 & 0.25 & 0 & 0 \\ 0 & 0.5 & 0.3 & 0.2 & 0 \\ 0 & 0 & 0.2 & 0.75 & 0.05 \end{pmatrix}$$

and the cost vectors for each action

$$g_{0.5h} = \begin{pmatrix} -4.55 \\ -0.9 \\ 1.9 \\ 6.65 \\ 10.5 \end{pmatrix}, \qquad g_{2h} = \begin{pmatrix} -5.75 \\ -3.8 \\ -0.25 \\ 4 \\ 6.5 \end{pmatrix}, \qquad g_{4h} = \begin{pmatrix} -5.1 \\ -3.6 \\ -2.55 \\ -0.9 \\ 2.95 \end{pmatrix}.$$

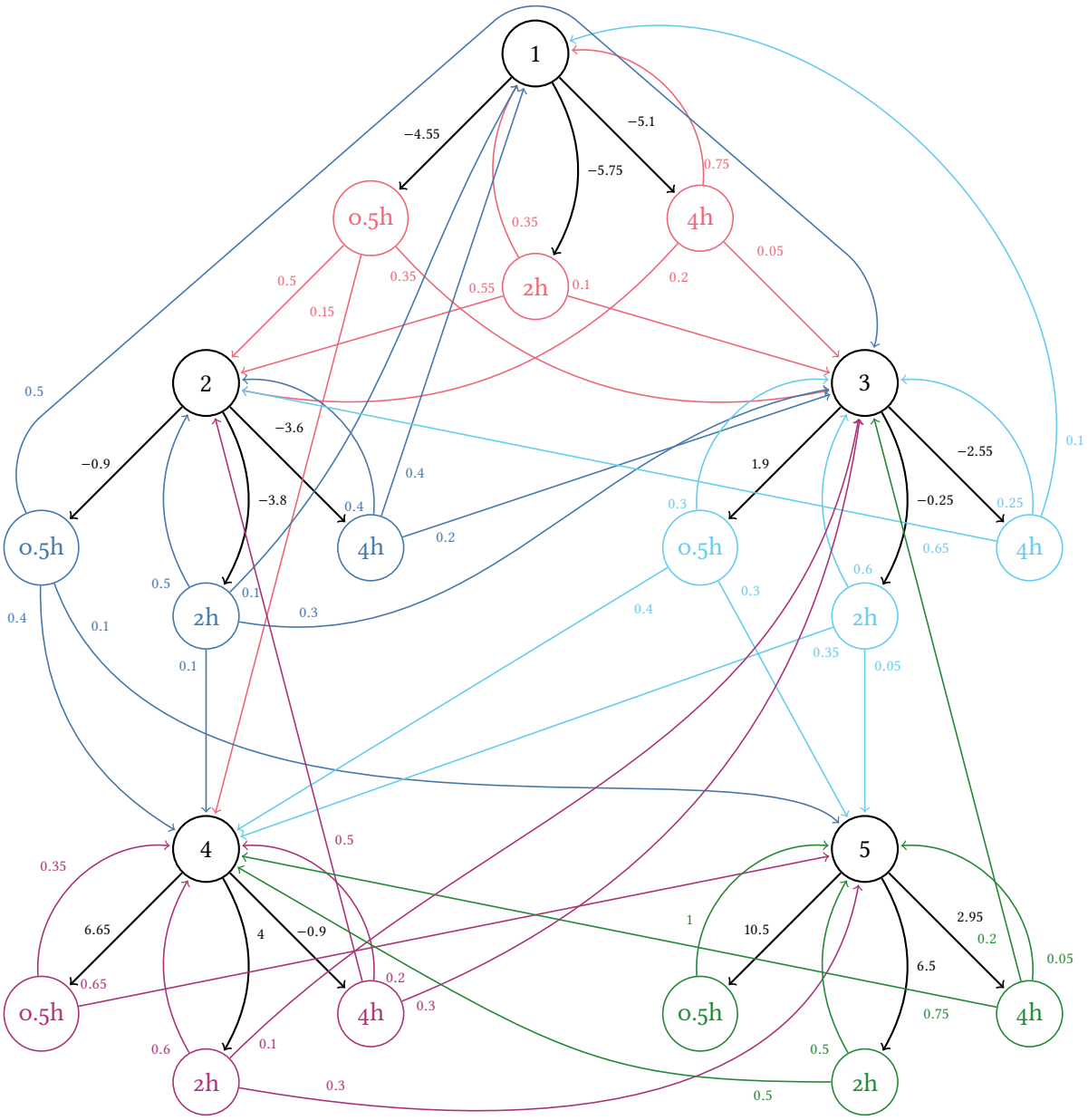


Figure 2.3: The MDP visualised with the grades being black and the study times in different colours

As we see, unlike our first example, the probabilities of going from one state to another now depend on the previous state and not just the action chosen, making the visualisation of this example a lot more complicated.

2.3 Policy Iteration

Now that we understand what Markov Decision processes are and how to model them, we discuss a method called *Policy Iteration* (PI) used to find an optimal policy for the finite-space MDPs, as introduced in Bertsekas 2012. The general idea of PI is to calculate the cost function J_μ for a given stationary policy μ by solving the Bellman Equation (2.6). Using this cost function we then compute an improved policy $\tilde{\mu}$, minimizing $T_{\tilde{\mu}}J_\mu = TJ_\mu$ with respect to J_μ. This equation comes from the necessary and sufficient condition for optimality as seen in Proposition 1.10. As we see in this section, the sequence consisting of the pairs $\{\mu_k, J_{\mu_k}\}$ will be monotonically improving the expected cost and will converge to the optimal solution.

Algorithm 1 Policy Iteration Algorithm from Bertsekas 2012, Sec. 2.3.1

Step 1: (Initialization) Guess an initial stationary policy μ_0

Step 2: (Policy Evaluation) Given the stationary policy μ_k, compute the corresponding cost function J_{μ_k} from the linear system of equations

$$(\mathbb{I} - \alpha P_{\mu_k})J_{\mu_k} = g_{\mu_k}$$

or equivalently

$$J_{\mu_k} = T_{\mu_k}J_{\mu_k}.$$

Step 3: (Policy Improvement) Obtain a new stationary policy μ_{k+1} satisfying

$$T_{\mu_{k+1}}J_{\mu_k} = TJ_{\mu_k}.$$

If $J_{\mu_k} = TJ_{\mu_k}$ stop; else return to step 2 and repeat the process for new policy μ_{k+1}.

To show that Policy Iteration does indeed converge to the optimal solution we prove the following proposition.

Proposition 2.1 (Policy Improvement Property from Bertsekas 2012, Prop. 2.3.1)**.** *Let μ and $\tilde{\mu}$ be stationary policies such that $T_{\tilde{\mu}}J_\mu = TJ_\mu$, or equivalently, for $i = 1, \ldots, n$,*

$$g(i, \tilde{\mu}(i)) + \alpha \sum_{j=1}^{n} p_{ij}(\tilde{\mu}(i))J_\mu(j) = \min_{u \in U(i)} \left(g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u)J_\mu(i) \right).$$

Then we have

$$J_{\tilde{\mu}}(i) \leq J_\mu(i), \ i = 1, \ldots, n. \tag{2.7}$$

Furthermore, if μ is not optimal, strict inequality holds in the above equation for at least one state i.

