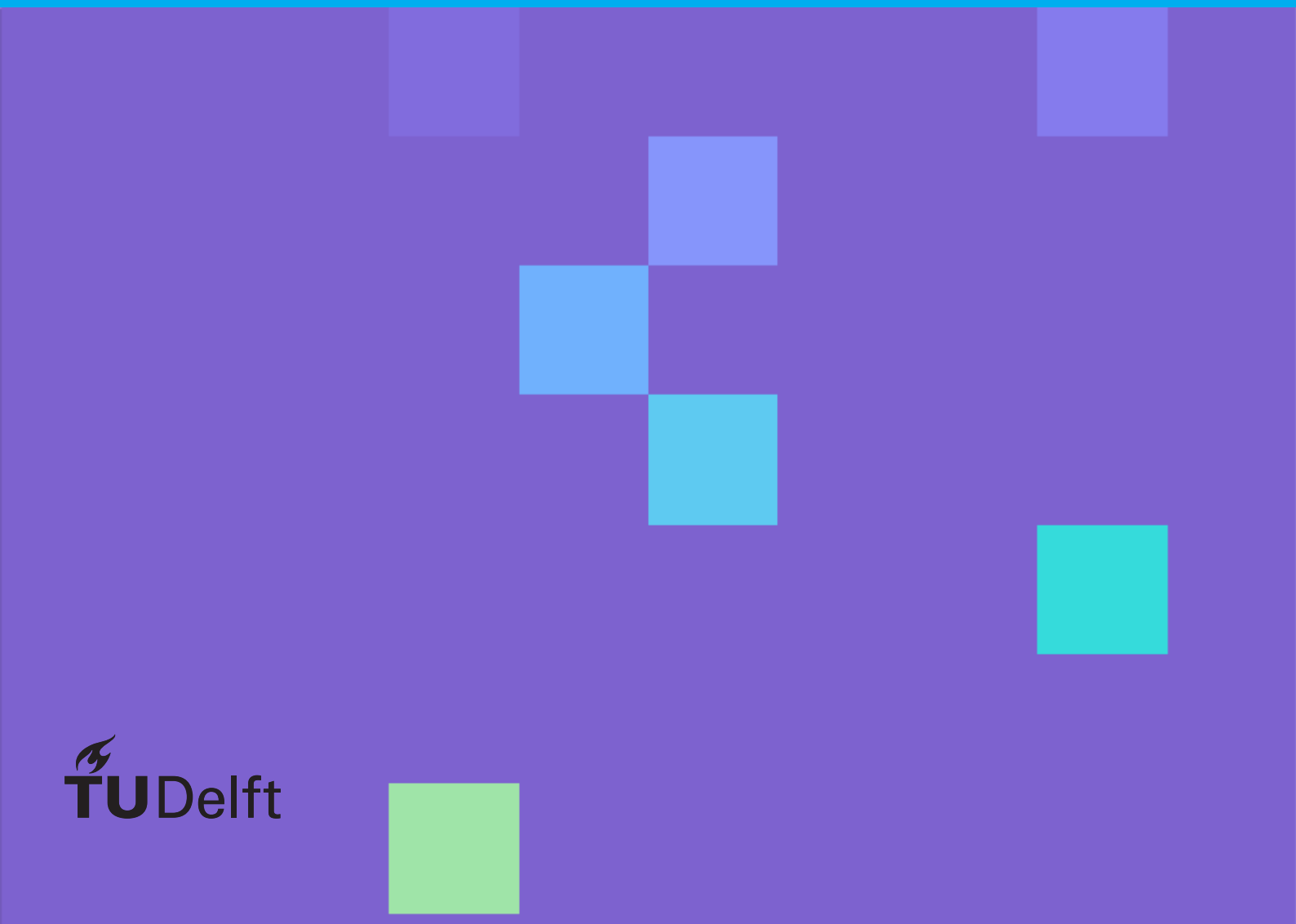
Discrete state-space active inference in nonstationary environments

Naresh Pancham

MSc Thesis



TU Delft

Discrete state-space active inference in nonstationary environments

by

Naresh Pancham

to obtain the degree of Master of Science in Mechanical Engineering
at the Delft University of Technology,
to be defended publicly on Tuesday April 6, 2021 at 10:00 AM.

Student number:	4235088	
Thesis committee:	Prof. dr. ir. M. Wisse,	TU Delft, supervisor
	Ir. A. Anil Meera,	TU Delft, supervisor
	Dr. Ing. J. Kober,	TU Delft

TU Delft

Preface

When I started my Master's programme, I would never have imagined that I would finish it by working mostly from home due to a worldwide pandemic. It was quite the contrast going from the crowded lecture halls, study spots and corridors to working in my own room. Even though the idea of not having to leave home sounds fantastic on paper, this also means working individually for the majority of time, which is not always as easy. But working on your own does not necessarily mean being left on your own. For that, I want to share my gratitude.

First and foremost, I want to thank my supervisors Martijn Wisse and Ajith Anil Meera for their supervision. Since most of our conversations were digital — whether it was by email exchange or video meeting — communication was not always as smooth and seamless as an in-person meeting. Nevertheless, the feedback I got from these sessions has been invaluable to me.

I want to thank Martijn specifically for his continuous supervision from the start to the end of the process. Because of the relatively large number of graduate students under your direct and indirect supervision, I can believe it is hard to know in detail what every student is working on. Despite this, you were always quick to grasp the essence of my questions. Doing my Master's thesis in your research group has been a very informative experience.

I want to thank Ajith for his supervision on a more practical level. The theory of active inference is not exactly known for being easily comprehensible to new researchers, and as such it is not always easy to get out of predicament. It was reassuring to know that I was able to contact someone with more experience on this topic whenever I got in such a situation.

Last but not least, I want to thank my family, especially my mother, for the continuous support I got. Without your help throughout life, I might not even have made it this far in the first place.

Naresh Pancham
March 2021

Abstract

Active inference is a neuroscientific theory, which states that all living systems (e.g. the human brain) minimize a quantity termed the free energy. By minimizing this free energy, living systems keep an accurate representation of the world in their internal model (learning), are provided with an optimal way of acting on the world (action selection), and can predict incoming sensory data (perception). Considering the fact that these properties are sought after in artificial intelligence systems as well, active inference has also become an interesting topic from an engineering point of view.

The application of active inference can be done with both a continuous and a discrete state-space framework. However, research on discrete state-space active inference has neglected the extension of its applicability to nonstationary environments. This work aims to fill that gap. More specifically, the goal of this research is to evaluate the performance of state-of-the-art discrete state-space active inference agents in nonstationary environments, and assess whether forgetting part of the agent's previous experiences can increase its performance.

The type of nonstationarity that is used in this work is cyclostationarity, and this nonstationarity will only be manifested in the transition process of the active inference task. Moreover, the specific type of task solved is one of planning and navigation in a gridworld. Since the agent has to deal with a planning and navigation task, performance is quantified by the number of steps the agent needs to take in order to reach its goal.

Three methods of forgetting are implemented and compared, inspired by techniques from reinforcement learning, deep learning and time series analysis respectively. These are: (1) the use of a constant forget rate, (2) the use of the updating mechanism of a long short-term memory (LSTM) cell applied to the updating of the generative model in active inference, and (3) the use of a memory window that stores experience only from a certain trial onwards and forgets experience from before this trial by utilizing a rolling summation of the concentration parameters.

The results show that forgetting with the use of a memory window can significantly improve performance, provided that the agent can reach the goal state from the initial state in one trial. When this is not the case, the use of a memory window does not (positively or negatively) influence performance. Both the implementation of forgetting based on the updating of an LSTM cell and the use of a constant forget rate have unanimously shown to decrease performance, and thus should not be implemented in active inference.

Contents

List of Figures

List of Tables

1 Introduction

This introductory chapter will start by putting this thesis and the research topic of active inference in context. Hereafter, a formal description of the problem statement will be given, along with the main contributions of this research. Finally, at the end of this chapter the thesis structure is provided, which will give a preview of the content that will be discussed in the upcoming chapters.

