Mathematical Statistics
Stockholm University

# Premium control with reinforcement learning

Lina Palmborg
Filip Lindskog

## Research Report 2022:7

ISSN 1650-0377

**Postal address:**
Mathematical Statistics
Dept. of Mathematics
Stockholm University
SE-106 91 Stockholm
Sweden

**Internet:**
http://www.math.su.se

Mathematical Statistics
Stockholm University
Research Report **2022:7**,
http://www.math.su.se

# Premium control with reinforcement learning

Lina Palmborg and Filip Lindskog

July 2022

**Abstract**

We consider a premium control problem in discrete time, inspired by [9] and [10], formulated in terms of a Markov decision process. In a simplified setting, the optimal premium rule can be derived with dynamic programming methods. However, these classical methods are not feasible in a more realistic setting due to the dimension of the state space. Hence, to combat the curse of dimensionality we explore reinforcement learning techniques, using linear function approximation. We illustrate the appropriateness of the approximate optimal premium rule compared with the true optimal premium rule in a simplified setting, and further demonstrate that the approximate optimal premium rule outperforms benchmark rules in a more realistic setting where classical approaches fail.

## 1 Introduction

An insurance company's claim costs and investment earnings fluctuate randomly over time. The insurance company needs to determine the premiums before the coverage periods start, i.e. before knowing knowing what claim costs will appear and without knowing how its invested capital will develop. Hence, the insurance company is facing a dynamic stochastic control problem. The problem is complicated because of delays and feedback effects: premiums are paid before claim costs materialise and premium levels affect whether the company attracts or loses customers.

An insurance company wants a steady high surplus. The optimal dividend problem introduced by de Finetti in [4] (and solved by Gerber in [5]) has the objective to maximise the expected present value of future dividends. Its solution takes into account that paying dividends too generously is suboptimal since a too high probability of default affects the expected present value of future dividends negatively. A practical problem with implementing the optimal premium rule, i.e. a rule that maps the state of the stochastic environment to a premium level, obtained from solving the optimal dividend problem is that the premiums would be fluctuating more than what would be feasible for a real insurance market with competition. A good premium rule needs to generate premiums that do not fluctuate wildly over time.

For a mutual insurance company, maximising dividends is not the main objective. Instead the premiums should be low and suitably averaged over time but also making sure that the surplus is sufficiently high to avoid a too high probability of default. This is the situation we study in the present paper. Similar premium control problems have been studied in the papers [9] and [10] by Martin-Löf and these papers have been a source of inspiration for our work.

The paper [9] carefully sets up the balance equations for the key economic variables of relevance for the performance of the insurance company and studies the premium control problem as a linear control problem under certain simplifying assumptions enabling application of linear control theory. The paper analyses the effects of delays in the insurance dynamical system on the linear control law with feedback and discusses designs of the premium control that ensure that the probability of default is small.

The paper [10] considers an application of general optimal control theory in a setting similar to, but simpler than, the setting considered in [9]. The paper derives and discusses the optimal premium rule that achieves low and averaged premiums and also targets sufficient solvency of the insurance company.

The literature on optimal control theory in insurance is vast, see e.g. the text book treatment by Schmidli [15] and references therein. Our aim is to provide solutions to realistic premium control problems in order to allow the optimal premium rule to be used with confidence by insurance companies. In particular, we avoid considering convenient stochastic models that may fit well with optimal control theory but fail to take key features of real dynamical insurance systems into account. Our aim is in particular to present methods for solving premium control problems that work well for a wide range of models for the stochastic environment of the insurance company and that may also be used directly on real insurance data. We do however acknowledge that short historical time series is a serious problem when attempting to solve a premium control problem on only real data.

Increased computing power and methodological advances during the recent decades make it possible to revisit the problems studied in [9] and [10] and in doing so allow for more complex and realistic dynamics of the insurance dynamical system. Allowing realistic complex dynamics means that optimal premium rules, if possible to be obtained, will allow insurance companies to not only be given guidance on how to set premiums but actually have premium rules that they can use with certain confidence. The methodological advances that we use in this work is reinforcement learning and in particular reinforcement learning combined with function approximation, see e.g. Bertsekas and Tsitsiklis [1] and Sutton and Barto [18] and references therein. By using reinforcement learning methods combined with function approximation we obtain premium rules in terms of Markovian controls for Markov decision processes whose state spaces are much larger/more realistic than what was considered in the premium control problem studied in [10].

The paper is organised as follows. Section 2 describes the relevant insurance economics by presenting the involved cash flows, key economic quantities such as surplus, earned premium, reserves and how such quantities are connected to each other and their dynamics or balance equations. Section 2 also introduces stochastic models giving a complete description of the stochastic environment in which the insurance company operates and aims to determine an optimal premium rule. The complete stochastic model will serve us by enabling simulation of data from which the reinforcement learning methods gradually learn the stochastic environment in the search for optimal premium rules. The model is necessarily somewhat complex if we require that delays between accidents and payments should be allowed as well as random fluctuations in the number of policyholders, partly due to varying premium levels. The model choices are intended to be realistic but other model choices can be made without affecting the feasibility of the methods and analysis considered in later sections.

Section 3 sets up the premium control problem we aim to solve in terms of a Markov decision process and standard elements of stochastic control theory such as the Bellman equation. Finding the optimal premium rule by directly solving the Bellman (optimality) equation numerically is not possible when considering state spaces for the Markov decision process matching a realistic model for the insurance dynamical system. Therefore, we introduce reinforcement learning methods in Section 4. In particular, we present basic theory for the temporal difference learning methods Q-learning and SARSA. We explain why these methods will not be able to provide us with reliable estimates of optimal premium rules unless we restrict ourselves to simplified versions of the insurance dynamical system. We argue that SARSA combined with function approximation of the so-called action-value function will allow us to determine optimal premium rules. We also highlight several pitfalls that the designer of the reinforcement learning method must be aware of and make sure to avoid.

Section 5 presents the necessary details in order to solve the premium control problem using SARSA with function approximation. We analyse the effects of different model/method choices on the performance of different reinforcement learning techniques and compare the performance of the optimal premium rule with those of simpler benchmark rules.

Finally, Section 6 concludes the paper. We emphasise that the premium control problem studied in the present paper is easily adjusted to fit the features of a particular insurance company and that the excellent performance of a carefully set up reinforcement learning method with function approximation provides the insurance company with an optimal premium rule that can be used in practice and communicated to stakeholders.

## 2 A stochastic model of the insurance company

The number of contracts written during year $t+1$ is denoted $N_{t+1}$, a quantity known at the end of year $t+1$. The premium per contract $P_t$ during year $t+1$ is decided at the end of year $t$. Hence $P_t$ is $\mathcal{F}_t$-measurable, where $\mathcal{F}_t$ denotes the $\sigma$-algebra representing the available information at the end of year $t$. Contracts are assumed to be written uniformly in time over the year, and therefore the earned premium during year $t+1$ is

$$\mathrm{EP}_{t+1} = \frac{1}{2}(P_t N_{t+1} + P_{t-1} N_t),$$

i.e. for contracts written during year $t+1$, on average half of the premium income $P_t N_{t+1}$ will be earned during year $t+1$, and half during year $t+2$. Since only half of the premium income $P_t N_{t+1}$ is earned during year $t+1$, the other half, which should cover claims during year $t+2$, will be stored in the premium reserve. The balance equation for the premium reserve is $V_{t+1} = V_t + P_t N_{t+1} - \mathrm{EP}_{t+1}$. Note that when we add cash flows or reserves occurring at time $t+1$ to cash flows or reserves occurring at time $t$, the time-$t$ amounts should be interpreted as adjusted for the time value of money. We choose not to write this out explicitly in order to simplify notation.

That contracts are written uniformly in time over the year means that $I_{t,k}$, the incremental payment to policyholders during year $t+k$ for accidents during year $t+1$, will consist partly of payments to contracts written during year $t+1$ and partly of payments to contracts written during year $t$. Hence we assume that $I_{t,k}$ depends on both $N_{t+1}$ and $N_t$. Table 1 shows a claims triangle with entries $I_{j,k}$ representing incremental payments to policyholders during year $j+k$ for accidents during year $j+1$. For ease of presentation, other choices could of course be made, we will assume that the maximum delay between an accident and a resulting payment is four years. Entries $I_{j,k}$ with $j+k \leq t$ are $\mathcal{F}_t$-measurable and coloured blue in Table 1. Let

$$\begin{aligned}
\mathrm{IC}_{t+1} &= I_{t,1} + \mathrm{E}[I_{t,2} + I_{t,3} + I_{t,4} \mid \mathcal{F}_{t+1}],\\
\mathrm{PC}_{t+1} &= I_{t,1} + I_{t-1,2} + I_{t-2,3} + I_{t-3,4},\\
\mathrm{RP}_{t+1} &= \mathrm{E}[I_{t-3,4} + I_{t-2,3} + I_{t-2,4} + I_{t-1,2} + I_{t-1,3} + I_{t-1,4} \mid \mathcal{F}_t]\\
&\quad - \mathrm{E}[I_{t-3,4} + I_{t-2,3} + I_{t-2,4} + I_{t-1,2} + I_{t-1,3} + I_{t-1,4} \mid \mathcal{F}_{t+1}],
\end{aligned}$$

where IC, PC and RP denote, respectively, incurred claims, paid claims and runoff profit. The balance equation for the claims reserve is $E_{t+1} = E_t + \mathrm{IC}_{t+1} - \mathrm{RP}_{t+1} - \mathrm{PC}_{t+1}$, where

$$\begin{aligned}
&\mathrm{IC}_{t+1} - \mathrm{RP}_{t+1} - \mathrm{PC}_{t+1}\\
&\quad = \mathrm{E}[I_{t,2} + I_{t,3} + I_{t,4} \mid \mathcal{F}_{t+1}] - \mathrm{E}[I_{t-1,2} + I_{t-2,3} + I_{t-3,4} \mid \mathcal{F}_t]\\
&\qquad + \mathrm{E}[I_{t-1,3} + I_{t-2,4} + I_{t-1,4} \mid \mathcal{F}_{t+1}] - \mathrm{E}[I_{t-1,3} + I_{t-2,4} + I_{t-1,4} \mid \mathcal{F}_t].
\end{aligned} \tag{1}$$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $t-2$ | $I_{t-3,1}$ | $I_{t-3,2}$ | $I_{t-3,3}$ | $I_{t-3,4}$ |
| $t-1$ | $I_{t-2,1}$ | $I_{t-2,2}$ | $I_{t-2,3}$ | $I_{t-2,4}$ |
| $t$ | $I_{t-1,1}$ | $I_{t-1,2}$ | $I_{t-1,3}$ | $I_{t-1,4}$ |
| $t+1$ | $I_{t,1}$ | $I_{t,2}$ | $I_{t,3}$ | $I_{t,4}$ |

Table 1: Incremental paid claim amounts from accidents during years $t-2,\dots,t+1$.

The dynamics of the surplus fund is

$$G_{t+1} = G_t + \mathrm{EP}_{t+1} + \mathrm{IE}_{t+1} - \mathrm{OE}_{t+1} - \mathrm{IC}_{t+1} + \mathrm{RP}_{t+1}, \tag{2}$$

where IE denotes investment earnings and OE denotes operating expenses.

We choose to model the key random quantities as integer-valued random variables with conditional distributions that are either Poisson or Negative Binomial distributions. Other choices of distributions on the integers are possible without any major effects on the analysis that follows. Let

$$\mathcal{L}(N_{t+1} \mid \mathcal{F}_t) = \mathcal{L}(N_{t+1} \mid P_t) = \mathsf{Pois}(aP_t^b), \tag{3}$$

where $a > 0$ is a constant, and $b < 0$ is the price elasticity of demand. The notation says that the conditional distribution of the number of contracts written during year $t+1$ given the information at the end of year $t$ depends on that information only through the premium decided at the end of year $t$ for those contracts.

Let $\widetilde{N}_{t+1} = (N_{t+1} + N_t)/2$ denote the number of contracts during year $t+1$ that provide coverage for accidents during year $t+1$. Let

$$\mathrm{OE}_{t+1} = \beta_0 + \beta_1 \widetilde{N}_{t+1}, \tag{4}$$

saying that the operating expenses have both a fixed part and a variable part proportional to the number of active contracts. The appearance of $\widetilde{N}_{t+1}$ instead of $N_{t+1}$ in the expressions above is due to the assumption that contracts are written uniformly in time over the year, and that accidents occur uniformly in time over the year.