Proof. Let the stationary policy μ and $\tilde{\mu}$ fulfil our assumption $T_{\tilde{\mu}}J_\mu = TJ_\mu$. We know from Proposition 1.8 $J_\mu = T_\mu J_\mu$. So we get the following equation for all $i \in X$

$$\begin{aligned} J_\mu(i) &= T_\mu J_\mu \\ &= g(i,\mu(i)) + \alpha \sum_{j=1}^{n} p_{ij}(\mu(i)) J_\mu(j) \\ &\geq \min_{u \in U(i)} \left(g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J_\mu(j) \right) \\ &= (TJ_\mu)(i) \\ &= (T_{\tilde{\mu}} J_\mu)(i). \end{aligned}$$

Applying Proposition 1.8 again we get the desired inequality $J_\mu(i) \geq J_{\tilde{\mu}}(i)$ for all $i \in X$. What is left to be shown is the strict inequality if μ is not an optimal policy. This can be shown analogue to $J_\mu(i) \geq J_{\tilde{\mu}}(i)$ only that as μ is not optimal there exists at least one $i \in X$ with

$$g(i,\mu(i)) + \alpha \sum_{j=1}^{n} p_{ij}(\mu(i)) J_\mu(j) > \min_{u \in U(i)} \left(g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J_\mu(j) \right). \tag{2.8}$$

□

Under the finiteness assumption on X and U, there are only a finite number of possible stationary policies. Hence, with monotonically improving steps, the Policy Iteration converges to an optimal solution in a finite amount of time, which is one of the advantaged of PI compared to other methods not covered in this thesis (e.g. Value Iteration from Bertsekas 2012). Of course, solving the Bellman equation $J_\mu = (\mathbb{I} - \alpha P_\mu) J_\mu$ in Step 2 of PI can be very computationally expensive for very large problems. We will analyse different solvers of this equation in chapter 3 and will also discuss what methods are more or less suitable for larger problems.

2.4 Applying Policy Iteration

Let us now apply the Polity Iteration algorithm to our example of Section 2.2.3. We assume $\alpha = 0.8$ and that the student decides to study more when they performed poorly on their previous exam, but does not study as much if their grade was satisfactory. This gives us our

initial policy $\mu_0 = (0.5h, 0.5h, 0.5h, 2h, 2h)^T$, meaning

$$P_{\mu_0} = \begin{pmatrix} 0 & 0.5 & 0.35 & 0.15 & 0 \\ 0 & 0 & 0.5 & 0.4 & 0.1 \\ 0 & 0 & 0.3 & 0.4 & 0.3 \\ 0 & 0 & 0.1 & 0.6 & 0.3 \\ 0 & 0 & 0 & 0.5 & 0.5 \end{pmatrix}$$

Leading us to the second step, the policy evaluation of $\left(\mathbb{I} - 0.8P_{\mu_0}\right) J_{\mu_0} = g_{\mu_0}$, which looks as follows:

$$\begin{pmatrix} 1 & -0.4 & -0.28 & -0.12 & 0 \\ 0 & 1 & -0.4 & -0.32 & -0.08 \\ 0 & 0 & 0.76 & -0.32 & -0.24 \\ 0 & 0 & -0.08 & 0.52 & -0.24 \\ 0 & 0 & 0 & -0.4 & 0.6 \end{pmatrix} J_{\mu_0} = \begin{pmatrix} -4.55 \\ -0.9 \\ 1.9 \\ 4 \\ 6.5 \end{pmatrix}.$$

To obtain the results of the Policy Iteration algorithm in each step, we utilize the implementation available at `https://github.com/PhilNeitzel/Policy_Iteration` which yields the cost function:

$$J_{\mu_0} = \begin{pmatrix} 10.54999927 \\ 16.64285649 \\ 20.35714229 \\ 22.85714237 \\ 26.07142807 \end{pmatrix}.$$

Which brings us to the next step, the policy improvement, where we need to find a μ_1 satisfying

$$\min_{\mu \in U^5} g_u + \alpha P_u J_{\mu_0} = g_{\mu_1} + \alpha P_{\mu_1} J_{\mu_0}.$$

The solution is the new, improved policy $\mu_1 = \{4h, 4h, 4h, 4h, 4h\}$ so $P_{\mu_1} = P_{4h}$. By repeating the process with μ_1 we receive another linear system:

$$\begin{pmatrix} 0.4 & -0.16 & -0.04 & 0 & 0 \\ -0.32 & 0.68 & -0.16 & 0 & 0 \\ -0.08 & -0.52 & 0.8 & 0 & 0 \\ 0 & -0.4 & -0.24 & 0.84 & 0 \\ 0 & 0 & -0.16 & -0.6 & 0.96 \end{pmatrix} J_{\mu_1} = \begin{pmatrix} -5.1 \\ -3.6 \\ -2.55 \\ -0.9 \\ 2.95 \end{pmatrix}.$$

Solving this linear equation then yields

$$J_{\mu_1} = \begin{pmatrix} -22.79891267 \\ -20.43478215 \\ -18.74999947 \\ -16.15941971 \\ -10.15172032 \end{pmatrix}.$$

As $\|TJ_{\mu_1} - J_{\mu_1}\| = 3.537217643680566^{-7}$ the optimality condition is satisfied and the algorithm stops with the optimal policy $\mu^* = \mu_1$. Meaning, the student should always study the full 4 hours for his next exam in order to optimize his results. Of course, in most Markov Decision Processes the optimal solution would not be the same for every state.

2.5 Convergence of Policy Iteration

Even though we assumed U to be a finite space at the beginning of this chapter it is also interesting to analyse the convergence rate of Policy Iteration for U being an infinite space of actions. For this, we use Bokanowski, Maroso, and Zidani 2009 as guidance. Our goal to find a solution to a non-linear problem stay the same, i.e., the goal is to

$$\text{find } J_\mu \in \mathbb{R}^n \text{ such that } \min_{\mu \in U}((\mathbb{I} - \alpha P_\mu)J_\mu - g_\mu) = 0, \tag{2.9}$$

which is equivalent to finding the optimal policy μ^* that satisfies $J_{\mu^*} = g_{\mu^*} + \alpha P_{\mu^*} J_{\mu^*}$. However, before we look at the convergence rate of the Policy Iteration we have to show the general convergence of the algorithm. In order to do so we recall the following definition.

Definition 2.2 (Monotonicity from Bokanowski, Maroso, and Zidani 2009, Sec. 2)**.** *A matrix $A \in \mathbb{R}^{n \times n}$ is said to be monotone if and only if A is invertible and $A^{-1} \geq 0$ (component wise).*

Throughout this section we use the following assumptions.

$$\text{For every } \mu \in U, \text{ the matrix } (\mathbb{I} - \alpha P_\mu) \text{ is monotone.} \tag{H1}$$

$$\begin{aligned} &\text{When } U(i) \text{ is an infinite compact set } \forall i \in \{1, \dots, n\}, \text{ the functions} \\ &\mu \in U \mapsto (\mathbb{I} - \alpha P_\mu) \in M \text{ and } \mu \in U \mapsto g_\mu \in \mathbb{R}^N \text{ are continuous.} \end{aligned} \tag{H2}$$

Note that U is simply $U = U(1) \times U(2) \times \cdots \times U(n)$ with $U(i)$ being the space of actions in state i. The following theorem summarizes the general convergence of Policy Iteration for different U.

Theorem 2.3 (Bokanowski, Maroso, and Zidani 2009, Thm. 2.1)**.** *Assume that (H1) - (H2) hold. Then there exists a unique J_{μ^*} in $\mathbb{R}^n$ satisfying $\min_{\mu \in U}((I - \alpha P_\mu)J_\mu - g_\mu) = 0$. Moreover, the sequence J_{μ_k} given by the Policy Iteration satisfies the following:*

(i) $J_{\mu_k} \geq J_{\mu_{k+1}}$ $\forall k \geq 0$.

(ii) When U is finite, the algorithm converges in at most card(U) iterations.

(iii) If U is an infinite compact set, then $J_{\mu_k} \to J_{\mu^*}$ when $k \to +\infty$.

Proof. We already showed that J_{μ^*} is a unique solution by Proposition 1.9.

(i) $J_{\mu_k} \geq J_{\mu_{k+1}}$ follows immediately from Proposition 2.1.