1.1. Context

The search for intelligent agents that match the natural intelligence displayed by humans is an ongoing one in the field of artificial intelligence. In recent years, machines have become increasingly more competent in solving tasks that typically require human intelligence, up to the point that it is not uncommon for these artificial agents to surpass humans in certain tasks. However, the way in which these artificial agents learn and solve their tasks might differ drastically from the way humans do. Instead of trying to solve one specific task in the best way possible, one can take another approach in designing artificial intelligence: trying to understand how biological agents learn and embedding artificial agents with a similar form of intelligence. One attempt at doing so has led to the emergence of the free energy principle, and in extension active inference, which is the main theory discussed in this thesis.

Active inference is a theory originating from the field of cognitive neuroscience, and is a corollary of the free energy principle. The free energy principle states that in agents, both biological and artificial, perception, action selection and learning are unified in the sense that all are the result of optimizing the same objective function, termed the free energy. By minimizing the free energy, agents keep an accurate representation of the world in their internal model (learning), are provided with an optimal way of acting on the world (action selection), and can predict incoming sensory data (perception). Therefore, the brain, as well as other autonomous agents, can essentially be seen as inference engines that seek to minimize prediction error, and have the tools to do so by their choice of action selection and by the updating of their internal model of the world. Because all of these elements that comprise active inference can be evaluated mathematically, active inference can find applications beyond neuroscience, making it also an interesting topic from an engineering point of view.

The theory of active inference is still a relatively new one, with the free energy principle having been coined by neuroscientist Karl Friston a little more than a decade ago [9]. Since then, examples of problems to which active inference has been applied to include planning and navigation [18], reading [16], visual attention [22] and the mountain car problem [15]. However, not all of these problems are solved in the same way. The generative processes encountered in active inference can entail either continuous or discrete state-spaces (or even a mixture of both in more complicated tasks), and the continuous and discrete state-space variants of active inference differ in their computational frameworks as well. Whether to utilize the continuous or discrete state-space framework is dependent on the task, with the former typically being used in lower level control and the latter typically being used in higher level decision making. This thesis will focus on the discrete representation of states, because it will be an extension to the research on planning and navigation tasks as will be seen later on. For these tasks, the processes encountered are generally modeled as slight variations of Markov decision processes.

Markov decision processes are widely used in the solving of optimization problems and are known

to have been used at least since the late 1950s [1]. However, with active inference being such a novel theory, a multitude type of problems are still unexplored, even if the scope is just restricted to active inference on Markov decision processes. One type of problem that is neglected in this scope of problems, but will be essential for active inference agents to deal with, is active inference in nonstationary environments. While with the current technological advances machines prosper in stationary, controlled environments, it is the reality that not all real-world environments can be regulated to ensure stationarity [6]. A common example of an environment that does not have the property of stationarity is simply the outside world disturbed by meteorological phenomena. As such, one can comprehend that nonstationarity arises in everyday situations. Eventually, true intelligent agents should (up to a certain extent) be able to adapt to nonstationary environments. This thesis aims to make a start in the research of discrete state-space active inference in nonstationary environments.

1.2. Problem statement

The first section outlined the context of and motivation behind this thesis. This section will build on that by giving a formal description of the problem to be solved and defining the corresponding research questions. The research questions that will be addressed in this thesis, are formulated as follows:

1. *How do state-of-the-art active inference agents perform in discrete state-space nonstationary environments?*

2. *Can the performance of discrete state-space active inference agents in nonstationary environments be improved by forgetting part of their previous experiences?*

3. *What are the most effective methods for implementing forgetting in discrete state-space active inference?*

The second research question is defined, because it is expected that with discrete state-space active inference as it currently is, the agent is not able to attain satisfactory performance. This expectation is based on the difficulties that nonstationarity has brought in related fields of research [3, 6]. This expectation is confirmed later when going through the results in chapter 5. The third research question is a natural extension to the second research question.

Since nonstationary processes encompass a large variety of subtypes, constraints have to be set in what type of nonstationarity will be researched. Being one of the most common instances of nonstationarity and providing a bridge between stationary processes and other more complicated nonstationary processes [24], the choice is made to focus on cyclostationary processes in this research. To constrain the problem further, in what follows, the nonstationarity will be exclusively manifested in the transition process. Aside from the fact that nonstationarity in the transition process has not been addressed in discrete state-space active inference research literature, the literature often neglects the transition model in favor of working on the observation model. Getting results that focus on the transition process will therefore be valuable even outside the scope of nonstationarity.

For creating a working environment of the active inference agent in this research, the main task to be solved by the agent will be a MATLAB simulation in which the agent has to traverse a maze-like gridworld, trying to find the shortest path from start to goal in a transition-nonstationary environment. Since the task to be solved is one of planning and navigation, the number of steps the agent needs to take in order to reach its goal will be used to quantify the performance. Regarding the nonstationarity in the transition process, in this task it will be modeled as an external perturbating force, that can be interpreted as wind, for which the direction changes over time.

1.3. Contribution

The contribution of this thesis to the active inference research will be to modify the updating scheme of the agent's generative model to better cope with nonstationary state transitions. As is evident from the last two research questions, this will be done by providing the agent with a way of forgetting information gained by previous experiences. This is currently not possible in active inference. Since this modification allows the agent to forget experiences that do not conform with the present, it is expected that this can improve the agent's performance in nonstationary environments.

Implementing forgetting in the updating scheme of the agent can be done in several ways, and as such different methods will be tested. Three methods for implementing forgetting will be discussed in this research:

1. The approach arguably closest to the conventional discrete state-space active inference updating of the generative model is to implement a forget rate parameter to the updating of the agent's transition model, thereby simply expanding the updating scheme of the generative model. The use of forget rate parameters in general is not new, as these have been implemented in reinforcement learning before [19, 20]. The implementation in active inference however, is novel and as such, this is one of the approaches that will be taken.

2. Besides expanding the agent's updating scheme of the transition model with a forget rate parameter, other approaches for implementing forgetting can be taken that are inherently different from the conventional updating of the generative model in active inference. One approach from the field of deep learning that has proven to be capable of forgetting is the application of the long short-term memory (LSTM) [17]. Therefore, it will be investigated whether the way of updating an LSTM cell can be translated to the updating of the generative model in active inference to accommodate forgetting.

3. The final approach that will be taken for implementing forgetting is to apply a rolling summation for the parameters of the generative model instead of a total summation. Similar to a rolling average, the rolling summation takes into account only the most recent information stored in the agent's generative model, the amount determined by the width of the summation window. Rolling averages have proven to be effective for detecting patterns and transient events in time series analysis [23]. As such, this could be the case for handling nonstationarity in active inference as well.

The resulting behavior of these agents embedded with different ways of forgetting will be compared to each other and to the behavior of an agent incapable of forgetting, when solving the task in the previously described MATLAB setup. This allows for the first-time evaluation of performance for active inference agents using novel updating schemes for the generative model.

1.4. Thesis structure

Succeeding this introductory chapter is chapter 2, which will give an introduction to the theory of discrete state-space active inference. This chapter contains the fundamental concepts and definitions, among which the generative model and the free energy, and will also provide an overview of the numerous symbols and notations that will be used throughout this work.

Chapter 3 will introduce nonstationarity in stochastic processes and describe the manifestations of nonstationarity in active inference. It will also discuss possible approaches to handle nonstationarity in active inference and zoom in on the implementation of the chosen methods.

Chapter 4 will describe the problem setup that will be used for testing the performance of discrete state-space active inference agents in this research. A detailed explanation of the experimental methodology will be given, along with all the variables that will influence the performance of the agents. For this, the metrics and statistics that will be used for quantifying the performance of the agents need to be declared, which will be done at the end of this chapter.

Chapter 5 will provide the results of the experiments and their statistical significance. The end of this chapter will also provide a tuning analysis by repeating the experiments with different values for the selected tunable parameters.

The penultimate chapter 6 will go deeper into the performance of the proposed methods in this research, by explaining the causes of the obtained results. This chapter will also discuss possible additions to the proposed methods and even alternatives to these methods.

Finally, chapter 7 will give a recap of the findings in this research and end with the conclusions that answer the research questions defined in this chapter.

2

Active inference

This chapter will give an introduction to the theory of active inference, focussing on its discrete state-space variant. After section 2.1 has provided the motivation behind active inference, sections 2.2 and 2.3 will introduce the general concepts of the agent-environment interface and the free energy respectively. The following sections, 2.4 and 2.5, will explain the fundamental concepts of discrete state-space active inference in detail. It is important to become acquainted with these concepts, as these will be further addressed in the subsequent chapters.