Let $\alpha_1, \dots, \alpha_4 \in [0, 1]$ with $\sum_{i=1}^4 \alpha_i = 1$. The constant $\alpha_k$ denotes the expected fraction of claim costs for a given accident year that generates payments during development year $k$. Let

$$\mathcal{L}(I_{t,k} \mid \mathcal{F}_t, \widetilde{N}_{t+1}) = \mathcal{L}(I_{t,k} \mid \widetilde{N}_{t+1}) = \mathsf{Pois}(\alpha_k \mu \widetilde{N}_{t+1}), \tag{5}$$

where $\mu$ denotes the expected claim cost per contract. We assume that different incremental claims payments $I_{j,k}$ are conditionally independent given information about the corresponding numbers of contracts written. Formally, the elements in the set

$$\{I_{j,k} : j \in \{t-l, \dots, t\}, k \in \{1, \dots, 4\}\}$$

are conditionally independent given $N_{t-l}, \dots, N_{t+1}$. Therefore, using (1) and (5),

$$\mathcal{L}(\mathrm{PC}_{t+1} \mid \mathcal{F}_t, \widetilde{N}_{t+1}) = \mathcal{L}(\mathrm{PC}_{t+1} \mid \widetilde{N}_{t-2}, \dots, \widetilde{N}_{t+1}) = \mathsf{Pois}(\alpha_1 \mu \widetilde{N}_{t+1} + \dots + \alpha_4 \mu \widetilde{N}_{t-2})$$

and

$$\mathrm{IC}_{t+1} - \mathrm{RP}_{t+1} = \mathrm{PC}_{t+1} + (\alpha_2 + \alpha_3 + \alpha_4)\mu \widetilde{N}_{t+1} - \alpha_2 \mu \widetilde{N}_t - \alpha_3 \mu \widetilde{N}_{t-1} - \alpha_4 \mu \widetilde{N}_{t-2}. \tag{6}$$

The model for the investment earnings $\mathrm{IE}_{t+1}$ is chosen so that $G_t \leq 0$ implies $\mathrm{IE}_{t+1} = 0$ since $G_t \leq 0$ means that nothing is invested. Moreover, we assume that

$$\mathcal{L}(\mathrm{IE}_{t+1} + G_t \mid \mathcal{F}_t, G_t > 0) = \mathcal{L}(\mathrm{IE}_{t+1} + G_t \mid G_t, G_t > 0) = \mathsf{NegBin}\left(\nu G_t, \frac{1+\xi}{1+\xi+\nu}\right), \tag{7}$$

where $\mathsf{NegBin}(r, p)$ denotes the Negative Binomial distribution with probability mass function

$$k \mapsto \binom{k+r-1}{k}(1-p)^r p^k$$

which corresponds to mean and variance

$$\mathrm{E}[\mathrm{IE}_{t+1} + G_t \mid G_t, G_t > 0] = \frac{p}{1-p} r = (1+\xi) G_t$$

and

$$\mathrm{Var}(\mathrm{IE}_{t+1} + G_t \mid G_t, G_t > 0) = \frac{p}{(1-p)^2} r = \frac{1+\xi+\nu}{\nu}(1+\xi) G_t.$$

Given a policy $\pi$ that given the state $S_t = (G_t, P_{t-1}, N_{t-3}, N_{t-2}, N_{t-1}, N_t)$ generates the premium $P_t$, the system $(S_t)$ evolves in a Markovian manner according to the transition probabilities that follows from (3)-(7) and (2). Notice that if we consider a less long-tailed insurance product so that $\alpha_3 = \alpha_4 = 0$ (at most one year delay from occurrence of the accident to final payment), then the dimension of the state space reduces to four, i.e. $S_t = (G_t, P_{t-1}, N_{t-1}, N_t)$.

Our objective is to determine an optimal policy optimising a function that describes well good and poor performance of an insurance company.

## 3 The control problem

We consider a set of states $\mathcal{S}^+$, a set of non-terminal states $\mathcal{S} \subseteq \mathcal{S}^+$, and for each $s \in \mathcal{S}$ a set of actions $\mathcal{A}(s)$ available from state $s$, with $\mathcal{A} = \cup_{s \in \mathcal{S}} \mathcal{A}(s)$. We assume that $\mathcal{S}^+$ and $\mathcal{A}$ are discrete (finite or countable). For each $s \in \mathcal{S}$, $s' \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$ we define the reward received after taking action $a$ in state $s$ and transitioning to $s'$, $-f(a, s, s')$, and the probability of transitioning from state $s$ to state $s'$ after taking action $a$, $p(s'|s, a)$. We assume that rewards and transition probabilities are stationary (time-homogeneous). This defines a Markov decision process (MDP). A policy $\pi$ specifies how to determine what action to take in each state. A stochastic policy describes, for each state, a probability distribution on the set of available actions. A deterministic policy is a special case of a stochastic policy, specifying a degenerate probability distribution, i.e. a one-point distribution.

Our objective is to find the premium policy that minimises the expected value of the premium payments over time, but that also results in $(P_t)$ being more averaged over time, and further ensures that the surplus $(G_t)$ is large enough so that the risk that the insurer cannot pay the claims costs and other expenses is small. We formulate this in terms of a MDP, i.e. we want to solve the following optimisation problem:

$$\underset{\pi}{\text{minimise}}\ \mathrm{E}_\pi\left[\sum_{t=0}^{T}\gamma^t f(P_t, S_t, S_{t+1}) \mid S_0 = s\right] \tag{8}$$

where $\pi$ is a policy generating the premium $P_t$ given the state $S_t$, $\mathcal{A}(s)$ is the set of premium levels available from state $s$, $\gamma$ is the discount rate, $f$ is the cost function, and $\mathrm{E}_\pi[\cdot]$ denotes the expectation given that policy $\pi$ is used. Note that the discount factor $\gamma^t$ should not be interpreted as the price of a zero-coupon bond maturing at time $t$, since the cost that is discounted does not represent an economic cost. Instead $\gamma$ reflects how much weight is put on costs that are immediate compared to costs further in the future. The transition probabilities are

$$p(s'|s,a) = \mathrm{P}(S_{t+1} = s' \mid S_t = s, P_t = a),$$

and we consider stationary policies, letting $\pi(a|s)$ denote the probability of taking action $a$ in state $s$ under policy $\pi$,

$$\pi(a|s) = \mathrm{P}_\pi(P_t = a \mid S_t = s).$$

If there are no terminal states, we have $T = \infty$, and $\mathcal{S}^+ = \mathcal{S}$. We want to choose $\mathcal{A}(s), s \in \mathcal{S}$, $f$, and any terminal states such that the objective discussed above is achieved. We will do this in two ways, see Sections 3.1 and 3.2.

The value function of state $s$ under a policy $\pi$ generating the premium $P_t$ is defined as

$$v_\pi(s) := \mathrm{E}_\pi\left[\sum_{t=0}^{T}\gamma^t(-f(P_t, S_t, S_{t+1})) \mid S_0 = s\right].$$

The Bellman equation for the value function is

$$v_\pi(s) = \sum_{a\in\mathcal{A}(s)}\pi(a|s)\sum_{s'\in\mathcal{S}}p(s'|s,a)\Big(-f(a,s,s') + \gamma v_\pi(s')\Big).$$

When the policy is deterministic, we let $\pi$ be a mapping from $\mathcal{S}$ to $\mathcal{A}$, and

$$v_\pi(s) = \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s))\Big( - f(\pi(s), s, s') + \gamma v_\pi(s')\Big).$$

The optimal value function is $v_*(s) := \sup_\pi v_\pi(s)$. When the action space is finite the supremum is attained, which implies the existence of an optimal deterministic stationary policy (see [13, Cor. 6.2.8], for other sufficient conditions for attainment of the supremum, see [13, Thm. 6.2.10]). Hence, if the transition probabilities are known, we can use the Bellman optimality equation to find $v_*(s)$:

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} p(s'|s, a)\Big( - f(a, s, s') + \gamma v_*(s')\Big).$$

We use policy iteration in order to find the solution numerically. Let $k = 0$, and choose some initial deterministic policy $\pi_k(s)$ for all $s \in \mathcal{S}$. Then

(i) Determine $V_k(s)$ as the unique solution to the system of equations

$$V_k(s) = \sum_{s' \in \mathcal{S}} p(s'|s, \pi_k(s))\Bigg( - f(\pi_k(s), s, s') + \gamma V_k(s')\Bigg).$$

(ii) Determine an improved policy $\pi_{k+1}(s)$ by computing

$$\pi_{k+1}(s) = \operatorname*{argmax}_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} p(s'|s, a)\Bigg( - f(a, s, s') + \gamma V_k(s')\Bigg).$$

(iii) If $\pi_{k+1}(s) \neq \pi_k(s)$ for some $s \in \mathcal{S}$, then increase $k$ by 1 and return to step (i).

Note that if the state space is large enough, solving the system of equations in step (i) directly might be too time-consuming. In that case, this step can be solved by an additional iterative procedure, called iterative policy evaluation, see e.g. [18, Ch. 4.1].

## 3.1 MDP with constraint on the action space

The premiums $(P_t)$ will be averaged if we minimise $\sum_t c(P_t)$, where $c$ is an increasing, strictly convex function. Thus for the first MDP we let $f(a, s, s') = c(a)$. To ensure that the surplus $(G_t)$ does not become negative too often, we combine this with the constraint saying that the premium needs to be chosen so that the expected value, given the current state, of the surplus stays nonnegative, i.e.

$$\mathcal{A}(S_t) = \{P_t : \mathrm{E}_\pi[G_{t+1} \mid S_t] \geq 0\}, \tag{9}$$

and the optimisation problem becomes

$$\underset{\pi}{\text{minimise}}\ \mathrm{E}_\pi\Big[\sum_{t=0}^{\infty}\gamma^t c(P_t) \mid S_0 = s\Big] \quad \text{subject to}\ \ \mathrm{E}_\pi[G_{t+1} \mid S_t] \geq 0 \text{ for all } t. \tag{10}$$

The choice of the convex function $c$, together with the constraint, will affect how quickly the premium can be lowered as the surplus or previous premium increases, and how quickly the premium must be increased as the surplus or previous premium decreases. Different choices of $c$ affect how well different parts of the objective are achieved. Hence one choice of $c$ might put a higher emphasis on the premium being more averaged over time but slightly higher, while another choice might promote a lower premium level that is allowed to vary a bit more from one time point to another. Furthermore, it is not clear from the start what choice of $c$ will lead to a specific result, thus designing the reward signal might require searching through trial and error for the cost function that achieves the desired result.

## 3.2 MDP with a terminal state

The constraint (9) requires a prediction of $N_{t+1}$ according to (3). However, estimating the price elasticity in (3) is difficult task, hence it would be desirable to solve the optimisation problem without having to rely on this prediction. To this end, we remove the constraint on the action space, i.e. we let $\mathcal{A}(s) = \mathcal{A}$ for all $s \in \mathcal{S}$, and instead introduce a terminal state which has a larger negative reward than all other states. This terminal state is reached when the surplus $G_t$ is below some predefined level, and it can be interpreted as the state where the insurer defaults and has to shut down. If we let $\mathcal{G}$ denote the set of non-terminal states for the first state variable (the surplus), then

$$f(P_t, S_t, S_{t+1}) = h(P_t, S_{t+1}) := \begin{cases} c(P_t), & \text{if } G_{t+1} \geq \min \mathcal{G}, \\ c(\max \mathcal{A})(1+\eta), & \text{if } G_{t+1} < \min \mathcal{G}, \end{cases}$$

where $\eta > 0$. The optimisation problem becomes

$$\underset{\pi}{\text{minimise}}\ \mathrm{E}_\pi\Big[\sum_{t=0}^{T}\gamma^t h(P_t, S_{t+1}) \mid S_0 = s\Big], \quad T := \min\{t : G_t < \min \mathcal{G}\}. \tag{11}$$

The reason for choosing $\eta > 0$ is to ensure that the reward when transitioning to the terminal state is lower than the reward when using action $\max \mathcal{A}$ (the maximal premium level), i.e. it should be more costly to terminate and restart compared with attempting to increase the surplus when the surplus is low. The particular value of the parameter $\eta > 0$ together with the choice of the convex function $c$ determine the reward signal, i.e. the compromise between minimising the premium, averaging the premium, and ensuring that the risk of default is low. One way of choosing $\eta$ is to set it high enough so that the reward when terminating is lower than the total reward using any other policy. Then we require that

$$(1+\eta)c(\max \mathcal{A}) > \sum_{t=0}^{\infty} \gamma^t c(\max \mathcal{A}) = \frac{1}{1-\gamma} c(\max \mathcal{A}),$$

i.e. $\eta > \gamma/(1-\gamma)$. This choice of $\eta$ will put a higher emphasis on ensuring that the risk of default is low, compared with using a lower value of $\eta$.