(ii) Assume that $U(i)$ is finite for all $i \in \{1, \ldots, n\}$. Hence, there are at most $\text{card}(U)$ different variables $\mu \in U$. Then there exist two indices k, l such that $0 \leq k \leq l \leq \text{card}(U)$ and $\mu_k = \mu_l$ (μ_k and μ_l being, respectively, the k-th and l-th iterate of PI). Hence, $J_{\mu_k} = J_{\mu_l}$, and since $J_{\mu_k} \geq J_{\mu_{k+1}} \geq \cdots \geq J_{\mu_l}$, we also obtain $J_{\mu_{k+1}} = J_{\mu_k}$. Therefore, the Policy Iteration stops at the $(k+1)$-th iteration. This proves that $J_{\mu_k} = J_{\mu_{k+1}}$ is a solution of Equation (2.9), since $J_{\mu_k} = T_{\mu_k} J_{\mu_k} = T_{\mu_{k+1}} J_{\mu_k} = T J_{\mu_k}$.

(iii) We now consider $U(i)$ as an infinite compact set for all $i \in \{1, \ldots, n\}$. By step 2 of the Algorithm 1 we have

$$\|J_{\mu_k}\| \leq \|(\mathbb{I} - \alpha P_{\mu_k})^{-1} g_{\mu_k}\| \leq \max_{\mu \in U} \|(\mathbb{I} - \alpha P_\mu)^{-1}\| \|g_\mu\|. \tag{2.10}$$

By assumptions (H1) and (H2), the function $(\mathbb{I} - \alpha P_\mu)^{-1}$ is continuous on U. Moreover, g_μ is continuous and U is a compact set.
Therefore, from the inequality in Equation (2.10), we deduce that $(J_{\mu_k})_{k \in \mathbb{N}}$ is bounded in $\mathbb{R}^n$. Here J_{μ_k} converges toward some $J_{\mu^*} \in \mathbb{R}^n$. Let us show that J_{μ^*} is the solution of Equation (2.9).
Define the function F for J_{μ_k} by

$$F(J_{\mu_k}) := \min_{\mu \in U}((\mathbb{I} - \alpha P_\mu) J_\mu - g_\mu)) \tag{2.11}$$

and let $F_i(J_{\mu_k})$ be the i-th component of $F(J_\mu)$, so

$$F_i(J_{\mu_k}) = \min_{\mu \in U}((\mathbb{I} - \alpha P_\mu) J_\mu - g_\mu)_i.$$

It is obvious that $\lim_{k \to +\infty} F_i(J_{\mu_k}) = F_i(J_{\mu^*})$. By the compactness of $U(i)$ and using a diagonal extraction argument, there exists a subsequence of $(\mu_k)_{k \in \mathbb{N}}$ denoted by μ_{ϕ_k} that converges toward some $\mu \in U$. Furthermore, with assumption (H2), we have

$$\lim_{k \to \infty} ((\mathbb{I} - \alpha P_{\mu_{\phi_k}}) J_{\mu_k})_i - ((\mathbb{I} - \alpha P_{\mu^*}) J_{\mu^*})_i.$$

Passing to the limit in $((\mathbb{I} - \alpha P_{\mu_{\phi_k}}) J_{\mu_{\phi_k}} - g_{\mu_{\phi_k}})_i = 0$, we deduce that $((\mathbb{I} - \alpha P_{\mu^*}) J_{\mu^*} - g_{\mu^*}))_i = 0$ for all $i \in \{1, \ldots, n\}$. On the other hand, we also have

$$\begin{aligned} F_i(J_{\mu^*}) &= \lim_{k \to \infty} F_i(J_{\mu_{\phi_k - 1}}) \\ &= \lim_{k \to +\infty} ((\mathbb{I} - \alpha P_{\mu_{\phi_k}}) J_{\mu_{\phi_k - 1}} - g_{\mu_{\phi_k}})_i \\ &= ((\mathbb{I} - \alpha P_{\mu^*}) J_{\mu^*} - g_{\mu^*})_i. \end{aligned}$$

Hence, $F_i(J_{\mu^*}) = 0$, which concludes the proof and implies that J_{μ^*} is a solution of Equation (2.9).

□

Having shown the general convergence of the Policy Iteration, for the rest of this chapter, we assume U to be an infinite compact set. To see if the algorithm converges superlinearly it is easiest to rewrite the Policy Iteration as a Newton-like method with the goal of finding the solution of the function $F : \mathbb{R}^n \to \mathbb{R}^n$, $F(J_\mu) := \min_{\mu \in U}((\mathbb{I} - \alpha P_\mu)J_\mu - g_\mu)$. So step 2 and 3 of Algorithm 1 can be rewritten as

$$(\mathbb{I} - \alpha P_{\mu_{k+1}})J_{\mu_{k+1}} - g_{\mu_{k+1}} = 0 \qquad \text{(Policy Evaluation)}$$

$$(\mathbb{I} - \alpha P_{\mu_{k+1}})J_{\mu_k} - g_{\mu_{k+1}} = F(J_{\mu_k}) \qquad \text{(Policy Improvement)}$$

Meaning, $(\mathbb{I} - \alpha P_{\mu_{k+1}})(J_{\mu_k} - J_{\mu_{k+1}}) = F(J_{\mu_k})$ leading to the iteration

$$J_{\mu_{k+1}} = J_{\mu_k} - (\mathbb{I} - \alpha P_{\mu_{k+1}})^{-1} F(J_{\mu_k}). \tag{2.12}$$

For the following result we define the set of minimizers μ for a given cost function $J \in \mathbb{R}^n$ as

$$U_J := \{\mu \in U | (\mathbb{I} - \alpha P_\mu)J - g_\mu = F(J)\}. \tag{2.13}$$

Since U is simply the cartesian product of $U(i)$ we define $U_J(i)$ as the minimizer for the i-th component

$$U_J(i) := \{\mu_i \in U(i) | (e_i^T - \alpha P_{\mu,i})J + g_\mu(i) = F_i(J)\}. \tag{2.14}$$

Furthermore, we denote μ^J if $\mu^J \in U_J$. Using these definitions we prove the following lemma.

Lemma 2.4 (Bokanowski, Maroso, and Zidani 2009, Lem. 3.2)**.** *Assume that (H1) - (H2) hold. For every $J \in \mathbb{R}^n$ and for all $i \in \{1, \dots, n\}$,*

$$d(\mu_i^{J+h}, U_J(i)) \to 0 \text{ as } h \in \mathbb{R}^n \text{ and } \|h\| \to 0, \text{ with } \mu_i^{J+h} \in U_{J+h}(i),$$

where d is the metric on $\mathbb{R}^n$.

Proof. Follows directly from assumptions (H1) - (H2) and the definition of U_J and $U_J(i)$. □

We now use this lemma to prove the superlinear convergence of the Policy Iteration.

Theorem 2.5 (Bokanowski, Maroso, and Zidani 2009, Thm. 3.4)**.** *Assume that (H1) - (H2) are satisfied. Then $\min_{\mu \in U}((\mathbb{I} - \alpha P_\mu)J_\mu + g_\mu) = 0$ has a unique solution $J_{\mu^*} \in \mathbb{R}^n$ and for any initial guess $\mu_0 \in U$, the Policy Iteration converges globally ($\lim_{k \to \infty} \|J_{\mu_k} - J_{\mu^*}\| = 0$) and super linearly i.e.,*

$$\|J_{\mu_{k+1}} - J_{\mu^*}\| = o(\|J_{\mu_k} - J_{\mu^*}\|) \text{ as } k \to \infty.$$

Proof. As existence, uniqueness and the global convergence toward J_{μ^*} have already been proven in Theorem 2.3, we only need to show the superlinear convergence of J_{μ_k}. First let us consider $h_k := J_{\mu_k} - J_{\mu^*}$ and denote $\mu_{k+1} := \mu^{J_{\mu_k}} = \mu^{J_{\mu^*}+h_k}$. Due to Lemma 2.4 we know there exists a $\mu^J \in U_J$ such that $d(\mu_i^{J+h}, \mu_i^J) \to 0$ as $\|h\| \to 0$ for all $i \in \{1, \dots, n\}$. So because of our assumption (H2) we know

$$\lim_{h \to 0} \|(\mathbb{I} - \alpha P_{\mu^{J+h}}) - (\mathbb{I} - \alpha P_{\mu^J})\| = 0.$$