2.1. The motivation behind active inference

Active inference is a neuroscientific theory based on the free energy principle, a statement introduced by neuroscientist Karl Friston, which proposes that everything in life minimizes a quantity termed the free energy [8]. The motivation behind this theory comes from the question how a biological system (e.g. the brain) can stay at equilibrium with its environment, even in a constantly changing environment. Equivalently, this question can be formulated as how these systems resist a natural tendency to disorder [7]. To answer this question, the free energy principle starts by stating that systems typically restrict themselves to a limited number of states. That is, for most of the part, these systems will find themselves in the same small number of states out of all the possible states they could occupy. Statistically speaking, this means that the probability is high that a system will be in any of the aforementioned small number of states, and low that it will be in the remaining states. This finding can be rephrased by saying that it would be suprising to find the system in any of the remaining states, and not so surprising to find the system in the smaller subset of states. A commonly used example of this would be a fish. For an ordinary fish, living its life in the water comes naturally: it is not surprising at all for the fish to find itself in the water. However, a fish will not want to find itself out of the water: this would be detrimental to its survival and can be regarded as a very surprising state. As can be seen from this simple example, avoiding surprising states seems beneficial to prolonging existence.

This is exactly what is done in active inference. The free energy that is minimized by active inference can actually be regarded as a measure of the 'surprisingness'. This minimization of free energy embeds biological systems with the behavior that characterizes their nature: action selection, perception and learning. By minimizing the free energy, they keep an accurate representation of the world (learning), are provided with an optimal way of acting on the world (action selection), and can predict incoming sensory data (perception). The use of the term 'minimization' here implicitly suggested that the free energy can be evaluated mathematically. Because of this property, active inference can find use in applications beyond neuroscience. After all, the features of active inference mentioned in the previous paragraphs do not solely hold for biological systems: they can be embedded in artificial systems as well. This makes active inference an interesting topic from an engineering point of view, and in particular makes it an attractive framework to pursue in the design of novel artificial intelligence systems and robots.

Any scientific theory trying to explain the working principles of the nervous system faces a dual challenge in quantifying behavior [28]. On the one hand, to act upon the world, it is necessary to make changes to continuous variables: an example is the change in length of the different muscles in the

arm when extending that arm. On the other hand, the preceding process, that of decision making, is intrinsically discrete: a choice is being made between a finite set of different possibilities. This duality is also present in the theory of active inference. Active inference can be applied to problems regardless of whether the state-space representation is continuous or discrete. However, as the continuous and discrete state-space variants of active inference differ in their computational frameworks, the nature of the problem will determine the framework it will be used in. Generally, the continuous state-space framework is used in lower-level tasks, e.g. when controlling forces or velocities, whereas the discrete state-space framework is used in higher-level tasks, e.g. in abstract problem solving [28]. As mentioned in the introduction, this thesis will focus on the discrete state-space representation of active inference. It will be seen in the remainder of this chapter that the processes encountered in this framework can be modeled as slight variations of Markov decision processes.

2.2. The agent and the environment

The start of this chapter questioned how some systems can stay in equilibrium with their environments. Such a system that is capable of interacting with its environment is referred to as an agent in artificial intelligence terminology [29].

In order to interact with its environment, an agent has to be able to perceive the environment, and also can be capable of learning from the interaction with the environment. This environment comprises everything outside the agent, and is represented by the set of possible states it can take on. Crucially, in an active inference task, the agent is not able to determine the state of the environment directly. Because of this, they are denoted as hidden states in the literature [13]. Nevertheless, the agent is able to observe the outcomes in a task through its sensors. A simple example of this would be an agent whose only sensor is a moisture sensor: through its measurements of the water content in the soil, the agent can determine whether it has rained or not, even though the agent is incapable of observing rainfall directly. Thus, the agent has to infer the states from the outcomes.

Additionally, the agent can interact with the environment through actions, whereby it transitions from one state to another. This closes the loop between agent and environment, as shown in figure 2.1. The result is an almost identical representation of the agent-environment interface used in standard Markov decision processes and reinforcement learning [32], the only difference being the absent notion of explicit rewards.

2.3. Free energy: a bound on surprisal

Active inference is a corollary of the free-energy principle: the behavior that characterizes active inference, that is perception, action selection and learning are all the result of minimizing the free energy. But what is free energy exactly? With just the concepts defined so far in this thesis, it is difficult to give an elaborate definition of free energy. What can already be defined however, is that free energy is a quantity from the field of information theory (it is different from the energy encountered in physics) that provides an upper bound on the surprisal. The surprisal of a state is a quantification of how surprising a certain state is. Remembering from section 2.1, the avoiding of surprising states by agents is beneficial to prolonging their existence. However, as mentioned in section 2.2, agents are not able to determine the state of the environment directly, let alone to avoid the surprising ones in this way. Although incapable of minimizing the surprisal of states, agent are able to approximate this minimization by minimizing the sensory surprisal, that is the suprisal of outcomes. Mathematically, the surprisal of a certain outcome o is given by the negative log-probability of that outcome, i.e. $-\ln p(o)$. Since probabilities exist in the range between 0 and 1, the surprisal of an outcome is a value between 0 (when the probability of that outcome is 1) and ∞ (which is approached when the probability of that outcome goes to 0).

Thus, to calculate the surprisal of a certain outcome o, one first needs to determine the probability of that outcome. However, this task is not as trivial as it seems. Using the law of total probability, it can be seen that determining the probability of a certain outcome o would entail summation over all possible hidden states $s \in \mathcal{S}$:

$$p(o) = \sum_{s \in \mathcal{S}} p(o|s)p(s) \tag{2.1}$$

The problem with this computation is that it becomes intractable as the state-space gets large, since the

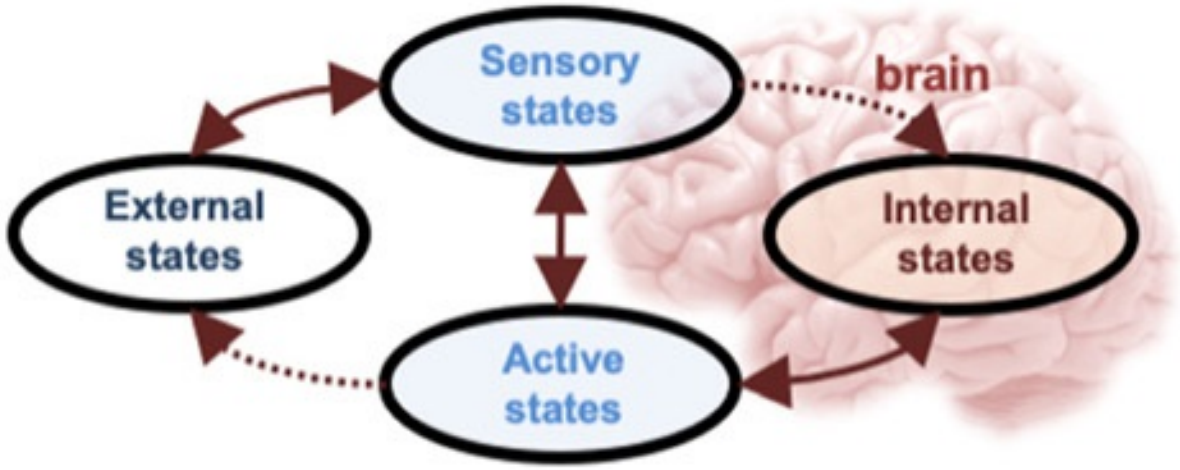


Figure 2.1: A schematic depiction of the agent-environment interface. The external states represent the states of the environment as they are in reality. The agent however, is not able to determine the external states directly, which is why these are often referred to as hidden states. What the agent can do, is observe the external states through sensory states. Therefore, these sensory states are commonly referred to as observations or outcomes. An agent can infer the external states, with the use of its sensory states, the result which is given by the internal states. These internal states represent the agent's beliefs about the environment. To close the loop, the agent can act on the environment, with the use of active states, often simply referred to as actions or control variables. (This figure is adapted from [4].)

number of calculations grows exponentially with the number of states [2]. Therefore, another approach has to be taken to minimize the surprisal.