## 3.3 Policy iteration for simplified model

If the state space is not too large, then we may solve both the optimisation problem with a constraint on the action space and with a terminal state numerically using policy iteration.

Consider the situation where the insurer has a fixed number $N$ of policyholders, who at some initial time point bought insurance policies with automatic contract renewal for the price $P_t$ year $t+1$. The state at time $t$ is $S_t = (G_t, P_{t-1})$. In this simplified setting, $\mathrm{OE}_{t+1} = \beta_0 + \beta_1 N$, all payments $I_{t,k}$ are independent, $\mathcal{L}(I_{t,k}) = \mathsf{Pois}(\alpha_k \mu N)$, $\mathrm{IC}_{t+1} - \mathrm{RP}_{t+1} = \mathrm{PC}_{t+1}$ and $\mathcal{L}(\mathrm{PC}_{t+1}) = \mathsf{Pois}(\mu N)$. With this simplified model we have

$$G_{t+1} = G_t + \frac{1}{2} N(P_t + P_{t-1}) - (\beta_0 + \beta_1 N) - \mathrm{PC}_{t+1} + \mathrm{IE}_{t+1},$$

where $\mathrm{IE}_{t+1} = 0$ if $G_t \le 0$ and otherwise distributed according to (7). The constraint (9) here means that $P_t$ must be sufficiently large to satisfy

$$(1 + \xi \mathbf{1}_{\{G_t > 0\}}) G_t + \frac{1}{2} N(P_t + P_{t-1}) - (\beta_0 + \beta_1 N + \mu N) \ge 0.$$

The transition probabilities are given by

$$\begin{aligned}
&\mathrm{P}(S_{t+1} = (k,p) \mid S_t = (g,q), P_t = p) \\
&\quad = \mathrm{P}(\mathrm{IE}_{t+1} + G_t - \mathrm{PC}_{t+1} = k + (\beta_0 + \beta_1 N) - \frac{1}{2} N(p+q) \mid (G_t, P_{t-1}, P_t) = (g,q,p)) \\
&\quad = \begin{cases} \mathrm{P}\left(\mathrm{PC}_{t+1} = g - m\right) & \text{if } g \le 0, \\ \sum_{\{l : m+l \ge 0\}} \mathrm{P}(\mathrm{PC}_{t+1} = l)\, \mathrm{P}\left(\mathrm{IE}_{t+1} + G_t = m + l \mid G_t = g\right) & \text{if } g > 0, \end{cases}
\end{aligned}$$

where $m = k + (\beta_0 + \beta_1 N) - N(p+q)/2$.

The parameter values used in Section 5 are $\mu = 5$, $\beta_0 = 10$ and $\beta_1 = 1$. With $N = 10$ this means that the expected yearly total cost for the insurer is 70 and the expected yearly cost per customer is 7. We emphasise that parameter values are meant to be interpreted in suitable units to fit the application in mind. As the model for the MDP is formulated, $\mathrm{P}(G_t = g) > 0$ for all integers $g$. However, for actions/premiums that are considered with the aim of solving the optimisation problem, it will be sufficient to only consider a finite range of integer values for $G_t$ since transitions to values outside this range will have negligible probability. Specifically, we will only consider values $\{-20, -19, \dots, 149, 150\}$ as possible values for the surplus. In order to ensure that transition probabilities sum to one, we must adjust the probabilities of transitions to the limiting surplus values according to the original probabilities of exiting the range of possible surplus values.

**Remark 3.1** *We have tested different values for the maximum value of the surplus process, with the conclusion that truncating the surplus process at 150 does not have a material effect on the optimal policy. However, the minimum value chosen for the surplus process (here -20), will have a larger effect on the optimal policy for the MDP with a terminal state, and should be seen as another parameter value that needs to be chosen to determine the reward signal, see Section 3.2.*

## 4 Reinforcement learning

If the model of the environment is not fully known, or if the state space or action space are not finite, the control problem can no longer be solved by classical dynamical programming approaches. Instead, we can utilise different reinforcement learning algorithms. We will begin by analysing the same simple model as in Section 3.3, but now assuming that the model of the environment is not fully known, by which we mean that we are not able to explicitly compute the transition probabilities. We will, however, still use the model to simulate the environment.

### 4.1 Temporal-difference learning

Temporal-difference (TD) methods can learn directly from real or simulated experience of the environment. Given a specific policy $\pi$ which determines the action taken in each state, and the sampled or observed state at time $t$, $S_t$, state at time $t+1$, $S_{t+1}$, and reward $R_{t+1}$, the iterative update for the value function, using the one-step TD method, is

$$V(S_t) \leftarrow V(S_t) + \alpha_t\Big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big),$$

where $\alpha_t$ is a step size parameter. Hence, the target for the TD update is $R_{t+1} + \gamma V(S_{t+1})$. Thus, we update $V(S_t)$, which is an estimate of $v_\pi(S_t) = \mathrm{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})]$, based on another estimate, namely $V(S_{t+1})$. The intuition behind using $R_{t+1} + \gamma V(S_{t+1})$ as the target in the update is that this is a slightly better estimate of $v_\pi(S_t)$, since it consists of an actual (observed or sampled) reward at $t+1$ and an estimate of the value function at the next observed state.

It has been shown in e.g. [3] that the value function (for a given policy $\pi$) computed using the one-step TD method converges to the true value function if the step size parameter $0 \leq \alpha_t \leq 1$ satisfies the following stochastic approximation conditions

$$\sum_{k=1}^{\infty} \alpha_{t^k(s)} = \infty, \quad \sum_{k=1}^{\infty} \alpha_{t^k(s)}^2 < \infty, \quad \text{for all } s \in \mathcal{S},$$

where $t^k(s)$ is the time step when state $s$ is visited for the $k$th time.

### 4.1.1 Q-learning

The one-step TD method described above gives us an estimate of the value function for a given policy $\pi$. To find the optimal policy using TD learning, a TD control algorithm can be used. One example of such an algorithm is Q-learning, which focuses on directly estimating the optimal action-value function:

$$q_*(s,a) := \max_\pi q_\pi(s,a),$$

where $q_\pi$ is the action-value function for policy $\pi$,

$$q_\pi(s,a) := \mathrm{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, A_0 = a\Big].$$

The iterative update searching for the optimal action-value function, using Q-learning, is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha_t\Big(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big). \tag{12}$$

Given that all state-action pairs continue to be updated, it has been shown in [20] that Q-learning converges to the true optimal action-value function if the step size parameter $0 \leq \alpha_t \leq 1$ satisfies the following stochastic approximation conditions

$$\sum_{k=1}^{\infty} \alpha_{t^k(s,a)} = \infty, \quad \sum_{k=1}^{\infty} \alpha_{t^k(s,a)}^2 < \infty, \quad \text{for all } s \in \mathcal{S}, a \in \mathcal{A}(s), \tag{13}$$

where $t^k(s,a)$ is the time step when a visit in state $s$ is followed by taking action $a$ for the $k$th time.

To use Q-learning, we need to generate transitions from state-action pairs $(S_t, A_t)$ to state-action pairs $(S_{t+1}, A_{t+1})$ and observe the rewards $R_{t+1}$ obtained during each transition. To do this, we need a behaviour policy, i.e. a policy that determines which action is taken in the state we are currently in when the transitions are generated. To ensure that all state-action pairs continue to be updated, this policy needs to be exploratory. At the same time, we want to exploit what we have learned so far by choosing actions that we believe will give us large future rewards. A common choice of policy that compromises in this way between exploration and exploitation is the $\varepsilon$-greedy policy, which with probability $1-\varepsilon$ chooses the action that maximises the action-value function in the current state; and with probability $\varepsilon$ chooses any other action uniformly at random. Hence, the Q-learning algorithm is an off-policy algorithm, since it uses one policy to generate the transitions, but updates a different policy (called the target policy), namely the greedy policy that maximises the action-value function in each state. If we let $\tilde{\pi}$ denote the behaviour policy and $\pi$ the target policy, then $\pi(s) = \operatorname{argmax}_a Q(s,a)$ and

$$\tilde{\pi}(a|s) = \begin{cases} 1-\varepsilon, & \text{if } a = \operatorname{argmax}_a Q(s,a), \\ \frac{\varepsilon}{|\mathcal{A}|-1}, & \text{otherwise.} \end{cases}$$

In Section 5 we use $\mathcal{A} = \{0.2, 0.4, \ldots, 19.8, 20.0\}$, hence $|\mathcal{A}| = 100$.

### 4.1.2 SARSA

Another example of a TD control algorithm is SARSA. In contrast to Q-learning, this is an on-policy algorithm, meaning that the same policy is used both as the behaviour policy and the target policy. The iterative update for the action-value function, using SARSA, is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha_t\Big(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big).$$

Thus, SARSA gives an estimate of the action-value function $q_\pi$ given the behaviour policy $\pi$. Since the same policy is used as both the behaviour policy and the target policy, this policy needs to be exploratory, e.g. $\varepsilon$-greedy, to ensure that all state action-pairs continue to be updated. At the same time, we need to ensure that the end result is the optimal policy, hence the policy needs to be changed over time towards the greedy policy that maximises the action-value function in each state. If the policy is $\varepsilon$-greedy, this can be accomplished by letting $\varepsilon$ slowly decay towards zero.

Under the condition that all state-action pairs continue to be updated, and that the behaviour policy is greedy in the limit, it has been shown in [16] that SARSA converges to the true optimal action-value function if the step size parameter satisfies the stochastic approximation conditions (13).

Since Q-learning directly learns the optimal policy, one might wonder why we have need for this second TD control algorithm that has a target policy that is not fully optimal since it needs to be exploratory. One reason has to do with the topic of the next section, namely function approximation. While there are some convergence results for SARSA with function approximation, there are none for standard Q-learning with function approximation. In fact, there are examples of divergence when combining off-policy training (as is done in Q-learning) with function approximation. For more on this, see e.g. [18, Ch. 11].

## 4.2 Function approximation

The methods discussed thus far are examples of tabular solution methods, i.e. methods where the value functions can be represented as tables. These methods are suitable when the state and action space are not too large, e.g. for the simplified model in Section 3.3. However, when the state space and/or action space is very large, or even continuous, these methods are not feasible, due to not being able to fit tables of this size in memory, and/or due to the time required to visit all state-action pairs multiple times. This is the case for the more realistic model presented in Section 2 where we allow the number of contracts written per year to vary. This has two effects that both increase the size of the state space: the dimension of the state space is larger; and the surplus process, depending on the parameter values chosen, can take non-integer values. For the simplified model $\mathcal{S} = \mathcal{G} \times \mathcal{A}$, and with the parameters chosen in Section 5 we have $|\mathcal{G}| = 171$ and $|\mathcal{A}| = 100$. In the more realistic setting, if we let $\mathcal{N}$ denote the set of integer values that $N_t$ is allowed to take values in, then $\mathcal{S} = \mathcal{G} \times \mathcal{A} \times \mathcal{N}^l$, where $l$ denotes the maximum number of development years. With the parameters chosen in Section 5 in the more realistic setting the total number of states is approximately $10^8$.