It follows that we can find a $\mu^{k,*} \in U_{J_{\mu^*}}$ such that

$$(\mathbb{I} - \alpha P_{\mu_{k+1}}) - (\mathbb{I} - \alpha P_{\mu_{k,*}}) \to 0 \text{ for } k \to \infty. \tag{2.15}$$

Using the definition of F it can be easily obtained that F is a concave function, as for any $\beta \in [0, 1]$:

$$\begin{aligned} F((1-\beta)x + \beta y) &= \min_{\mu \in U} (\mathbb{I} - \alpha P_\mu)((1-\beta)x + \beta y) - g_\mu \\ &= \min_{\mu \in U} (1-\beta)((\mathbb{I} - \alpha P_\mu)x - g_\mu) + \beta((\mathbb{I} - \alpha P_\mu)y - g_\mu) \\ &\geq \min_{\mu \in U} (1-\beta)((\mathbb{I} - \alpha P_\mu)x - g_\mu) + \min_{\mu \in U} \beta((\mathbb{I} - \alpha P_\mu)y - g_\mu) \\ &= (1-\beta)F(x) + \beta F(y) \end{aligned}$$

Additionally, we know $F(J_{\mu^*}) = 0$ and hence $F(J_{\mu_k}) \leq (\mathbb{I} - \alpha P_{\mu_{k+1}})(J_{\mu_k} - J_{\mu^*})$. Using Equation (2.12) we have

$$\begin{aligned} J_{\mu_{k+1}} &= J_{\mu_k} - (\mathbb{I} - \alpha P_{\mu_{k+1}})^{-1} F(J_{\mu_k}) \\ &\geq J_{\mu_k} - (\mathbb{I} - \alpha P_{\mu_{k+1}})^{-1} (\mathbb{I} - P_{\mu^{k,*}})(J_{\mu_k} - J_{\mu^*}) \end{aligned}$$

and subtracting J_{μ^*} on both sides yields

$$0 \geq J_{\mu_{k+1}} - J_{\mu^*} \geq \left(\mathbb{I} - (\mathbb{I} - \alpha P_{\mu_{k+1}})^{-1}(\mathbb{I} - \alpha P_{\mu_{k,*}})\right)(J_{\mu_k} - J_{\mu^*}). \tag{2.16}$$

By Equation (2.15) and (H1), we obtain $\mathbb{I} - (\mathbb{I} - P_{\mu_{k+1}})^{-1}(\mathbb{I} - \alpha P_{\mu_{k,*}}) \overset{k \to +\infty}{\longrightarrow} 0$. Giving us the result

$$0 \geq J_{\mu_{k+1}} - J_{\mu^*} \geq o(J_{\mu_k} - J_{\mu^*}),$$

which concludes the proof of superlinear convergence. □

3 Methods for Solving Linear Systems

As discussed in the last chapter, to complete Step 2 of the Policy Iteration Algorithm, a system of linear equations needs to be solved. In this chapter we discuss different iterative solving algorithms and their efficiency. It might also be interesting to see if and how those algorithms can be interpreted when applied to a MDP.

3.1 Gauss-Seidel and Jacobi Method

Beginning with some well-known algorithms, this section discusses the Jacobi and Gauss-Seidel method. Before applying those methods to a MDP, the methods will be introduced for the linear system $Ax = b$. Both methods will be described similarly as in Herzog 2022. We denote

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}, \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}, \qquad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

In the *Jacobi method* A will be decomposed into a diagonal component D, a lower triangular component L and an upper triangular component U. Leading to the following matrices

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}, \qquad L + U = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{pmatrix}. \tag{3.1}$$

For an initial guess $x^{(0)}$ we can solve $Ax = b$ iteratively with

$$x^{(k+1)} = D^{-1}(b - (L + U)x^{(k)}). \tag{3.2}$$

Of course, in order for the Algorithm to be well-defined D needs to be invertible. Meaning, A cannot have any zeros on its diagonal.

Algorithm 2 Jacobi Method from Herzog 2022, Sec. 25

$x := x^{(0)}$
while convergence not reached **do**
  $y := x$
  for $i = 1, \dots, n$ **do**
    $x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{n} a_{ij} y_j \right)$
  end for
end while

The *Gauss-Seidel method* works very similar to the Jacobi method, with the difference being that A is decomposed into $L^* = L + D$ and U as defined in Equation (3.1). So the iteration looks as follows

$$x^{(k+1)} = (L^*)^{-1}(b - Ux^{(k)}). \tag{3.3}$$

Algorithm 3 Gauss-Seidel Method

$x := x^{(0)}$
while convergence not reached **do**
  for $i = 1, \dots, n$ **do**
    $x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} x_j \right)$
  end for
end while

The goal now is to look at the algorithms in the environment of MDPs. Applying the Policy Iteration algorithm to the model described in Section 2.1 leads to the following *Policy Evaluation*

$$(\mathbb{I} - \alpha P_\mu) J_\mu = g_\mu.$$

Meaning, the general linear system $Ax = b$ is replaced as follows

$$A = \mathbb{I} - \alpha P_\mu, \qquad x = J_\mu, \qquad b = g_\mu.$$

Note because $\mathbb{I}$ is the identity matrix it is enough to only divide $P_\mu = P_\mu^D + P_\mu^L + P_\mu^U$ so with this notation the iterators can be written as

$$\begin{aligned} & (\mathbb{I} - \alpha P_\mu^D) J_\mu^{(k+1)} = g_\mu + \alpha P_\mu^{L+U} J_\mu^{(k)} && \text{(Jacobi)} \\ \Leftrightarrow \quad & J_\mu^{(k+1)} = g_\mu + \alpha P_\mu^{L+U} J_\mu^{(k)} + \alpha P_\mu^D J_\mu^{(k+1)}, \end{aligned}$$

$$(\mathbb{I} - \alpha P_\mu^{L^*}) J_\mu^{(k+1)} = g_\mu + \alpha P_\mu^U J_\mu^{(k)} \qquad \text{(Gauss-Seidel)}$$
$$\Leftrightarrow \qquad J_\mu^{(k+1)} = g_\mu + \alpha P_\mu^U J_\mu^{(k)} + \alpha P_\mu^{L^*} J_\mu^{(k+1)}.$$

We understand J_μ to be the overall cost function calculated for a stationary policy μ. It can be seen that for every iterator in the Jacobi method the cost function changes only for the diagonal part D of matrix P_μ, as we calculate the expected cost for the rest of the problem using the previous cost function with $\alpha P_\mu J_\mu^(k)$. Meaning, depending on whether $J_\mu^{(k+1)} \leq J_\mu^{(k)}$ or $J_\mu^{(k+1)} \geq J_\mu^{(k)}$ the cost for staying in the same state falls or rises. Meanwhile, the cost for going back or forth to a different state still uses the older cost function, so the cost is assumed to stay the same.
Similarly to the Jacobi method, the Gauss-Seidel method keeps the previous cost function for going a state forward but if one stays in the same state or goes back to an earlier state the cost function is set to be the new one. So again it might become more or less expensive to move backward or stay in the same place. Of course, the new cost function still needs to be the expected cost of the overall problem in order for it to be used as the next iterate.

Furthermore, we do not have to worry about the general convergence of Jacobi and Gauss-Seidel method as both converge when the matrix A is *strictly diagonally dominant*. Meaning,

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \text{ for all } i \in \{1, \ldots, n\}.$$

Due to the properties of αP_μ, we can easily see that $(\mathbb{I} - \alpha P_\mu)$ is always strictly diagonally dominant.

The following theorem compares the speed of convergence of both methods.

Theorem 3.1 (Bradie 2006, p. 233)**.** *Suppose A is an $(n \times n)$ matrix. If $a_{ii} > 0$ for each i and $a_{ij} \leq 0$ for every $i \neq j$, then one and only one of the following statements holds*

(i) $0 \leq \rho(T_{gs}) < \rho(T_{jac}) < 1$,

(ii) $1 < \rho(T_{jac}) < \rho(T_{gs})$,

(iii) $\rho(T_{jac}) = \rho(T_{gs}) = 1$,

(iv) $\rho(T_{jac}) = \rho(T_{gs}) = 0$.

where T is the iteration matrix that arises for each method and $\rho(T)$ is the spectral radius.