A good starting point when trying to minimize the surprisal of states, is to have a way of determining the states first. Since hidden states can be inferred from outcomes, Bayes' rule can be utilized to calculate the posterior probability of the hidden states $p(s|o)$, given the prior probability of the hidden states $p(s)$ and the likelihood $p(o|s)$:

$$p(s|o) = \frac{p(o|s)p(s)}{p(o)} \tag{2.2}$$

Unfortunately, as can be seen from equation 2.2, this also requires knowledge about the marginal likelihood $p(o)$, which was the reason to try another approach in the first place.

Since it seems to be impossible to solve this type of problem with traditional Bayesian inference, variational Bayesian inference provides an alternative. Variational Bayesian inference uses an approximation of the posterior probability over states, instead of the exact calculation in equation 2.2, thereby avoiding the need to marginalize out the probability of all states s in set $\mathcal{S}$. This approximate posterior is a distribution parameterized by the parameters of the agent's generative model. By closely approximating the real posterior distribution with this approximate distribution, the agent obtains the best estimation of the sensory surprisal. This will be mathematically proven in section 2.5. But first, section 2.4 will provide the mathematical concepts that underlie this proof.

2.4. The generative process, generative model and approximate posterior

With the concepts introduced thus far, a formal description of the discrete state-space representation of active inference can be given by the following list. There exist:

- Hidden states s forming the set of all possible hidden states $\mathcal{S}$, that is $s \in \mathcal{S}$. The states are called hidden, because the agent cannot observe them directly.
- Outcomes o forming the set of all possible outcomes $\mathcal{O}$, that is $o \in \mathcal{O}$. Outcomes are also referred to as observations, because these are what the agent observes directly.
- Actions u forming the set of all possible actions $\mathcal{U}$, that is $u \in \mathcal{U}$. Actions are also referred to as control variables.
- A generative process, that is the real world dynamics as they are specified by the environment. It is denoted by $R(\tilde{o}, \tilde{s}, \tilde{u})$.
- A generative model, that is the agent's beliefs about the real world dynamics. It is denoted by $P(\tilde{o}, \tilde{s}, \pi, \eta)$.
- An approximate posterior, which is denoted by $Q(\tilde{s}, \pi, \eta)$.

The first three items in this list have already been introduced in the previous sections in this chapter. The last three items in this list are the topic of this section and will be explained here in more detail. For an overview of the numerous symbols and notations used throughout this chapter, refer to table 2.1.

2.4.1. The generative process

The generative process $R(\tilde{o}, \tilde{s}, \tilde{u})$ describes the world by generating outcomes from hidden states and actions, and state transitions in time. To exemplify this, if the agent takes a certain action in a certain state at the current timestep, the generative process will determine what the state will be at the next timestep and what will be observable for the agent. Since the generative process samples states given the previous state and current action, and outcomes given the current state, it is also known as the sampling probability in earlier literature [10]. Important to note here is that the agent does not have access to the information contained within the generative process. This has important consequences depending on the nature of the active inference task. In real-world tasks, this means that the generative process is not explicitly known at all. In simulated tasks, this means that the designer has to specify the generative process to simulate the environmental dynamics, but must ensure that the agent is precluded from this information.

Table 2.1: List of frequently used symbols and notations used in discrete state-space active inference.

(a) List of frequently used symbols and their meaning.

Symbol	Meaning
τ	The general notation of a timestep, usually used for indexing other variables.
T	The time horizon, which specifies the total number of timesteps in a trial.
s_τ s_τ^π	A vector containing the probabilities of hidden states at timestep τ. A vector containing the probabilities of hidden states at timestep τ, according to policy π.
o_τ o_τ^π	A vector containing the probabilities of outcomes or observations at timestep τ. A vector containing the probabilities of outcomes or observations at timestep τ, according to policy π.
u	The general notation of an action or control variable.
π	The general notation of a policy, which specifies a sequence of actions.
A a	The likelihood or observation matrix, which specifies the probabilities of outcomes given hidden states. The concentration parameters of the likelihood matrix.
B_τ^π b	The transition matrix, which specifies the probabilities of the next hidden state given the current hidden state and policy. The concentration parameters of the transition matrix.
D	The initial state vector, which specifies the probabilities over initial states.
U	The utility vector, which specifies how the agent prefers possible outcomes over one another.
C	The prior preference vector, which is the utility vector normalized into a probability distribution.
γ β	The precision, which refers to the agent's randomness in policy selection. The inverse precision, which is used for easier updating of the precision.
η	The parameters of the generative model, composed of the set of concentration parameters and inverse precision, i.e. $\eta = \{a, b, d, \beta\}$.
x	The hidden causes, composed of the set of hidden states, policies and model parameters, i.e. $x = \{\tilde{s}, \pi, \eta\}$.
F	The variational free energy. The agent infers the hidden causes by minimizing this quantity with respect to those hidden causes.
G	The expected free energy. The agent evaluates policies by minimizing this quantity with respect to those policies.

(b) List of frequently used notations and their meaning.

Notation	Meaning
$subscript$	The subscript notation is commonly used for the time indexing of variables, e.g. s_τ.
$superscript$	The superscript notation is commonly used for indicating that a distribution is specific for a policy, e.g. s_τ^π.
Bold	When a distribution is depicted in boldface font, it concerns the posterior expectation of that distribution, e.g. $\mathbf{s}_\tau^\pi$.
$\hat{hat}$	The hat notation indicates the natural logarithm of the corresponding distribution, e.g. $\hat{\mathbf{s}}_\tau^\pi = \ln \mathbf{s}_\tau^\pi$.
$\tilde{tilde}$	The tilde notation indicates a sequence, e.g. $\tilde{s} = (s_1, ..., s_T)$.

2.4.2. The generative model

Although the agent does not have access to the information in the generative process, it does have its own generative model of the world. The generative model $P(\tilde{o}, \tilde{s}, \pi, \eta)$ is a model that describes the dynamics of the world according to the agent's beliefs. In general, these beliefs are different from the actual dynamics generated by the generative process (except in the unlikely case where the agent has a perfect model of the world). The generative model is used to infer hidden causes from outcomes, and is simply the joint probability of outcomes and their hidden causes. Causes can be defined as the set of states, policies and model parameters. The term causes is used, because hidden states are not the only quantities that determine observations: the different policies the agent can follow also influence what the agent will observe (e.g. the choice of where to look plays a role in what will be seen), as do the parameters of the model (e.g. past experiences play a role in how something is perceived). This means that — besides inferring the hidden states from the observations — the agent also has to select policies and learn the model parameters as will be seen later. As the generative model is a joint probability of outcomes and their causes, it can be parameterized in the following way:

$$\begin{aligned} P(\tilde{o}, \tilde{s}, \pi, \eta) &= P(\tilde{o}|\tilde{s})P(\tilde{s}|\pi)P(\pi|\gamma)P(\gamma)P(A)P(B)P(D) \\ &= \prod_{\tau=1}^{T} P(o_\tau|s_\tau) \prod_{\tau=2}^{T} P(s_\tau|s_{\tau-1}, \pi)P(s_1)P(\pi|\gamma)P(\gamma)P(A)P(B)P(D) \end{aligned} \tag{2.3}$$

The meaning of each of the terms in this equation will be clarified in the remaining paragraphs of this subsection.

In most of the terms in equation 2.3, a probabilistic mapping is made from one variable to another. For example, $P(o|s)$ denotes the probability of the observation being o when the state is s. Naturally, the observations and states (along with the other variables) can take on different values. Therefore, the values of all combinations in the probabilistic mapping are typically stored in matrices:

$$P(o_\tau|s_\tau) = Cat(A) \tag{2.4}$$

$$P(s_{\tau+1}|s_\tau, \pi) = Cat(B(u = \pi(\tau))) \tag{2.5}$$

These matrices, specified by Cat, are categorical distributions. The categorical distribution is nothing more than a discrete probability distribution that describes the probability of all the possible categories that the random variable can result in. When dealing with two random variables (as is the case here), each element in a column of the matrix specifies the probability that the dependent variable takes on that value, whereas every different column represents the categories the independent variables can take on. Or mathematically stated, $P(o|s) = Cat(A)$ implies $P(o = i|s = j) = Cat(A_{ij})$. A graphical example is given in figure 2.2.