Thus, to solve the optimisation problem in the more realistic setting, we need approximate solution methods, in order to generalise from the states that have been experienced to other states. In approximate solution methods the value function $v_\pi(s)$ (or action-value function $q_\pi(s, a)$) is approximated by a parameterised function, $\hat{v}(s; w)$ (or $\hat{q}(s, a; w)$). When the state space is discrete, it is common to minimise the following objective function,

$$J(w) := \sum_{s\in\mathcal{S}} \mu_\pi(s)\Big(v_\pi(s) - \hat{v}(s;w)\Big)^2, \tag{14}$$

where $\mu_\pi(s)$ is the fraction of time spent in state $s$. For the model without terminal states, $\mu_\pi$ is the stationary distribution under policy $\pi$. For the model with terminal states, to determine the fraction of time spent in each transient state, we need to compute the expected number of visits $\eta_{\lambda,\pi}(s)$ to each transient state $s \in \mathcal{S}$ before reaching a terminal (absorbing) state, where $\lambda(s) = \mathrm{P}(S_0 = s)$ is the initial distribution. For ease of notation, we omit $\lambda$ from the subscript below, and write $\eta_\pi$ and $\mathrm{P}_\pi$ instead of $\eta_{\lambda,\pi}$ and $\mathrm{P}_{\lambda,\pi}$. Let $p(s|s')$ be the probability of transitioning from state $s'$ to state $s$ under policy $\pi$, i.e.

$$p(s|s') = \mathrm{P}_\pi(S_t = s \mid S_{t-1} = s').$$

Then

$$\begin{aligned}\eta_\pi(s) &:= \mathrm{E}_\pi\left[\sum_{t=0}^{\infty} \mathbf{1}_{\{S_t=s\}}\right] = \lambda(s) + \sum_{t=1}^{\infty} \mathrm{P}_\pi(S_t = s)\\ &= \lambda(s) + \sum_{t=1}^{\infty}\sum_{s'\in\mathcal{S}} p(s|s')\,\mathrm{P}_\pi(S_{t-1} = s') = \lambda(s) + \sum_{s'\in\mathcal{S}} p(s|s') \sum_{t=0}^{\infty} \mathrm{P}_\pi(S_t = s')\\ &= \lambda(s) + \sum_{s'\in\mathcal{S}} p(s|s')\eta_\pi(s'),\end{aligned}$$

or, in matrix form, $\eta_\pi = \lambda + P^\top \eta_\pi$, where $P$ is the part of the transition matrix corresponding to transitions between transient states, i.e. if we label the states $0, 1, \dots, |\mathcal{S}|$ (where state 0 represents all terminal states), then $P = (p_{ij} : i, j \in \{1, 2, \dots, |\mathcal{S}|\})$, where $p_{ij} = p(j \mid i)$. After solving this system of equations, the fraction of time spent in each transient state under policy $\pi$ can be computed according to

$$\mu_\pi(s) = \frac{\eta_\pi(s)}{\sum_{s'\in\mathcal{S}} \eta_\pi(s')}, \quad \text{for all } s \in \mathcal{S}.$$

This computation of $\mu_\pi$ relies on the model of the environment being fully known and the transition probabilities explicitly computable, as is the case for the simplified model in Section 3.3. However, for the situation at hand, where we need to resort to function approximation and determine $\hat{v}(s;w)$ (or $\hat{q}(s,a;w)$) by minimising (14), we cannot explicitly compute $\mu_\pi$. Instead, $\mu_\pi$ in (14) is captured by learning incrementally from real or simulated experience, as in semi-gradient TD learning. Using semi-gradient TD learning, the iterative update for the weight vector $w$ becomes

$$w_{t+1} = w_t + \alpha_t\Big(R_{t+1} + \gamma\hat{v}(S_{t+1};w_t) - \hat{v}(S_t;w_t)\Big)\nabla\hat{v}(S_t;w_t).$$

This update can be used to estimate $v_\pi$ for a given policy $\pi$, generating transitions from state to state by taking actions according to this policy. Similarly to standard TD learning (Section 4.1), the target $R_{t+1} + \gamma \hat{v}(S_{t+1}; w_t)$ is an estimate of the true (unknown) $v_\pi(S_{t+1})$. The name 'semi-gradient' comes from the fact that the update is not based on the true gradient of $(R_{t+1} + \gamma \hat{v}(S_{t+1}; w_t) - \hat{v}(S_t; w_t))^2$, instead the target is seen as fixed when the gradient is computed, despite the fact that it depends on the weight vector $w_t$.

As in the previous section, estimating the value function given a specific policy is not our final goal - instead we want to find the optimal policy. Hence, we need a TD control algorithm with function approximation. One example of such an algorithm is semi-gradient SARSA, which estimates $q_*$. The iterative update for the weight vector is

$$w_{t+1} = w_t + \alpha\Big(R_{t+1} + \gamma_t \hat{q}(S_{t+1}, A_{t+1}; w_t) - \hat{q}(S_t, A_t; w_t)\Big)\nabla \hat{q}(S_t, A_t; w_t). \tag{15}$$

As with standard SARSA, we need a behaviour policy that generates transitions from state-action pairs to state action-pairs, e.g. an $\varepsilon$-greedy policy.

### 4.2.1 Linear function approximation

The simplest form of function approximation is linear function approximation. Using linear function approximation for estimating the value function means that $\hat{v}(\cdot; w)$ is a linear function of $w$:

$$\hat{v}(s; w) := w^\top x(s),$$

where $x(s)$ are basis functions. Using the Fourier basis as defined in [7], the $i$th basis function for the Fourier basis of order $n$ is

$$x_i(s) = \cos(\pi s^\top c^{(i)}),$$

where $s = (s_1, s_2, \ldots, s_k)^\top$, $c^{(i)} = (c_1^{(i)}, \ldots, c_k^{(i)})^\top$, and $k$ is the dimension of the state space (here $\pi$ is the number $\pi \approx 3.14$, not the policy). The $c^{(i)}$'s are given by the $k$-tuples over the set $\{0, \ldots, n\}$, hence $i = 1, \ldots, (n+1)^k$. This means that $x(s) \in \mathbb{R}^{(n+1)^k}$, i.e. the dimension of $x$ depends both on the dimension of the state space and the order of the Fourier basis.

One-step semi-gradient TD-learning with linear function approximation has been shown to converge to a weight vector $w^*$, however, this is not necessarily the weight vector that minimises $J$. [19] provide a bound on $J$ evaluated at $w^*$ in terms of the minimum of $J$:

$$J(w^*) \leq \frac{1}{1-\gamma} \min_w J(w).$$

However, since $\gamma$ is often close to one, this bound can be quite large.

When using linear function approximation for estimating the action-value function, we have

$$\hat{q}(s, a; w) := w^\top x(s, a),$$

and the $i$th basis function for the Fourier basis of order $n$ is

$$x_i(s, a) = \cos(\pi(s^\top c_{1:k}^{(i)} + a c_{k+1}^{(i)})),$$

where $s = (s_1, \dots, s_k)^\top$, $c_{1:k}^{(i)} = (c_1^{(i)}, \dots, c_k^{(i)})^\top$, $c_j^{(i)} \in \{0, \dots, n\}$, $j = 1, \dots, k+1$, and $i = 1, \dots, (n+1)^{k+1}$.

The convergence results for semi-gradient SARSA with linear function approximation depend on what type of policy is used in the algorithm. When using an $\varepsilon$-greedy policy, the weights have been shown to converge to a bounded region, and might oscillate within that region [6]. Furthermore, [12] has shown that there exists $L < 0$ such that if the policy improvement operator $\Gamma$ is Lipschitz continuous with constant $L$ and $\varepsilon$-soft, then SARSA will converge to a unique policy. The policy improvement operator maps every $q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ to a stochastic policy, and gives the updated policy after iteration $t$ as $\pi_{t+1} = \Gamma(q^{(t)})$, where $q^{(t)}$ corresponds to a vectorised version of the state-action-values after iteration $t$, i.e. $q^{(t)} = xw_t$ for the case where we use linear function approximation, where $x \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times d}$ is a matrix with $x(s, a)^\top$ for each $s \in \mathcal{S}, a \in \mathcal{A}$ as rows, and $d$ is the number of basis functions. That $\Gamma$ is Lipschitz continuous with constant $L$ means that

$$\|\Gamma(q) - \Gamma(q')\|_2 \leq L\|q - q'\|_2, \quad \text{for all } q, q' \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}.$$

That $\Gamma$ is $\varepsilon$-soft means that it produces a policy $\pi = \Gamma(q)$ that is $\varepsilon$-soft, i.e. $\pi(a|s) \geq \varepsilon/|\mathcal{A}|$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. In both [6, 12] the policy improvement operator was not applied at every time step, hence it is not the online SARSA-algorithm considered in the present paper that was investigated. The convergence of online SARSA under the assumption that the policy improvement operator is Lipschitz continuous with a small enough constant $L$ was later shown in [11].

Let $q_{s,a} \in \mathbb{R}$ denote the element of $q$ that correspond to state $s \in \mathcal{S}, a \in \mathcal{A}$, i.e. for the case where we use linear function approximation, $q_{s,a} = w^\top x(s, a)$. Furthermore, let $q_s := (q_{s,a})_{a \in \mathcal{A}} \in \mathbb{R}^{|\mathcal{A}|}$. An example of a Lipschitz continuous policy is the softmax policy. In this case, the policy improvement operator is given by $\Gamma(q) = (\sigma(q_s))_{s\in\mathcal{S}}$, where $\sigma$ is the softmax function,

$$\sigma(q_s) := \frac{\exp\left\{\frac{1}{\tau}q_s\right\}}{\sum_{a\in\mathcal{A}}\exp\left\{\frac{1}{\tau}q_{s,a}\right\}}.$$

To see that this policy improvement operator is Lipschitz continuous, first note that the softmax function $\sigma$ is $1/\tau$-Lipschitz. The softmax function $\sigma$ is differentiable, hence (see e.g. [14, Thm 9.19])

$$\|\sigma(q_s) - \sigma(q_s')\|_2 \leq \sup_{q_s}\|D\sigma(q_s)\|_2\|q_s - q_s'\|_2,$$

where $D\sigma(q_s)$ denotes the Jacobian matrix of $\sigma$ with respect to $q_s$, and $\|D\sigma(q_s)\|_2$ is the spectral norm of $D\sigma(q_s)$. Let $\sigma_a$ denote the component of $\sigma(q_s)$ that corresponds to action $a \in \mathcal{A}$. Then

$$\frac{\partial\sigma_a}{\partial q_{s,a}} = \frac{1}{\tau}\sigma_a(1-\sigma_a), \quad \frac{\partial\sigma_a}{\partial q_{s,a'}} = -\frac{1}{\tau}\sigma_a\sigma_{a'}, \quad \text{for } a \neq a'.$$

It is easy to verify that $D\sigma(q_s)$ is positive semi-definite [2, p. 74]. Hence all eigenvaues of $D\sigma(q_s)$ are non-negative, and

$$\begin{aligned}\|D\sigma(q_s)\|_2 = \lambda_{\max}(D\sigma(q_s)) &\leq \sum_i \lambda_i(D\sigma(q_s)) = \operatorname{tr}(D\sigma(q_s)) = \frac{1}{\tau}\sum_i \sigma_i(1-\sigma_i) \\ &\leq \frac{1}{\tau}\sum_i \sigma_i = \frac{1}{\tau},\end{aligned}$$

where $\lambda_{\max}(D\sigma(q_s))$, $\lambda_i(D\sigma(q_s))$, and $\operatorname{tr}(D\sigma(q_s))$ denote respectively the largest eigenvalue, the $i$th eigenvalue, and the trace of $D\sigma(q_s)$. Now,

$$\|\Gamma(q) - \Gamma(q')\|_2^2 = \sum_{s\in\mathcal{S}}\|\sigma(q_s) - \sigma(q_s')\|_2^2 \leq \sum_{s\in|\mathcal{S}|}\frac{1}{\tau^2}\|q_s - q_s'\|_2^2 = \frac{1}{\tau^2}\|q - q'\|_2^2,$$

i.e. $\Gamma$ is Lipschitz continuous with constant $L = 1/\tau$.

However, the value of the Lipschitz constant $L$ that ensures convergence depends on the problem at hand, and there is no guarantee that the policy the algorithm converges to is optimal. Furthermore, for SARSA to approximate the optimal action-value function, we need the policy to get closer to the greedy policy over time, e.g. by decreasing the temperature parameter when using a softmax policy. Thus, the Lipschitz constant $L$, which is inversely proportional to the temperature parameter, will increase as the algorithm progresses, making the convergence results in [12, 11] less likely to hold. As discussed in [11] this is not an issue specific to the softmax policy. Any Lipschitz continuous policy that over time gets closer to the greedy policy will in fact approach a discontinuous policy, and hence the Lipschitz constant of the policy might eventually become too large for the convergence result to hold. Furthermore, the results in [12, 11] are not derived for a Markov decision process with an absorbing state. Despite this, it is clear from the numerical results in Section 5 that a softmax policy performs substantially better compared to an $\varepsilon$-greedy policy, and for the simplified model approximates the true optimal policy well.

The convergence results in [6, 12, 11] are based on the stochastic approximation conditions

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty, \tag{16}$$

where $\alpha_t$ is the step size parameter used at time step $t$. Note that here we do not have a separate vector of step sizes for each state-action pair. This is since function approximation in general is used when the state space and/or action space is very large, hence it might not be possible to fit one step size parameter per state-action pair in memory (if this is possible then tabular methods are more suitable, see Section 4.1). Furthermore, the idea behind function approximation is to generalise from the state-action pairs visited to other state-action pairs, since some state-action pairs might never be visited, or visited very rarely, but can be more or less similar to other state-action pairs. Hence to count the number of times action $a$ has been taken after a visit to state $s$ is not as relevant as in the tabular case, even if memory usage was not an issue.

## 5 Numerical illustrations

### 5.1 Simplified model

We use the following parameter values: $\gamma = 0.9$, $N = 10$, $\mu = 5$, $\beta_0 = 10$, $\beta_1 = 1$, $\xi = 0.05$, $\nu = 1$, and cost function

$$c(p) = p + c_1(c_2^p - 1), \tag{17}$$

with $c_1 = 1$ and $c_2 = 1.2$. For the model with a terminal state, we use $\eta = 10 > \gamma/(1-\gamma)$, as suggested in Section 3.2. The surplus process and premium level are truncated and discretised according to $\mathcal{G} = \{-20, -19, \dots, 150\}$ and $\mathcal{A} = \{0.2, 0.4, \dots, 19.8, 20.0\}$.