We are able to apply this theorem to our MDPs as $1 - \alpha(P_\mu)_{ii} > 0$ and $-\alpha(P_\mu)_{ij} \leq 0$ for all $i \neq j$. This theorem in general provides the information that the methods either both converge or

both diverge. However, we already know that both methods converge so from this theorem we can see that the Gauss-Seidel method generally converges faster than the Jacobi method as $0 \leq \rho(T_{gs}) \leq \rho(T_{jac}) < 1$. This means we would always choose Gauss-Seidel over Jacobi when it comes to MDPs.

3.2 Richardson Iteration

Another stationary method that may come to mind is the so called *Richardson Iteration*. The idea is to simply rewrite the linear system $Ax = b$ as a fixed point equation in order to derive a fixed-point iteration. We write the problem as follows

$$x = x + b - Ax = b + (\mathbb{I} - A)x$$

and receive the iterators

$$x^{(k+1)} = b + (\mathbb{I} - A)x^{(k)}.$$

Applying this to the model of MDPs with $(\mathbb{I} - \alpha P_\mu)J_\mu = g_\mu$ the iterators $J_\mu^{(k)}$ will be calculated with

$$\begin{aligned} J_\mu^{(k+1)} &= g_\mu + (\mathbb{I} - (\mathbb{I} - \alpha P_\mu))J_\mu^{(k)} \\ &= g_\mu + \alpha P_\mu J_\mu^{(k)}. \end{aligned}$$

We can see immediately that this is the fixed-point iteration for Equation (2.5) and by rewriting $J_\mu^{(k+1)}$ as

$$\begin{aligned} J_\mu^{(k+1)} &= g_\mu + \alpha P_\mu (g_\mu + \alpha P_\mu J_\mu^{(k-1)}) \\ &= g_\mu + \alpha P_\mu g_\mu + \alpha^2 P_\mu^2 J_\mu^{(k-1)} \\ &= \dots \\ &= \sum_{i=0}^{k} \alpha^j P_\mu^i g_\mu + \alpha^{k+1} P_\mu^{k+1} J_\mu^{(0)}. \end{aligned}$$

We can see that $J_\mu^{(k+1)}$ is nothing else than the $k+1$-stage expected cost as in Equation (1.8), where the terminal cost is our initial guess $J_\mu^{(0)}$. However, as k grows larger and $0 < \alpha < 1$ this initial guess will become less and less relevant for the next iteration. So the Richardson iteration can be interpreted as calculating the k-th stage expected cost.

3.3 Krylov Subspace Methods

This part discusses the idea of Krylov subspace methods (as presented in Saad 2003) and their application on MDPs. As Krylov subspace methods consist of the idea of applying projection

methods to a Krylov subspace, those projection methods will be introduced first. In general, a projector P is a mapping from $\mathbb{C}^n$ to itself with the idempotence property

$$P^2 = P.$$

Furthermore, the projection can be split into two subspaces

$$\begin{aligned} \text{Null}(P) &= \{x \in \mathbb{C}^n | Px = 0\}, \\ \text{Ran}(P) &= \{Px | x \in \mathbb{C}^n\}. \end{aligned}$$

Of course, those subspaces only intersect each other at the element zero. Meaning, $\mathbb{C}^n$ can be written as the direct sum of both of them

$$\mathbb{C}^n = \text{Null}(P) \oplus \text{Ran}(P).$$

Simultaneously, every two subspaces M and S with $\mathbb{C}^n = S \oplus M$ define a unique projector with $\text{Null}(P) = S$ and $\text{Ran}(P) = M$. Additionally, we can easily see that $\text{Null}(P) = \text{Ran}(\mathbb{I} - P)$. Since if P is a projector, then $\mathbb{I} - P$ is a projector too. Knowing this S can be rewritten as $S = \text{Ran}(\mathbb{I} - P)$ with the projector P mapping any $x \in \mathbb{C}^n$ onto the unique decomposition $x = x_1 + x_2$, where x_1 is the M-component and x_2 the S-component. Thus, any vector x can be written as $x = Px + (\mathbb{I} - P)x$ and hence Px satisfies the following conditions

$$\begin{aligned} Px &\in M, \\ x - Px &\in S. \end{aligned}$$

Assuming P is of rank m, the range of $\mathbb{I} - P$ will be of dimension $n - m$. By using the orthogonal complement $L = S^{\perp}$, where every vector in L is orthogonal to every vector in S. The above conditions can be rewritten as

$$Px \in M, \tag{3.4}$$

$$x - Px \perp L. \tag{3.5}$$

These conditions define a projector P onto M and orthogonal to the subspace L. The following Lemma proves it is indeed always possible to define a projection through the conditions (3.4) and (3.5).

Lemma 3.2 (Saad 2003, Lem. 1.36)**.** *Given two subspaces M and L of the same dimension m, the following two conditions are mathematically equivalent.*

(i) No non-zero vector of M is orthogonal to L.

(ii) For any x in $\mathbb{C}^n$ there exists a unique vector u which satisfies the conditions (3.4) and (3.5).

Proof. The proof follows from the definitions given above. □

It is useful to represent the projector P via a basis $V = [v_1, \ldots, v_m]$ of the subspace $M = Ran(P)$ and the basis of the subspace L as $W = [w_1, \ldots, w_m]$. This leads to projection methods where the goal is to find an approximate solution to the linear system $Ax = b$ on a subspace of $\mathbb{R}^n$. This subspace will be denoted as K and $\tilde{x} \in K$ can be thought of as a candidate for approximation. Considering the dimension of K is equal to m, m constraints should be sufficient to be able to obtain an approximation. A typical way of describing these constraints is to impose m (independent) orthogonality conditions. Specifically, the residual vector $b - Ax$ is constrained to be orthogonal to m linearly independent vectors. This defines another subspace L of dimension m called the subspace of constraints. This simple framework is common to many different mathematical methods and is known as the *Petrov Galerkin condition*. There are two broad classes of projection methods: orthogonal and oblique. In an orthogonal projection technique, the subspace of L is the same as K. In an oblique projection method, L is different from K and may not have any relation to it. However, in this thesis the focus will lie on the oblique projection technique $L = AK$. In total, for a matrix $A \in R^{n \times n}$ and K and L being two m-dimensional subspaces of $\mathbb{R}^n$ the following conditions apply for an approximate solution $\tilde{x}$.

$$\text{find } \tilde{x} \in K, \text{ such that } b - A\tilde{x} \perp L.$$

Applying an initial guess to those conditions $\tilde{x}$ can be rewritten as $\tilde{x} = x_0 + \delta$ where $\delta \in K$ and $x_0 \in \mathbb{R}^n$. With the residual vector r_0 being defined as $r_0 = b - Ax_0$ the condition can be written as

$$\text{find } \tilde{x} \in x_0 + K, \text{ such that } r_0 - A\delta \perp L.$$

Using the basis matrix representation V and W introduced above, the approximate solution can also be seen as

$$\tilde{x} = x_0 + Vy,$$

with the orthogonality condition leading to

$$W^T AVy = W^T r_0.$$

In general, this is considered to be the basic step of the projection method. Most methods will apply this step multiple times updating the residual r_0 and the initial guess x_0 in each step to find an approximation of the solution x.
This leads us to the Krylov subspace methods which define K_m as

$$K_m = span\{r_0, Ar_0, A^2 r_0, \ldots, A^{m-1} r_0\}, \tag{3.6}$$

with r_0 being the residual and x_0 an arbitrary initial guess. As mentioned above L_m is chosen as $L_m = AK_m$ in this thesis. Of course, different Krylov subspace methods arise when using a different subspace L_m. In general, the method approximates x as x_m through a polynomial q_{m-1} of degree $m - 1$

$$A^{-1}b \approx x_m = x_0 + q_{m-1}(A)r_0.$$

3.3.1 Arnoldi's Method

The Arnoldi method introduces one of the basic Krylov subspace methods, laying the foundation for the algorithms discussed later in this chapter. The goal of this method is to reduce

the matrix A to a Hessenberg matrix H and to calculate an orthonormal basis $v_1, \ldots, v_m$ of the Krylov subspace. Note that a square $n \times n$ matrix H is said to be an upper Hessenberg matrix if $a_{ij} = 0$ for all i, j with $i > j + 1$. In this thesis an upper Hesseberg matrix will be simply referred to as a *Hessenberg matrix*. In general, the algorithm works very similar to the Gram-Schmidt method when it comes to the orthonormalization and looks as follows