The matrix described in equation 2.4, A, is known as the likelihood matrix or observation model. Each element in A specifies the likelihood of a certain observation given a certain state. The columns in the A-matrix correspond to all possible hidden states, whereas the rows correspond to all possible observations. Since the A-matrix takes on a categorical distribution (it is a probabilistic mapping from hidden states to outcomes), the elements in each column sum to one.

The matrix described in equation 2.5, $B(\pi(\tau))$, is known as the transition matrix or model, for which each element in the matrix gives the probability of transitioning to a next state given a current state and policy. Note that there is a separate B-matrix for every possible control variable, since the state transition probabilities are conditional on the chosen policy. In the B-matrix for a specific control variable, the columns correspond to all possible current hidden states, whereas the rows correspond to all possible next hidden states. Similar to the A-matrix, the elements in the columns of the B-matrices sum to one.

A special case of the transition model exists when 'transitioning' to the initial state:

$$P(s_1|s_0, \pi) = P(s_1) = Cat(D) \tag{2.6}$$

Since the initial state is not actually dependent on a previous state or policy, it is not a conditional probability. As such, the categorical distribution in equation 2.6 takes not the form of a two-dimensional matrix, but rather a one-dimensional vector. It is therefore known as the initial state vector or D-vector. Each element in the D-vector specifies the agent's beliefs about the probability of that state being the initial state.

Returning to equation 2.3, the third term $P(\pi|\gamma)$ specifies the agent's probability of following policy π, dependent on the agent's precision γ. The precision refers to the degree of randomness in the agent's behavior. Simply stated, if an agent believes a certain policy is best (how the agent actually determines which policy is best will be discussed in 2.5) there remains a chance that the agent will not follow that policy if the precision is low.

The agent has prior beliefs over the precision γ itself. The prior on the precision takes the form of a gamma distribution $P(\gamma) = Gamma(1, \beta)$. Even though the gamma distribution is a two-parameter continuous probability distribution, when the first parameter is equal to one (like in this case), the distribution is essentially reduced to an exponential distribution [5]. Therefore, the precision is effectively parameterized only by the variable β, which is called the temperature or inverse precision [13]. This comes from the fact that the expected value of the precision γ equals $1/\beta$ (since the expected value of an exponential distribution $f(x; \lambda)$ is equal to $1/\lambda$).

The final three terms in equation 2.3 are the priors over the observation matrix, transition matrix and initial state vector respectively:

$$P(A) = Dir(a) \tag{2.7}$$

$$P(B) = Dir(b) \tag{2.8}$$

$$P(D) = Dir(d) \tag{2.9}$$

These priors are specified by a Dirichlet distribution of so-called concentration parameters a, b and d. In a Dirichlet distribution, the input concentration parameters take the form of an array, whose size is equal to the array of the conjugate distribution (which in this case is the categorical distribution). For example, concentration parameter matrix a is a matrix equal in size to observation matrix A. In fact, since matrix A represents a probability distribution, it is the column-normalized version of matrix a (the same relation holds between matrices B and b, and vectors D and d). These concentration parameters describe the confidence of the probabilities specified by their normalized counterparts. Again taking the observation matrix as an example, when a specific outcome is observed more often in a specific state, the agent becomes increasingly certain that that particular state will give that particular outcome as observation, and the value for the corresponding element in the concentration parameter matrix increases. Intuitively, the concentration parameters can be seen as counts that keep track of how often state-outcome pairs (for a) or state transition pairs (for b) have occurred, or how often states have been encountered as the initial state (for d).

Readers with a keen eye will have noticed that while the arrays A, B and D have all been mentioned, a possible C array is missing. A vector C actually exists in discrete state-space active inference, although it does not appear in equation 2.3:

$$P(o_\tau) = Cat(C) \tag{2.10}$$

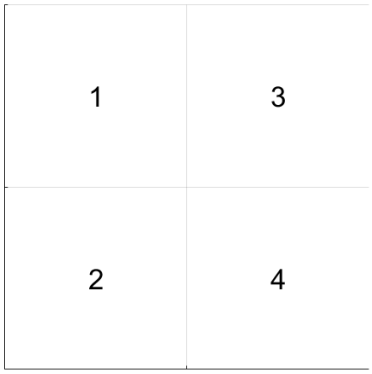
The reason for this omission, is that the elements in the C-vector do not represent actual probabilities. Rather, they represent the prior probabilities over outcomes as specified (indirectly) by the designer. As such, the amount of possible values the elements in the C-vector can take on are endless in theory. In practice however, the priors over outcomes are usually set to reflect the agent's preferences over outcomes [14]. Since the elements in the C-vector must sum to one, it is obtained by applying the softmax function on the so-called utility vector U:

$$P(o_\tau) = \sigma(U(o_\tau)) \tag{2.11}$$

It is this utility vector that is directly specified by the designer, which tells how the agent rates its preferences over outcomes. That is, higher values in the utility vector simply correspond to outcomes that are more preferred by the agent. Prior preferences over outcomes are used in the evaluation of policies the agent can select, and will be further encountered in section 2.5.

2.4.3. The approximate posterior

Section 2.3 concluded by stating the need for approximating the real posterior distribution with an approximate posterior distribution, as this way, the agent obtains the best estimation of the sensory surprisal. In variational Bayesian inference, one first has to specify the form of the approximate posterior distribution. In active inference this can be done with the mean field approximation. By factorizing the approximate posterior into a product of smaller terms, the dependency structure can be simplified. Even



$$A = p(o_\tau|s_\tau) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B(west) = p(s_{\tau+1}|s_\tau) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B(east) = p(s_{\tau+1}|s_\tau) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$B(north) = p(s_{\tau+1}|s_\tau) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B(south) = p(s_{\tau+1}|s_\tau) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$C = p(o_\tau) = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.7 \\ 0.1 \end{bmatrix}$$

$$D = p(s_1) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Figure 2.2: Example of the arrays used in the generative model. There exists a 2×2 gridworld, in which each cell represents one specific state, for which the state number is shown in the corresponding cell. These state numbers are used for indexing the arrays of the generative model. In this example, the agent is able to directly observe the states, because there is a one-to-one mapping from states to observations, as is seen from the A-matrix. To transition between states, the agent has four control variables at its disposal: $west$, $east$, $north$ and $south$. In the B-matrices, it can be seen how taking one action in a specific state leads to what state transition. For example, in the transition matrix of the west control variable, the value of 1 in row 2 column 4, means that taking the action to the west in state 4 leads to a transition to state 2. The C-vector tells how the agent prefers its outcomes: in this example, the agent prefers to observe (in these kind of tasks, to be in) state 3 over the other states. Finally, the D-vector specifies the probabilities of the agent's initial state: in this example, the agent beliefs that it will certainly start in state 2.

though the structure of the model becomes simpler with this factorization, it has a sound mathematical foundation and has been used with success in many previous applications of variational methods [34]. The result is the following equation for the approximate posterior:

$$\begin{aligned} Q(\tilde{s}, \pi, \eta) &= Q(\tilde{s}|\pi)Q(\pi)Q(\gamma)Q(A)Q(B)Q(D) \\ &= \prod_{\tau=1}^{T} Q(s_\tau|\pi)Q(\pi)Q(\gamma)Q(A)Q(B)Q(D) \end{aligned} \tag{2.12}$$

A description of the meaning of the terms that appear in equation 2.12 has been provided in the previous subsection. The difference now however, is that the probabilities in equation 2.12 concern posterior probabilities, which was not the case for equation 2.3. That is, equation 2.12 relates to the probabilities after the agent's observations have been taken into account. How each of the factors in the approximate posterior is calculated and updated, will be explained in section 2.6. Although, that requires understanding of the mathematical evaluation of the free energy first.

2.5. Variational and expected free energy

The minimization of free energy lies at the heart of the theory of active inference. In the discrete state-space active inference literature, actually two forms of free energy are mentioned, namely the variational free energy and the expected free energy. Both forms of the free energy will be discussed in this section.