**Remark 5.1 (Cost function)** *The cost function* (17) *was suggested in [10] since it is an increasing, convex function, and thus will lead to the premium being more averaged over time. However, in [10]* $c_2 = 1.5$ *was used in the calculations. We have chosen a slightly lower value of* $c_2$ *due to that too extreme rewards can lead to numerical problems when using SARSA with linear function approximation. It is possible that non-linear function approximation might alleviate this problem, but this was not investigated in the present paper.*

### 5.1.1 Policy iteration

The top row in Figure 1 shows the optimal policy and the stationary distribution under the optimal policy, for the simplified model with a constraint on the action space (Section 3.1) using policy iteration. The bottom row in Figure 1 shows the optimal policy and the fraction of time spent in each state under the optimal policy, for the simplified model with terminal state (Section 3.2) using policy iteration. In both cases, the premium charged increases as the surplus or the previously charged premium decreases. Based on the fraction of time spent in each state under each of these two policies, we note that in both cases the average premium level is close to the expected cost per contract (7), but the average surplus level is slightly lower when using the policy for the model with a constraint on the action space compared to when using the policy for the model with the terminal state. However, the policies obtained for these two models are quite similar, and since (as discussed in Section 3.2) the model with the terminal state is more appropriate in the more realistic setting, we focus the remainder of the analysis only on the model with the terminal state.

### 5.1.2 Q-learning

We use the following step size parameter after the $k$th time a visit in state $s$ is followed by taking action $a$,

$$\alpha_{t^k(s,a)} = \frac{1}{k^{0.5+\theta}}, \quad \theta = 0.001.$$

This ensures that the stochastic approximation conditions (13) are satisfied, while still allowing for larger step sizes compared to the more standard choice $\alpha_t(s,a) = 1/t$. For the behaviour policy, we set $\varepsilon = 0.2$. The starting state is chosen uniformly at random from the state space. $Q(s,a)$ is initialised to zero for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$ to encourage initial exploration. Since all rewards are negative the true action-value function must be negative for all state-action pairs, hence setting the initial value

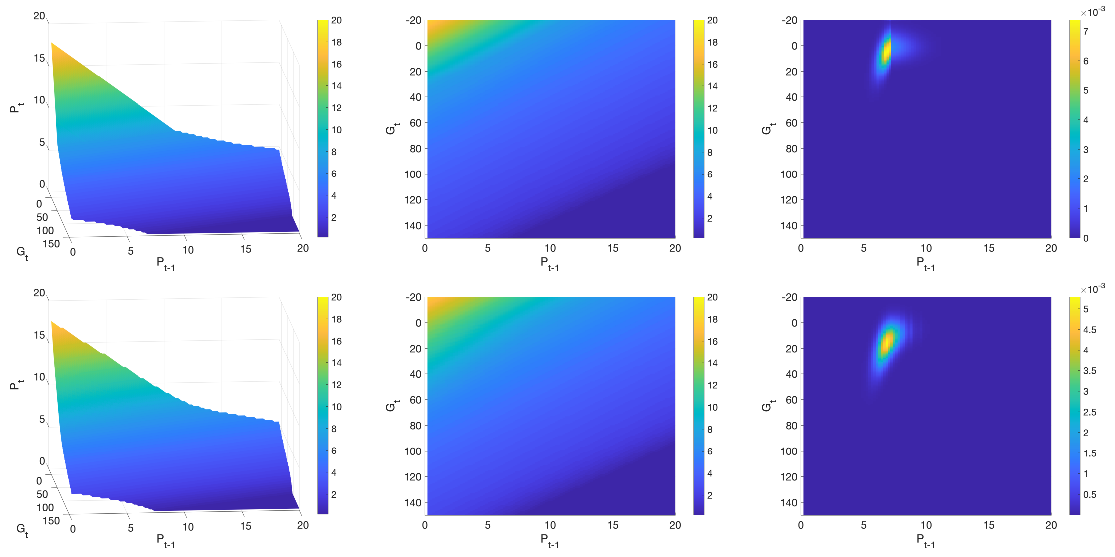

Figure 1: Simplified model using policy iteration. Top: with constraint. Bottom: with terminal state. First and second column: optimal policy. Third column: fraction of time spent in each state under the optimal policy.

to zero will encourage that all actions are tried early on. This technique for setting the initial values is called "optimistic initial values" in [18, Ch. 2.6]. To further encourage exploration of the state space, since discounting will lead to rewards after a large number of steps having a very limited effect on the total reward, we run each episode for at most 100 steps, before resetting to a starting state, again selected uniformly at random from $\mathcal{S}$.

Figure 2 shows the optimal policy for the simplified model using Q-learning. As can be seen in the figures, the Q-learning algorithm has not fully converged to the true optimal policy, despite having been run for a very large number of iterations. This is not too surprising when one considers the fraction of time spent in each state under the optimal policy, see Figure 1. The $\varepsilon$-greedy policy and restarting each episode after at most 100 steps ensures that exploration continues when using Q-learning, hence the fraction of time spent in each state during the Q-learning algorithm will not be quite as extreme as in Figure 1, but there are still many states that will be visited very rarely. Consider e.g. the probability of getting a negative surplus after charging a very high premium (i.e. ending up in the upper right corner of Figure 2); the claims payment in the period needs to be quite extreme for this state to be visited, unless the process starts in this state. We have used a step size that guarantees convergence of the algorithm, however, it is possible that a suitably chosen constant step size might lead to the algorithm converging faster.

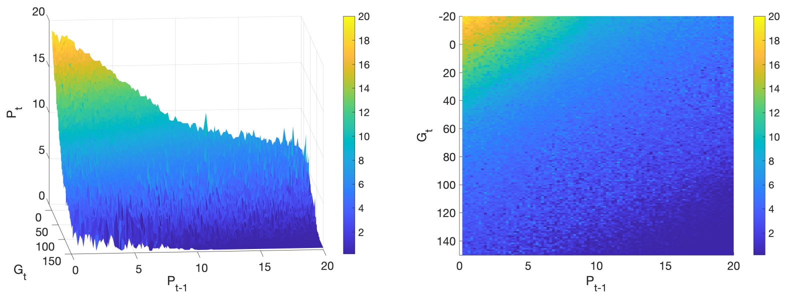

Figure 2: Optimal policy for simplified model with terminal state using Q-learning.

### 5.1.3 Linear function approximation

We have a 2-dimensional state space, hence $k+1=3$. When using the Fourier basis we should have $s \in [0,1]^k, a \in [0,1]$, hence we rescale the inputs according to

$$\tilde{s}_1 = \frac{s_1 - \min \mathcal{G}}{\max \mathcal{G} - \min \mathcal{G}}, \quad \tilde{s}_2 = \frac{s_2 - \min \mathcal{A}}{\max \mathcal{A} - \min \mathcal{A}}, \quad \tilde{a} = \frac{a - \min \mathcal{A}}{\max \mathcal{A} - \min \mathcal{A}},$$

and use $(\tilde{s}_1, \tilde{s}_2, \tilde{a})^\top$ as input. We use a softmax policy, i.e.

$$\pi(a|s) := \frac{e^{\hat{q}(s,a;\boldsymbol{w})/\tau}}{\sum_{a \in \mathcal{A}(s)} e^{\hat{q}(s,a;\boldsymbol{w})/\tau}},$$

where $\tau$ is slowly decreased according to

$$\tau_t = \max\{\tau_{\min}, \tau_0 d^{t-1}\}, \quad \tau_0 = 2, \quad \tau_{\min} = 0.02, \quad d = 0.99999,$$

where $\tau_t$ is the parameter used during episode $t$. This schedule for decreasing the temperature parameter is somewhat arbitrary and the parameters have not been tuned. It is hence possible that other ways of decaying $\tau$ and/or different values of $\tau_0$ and $\tau_{\min}$ might lead to improved results. The choice of a softmax policy is based on the results in [12, 11], discussed in Section 4.2.1. Since a softmax policy is Lipschitz continuous, convergence of SARSA to a unique policy is guaranteed, under the condition that the policy is also $\varepsilon$-soft and that the Lipschitz constant $L$ is small enough. However, since the temperature parameter $\tau$ is slowly decreased, the policy chosen is not necessarily $\varepsilon$-soft for all states and time steps, and the Lipschitz constant increases as $\tau$ decreases. Despite this, our results show that the algorithm converges to a policy that approximates the optimal policy derived with policy iteration well when using a 3rd order Fourier baisis, see the top row in Figure 3. The same cannot be said for an $\varepsilon$-greedy policy. In this case the algorithm converges to a policy that in general charges a higher premium than the optimal policy derived with policy iteration, see the bottom row in Figure 3. For the $\varepsilon$-greedy policy we decrease the parameter according to

$$\varepsilon_t = \max\{\varepsilon_{\min}, \varepsilon_0 d^{t-1}\}, \quad \varepsilon_0 = 0.2, \quad \varepsilon_{\min} = 0.01,$$

where $\varepsilon_t$ is the parameter used during episode $t$.

The starting state is selected uniformly at random from $\mathcal{S}$. Furthermore, since discounting will lead to rewards after a large number of steps having a very limited effect on the total reward, we run each episode for at most 100 steps, before resetting to a starting state, again selected uniformly at random from $\mathcal{S}$. This has the benefit of diversifying the states experienced, enabling us to achieve an approximate policy that is closer to the policy derived with dynamic programming as seen over the whole state space. The step size parameter used is

$$\alpha_t = \min\left\{\alpha_0, \frac{1}{t^{0.5+\theta}}\right\}, \tag{18}$$

where $\alpha_t$ is the step size parameter used during episode $t$, and $0 < \theta \leq 0.5$. An appropriate value of $\alpha_0$ can be find via trial end error, searching for the largest $\alpha_0$ that ensures that the weights do not explode. However, the value of $\alpha_0$ obtained in this way coincides with the the "rule of thumb" for setting the step-size parameter suggested in [18, Ch. 9.6], namely

$$\alpha_0 = \frac{1}{\mathrm{E}_\pi[x^\top x]}, \quad \mathrm{E}_\pi[x^\top x] = \sum_{s,a} \mu_\pi(s)\pi(a|s)x(s,a)^\top x(s,a).$$

If $x(S_t, A_t)^\top x(S_t, A_t) \approx \mathrm{E}_\pi[x^\top x]$, then this step size ensures that the error (i.e. the difference between the updated estimate $w_{t+1}^\top x(S_t, A_t)$ and the target $R_{t+1} + \gamma w_t^\top x(S_{t+1}, A_{t+1})$) is reduced to zero after one update. Hence, using a step size larger than $\alpha_0 = (\mathrm{E}_\pi[x^\top x])^{-1}$ risks overshooting the optimum, or even divergence of the algorithm. When using the Fourier basis of order $n$, this becomes

$$\begin{aligned}\mathrm{E}_\pi[x^\top x] &= \sum_{s,a} \mu_\pi(s)\pi(a|s) \sum_{i=1}^{(n+1)^{k+1}} \cos^2\left(\pi(sc_{1:k}^{(i)} + ac_{k+1}^{(i)})\right) \\ &= \frac{(n+1)^{k+1}}{2} + \frac{1}{2}\sum_{s,a} \mu_\pi(s)\pi(a|s) \sum_{i=1}^{(n+1)^{k+1}} \cos\left(2\pi(sc_{1:k}^{(i)} + ac_{k+1}^{(i)})\right),\end{aligned}$$

where the last term is approximately zero for the examples we have studied. For the simplified model we have used $\alpha_0 = 0.2$ for $n = 1$, $\alpha_0 = 0.07$ for $n = 2$, and $\alpha_0 = 0.03$ for $n = 3$. For the more realistic model we used $\alpha_0 = 0.06$ for $n = 1$, $\alpha_0 = 0.008$ for $n = 2$, and $\alpha_0 = 0.002$ for $n = 3$. In all cases this means that $\alpha_0 \approx (\mathrm{E}_\pi[x^\top x])^{-1}$. For $\theta$ we tried values in the set $\{0.001, 0.1, 0.2, 0.3, 0.4, 0.5\}$. For the simplified model, the best results were obtained with $\theta = 0.001$ irrespective of $n$. For the more realistic model we used $\theta = 0.5$ for $n = 1$, $\theta = 0.2$ for $n = 2$, and $\theta = 0.3$ for $n = 3$.