Algorithm 4 Arnoldi from Saad 2003, Alg. 6.1

choose a vector v_1 such that $||v_1||_2 = 1$
for $j = 1, 2, \ldots, m$ **do**
  for $i = 1, 2, \ldots, j$ **do**
    $h_{ij} = (Av_j, v_i)$
  end for
  $w_j := Av_j - \sum_{i=1}^{j} h_{ij} v_i$
  $h_{j+1,j} = ||w_j||_2$
  if $h_{j+1,j} = 0$ **then**
    Stop
  end if
  $v_{j+1} = \frac{w_j}{h_{j+1,j}}$
end for

We now show this method does indeed create an orthonormal basis.

Proposition 3.3 (Saad 2003, Prop. 6.4)**.** *Assume that Algorithm 4 does not stop before the m-th step. Then the vectors $v_1, v_2, \ldots, v_m$ form an orthonormal basis of the Krylov subspace*

$$K_m = span\{v_1, Av_1, A^2 v_1, \ldots, A^{m-1} v_1\}.$$

Proof. By the construction of v_i with $i \in \{1, \ldots, m\}$ the orthonormality follows immediately. So we only need to prove that the vectors $v_1, \ldots v_m$ do indeed form a basis of K_m. Of course, for $m = 1$ $K_1 = span\{v_1\} \implies v_1$ is the basis of K_m. Now let us assume $v_1, \ldots, v_m$ to be the basis of K_m. For $m + 1$ we get $K_{m+1} = span\{v_1, Av_1, \ldots, A^m v_1\}$ and because of our assumption K_{m+1} can be rewritten as $K_{m+1} = span\{v_1, v_2, \ldots, A^m v_1\}$. Applying the Arnoldi method we receive $h_{im} = (Av_m, v_i)$ for $i \in \{1, \ldots, m\}$, $w_m = Av_m - \sum_{i=1}^{m} h_{im} v_i$ and $v_{m+1} = \frac{w_m}{\|w_m\|}$ This leads to the following equations

$$\begin{aligned} \|w_m\| v_{m+1} &= Av_m - \sum_{i=1}^{m} h_{im} v_i \\ &= Aq_{m-1}(A)v_1 - \sum_{i=1}^{m} h_{im} q_{i-1}(A) v_1. \end{aligned}$$

Which means v_{m+1} can be rewritten as some polynomial of degree m denoted as $q_m(A)v_1$. Proving that $v_1, \ldots, v_{m+1}$ is indeed a basis of K_{m+1}. □

Proposition 3.4 (Saad 2003, Prop. 6.5)**.** *Denote by V_m the $n \times m$ matrix with column vectors $v_1, v_2, \ldots, v_m$, by $\overline{H}_m$ the $(m+1) \times m$ Hessenberg matrix whose non-zero entries h_{ij} are defined by Algorithm 4, and by H_m the matrix obtained from $\overline{H}_m$ by deleting its last row. Then the following relation holds*

$$\begin{aligned} AV_m &= V_m H_m + w_m e_m^T \\ &= V_{m+1}\overline{H}_m, \\ V_m^T A V_m &= H_m. \end{aligned}$$

Proof. Applying the Arnoldi method we can rewrite AV_m as follows

$$\begin{aligned} AV_m &= [Av_1, \ldots, Av_m] \\ &= \left[\sum_{i=1}^{2} h_{i1} v_i, \ldots, \sum_{i=1}^{m+1} h_{im} v_i \right] \\ &= \left[\sum_{i=1}^{2} h_{i1} v_i, \ldots, \sum_{i=1}^{m} h_{im} v_i \right] + \left[0, \ldots, 0, h_{m+1,m} v_{m+1} \right] \\ &= V_m H_m + w_m e_m^T \\ &= V_{m+1}\overline{H}_m. \end{aligned}$$

Multiplying V_m^T to both sides gives us

$$V_m^T A V_m = V_m^T V_m H_m + V_m^T w_m e_m^T.$$

As V_m is an orthonormal basis and v_m is orthogonal to w_m by construction. We get

$$V_m^T V_m = \mathbb{I} \qquad\qquad V_m^T w_m e_m^T = \left[0, \ldots, 0, v_m^T w_m \right] = [0, \ldots, 0, 0] .$$

Giving us the desired equation $V_m^T A V_m = H_m$. □

3.3.2 The GMRES Algorithm

We now introduce the *Generalized Minimal Residual Method (GMRES)*. As discussed at the beginning of this chapter we choose $K = K_m$ and $L = AK_m$ with K_m being the Krylov subspace as in Equation (3.6) with $v_1 = \frac{r_0}{\|r_0\|}$. The idea is that GMRES minimizes the norm of the residual on $x_0 + K_m$ for an initial guess x_0 by exploiting the relation of Proposition 3.4. Of course, for every vector $x \in x_0 + K_m$ we can write

$$x = x_0 + V_m y, \tag{3.7}$$

where we define y by the following function

$$\mathcal{J}_m(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2. \tag{3.8}$$

We denote the initial residual $r_0 := b - Ax_0$, $\beta := \|r_0\|_2$ and $v_1 := \frac{r_0}{\beta}$. By then applying Proposition 3.4 we can simplify our problem as follows

$$\begin{aligned} b - Ax &= b - A\left(x_0 + V_m y\right) && (3.9)\\ &= r_0 - AV_m y && (3.10)\\ &= \beta v_1 - V_{m+1}\overline{H}_m y && (3.11)\\ &= V_{m+1}\left(\beta e_1 - \overline{H}_m y\right). && (3.12) \end{aligned}$$

As the vectors of V_{m+1} are orthonormal to one another the norm is independent of V_{m+1}, leading to

$$\mathcal{J}_m(y) \equiv \|b - A\left(x_0 + V_m y\right)\|_2 = \|\beta e_1 - \overline{H}_m y\|_2. \tag{3.13}$$

In general, GMRES approximates the solution of $Ax = b$ by finding the minimizer y_m of $\mathcal{J}_m(y)$ and calculating the unique approximate x_m using y_m. The problem is written as

$$\begin{aligned} x_m &= x_0 + V_m y_m, \quad \text{where} && (3.14)\\ y_m &= arg\min_y \|\beta e_1 - \overline{H}_m y\|_2. && (3.15) \end{aligned}$$

The advantage of this method is that it is typically inexpensive to find the solution y_m as it is a $(m+1) \times m$ least square problem. The GMRES algorithm is summarized as follows

Algorithm 5 GMRES from Saad 2003, Alg. 6.9

Compute $r_0 := b - Ax_0, \beta := \|r_0\|_2$, and $v_1 := \frac{r_0}{\beta}$
for $j = 1, 2, \ldots, m$ **do**
  $w_j := Av_j$
  for $i = 1, \ldots, j$ **do**
    $h_{ij} := (w_j, v_i)$
    $w_j := w_j - h_{ij}v_i$
  end for
  $h_{j+1,j} := \|w_j\|_2$
  if $h_{j+1,j} = 0$ **then**
    $m := j$
  else
    $v_{j+1} := \frac{w_j}{h_{j+1,j}}$
  end if
end for
Define the $(m+1) \times m$ Hessenberg matrix $\overline{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$
Compute y_m the minimizer of $\|\beta e_1 - \overline{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$

The algorithm uses the Arnoldi method to calculate V_{m+1} and the Hessenberg matrix $\overline{H}_m$. In the last step it calculates the minimizer y_m, and with that, the unique approximation $x_m \in x_0 + K_m$.