2.5.1. Variational free energy

The variational free energy is used by the agent for making optimal inferences about the hidden states, policies and model parameters. The starting point for the equation of the variational free energy is usually the expression of the expected log difference between the approximate posterior (what the agent beliefs that has happened) and the joint distribution of the generative model (what the agent expected to happen), after which the variational free energy can be expressed further in different ways:

$$\begin{aligned} F &= \mathrm{E}_Q[\ln Q(s, \pi, A, B, D) - \ln P(s, \pi, A, B, D, \tilde{o})] \\ &= \mathrm{E}_Q[\ln Q(s, \pi, A, B, D) - \ln P(s, \pi, A, B, D|\tilde{o}) - \ln P(\tilde{o})] \\ &= D_{\mathrm{KL}}[Q(s, \pi, A, B, D)|P(s, \pi, A, B, D|\tilde{o})] - \ln P(\tilde{o}) \end{aligned} \tag{2.13}$$

Here, D_{KL} is the Kullback-Leibler (KL) divergence. The KL divergence is used for measuring how one probability distribution is quantitatively different from another one. For two discrete probability distributions $Q(x)$ and $P(x)$, that is:

$$D_{\mathrm{KL}}[Q(x)|P(x)] = \sum_{x \in X} Q(x) \ln\left(\frac{Q(x)}{P(x)}\right) \tag{2.14}$$

A useful property of the KL divergence is that its output is always a nonnegative value [21]. Taking this property into account, it can be seen from the last equality in equation 2.13 that the variational free energy is minimized when the KL divergence is zero. As can be seen from equation 2.14, this is the case when Q is equal to P. Moving back to 2.13, it can then be seen that in the case of Q being equal to P, the free energy becomes equal to the sensory surprisal $-\ln P(\tilde{o})$. This also means that when Q and P are not equal, the free energy will always be larger than the surprisal, as the KL divergence will then be some positive number. As such, the free energy will always be larger than or equal to the surprisal. This proves why the free energy is an upper bound on the surprisal, as was concluded in section 2.3.

2.5.2. Expected free energy

With the theory described thus far, it is known that the agent wants to minimize the variational free energy as this allows the agent to make optimal inferences. However, this is just one side of active inference. As mentioned at the start of this chapter, free energy minimization also provides the agent with an optimal way of acting on the world. However, the minimization of equation 2.13 is of little use for taking future actions (policy selection), since the variational free energy is to be calculated only after

the agent has observed the outcomes. To define the value of policies, the agent has to consider the free energy over (potential) future outcomes and hidden states under a given policy. This free energy expected under a policy is termed the expected free energy. The equation for the expected free energy of a policy π is given by:

$$G(\pi) = \sum_{\tau=1}^{T} G(\pi, \tau) \tag{2.15}$$

$$\begin{aligned} G(\pi,\tau) &= \mathrm{E}_{\tilde{Q}}[\ln Q(s_\tau|\pi) - \ln Q(s_\tau|o_\tau,\pi) - \ln P(o_\tau)] \\ &= \mathrm{E}_{\tilde{Q}}[\ln Q(s_\tau|\pi) - \ln Q(s_\tau|o_\tau,\pi)] - \mathrm{E}_{\tilde{Q}}[\ln P(o_\tau)] \\ &= \mathrm{E}_{\tilde{Q}}[\ln Q(o_\tau|\pi) - \ln Q(o_\tau|s_\tau,\pi)] - \mathrm{E}_{\tilde{Q}}[\ln P(o_\tau)] \\ &= D_{\mathrm{KL}}[Q(o_\tau|\pi)|P(o_\tau)] + \mathrm{E}_{\tilde{Q}}[H[P(o_\tau|s_\tau)]] \end{aligned} \tag{2.16}$$

where $\tilde{Q} = Q(o_\tau, s_\tau|\pi)$ is a posterior predictive distribution. This means that unlike the expectations in the variational free energy equation, where the outcomes were actually observed (i.e. according to the generative process), the expectations here consider outcomes that the agent would observe according to its generative model. The final expression in equation 2.16 can be rewritten in terms of the agent's generative model, which simplifies the calculation:

$$\begin{aligned} G(\pi,\tau) &= D_{\mathrm{KL}}[Q(o_\tau|\pi)|P(o_\tau)] + \mathrm{E}_{\tilde{Q}}[H[P(o_\tau|s_\tau)]] \\ &= \mathbf{o}_\tau^\pi \cdot (\hat{\mathbf{o}}_\tau^\pi - U) + \mathbf{s}_\tau^\pi \cdot H \end{aligned} \tag{2.17}$$

In the equation above, U is the utility vector and H is a vector encoding the entropy or ambiguity over outcomes:

$$H = -diag(A \cdot \hat{A}) \tag{2.18}$$

Also seen from equation 2.17 is the subscript-superscript notation for states and outcomes (e.g. $\mathbf{o}_\tau^\pi$ denotes the probability over outcomes at time τ when policy π is followed), which is commonly used in literature. In this notation, the boldface symbols indicate that the distributions in question are posterior distributions (or in this case posterior predictive distributions). Additionally, the hat notation in equations above is a shorthand notation for the natural logarithm of the corresponding distribution, e.g. $\hat{\mathbf{o}}_\tau^\pi = \ln \mathbf{o}_\tau^\pi$. These notations will be used throughout the rest of this report.

The agent performs the calculation in equation 2.15 for every possible policy. When this is done, the probability of pursuing a certain policy is given by the following equation:

$$P(\pi) = \pi_0 = \sigma(-\gamma \cdot G(\pi)) \tag{2.19}$$

This means that the probability of pursuing a policy is proportional to the negative expected free energy of that policy. This shows how the minimization of the free energy of beliefs about the future shapes the agent's behavior.

2.6. Belief updating

The previous section ended with evaluating the expected free energy of each policy, which allows the agent to select what policy to follow. The policy specifies an action at each timestep, which will result in a state transition and the generation of a new outcome per timestep as well. Just like the agent predicted what would happen by evaluating the expected free energy before taking action, the agent will infer what actually has happened after observing the outcomes, and learn from its inferences. The agent does this by minimizing the variational free energy (equation 2.13) with respect to hidden states, precision of action selection, policies and model parameters.

2.6.1. States, policies and precision (inference)

The agent can infer the most likely states per policy, based on the observed outcomes in the trial and the parameters of the generative model. The posterior beliefs over the hidden states are given by the following equation:

$$Q(s_\tau|\pi) = \mathbf{s}_\tau^\pi = \sigma(\hat{A} \cdot o_\tau + \hat{B}_{\tau-1}^\pi \mathbf{s}_{\tau-1}^\pi + \hat{B}_\tau^\pi \cdot \mathbf{s}_{\tau+1}^\pi) \tag{2.20}$$

As seen from this equation, the posterior beliefs about a state at one timestep depends on beliefs about both the state at the previous timestep and the state at the next timestep. As such, this requires for

each timestep equation 2.20 to be iterated until convergence. However, this iteration until convergence is typically not the approach taken in active inference, as it does not seem to be in line with biological plausibility in cognitive neuroscience. Therefore, active inference takes a biologically more plausible approach by doing a gradient descent on the free energy [13]:

$$\dot{\hat{\mathbf{s}}}_\tau^\pi = \partial_{\hat{\mathbf{s}}} \mathbf{s}_\tau^\pi \cdot \varepsilon_\tau^\pi \tag{2.21}$$

$$\mathbf{s}_\tau^\pi = \sigma(\hat{\mathbf{s}}_\tau^\pi) \tag{2.22}$$

$$\varepsilon_\tau^\pi = (\hat{A} \cdot o_\tau + \hat{B}_{\tau-1}^\pi \mathbf{s}_{\tau-1}^\pi + \hat{B}_\tau^\pi \cdot \mathbf{s}_{\tau+1}^\pi) - \hat{\mathbf{s}}_\tau^\pi \tag{2.23}$$

The solving of these equation results in the posterior expectations of the hidden states that minimize the total variational free energy.

At the end of the trial, the agent also infers the policies. The inference of policies at the end of a trial is very similar to the selection of policies at the start of a trial (equation 2.19), with the difference being that the agent now also takes into account the variational free energy:

$$Q(\pi) = \boldsymbol{\pi} = \sigma(-F - \gamma \cdot G(\pi)) \tag{2.24}$$

Important to emphasize is that equation 2.24 is not used to select policies, since the policy for the trial has already been executed at this point. Rather, it is used to infer the policy that has been executed.