**Remark 5.2 (Step size)** *To decrease the amount of tuning required there are automatic methods for adapting the step size. One such method is the Autostep method from [8], which is a tuning-free version of the Incremental Delta-Bar-Delta (IDBD) algorithm from [17]. When using this method for setting the step size, the algorithm performs marginally worse compared to the results below, when the parameters of the method are simply set as suggested in [8], without any tuning.*

Figure 3 shows the optimal policy for the simplified model with terminal state using linear function approximation with 3rd, 2nd, and 1st order Fourier basis using a softmax policy, and with 3rd order Fourier basis using an $\varepsilon$-greedy policy. Comparing the 3rd order Fourier basis with Figure 1 the approximate optimal policy is close to the optimal policy derived wih policy iteration. The 2nd order Fourier basis also gives a reasonable approximation of the optimal policy, but performs a bit worse than the 3rd order Fourier basis. The 1st order Fourier basis performs slightly worse, but does not look unreasonable, while the 3rd order Fourier basis using an $\varepsilon$-greedy policy performs considerably worse.

The same conclusions can be drawn from Table 2, where we see the expected total discounted reward per episode for these policies, together with the results for the optimal policy derived with policy iteration, the policy derived with Q-learning, and several benchmark policies (see Section 5.1.4). Clearly the performance of 3rd order Fourier basis is very close to the performance of the optimal policy derived with policy iteration, hence we conclude that the linear function approximation with 3rd order Fourier basis using a softmax policy appears to converge to approximately the optimal policy. The policy derived with Q-learning shows worse performance than both the 3rd and 2nd order Fourier basis, while the number of episodes run for the Q-learning algorithm is approximately a factor 30 bigger than the number of episodes run before convergence of SARSA with linear function approximation. Hence, even for this simplified model, the number of states is too large for the Q-learning algorithm to converge within a reasonable amount of time, and linear function approximation is thus preferred. Furthermore, we see that all policies derived with linear function approximation using a softmax policy outperform the benchmark policies.

Note that the optimal policy derived with policy iteration, the best constant policy, and the myopic policy with the terminal state requires full knowledge of the underlying model and the transition probabilities, and the myopic policy with the constraint requires an estimate of the expected surplus one time-step ahead, while the policies derived with function approximation or Q-learning only require real or simulated experience of the environment.

To analyse the difference between some of the policies, we simulate 300 episodes for the policy with the 3rd order Fourier basis, the best constant policy, and the two myopic policies with the terminal state, for a few different starting states, two of which can be seen in Figure 4. Note that each star in figures correspond to one or more terminations at that time point. The total number of terminations (of 300 episodes) are: $S_0 = (-10, 2)$: Fourier 3: 1, best constant: 291, myopic $p_{\min} = 5.8$: 20, myopic $p_{\min} = 0.2$: 30. $S_0 = (50, 7)$: Fourier 3: 1, best constant: 0, myopic $p_{\min} = 5.8$: 20, myopic $p_{\min} = 0.2$: 29. For other starting states, the comparison is similar to that in Figure 4. We see that the policy with the 3rd order Fourier basis appears to outperform the myopic policies in all respects, i.e. on average the premium is lower, the premium is more stable over time, and we have very few defaults. The best constant policy naturally is the most stable, but leads to in general a higher premium compared to the other policies, and will for more strained starting states quickly lead to a large number of terminations. The myopic policy with $p_{\min} = 0.2$ leads to a premium that varies wildly, often jumping from charging the lowest premium of 0.2 to a very high premium in the next time step. The myopic policy with $p_{\min} = 5.8$ is more stable. It does, however, lead to a much larger number of terminations compared to the policy with the 3rd order Fourier basis, as it in general tends to charge a slightly too low premium over time.

### 5.1.4 Benchmark policies

**Best constant policy.** The best constant policy is the solution to the optimisation problem

$$\underset{p}{\text{minimise}} \ \mathrm{E}\Big[\sum_{t=0}^{T} \gamma^t h(p, S_{t+1})\Big].$$

For both the simplified model and in the more realistic setting, $p = 7.4$ solves this optimisation problem. For details on solving this problem, see Appendix A.1.

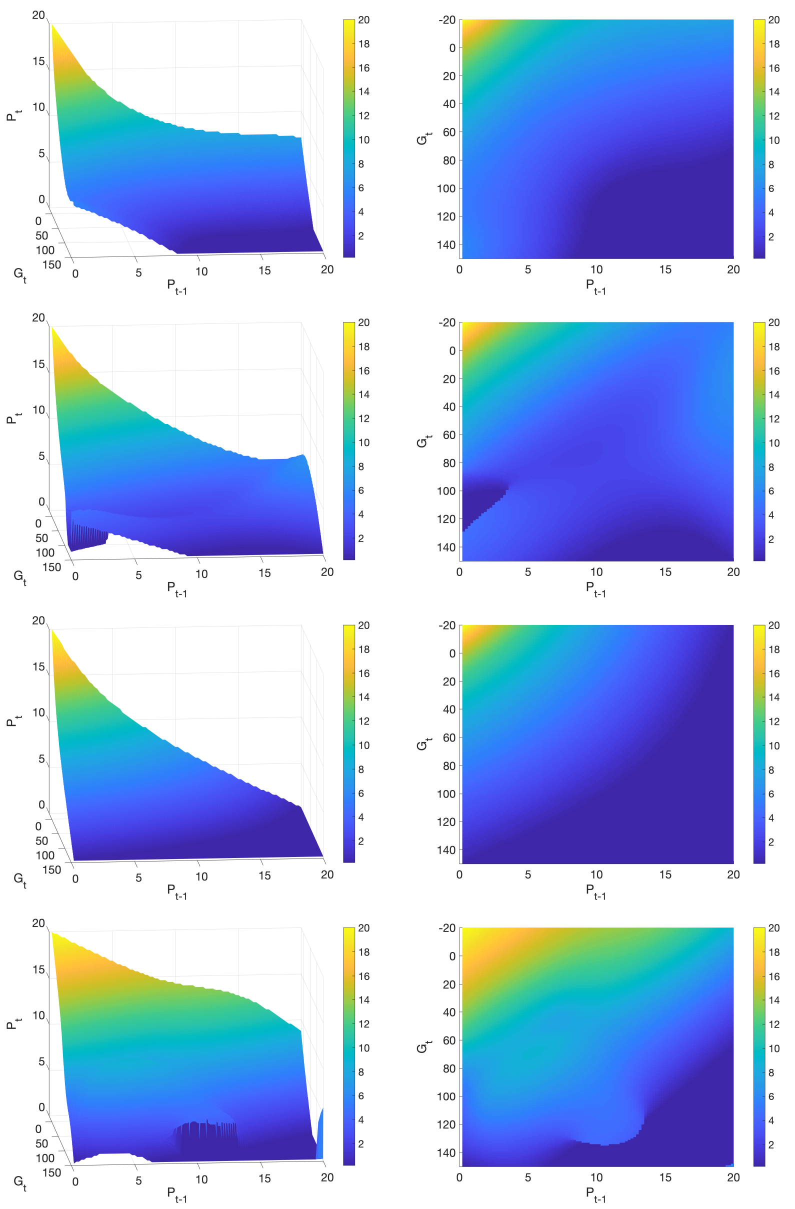

Figure 3: Optimal policy for simplified model with terminal state using linear function approximation. First row: 3rd order Fourier basis. Second row: 2nd order Fourier basis. Third row: 1st order Fourier basis. Fourth row: 3rd order Fourier basis with $\varepsilon$-greedy policy.

| | Expected reward | Terminations |
|---|---|---|
| Policy iteration | $-85.91$ | 0.006 |
| Q-learning | $-86.50$ | 0.002 |
| Fourier 3 with softmax policy | $-86.11$ | 0.006 |
| Fourier 2 with softmax policy | $-86.30$ | 0.007 |
| Fourier 1 with softmax policy | $-86.59$ | 0.011 |
| Fourier 3 with $\varepsilon$-greedy policy | $-92.74$ | 0.000 |
| Best constant policy | $-122.70$ | 0.040 |
| Myopic policy with terminal state, $p_{\min} = 0.2$ | $-97.06$ | 0.133 |
| Myopic policy with terminal state, $p_{\min} = 5.8$ | $-90.40$ | 0.096 |
| Myopic policy with constraint, $p_{\min} = 0.2$ | $-121.52$ | 0.337 |
| Myopic policy with constraint, $p_{\min} = 6.4$ | $-93.58$ | 0.132 |

Table 2: Expected discounted total reward, terminations, and average premium (uniformly distributed starting states) for simplified model with terminal state. The column terminations shows the fraction of the simulated episodes that end in the terminal state, within 100 time steps.

**Myopic policy for MDP with constraint.** The myopic policy is the policy that maximises immediate (next-step) rewards. For the model with a constraint on the action space, the myopic policy is the solution to the following optimisation problem

$$\underset{p}{\text{minimise}}\ \mathrm{E}[c(p) \mid S_0 = s, P_0 = p] \quad \text{subject to}\ \mathrm{E}[G_1 \mid S_0 = s, P_0 = p] \geq 0. \tag{19}$$

Since $c$ is an increasing function, it is easy to compute the myopic policy; it is given by the lowest premium level that satisfies the constraint. The myopic policy for the MDP with a constraint on the action space can be seen in Figure 5 for the simplified model. For more details on how (19) is solved, see Appendix A.2.

For both the simplified model and in the more realistic setting, the myopic policy charges the minimum premium level of 0.2 for a large number of states. Since this policy so quickly reduces the premium to 0.2 as the surplus or previously charged premium increases, it is not likely to work that well. Hence, we suggest an additional benchmark policy where we set the minimum premium level to a higher value, $p_{\min}$. Thus this adjusted myopic policy is given by $\tilde{\pi}(s) = \max\{\pi(s), p_{\min}\}$, where $\pi$ denotes the policy that solves (19). Based on simulations of the total expected discounted reward per episode for different values of $p_{\min}$, we conclude that $p_{\min} = 6.4$ achieves the best results in both the simplified and the more realistic setting.

**Myopic policy for MDP with terminal state.** For the model with a terminal

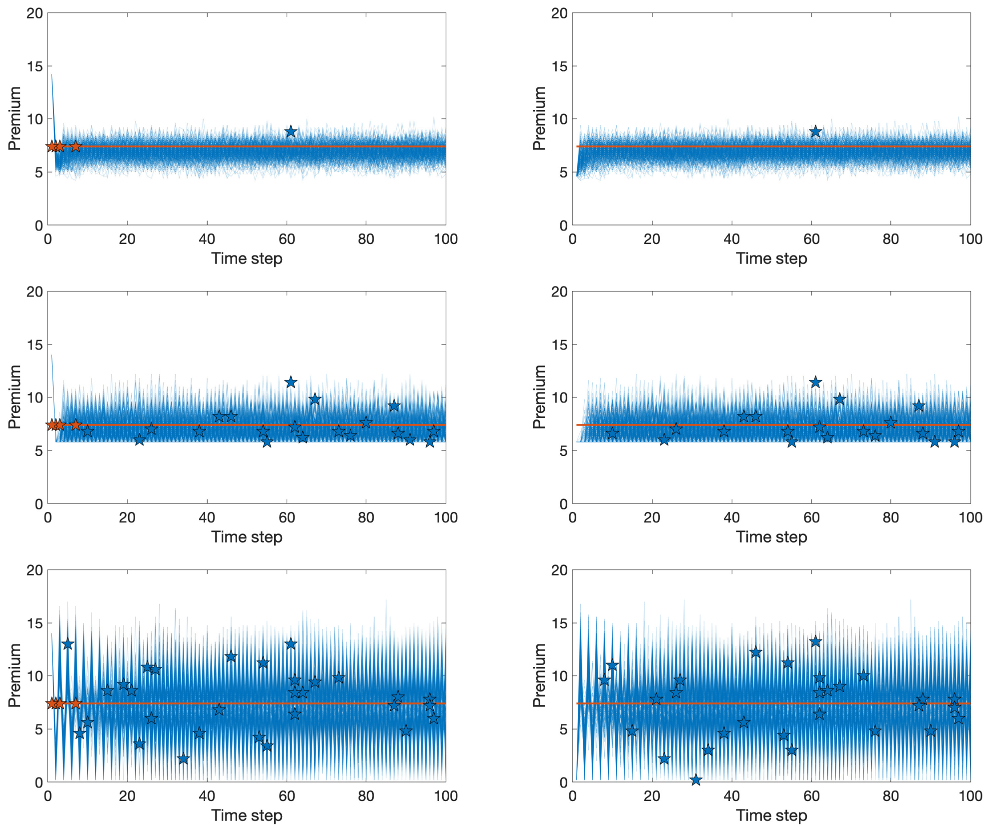

Figure 4: Simplified model. First row: policy with 3rd Fourier basis. Second row: myopic policy with terminal state, $p_{\min} = 5.8$. Third row: myopic policy with terminal state, $p_{\min} = 0.2$. Left: starting state $S_0 = (-10, 2)$. Right: starting state $S_0 = (50, 7)$. The red line shows the best constant policy. Each star indicates at least one termination at that time step.

state, the myopic policy is the solution to the optimisation problem

$$\underset{p}{\text{minimise}}\ \mathrm{E}[h(p, S_1) \mid S_0 = s, P_0 = p]. \tag{20}$$

The myopic policy for the MDP with a terminal state can be seen in Figure 5 for the simplified model. For more details on how (20) is solved, see Appendix A.3. As in the previous section, we also suggest an additional benchmark policy where the minimum premium level has been set to $p_{\min} = 5.8$, which is the minimum premium level that achieves the best results based on simulations of the total expected discounted reward per episode. For the more realistic model this myopic policy is too complex and is therefore not a good benchmark.