The *Minimal Residual Method (MINRES)*, this is another algorithm based on a Krylov subspace, which is algebraically equivalent to the GMRES. However for this algorithm we assume A to be a hermitian matrix. Which makes this implementation typically faster than GMRES. As the numerical details of this are beyond the scope of this thesis we are simply taking a look at the implementation given by Günnel, Herzog, and Sachs 2014.

Algorithm 6 MINRES from Günnel, Herzog, and Sachs 2014, Alg. 3.1

Set $v_0 := 0$ and $w_0 := w_1 := 0$
Set $v_1 := b - Ax_0$
Set $\gamma_1 := \|v_1\|_2$
Set $v_1 := \frac{v_1}{\gamma_1}$
Set $\eta_0 := \gamma_1, s_0 := s_1 := 0, c_0 := c_1 := 1$
Set $k := 1$
while not converged **do**
  Set $\delta_k := \langle Av_k, v_k \rangle$
  Set $v_{k+1} := Av_k - \delta_k v_k - \gamma_k v_{k-1}$
  Set $\gamma_{k+1} := \|v_{k+1}\|_2$
  Set $v_{k+1} := \frac{v_{k+1}}{\gamma_{k+1}}$
  Set $\alpha_0 := c_k\delta_k - c_{k-1}s_k\gamma_k$ and $\alpha_1 := \left(\alpha_0^2 + \gamma_{k+1}^2\right)^{\frac{1}{2}}$
  Set $\alpha_2 := s_k\delta_k + c_{k-1}c_k\gamma_k$ and $\alpha_3 := s_{k-1}\gamma_k$
  Set $c_{k+1} := \frac{\alpha_0}{\alpha_1}$ and $s_{k+1} := \frac{\gamma_{k+1}}{\alpha_1}$
  Set $w_{k+1} := \frac{1}{\alpha_1}(v_k - \alpha_3 w_{k-1} - \alpha_2 w_k)$
  Set $x_{k+1} := x_{k-1} + c_{k+1}\eta_{k-1}w_{k+1}$
  Set $\eta_k := -s_{k+1}\eta_{k-1}$
  Set $k := k + 1$
end while

3.3.3 Krylov Subspace Methods on MDPs

Now we take a look at Krylov subspace methods applied to MDPs. To make some sort of interpretation possible we keep the subspace a little simpler by only looking at the orthogonal subspace. So we define the Krylov subspace as follows

$$K_m = span\{r_0, (\mathbb{I} - \alpha P_\mu)r_0, \ldots, (\mathbb{I} - \alpha P_\mu)^{m-1}r_0\},$$

with the residual $r_0 = g_\mu - (\mathbb{I} - \alpha P_\mu)J_0$ and an initial guess J_0. A first idea on the interpretation comes from Bertsekas 2005, Chapter 6 who rewrites the cost vector defined by

$$J_\mu = (\mathbb{I} - \alpha P_\mu)^{-1} g_\mu,$$

as

$$J_\mu = \sum_{t=0}^{\infty} \alpha^t P_\mu^t g_\mu. \tag{3.16}$$

Of course, this property can be easily seen by the Neuman series (a generalized form of the geometric series) but it is also a very intuitive result. As the overall cost vector is simply the expected cost for each time t and P_μ^t is the probability matrix consisting of probabilities p_{ij} for going from state i to state j in exactly t time, we can interpret Equation (3.16) as calculating the expected cost for all time $t \geq 0$ and discounting it by α^t as costs or benefits are of less

interest the further they lie in the future. We can rewrite this function even further using our definition of the residual r_0 and replacing g_μ with it, which results in

$$J_\mu = \sum_{t=0}^{\infty} \alpha^t P_\mu^t (r_0 + (\mathbb{I} - \alpha P_\mu) J_0) \tag{3.17}$$

$$= \sum_{t=0}^{\infty} \alpha^t P_\mu^t r_0 + \sum_{t=0}^{\infty} \alpha^t P_\mu^t (\mathbb{I} - \alpha P_\mu) J_0 \tag{3.18}$$

$$= \sum_{t=0}^{\infty} \alpha^t P_\mu^t r_0 + (\mathbb{I} - \alpha P_\mu)^{-1} (\mathbb{I} - \alpha P_\mu) J_0 \tag{3.19}$$

$$= J_0 + \sum_{t=0}^{\infty} \alpha^t P_\mu^t r_0. \tag{3.20}$$

Using Equation (3.20), given a J_0, an approximation of J_μ can be obtained by truncating the infinite sum at some $m > 0$. Thus, it would make sense if we can rewrite K_m as

$$K_m = span\{r_0, P_\mu r_0, \ldots, P_\mu^{m-1} r_0\}$$

and the following theorem shows that both subspaces are indeed equivalent.

Theorem 3.5. *For all $n \in \mathbb{N}$ and $\alpha \neq 0$ the following holds*

$$x \in span\{r_0, P_\mu r_0, \ldots, P_\mu^n r_0\} \Leftrightarrow x \in span\{r_0, (\mathbb{I} - \alpha P_\mu) r_0, \ldots, (\mathbb{I} - \alpha P_\mu)^n r_0\}. \tag{3.21}$$

Proof. For $n = 1$:

$$\begin{aligned}
& x \in span\{r_0, P_\mu r_0\} \\
\Leftrightarrow \quad & x = \lambda_0 r_0 + \lambda_1 P_\mu r_0 && \text{for some } \lambda_0, \lambda_1 \in \mathbb{R} \\
\Leftrightarrow \quad & x = (\lambda_0 + \frac{\lambda_1}{\alpha} - \frac{\lambda_1}{\alpha}) r_0 - \frac{\lambda_1}{\alpha}(-\alpha) P_\mu r_0 \\
\Leftrightarrow \quad & x = (\lambda_0 + \frac{\lambda_1}{\alpha}) r_0 - \frac{\lambda_1}{\alpha}(\mathbb{I} - \alpha P_\mu) r_0 \\
\Leftrightarrow \quad & x = \beta_0 r_0 + \beta_1 (\mathbb{I} - \alpha P_\mu) r_0 && \text{for } \beta_0 := \lambda_0 + \frac{\lambda_1}{\alpha} \text{ and } \beta_1 := \frac{\lambda_1}{\alpha} \\
\Leftrightarrow \quad & x \in span\{r_0, (\mathbb{I} - \alpha P_\mu) r_0\}
\end{aligned}$$

Assuming Equation (3.21) to be true for a chosen $n \in \mathbb{N}$. The goal is to prove it also holds true for $n + 1$

Recalling $(\mathbb{I} - \alpha P_\mu)^{n+1} = \sum_{i=0}^{n+1} \binom{n+1}{i} (-\alpha P_\mu)^i$ leads to the following result

$$P_\mu^{n+1} = \frac{1}{(-\alpha)^{n+1}} \left((\mathbb{I} - \alpha P_\mu)^{n+1} - \sum_{i=0}^{n} \binom{n+1}{i} (-\alpha P_\mu)^i \right). \tag{$*$}$$