In addition to the states and policies, the agent infers the precision of policy selection γ, or more precise, its inverse β (see section 2.4):

$$\boldsymbol{\beta} = \beta + (\boldsymbol{\pi} - \pi_0) \cdot G(\pi) \tag{2.25}$$

Similar to the inference of states, the updated value of the inverse precision can be obtained by doing a gradient descent on the free energy:

$$\dot{\boldsymbol{\beta}} = \boldsymbol{\gamma}^2 \varepsilon^\gamma \tag{2.26}$$

$$\varepsilon^\gamma = (\beta - \boldsymbol{\beta}) + (\boldsymbol{\pi} - \pi_0) \cdot G(\pi) \tag{2.27}$$

Finally, with the expected states per policy $\mathbf{s}_\tau^\pi$ and the posterior over policies $\boldsymbol{\pi}$ now computed, the agent can obtain a posterior distribution over states, independent of policies, by creating a Bayesian model average of the states:

$$\mathbf{s}_\tau = \sum_{\pi \in \Pi} \boldsymbol{\pi}_\pi \cdot \mathbf{s}_\tau^\pi \tag{2.28}$$

This distribution is used by the agent in the updating of the parameters of its generative model.

2.6.2. Model parameters (learning)

After having inferred the hidden causes at every timestep in a trial, the agent is able to update the parameters of its generative model at the end of that trial. The agent does this by the accumulation of concentration parameters:

$$\mathbf{a} = a + \sum_{\tau=1}^{T} o_\tau \otimes \mathbf{s}_\tau \tag{2.29}$$

$$\mathbf{b}(u) = b(u) + \sum_{\pi \in \Pi} \sum_{\tau=1}^{T-1} \delta_{\pi(\tau),u} \boldsymbol{\pi}_\pi \mathbf{s}_{\tau+1}^\pi \otimes \mathbf{s}_\tau^\pi \tag{2.30}$$

where δ is the Kronecker delta. Similar to how a boldface symbol for the state distribution represents a posterior distribution, a boldface symbol for the concentration parameters represents those concentration parameters after accumulation at the end of the trial.

For simplifying the cross-referencing later on, the second term in equation 2.30 will be given its own variable:

$$\mathscr{b}(u) = \sum_{\pi \in \Pi} \sum_{\tau=1}^{T-1} \delta_{\pi(\tau),u} \boldsymbol{\pi}_\pi \mathbf{s}_{\tau+1}^\pi \otimes \mathbf{s}_\tau^\pi \tag{2.31}$$

This variable $\mathscr{b}(u)$ represents the concentration parameters accumulated in one trial, and can be interpreted as the experience the agent has gained from that trial.

After the accumulation of concentration parameters, the generative model is updated as follows:

$$\hat{\mathbf{A}} = \psi(\mathbf{a}) - \psi(\mathbf{a}_0) \tag{2.32}$$

$$\hat{\mathbf{B}} = \psi(\mathbf{b}) - \psi(\mathbf{b}_0) \tag{2.33}$$

where ψ is the digamma function. In the above equations, $\mathbf{a}_0$ and $\mathbf{b}_0$ are matrices in which each element in the same column is equal to the sum of elements of the corresponding column in the $\mathbf{a}$ and $\mathbf{b}$ matrix respectively. That is:

$$\mathbf{a}_{0_{ij}} = \sum_i \mathbf{a}_{ij} \tag{2.34}$$

$$\mathbf{b}_{0_{ij}} = \sum_i \mathbf{b}_{ij} \tag{2.35}$$

In the end, what equations 2.32 and 2.33 boil down to, is that the resulting matrices $\mathbf{A}$ and $\mathbf{B}$ become column-normalized versions of their concentration parameter matrix counterparts $\mathbf{a}$ and $\mathbf{b}$.

3

Nonstationarity in active inference

The previous chapter gave a condensed overview of the fundamental concepts and working principles in discrete state-space active inference. As mentioned in the introduction, these principles have shown to be capable of solving a wide range of problems, but these implementations had in common that the problems to be solved were stationary in nature. Therefore, this chapter will give an introduction to nonstationarity in stochastic processes, zoom in on the type of nonstationarity applied in this research (cyclostationarity) and describe the manifestations of nonstationarity in active inference. The remainder of this chapter will concern possible methods to cope with nonstationarity in active inference, from which the focus will be set on forgetting. Consequently, this chapter will conclude with the elaboration of three different ways for implementing forgetting in discrete state-space active inference.

3.1. Stationary and nonstationary processes

Stochastic processes can be classified based on many different basic properties. One such property that can be used in the classification of stochastic processes is the distinction between stationarity and nonstationarity. The literature on stochastic processes makes a further division in stationary processes between strict sense stationary processes and wide sense stationary processes [26]. Formally, a strict sense stationary process is a stochastic process for which all statistical properties are identical at any point in time. Mathematically, a stochastic process X is a strict sense stationary process if:

$$P(X(t)) = P(X(t + \Delta t) \quad \forall t, \Delta t \tag{3.1}$$

For a wide sense stationary process, this identity over time is restricted to just the mean and autocovariance of that process:

$$\mathrm{E}[X(t_1)] = \mathrm{E}[X(t_2)] \quad \forall t_1, t_2 \tag{3.2}$$

$$C_{XX}[t_1, t_2] = C_{XX}[t_1 - t_2, 0] \quad \forall t_1, t_2 \tag{3.3}$$

In general, it is valid to identify stationarity processes in discrete state-space active inference as strict sense stationary, since the probability distribution of the generative process is usually explicitly defined.

The property of stationarity is very desirable when dealing with highly automated tasks. After all, with just the knowledge about the process obtained from the past, predictions can be made about the process in the future. Unfortunately, real-world processes are often not stationary. Any process for which equation 3.1 does not hold, is said to be nonstationary. Nonstationarity poses an extra challenge when dealing with stochastic processes, since the knowledge obtained in the past might be insufficient in predicting the process in the future. In the next section, it will be seen what kind of knowledge this concerns when dealing with nonstationary processes in an active inference task. But first, a brief overview will be given of the specific type of nonstationarity that will be focussed on in this research.

3.1.1. Cyclostationary processes

When dealing with nonstationary processes, it is helpful to identify the nature of the nonstationarity to determine the constraints of the process, if possible. An exhaustive list on types of nonstationarity

will not be provided here, since the scope of this research will be restricted to cyclostationarity. The choice for focussing on cyclostationarity is substantiated by the fact that it provides a bridge between stationary processes that have already been studied in discrete state-space active inference in the past, and other more complicated nonstationary processes that can be a topic for further research in the future. In fact, a discrete-time cyclostationary process can be expressed in terms of a finite number of nonoverlapping stationary processes [24].

Cyclostationary processes are stochastic processes that have statistical properties that change over time, but repeat after a certain period. Just as with stationary processes, a division can be made between strict sense cyclostationary processes and wide sense cyclostationary processes. Mathematically, a stochastic process X is said to be strict sense cyclostationary if there exists a period T for which:

$$P(X(t)) = P(X(t+T) \quad \forall t \tag{3.4}$$

Cyclostationarity is commonly encountered in real-world processes. An example of a cyclostationary process is the average measured temperature per month: in a specfic year large differences can exist in the average temperature per month between different months, but the average temperatures of the months in one year will be approximately equal to the average temperatures of the same months in the next the year.

3.2. Manifestations of nonstationarity in active inference

Nonstationarity can manifest itself in multiple ways in an active inference task. As mentioned in section 2.4, the generative process in active inference describes the world by generating outcomes from hidden states and actions, and state transitions in time. This implies two distinct probabilistic mappings as mentioned in that same section: a mapping from current states to observations (i.e. the likelihood matrix) and a mapping from current states to next states (i.e. the transition matrix). As such, these two probabilistic mappings can be exhibitions of nonstationarity. Additionally, nonstationarity can even be manifested in the state-space itself, but that is outside the scope of this research. To constrain the problem of nonstationarity in active inference, the scope will be entirely restricted to the transition process, as this will be the point of focus of the experiments in chapter 4.