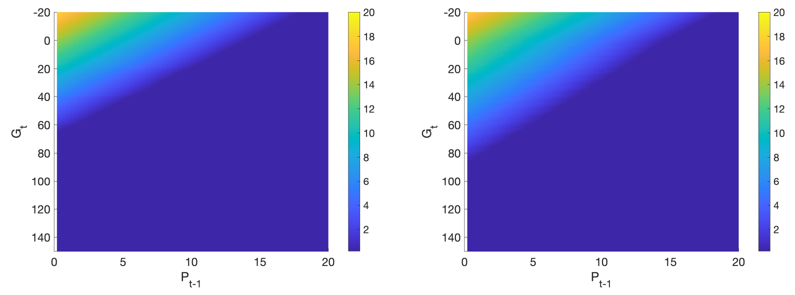

Figure 5: Left: myopic policy for simplified model with constraint. Right: myopic policy for simplified model with terminal state.

## 5.2 More realistic model

We use the following parameter values: $\gamma = 0.9$, $\mu = 5$, $\beta_0 = 10$, $\beta_1 = 1$, $\xi = 0.05$, $\nu = 1$, $\alpha_1 = 0.7$, $\alpha_2 = 0.3$, $\eta = 10$, $a = 18$, $b = -0.3$, and cost function (17) with $c_1 = 1$ and $c_2 = 1.2$. The premium level and number of contracts written are truncated and discretised according to $\mathcal{A} = \{0.2, 0.4, \ldots, 19.8, 20.0\}$ and $\mathcal{N} = \{0, 1, \ldots, 30\}$. The surplus process no longer only takes integer values (as in the simplified model), instead the values that the surplus process can take are determined by the parameter values chosen. However, it is still truncated to lie between -20 and 150. For the parameter values above, we have $\mathcal{G} = \{-20.00, -19.95, \ldots, 149.95, 150.00\}$.

Figure 6 shows the optimal policy in the more realistic setting using linear function approximation, with 3rd, 2nd, and 1st order Fourier basis, for $N_t = N_{t-1} = 10$. Comparing the policy with the 3rd order Fourier basis with the policy with the 2nd order Fourier basis, the former appears to require a slightly lower premium when the surplus or previously charged premium is very low. The policy with the 1st order Fourier basis appears quite extreme compared to the other two policies. Comparing the policy with the 3rd order Fourier basis for $N_t = N_{t-1} = 10$ with the optimal policy for the simplified model (bottom row in Figure 1), we note that $\pi^r(g, p, 10, 10) \neq \pi^s(g, p)$, where $\pi^s$ and $\pi^r$ denotes respectively the policy in the simplified setting and the policy in the more realistic setting. There is a qualitative difference between these policies, since even given that we are in a state where $N_t = N_{t-1} = 10$ in the more realistic setting, the policy from the simplified model does not take into account the affect that the premium charged will have on the number of contracts issued at time $t+1$, hence it e.g. charges too low of a premium when the surplus or previously charged premium are high, compared with the approximate optimal policy in the more realistic setting. The policy with 3rd order Fourier basis for $N_t, N_{t-1} \in \{5, 10, 15\}$ can be seen in Figure 7.

To determine the performance of the policies in the more realistic setting, we simulate the expected total discounted reward per episode for these policies. The results can be seen in Table 3, together with the results for some benchmark policies. Here we clearly see that the policy with 3rd order Fourier basis outperforms the other policies, and that the policy with 1st order Fourier basis performs quite badly, indicating that the 1st order Fourier basis is not flexible enough to be used in this more realistic setting. We also compare the policies with the benchmark policy of using the optimal policy derived with policy iteration from the simplified model in this more realistic setting, i.e. this policy is used without taking the number of contracts into account. Though this policy performs worse compared to the policy with 3rd and 2nd order Fourier basis, it does in fact clearly outperform the policy with 1st order Fourier basis. Note that the policies derived with function approximation only require real or simulated experience of the environment. The results for the myopic policy in Table 3 are based on using the true parameters when computing the expected value of the surplus. For a more fair comparison, these parameters should instead be estimated based on real or simulated experience of the environment, which would likely degrade the performance of the myopic policy somewhat. Despite this, the policy derived with the 3rd order Fourier basis clearly outperforms the myopic policy with the terminal state.

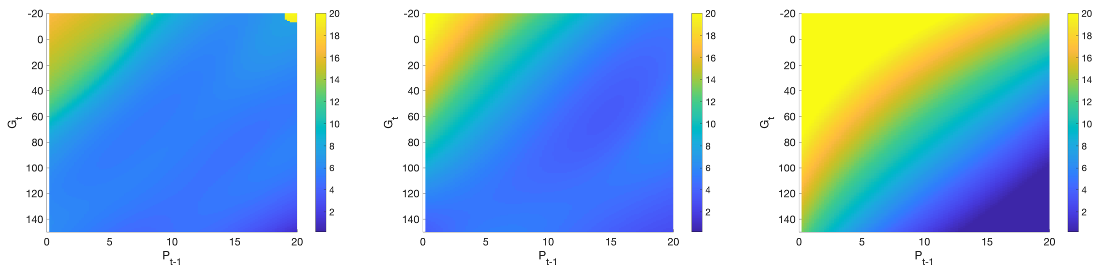

Figure 6: Optimal policy in more realistic setting with terminal state using linear function approximation, for $N_t = N_{t-1} = 10$. Left: 3rd order Fourier basis. Middle: 2nd order Fourier basis. Right: 1st order Fourier basis.

To analyse the difference between some of the policies, we simulate 300 episodes for the policy with the 3rd order Fourier basis, the best constant policy, the policy from the simplified model, and the myopic policy with $p_{\min} = 6.4$, for a few different

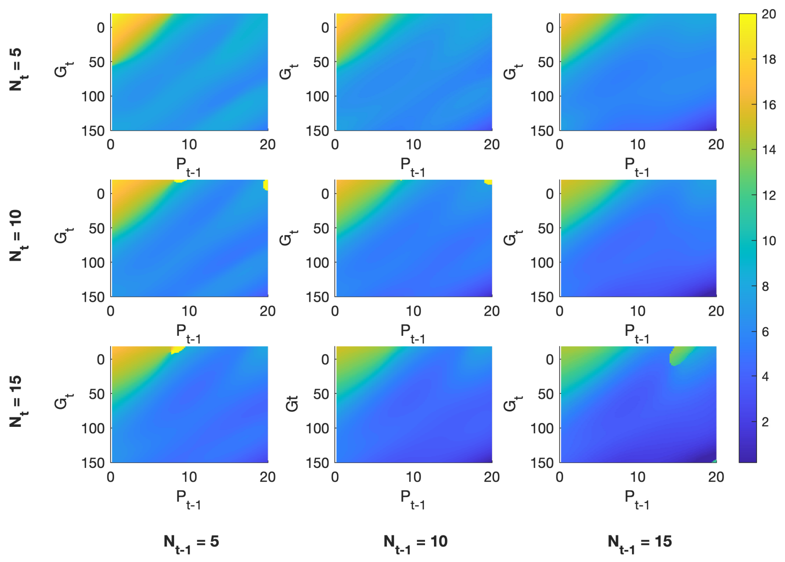

Figure 7: Optimal policy in more realistic setting with terminal state using linear function approximation with 3rd order Fourier basis, for $N_t, N_{t-1} \in \{5, 10, 15\}$.

| | Expected reward | Terminations |
|---|---|---|
| Fourier 3 | $-97.17$ | 0.015 |
| Fourier 2 | $-104.41$ | 0.025 |
| Fourier 1 | $-128.83$ | 0.036 |
| Policy from simplified model | $-116.70$ | 0.075 |
| Myopic policy with constraint, $p_{\min} = 0.2$ | $-360.69$ | 0.988 |
| Myopic policy with constraint, $p_{\min} = 6.4$ | $-100.92$ | 0.169 |
| Best constant policy | $-131.85$ | 0.060 |

Table 3: Expected discounted total reward based on simulation, (uniformly distributed starting states). The column terminations shows the fraction of the simulated episodes that end in the terminal state, within 100 time steps.

starting states, two of which can be seen in Figure 8. Note that each star in figures correspond to one or more terminations at that time point. The total number of terminations (of 300 episodes) are: $S_0 = (0, 7, 10, 10)$: Fourier 3: 1, best constant: 13, simplified: 5, myopic $p_{\min} = 6.4$: 66. $S_0 = (100, 15, 5, 5)$: Fourier 3: 0, best constant: 0, simplified: 10, myopic $p_{\min} = 6.4$: 35. Comparing the policy with the 3rd order Fourier basis with the policy from the simplified model, we see that it tends to on average give a lower premium and leads to very few defaults, but is slightly more variable compared to the premium charged by the simplified policy. This is not surprising, since the simplified policy does not take the variation in the number of contracts issued into account, and hence does not need to adjust the premium level for these changes. At the same time, this is to the detriment of the simplified policy, since it cannot correctly take the risk of the varying number of contracts into account, hence leading to more defaults. For example, for the more strained starting state $S_0 = (-10, 2, 20, 20)$ (not shown in figure), the number of defaults for the policy with the 3rd order Fourier basis is 91 of 300, and for the simplified policy it is 213 of 300. Similarly, for starting state $S_0 = (100, 15, 5, 5)$ (second column in Figure 8), the simplified policy will tend set the premium much too low during the first time step, not taking into account the effect this will have on the number of contracts, hence leading to more defaults very early on compared to e.g. starting state $S_0 = (0, 7, 20, 20)$ (first column in Figure 8), despite the fact that this starting state has a much lower starting surplus. This also explains why the simplified policy performs worse compared to the myopic policy with $p_{\min} = 6.4$ in Table 3: the early terminations for the simplified policy in e.g. state $S_0 = (100, 15, 5, 5)$ have a larger effect on the total expected reward than the large number of terminations at later time steps for the myopic policy.

## 6 Conclusion

Classical methods for solving premium control problems are suitable for dynamical insurance systems that are not too complex, and the model choice must to a large extent be based on how to make the problem solvable, rather than reflecting the real dynamics of the stochastic environment. For this reason, the practical use of the optimal premium rules derived with classical methods is often limited.

Reinforcement learning methods enable us to solve premium control problems in more realistic settings that adequately capture the complex dynamics of the system. Since these techniques can learn directly from real or simulated experience of the stochastic environment, they do not require explicit expressions for transition probabilities. Further, these methods can be combined with function approximation in order to overcome the curse of dimensionality as the state space tends to increase in more realistic settings. This makes it possible to take key features of real dynamical insurance systems into account, e.g. how the number of contracts issued in the future will vary depending on the premium rule. Hence, the optimal policies derived with these techniques can be used as a basis for decisions on how to set the premium for insurance companies, since it is not based on an overly simplistic model of reality.