Giving us

$$
\begin{aligned}
& x \in span\{r_0, P_\mu r_0, \ldots, P_\mu^{n+1} r_0\} \\
\Leftrightarrow \quad & x = \sum_{i=0}^{n+1} \lambda_i P_\mu^i r_0 \\
\Leftrightarrow \quad & x = \sum_{i=0}^{n} \lambda_i P_\mu^i r_0 + \lambda_{n+1} P_\mu^{n+1} r_0 \\
\Leftrightarrow \quad & x = \sum_{i=0}^{n} \tilde{\beta}_i (\mathbb{I} - \alpha P_\mu)^i r_0 + \lambda_{n+1} P_\mu^{n+1} r_0 && \text{Ass.} \\
\Leftrightarrow \quad & x = \sum_{i=0}^{n} \tilde{\beta}_i (\mathbb{I} - \alpha P_\mu)^i r_0 + \frac{\lambda_{n+1}}{(-\alpha)^{n+1}} \left((\mathbb{I} - \alpha P_\mu)^{n+1} - \sum_{i=0}^{n} \binom{n+1}{i} (-\alpha P_\mu)^i \right) r_0 && (*) \\
\Leftrightarrow \quad & x = \sum_{i=0}^{n} \tilde{\beta}_i (\mathbb{I} - \alpha P_\mu)^i r_0 + \sum_{i=0}^{n} \hat{\beta}_i (\mathbb{I} - \alpha P_\mu)^i r_0 + \frac{\lambda_{n+1}}{(-\alpha)^{n+1}} (\mathbb{I} - \alpha P_\mu)^{n+1} r_0, && \text{Ass.} \\
\Leftrightarrow \quad & x = \sum_{i=0}^{n} \beta_i (\mathbb{I} - \alpha P_\mu)^i r_0 + \beta_{n+1} (\mathbb{I} - \alpha P_\mu)^{n+1} r_0 \qquad \text{for } \beta_{n+1} := \frac{\lambda_{n+1}}{(-\alpha)^{n+1}} \text{ and } \beta_i := \tilde{\beta}_i + \hat{\beta}_i \\
\Leftrightarrow \quad & x \in span\{r_0, (\mathbb{I} - \alpha P_\mu) r_0, \ldots, (\mathbb{I} - \alpha P_\mu)^{n+1} r_0\}.
\end{aligned}
$$

□

Now it becomes apparent that the Krylov subspace can be interpreted as $K_m = span\{r_0, P_\mu r_0, P_\mu^2 r_0, \ldots, P_\mu^{m-1} r_0\}$. As mentioned before P_μ^k can be interpreted as the probability for going from one state to another in k steps. Using this slightly altered Krylov subspace and taking a look at the general method used to solve linear systems as in Equation (3.14) $x_m = x_0 + V_m y_m$ it can be seen that for MDPs this yields the following equation

$$
\begin{aligned}
J_m &= J_0 + V_m y_m && (3.22) \\
&= J_0 + \sum_{i=1}^{k} (y_m)_i P_\mu^{i-1} r_0, && (3.23)
\end{aligned}
$$

where we define V_m by each column $i \in \{1, \ldots, m\}$ as $v_i = P_\mu^{i-1} r_0$ The question now is if there is something that can be said about y_m. One might notice the similarity between Equation (3.23) and Equation (3.20), which leads us to the hypothesis of the discount factor α being connected to y_m. Indeed y_m is defined by minimizing $\|r_0 - AV_m y\|$ as in Equation (3.10) or in this case

$$
y_m = arg \min_y \|r_0 - (\mathbb{I} - \alpha P_\mu) V_m y\|. \tag{3.24}
$$

It is easily shown that $(y_m)_i := \alpha^{i-1}$ fulfils Equation (3.24) for $m \to \infty$ as

$$\begin{aligned}
\lim_{m\to\infty} \|r_0 - \left(\mathbb{I} - \alpha P_\mu\right) V_m y_m\| = & \quad \lim_{m\to\infty} \|r_0 - \left(\mathbb{I} - \alpha P_\mu\right) \sum_{i=1}^{m} (y_m)_i P_\mu^{i-1} r_0\| \\
= & \quad \lim_{m\to\infty} \|r_0 - \left(\mathbb{I} - \alpha P_\mu\right) \sum_{i=1}^{m} \alpha^{i-1} P_\mu^{i-1} r_0\| && \text{Ass.} \\
= & \quad \|r_0 - \left(\mathbb{I} - \alpha P_\mu\right) \sum_{i=0}^{\infty} \alpha^{i} P_\mu^{i} r_0\| \\
= & \quad \|r_0 - (\mathbb{I} - \alpha P_\mu)(J_\mu - J_0)\| && \text{by (3.23)} \\
= & \quad \|g_\mu + \alpha P_\mu J_0 - J_0 - J_\mu + \alpha P_\mu J_\mu + J_0 - \alpha P_\mu J_0\| \\
= & \quad \| \underbrace{g_\mu - (\mathbb{I} - \alpha P_\mu) J_\mu}_{=0} \|.
\end{aligned}$$

The problem however is that a unique solution for y_m can only be guaranteed for $m \leq n$ and as it is not possible to calculate y_m for the generalized problem we cannot say anything of interest about y_m. On the contrary, we can assume it is very unlikely for y_m to behave similar to the discount factor as it is not restricted to be positive and will take negative values as well, unlike α.

When it comes to the convergence of Krylov subspace methods they are usually considered most efficient when applied to systems with large sparse matrices. Especially for large models of MDPs matrix P is often sparse, as we usually cannot reach most of the states from the state one is currently in. Of course, all the mentioned methods can always be made more efficient with different preconditioners. However, from all the different methods introduced in this chapter, the Krylov subspace methods seem to be the most promising when it comes to efficiency.

4 Conclusion

In conclusion, Markov Decision Processes can be a useful tool for decision making and we can optimize our MDP using the Policy Iteration algorithm which requires us to solve a linear system. Depending on the size of our problem we saw that different iterative solvers might perform better than others. Jacobi method and Richardson iteration, for example, converge rather slow for large matrices. While, Krylov subspace methods converge a lot faster for large and sparse matrices. Furthermore, we tried to interpret the iterators of those methods when applied to MDPs in order to understand the process of calculating the expected cost vector in a more intuitive way. However, due to the complexity of Krylov subspace methods we were unable to discover anything of interest. Nevertheless, with the information provided in this thesis we are able to fully implement the Policy Iteration and apply it to different MDPs, while also being able to choose between different solvers in order to solve the problems efficiently. For a full implementation of MDPs and the different linear solvers discussed in this thesis see `https://github.com/PhilNeitzel/Policy_Iteration`.

Bibliography

Bertsekas, Dimitri P. (2005). *Dynamic Programming and Optimal Control Volume 1*. 3rd ed. Athena Scientific.

– (2012). *Dynamic Programming and Optimal Control Volume 2*. 4th ed. Athena Scientific.

Bokanowski, Olivier, Stefania Maroso, and Hasnaa Zidani (2009). "Some Convergence Results for Howard's Algorithm". In: *SIAM Journal on Numerical Analysis* 47, pp. 3001–3026.

Bradie, Brian (2006). *A Friendly Introduction to Numerical Analysis*. Pearson.

Günnel, Andreas, Roland Herzog, and Ekkehard Sachs (2014). "A note on preconditioners and scalar products in Krylov subspace methods for self-adjoint problems in Hilbert space". In: *Electronic Transactions on Numerical Analysis* 41, pp. 13–20.

Herzog, Roland (2022). *Einführung in die Numerik*. Last accessed 28th March 2023. URL: https://scoop.iwr.uni-heidelberg.de/teaching/2022ss/lecture-einfuehrung-in-die-numerik/einf%c3%bchrung-in-die-numerik-skript-20220717.pdf.

Johannes, Jan (2021). *Wahrscheinlichkeitstheorie 1*. Last accessed 6th Febuary 2023. URL: https://sip.math.uni-heidelberg.de/vl/wt1-ss21/src/Skript-WT1-SS21-%C2%A701-%C2%A719.pdf.

Saad, Yousef (2003). *Iterative Methods for Sparse Linear Systems*. 2nd ed. Society for Industrial and Applied Mathematics.

5 Notation

u	chosen action
U	space of all actions
w	disturbance that may occur
W	space of all disturbances
x	current state
X	space of all states
$\mathbb{I}$	Identity matrix
$\mathbb{E}$	Expected value
α	discount factor
μ	policy that determines the action
g_μ	stationary cost vector for a given policy
P_μ	transition matrix
J_μ	total cost vector for a given policy
TJ	mapping for minimal expected cost given a cost vector J
$T_\mu J$	mapping for expected cost given a policyμ and cost vector J_μ
K_m	Krylov subspace
V_m	orthognormal basis of the Krylov subspace
H_m	upper Hessenberg matrix