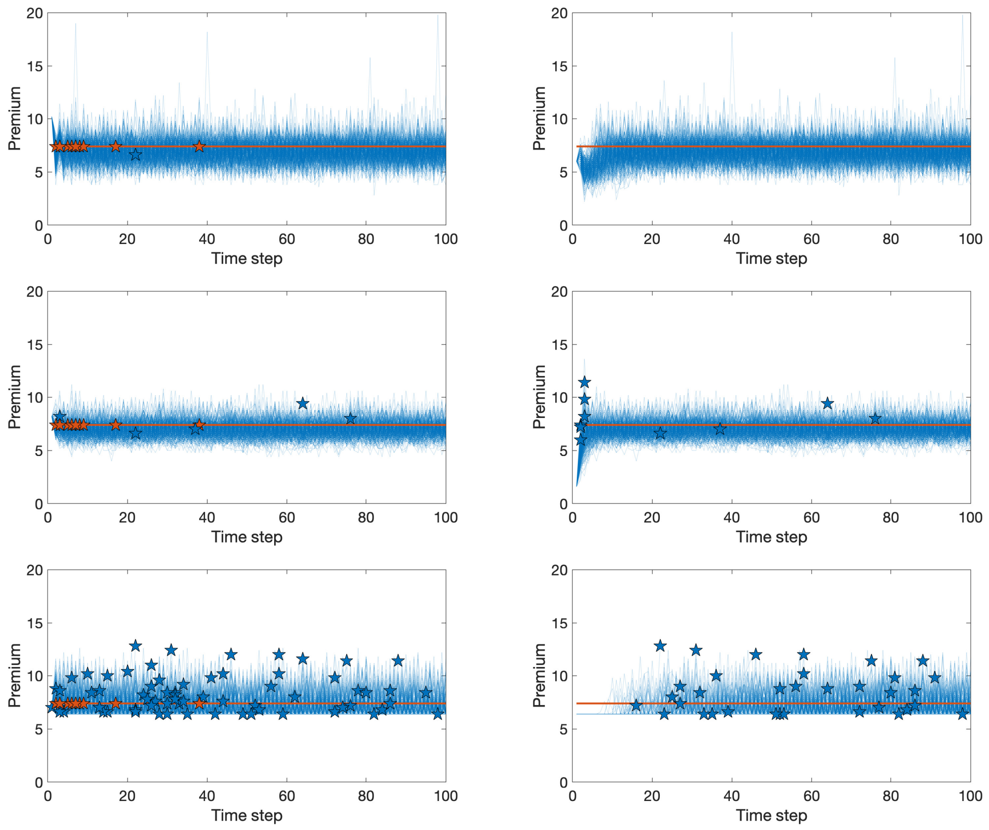

Figure 8: More realistic setting. First row: policy with 3rd Fourier basis. Second row: policy from the simplified model. Third row: myopic policy with constraint, $p_{\min} = 6.4$. Left: starting state $S_0 = (0, 7, 20, 20)$. Right: starting state $S_0 = (100, 15, 5, 5)$. The red line shows the best constant policy. Each star indicates at least one termination at that time step.

We have illustrated strengths and weaknesses of different methods for solving the premium control problem for a mutual insurer, and demonstrated that given a more complex dynamical system, the approximate policy derived with SARSA using function approximation outperforms several benchmark policies. In particular, it clearly outperforms the policy derived with classical methods based on a more simplistic model of the stochastic environment, which fails to take important aspects of a real dynamical insurance system into account. Furthermore, the use of these methods is not specific to the model choices made in Section 2. The present paper provides guidance on how to carefully design a reinforcement learning method with function approximation for the purpose of obtaining an optimal premium rule, which together with models that fit the experience of the specific insurance company allows for optimal premium rules that can be used in practice.

## 7 Acknowledgements

Filip Lindskog would like to acknowledge financial support from the Swedish Research Council, Project 2020-05065, and from Länsförsäkringars Forskningsfond, Project P9.20. We would also like to thank Mathias Lindholm for many interesting discussions.

## References

[1] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming.* Athena Scientific, 1996.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[3] Peter Dayan. The convergence of TD($\lambda$) for general $\lambda$. *Machine learning*, 8(3-4):341–362, 1992.

[4] Bruno de Finetti. Su un' impostazione alternativa della teoria collettivadel rischio. *Transactions of the XVth International Congress of Actuaries*, 1957(2):433–443, 1957.

[5] Hans U Gerber. *Entscheidungskriterien für den zusammengesetzten Poisson-Prozess.* PhD thesis, ETH Zurich, 1969.

[6] Geoffrey J Gordon. Reinforcement learning with function approximation converges to a region. In *Advances in neural information processing systems*, pages 1040–1046, 2001.

[7] George Konidaris, Sarah Osentoski, and Philip Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Twenty-fifth AAAI conference on artificial intelligence*, 2011.

[8] Ashique Rupam Mahmood, Richard S Sutton, Thomas Degris, and Patrick M Pilarski. Tuning-free step-size adaptation. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2121–2124. IEEE, 2012.

[9] Anders Martin-Löf. Premium control in an insurance system, an approach using linear control theory. *Scandinavian Actuarial Journal*, 1983(1):1–27, 1983.

[10] Anders Martin-Löf. Lectures on the use of control theory in insurance. *Scandinavian Actuarial Journal*, 1994(1):1–25, 1994.

[11] Francisco S Melo, Sean P Meyn, and M Isabel Ribeiro. An analysis of reinforcement learning with function approximation. In *Proceedings of the 25th international conference on Machine learning*, pages 664–671, 2008.

[12] Theodore J Perkins and Doina Precup. A convergent form of approximate policy iteration. In *Advances in neural information processing systems*, pages 1627–1634, 2003.

[13] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2005.

[14] Walter Rudin. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1976.

[15] Hanspeter Schmidli. *Stochastic control in insurance.* Springer, 2008.

[16] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308, 2000.

[17] Richard S Sutton. Adapting bias by gradient descent: An incremental version of Delta-Bar-Delta. In *AAAI*, pages 171–176. San Jose, CA, 1992.

[18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[19] John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE transactions on automatic control*, 42(5):674–690, 1997.

[20] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

## A Benchmark policies

### A.1 Best constant policy

When using a constant policy $p$ irrespective of state, $S_t = (G_t, p)$ for all $t > 0$, for the simplified model. We want to find $p$ that minimises

$$\begin{aligned} \mathrm{E}\left[\sum_{t=0}^{T} \gamma^t h(p, S_{t+1})\right] &= \mathrm{E}\left[\sum_{t=0}^{T-1} \gamma^t c(p) + \gamma^T (1+\eta) c(\max \mathcal{A})\right] \\ &= \mathrm{E}\left[c(p)\frac{1-\gamma^T}{1-\gamma} + \gamma^T(1+\eta)c(\max\mathcal{A})\right] \\ &= \frac{c(p)}{1-\gamma} + \left((1-\eta)c(\max\mathcal{A}) - \frac{c(p)}{1-\gamma}\right)\mathrm{E}[\gamma^T], \end{aligned} \tag{21}$$

where $\mathrm{E}[\gamma^T] = \sum_{t=1}^{\infty} \mathrm{P}(T = t)\gamma^t$ and

$$\mathrm{P}(T \le t) = \sum_{s' \in \mathcal{S}^+ \setminus \mathcal{S}} \mathrm{P}(S_t = s') = \mathrm{P}(G_t < \min \mathcal{G}).$$

Since the state space is finite, we can label the states $0, 1, \ldots, |\mathcal{G}||\mathcal{A}|$ (where state 0 represents all terminal (absorbing) states. Let $P = (p_{ij} : i, j \in \{0, 1, \ldots, |\mathcal{G}||\mathcal{A}|\})$, where $p_{ij} = \mathrm{P}(S_t = j \mid S_{t-1} = i)$, and $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{|\mathcal{G}||\mathcal{A}|})^\top$, where $\lambda_j = \mathrm{P}(S_0 = j)$ and $\lambda_0 = 0$. Then

$$\mathrm{P}(G_t = k) = \sum_j (\lambda^\top P^t)_{\{j : G_t = k\}}.$$

Based on this we can compute (21) for each $p$ (computing $\mathrm{E}[\gamma^T]$ by truncating the sum at some large value) from which we determine that $p = 7.4$ minimises (21), for the simplified model.

In the more realistic setting, we are not able to compute the best constant policy as above due to the dimension of the state space. Instead we can simulate the total expected discounted reward per episode for different values of the constant policy $p$. Based on these simulations we can again conclude that $p = 7.4$ minimises (21) also in the more realistic setting.

### A.2 Myopic policy for MDP with constraint

The myopic policy is the policy that maximises immediate (next-step) rewards. For the model with a constraint on the action space, the myopic policy is the solution to the following optimisation problem

$$\underset{p}{\text{minimise}}\ \mathrm{E}[c(p) \mid S_0 = s, P_0 = p] \quad \text{subject to}\ \mathrm{E}[G_1 \mid S_0 = s, P_0 = p] \geq 0.$$

Since $c$ is an increasing function, it is easy to compute the myopic policy; it is given by the lowest premium level that satisfies the constraint. For the simplified model we have

$$G_{n+1} = G_n + \frac{1}{2}N(P_n + P_{n-1}) - \beta_1 N - \beta_0 - \mathrm{PC}_{n+1} + \mathrm{IE}_{n+1}.$$

Hence, for each $s = (g, q)$, we need to find the lowest premium level $p = \pi(g, q)$ that satisfies

$$(1 + \xi \mathbf{1}_{\{g>0\}})g + \frac{1}{2}N(p + q) - (\beta_0 + (\beta_1 + \mu)N) \geq 0,$$

hence

$$\pi(g, q) = \min \Big\{ p \in \mathcal{A} : p \geq 2\Big(\beta_1 + \mu + \frac{\beta_0 - (1 + \xi \mathbf{1}_{\{g>0\}})g}{N} - q\Big)\Big\}.$$

For the more realistic model we have

$$\begin{aligned} G_{n+1} = G_n &+ \frac{1}{2}(N_{n+1}P_n + N_n P_{n-1}) - (\alpha_2\mu + \beta_1)\frac{1}{2}(N_{n+1} + N_n) + \alpha_2\mu\frac{1}{2}(N_n + N_{n-1}) - \beta_0 \\ &- \mathrm{PC}_{n+1} + \mathrm{IE}_{n+1}. \end{aligned}$$

Hence, for each $s = (g, q, n_0, n_{-1})$, we want to find the lowest premium level $p = \pi(g, q, n_0, n_{-1})$ that satisfies

$$(1 + \xi \mathbf{1}_{\{g>0\}})g + \frac{1}{2}ap^b(p - \beta_1 - \mu) + \frac{1}{2}n_0(q - \beta_1 - \mu) - \beta_0 \geq 0. \tag{22}$$

Note that $\pi(g, q, n_0, n_{-1})$ does not depend on $n_{-1}$. Let $\mathcal{P}(g, q, n_0)$ be the set of premium levels such that (22) is satisfied. Note that for our choice of $\mathcal{A}$ and $\mathcal{S}$, there exist $(g, q, n_0, n_{-1}) \in \mathcal{S}$ such that $\mathcal{P}(g, q, n_0) \cap \mathcal{A} = \emptyset$. Hence we let the myopic policy for the more realistic model with a constraint on the action space be given by

$$\pi(g, q, n_0, n_{-1}) = \begin{cases} \min\{p \in \mathcal{A} : p \in \mathcal{P}(g, q, n_0)\}, & \text{if } \mathcal{P}(g, q, n_0) \cap \mathcal{A} \neq \emptyset, \\ \max \mathcal{A}, & \text{if } \mathcal{P}(g, q, n_0) \cap \mathcal{A} = \emptyset, \end{cases}$$

i.e. if the constraint are not satisfied for any $p \in \mathcal{A}$, then the maximum premium level is chosen.

## A.3 Myopic policy for MDP with terminal state

For the model with a terminal state, for each state $s \in \mathcal{S}$ we want to find $p \in \mathcal{A}$ that minimises

$$\begin{aligned}\mathrm{E}[h(p, S_1) \mid S_0 = s, P_0 = p] = c(p)\,\mathrm{P}(G_1 \geq \min \mathcal{G} \mid S_0 = s, P_0 = p) \\ + c(\max \mathcal{A})(1+\eta)\,\mathrm{P}_\pi(G_1 < \min \mathcal{G} \mid S_0 = s, P_0 = p),\end{aligned} \tag{23}$$

hence for this model the myopic policy is not quite as easy to compute as for the case when we have a constraint on the action space, since we now need to determine

$$\mathrm{P}(G_1 \geq \min \mathcal{G} \mid S_0 = s, P_0 = p) = \sum_{k=\min \mathcal{G}}^{\infty} \mathrm{P}(G_1 = k \mid S_0 = s, P_0 = p)$$

instead of the expectation in (19). From Section 3.3 we see that for the simplified model

$$\begin{aligned}&\mathrm{P}(G_1 = k \mid S_0 = (g, q), P_0 = p) \\ &\quad = \begin{cases} \mathrm{P}\left(\mathrm{PC}_1 = g - m\right) & \text{if } g \leq 0, \\ \sum_{\{l : m+l \geq 0\}} \mathrm{P}(\mathrm{PC}_1 = l)\,\mathrm{P}\left(\mathrm{IE}_1 + G_0 = m + l \mid G_0 = g\right) & \text{if } g > 0, \end{cases}\end{aligned}$$

where $m = k + (\beta_0 + \beta_1 N) - N(p+q)/2$. Based on this we can compute the expectation in (23) for each premium level.