3.2.1. Nonstationarity in the transition process

The transition process specifies what the next state of the world will be, given the current state of the world and the most recent action taken by the agent. Nonstationarity in the transition process implies that the probability distribution of the possible next states given a certain current state and a certain action taken at one point in time, will be different from the probability distribution of the possible next states given the same current state and same action taken at a different point in time. This means that the transition process B is stationary only if:

$$B_t = B_{t+\Delta t} \quad \forall t, \Delta t \tag{3.5}$$

Important to note is that the B-matrix in equation 3.5 concerns the transition matrix of the generative process, which is different than the transition matrix of the generative model. As mentioned in chapter 2, the latter matrix gets updated at the end of every trial, so it does not make sense to speak about a stationary or nonstationary transition model. To make this more explicit, even in the case when the transition process is stationary, the transition model will be different between trials due to the accumulation of concentration parameters.

Large differences in the values of the transition process over time (i.e. a highly nonstationary transition process) will result in the agent using outdated information for its own transition model. At any moment in time, the agent therefore needs a way to only use that information in its generative model that is still relevant to the agent's explaining of the current real world dynamics.

3.3. Suggested resolution

The solution to using only that information in the generative model that is still relevant to the agent, and as such having better performance in nonstationary environments, is unlikely to be onefold. In theory, multiple solutions exist. A register of possible resolutions that can be considered will be discussed in chapter 6. One possible resolution is to expand the generative model with a higher level that represents

a collection of possible variations the dynamics of the world can take on. While this seems an excellent option on paper, a problem that arises is that the designer manually has to partition all possible variations of the dynamics the world can take on (or at least the most important ones) into a finite set, and this has some important consequences: (1) it requires extra prior knowledge from the designer about the world (that is not always available), and (2) it requires extra computational resources, which can be problematic when dealing with already large state- and policy-spaces. Another approach that is much less dependent on the prior knowledge of the designer and less computationally expensive as well (and thus more generally applicable), is to implement a way of forgetting part of the information in the agent's generative model. This will be the approach taken in this thesis to cope with nonstationary state transitions.

Although active inference is said to be a candidate theory of explaining brain function, the existing updating scheme of discrete state-space active inference is incapable of forgetting — a phenomenon common in cognition. The need for a forget mechanism in active inference can be reinforced by the fact that the standard discrete state-space formulation of active inference does not take into account the uncertainty in the transition model for the evaluation of policies [12, 13, 18]. While uncertainty in policy selection itself is mentioned in literature (usually the inverse of uncertainty is mentioned, i.e. precision as mentioned in chapter 2), uncertainty in the transition model does not influence the agent's policy selection. Uncertainty in the transition model is not a completely unexplored topic in active inference though, as it has been discussed before in [27]. This paper tries to equip the agent with beliefs about the uncertainty in the transitions of hidden states, by introducing a precision parameter for the state transitions. However, this parameter is subsequently only used to determine the weight the agent gives to the knowledge contained in the transition matrix when performing state estimation (the agent also weighs other knowledge, e.g. that contained in the likelihood matrix). While this can be useful to infer the agent's state history in cases where the agent needs to know whether the knowledge in its likelihood matrix or transition matrix is more reliable, it does not solve the problem of actually getting an accurate model of the transition dynamics (especially in nonstationary environments), since the learning scheme remains unchanged. Forgetting can be a solution to this problem.

For this work, inspiration for the mathematical implementation of forgetting will be drawn from reinforcement learning, deep learning and time series analysis. As such, three main ways of implementing forgetting will be described in this section: (1) the addition of a forget rate parameter to the existing updating scheme of the generative model, (2) an alternative updating scheme for the generative model based on the updating of an LSTM cell, and (3) an updating scheme for the generative model using a rolling summation for the accumulation of concentration parameters.

3.3.1. Forget rate parameter

Forget rate parameters have previously been used in reinforcement learning as an extension to temporal difference learning methods [19, 20]. These forget rates can be seen as counterparts to the traditional learning rates in temporal difference learning: whereas learning rates are used to scale the value of the temporal difference error (the difference between the estimated value at the current step and the estimated value at the next step) like in equation 3.6, forget rates can be used to scale just the estimated value at the current step like in equation 3.7:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_L \delta(t) \tag{3.6}$$

$$Q(s_t, a_t) \leftarrow (1 - \alpha_F) Q(s_t, a_t) \tag{3.7}$$

Equation 3.6 is commonly used in reinforcement learning, whereas equation 3.7 has only been used in certain papers to update the action value of the unchosen actions [19, 20]. Since learning rates are more commonly used than forget rates in reinforcement learning, a question that arises is why to not simply use a learning rate in active inference instead of a forget rate when dealing with nonstationarity. As a matter of fact, learning rates have been mentioned in the discrete state-space active inference literature before [31], but there is a major caveat why using a learning rate in active inference might not be sufficient. This has to do with the fact that in reinforcement learning, the updating of the estimated value $Q(s_t, a_t)$ can go in two directions, that is the temporal difference error $\delta(t)$ can be either positive or negative. Active inference does not make use of value functions, and updates the generative model with the use of concentration parameters, as mentioned in chapter 2. Since these concentration parameters can be seen as a form of counting, by definition these parameters are always nonnegative and thus, will

never decrease. The updating of the generative model therefore always happens in one direction. By adding a forget rate parameter instead, the generative model can actually be updated in two directions.

Moving on to the implementation, with the forget rate parameter approach, a forget rate parameter α_F will be applied to the old concentration parameter matrix b of the transition model at the time when the new values for this concentration parameter matrix $\mathbf{b}$ are calculated. Mathematically, the concentration parameters for the transition model will be updated as follows:

$$\begin{aligned}\mathbf{b}(u) &= (1-\alpha_F)\cdot b(u) + \mathscr{b}(u) \\ &= (1-\alpha_F)\cdot b(u) + \sum_{\pi\in\Pi}\sum_{\tau=1}^{T-1}\delta_{\pi(\tau),u}\boldsymbol{\pi}_\pi \mathbf{s}^\pi_{\tau+1}\otimes\mathbf{s}^\pi_\tau\end{aligned} \tag{3.8}$$

with δ being the Kronecker delta. Here, the forget rate α_F takes a value between 0 and 1. When $\alpha_F = 0$, the agent will not forget any part of its previously accumulated concentration parameters, and equation 3.8 effectively reduces to equation 2.30. At the other extreme, when $\alpha_F = 1$, the agent will forget all of its previously accumulated concentration parameters. This has as consequence that the generative model of the agent at the next trial will exclusively be comprised of the experience gained at the current trial. A value of 1 for the forget rate is something to avoid: since it is very unlikely that the agent can take all possible actions in every possible state at least once in a single trial, this will leave the agent with large knowledge gaps in its generative model.

This leaves the question what values for the forget rate are good values to use in practice. This will usually depend on the dynamics of the task to be solved by the agent, although the simplest option is to set the forget rate to a small constant number. To stay in line with previous implementations in reinforcement learning [19], this will be the approach taken.

Algorithm 1 shows the pseudocode for implementing the forget rate parameter update scheme applied to the transition model in active inference.

Algorithm 1 Implementation of the forget rate parameter update scheme in active inference.

```
initialize b, forgetRate, ...
for nTrials do
    procedure ActiveInference(b, ...)
        ...
        return trialExperience, ...
    end procedure
    procedure updateSchemeForgetRate(b, trialExperience, forgetRate)
        b ← (1 − forgetRate) ∗ b + trialExperience
        return b
    end procedure
end for
```

3.3.2. LSTM cell

The second main approach taken in this section for implementing forgetting in active inference is to use an updating scheme inherently different from the existing updating scheme in active inference. This subsection will describe an update method for active inference based on the updating of an LSTM cell.

A long short-term memory (LSTM) is a type of recurrent neural network (RNN), which is often used for handling temporal sequences of data. However, unlike regular RNNs, LSTMs are equipped with a so-called cell state, which embodies the network with a form of memory. At every timestep, the cell state is updated by forgetting part of the old information in the cell state and adding new information from the input. The notion of forgetting old and adding new information is exactly what is desired for the updating of an active inference agent's generative model when dealing with nonstationarity. Therefore, adding elements of the updating mechanism of an LSTM cell to the updating mechanism of the generative model in active inference can result in the agent dealing better with nonstationarity.

A diagram depicting the control flow of an LSTM cell can be found in figure 3.1. The main signals in this diagram are the cell state c_t, the hidden state h_t (not to be confused with the hidden states used in active inference) and the input x_t. Besides these signals, additional signals f_t, i_t, $\tilde{c}_t$ and o_t are used to

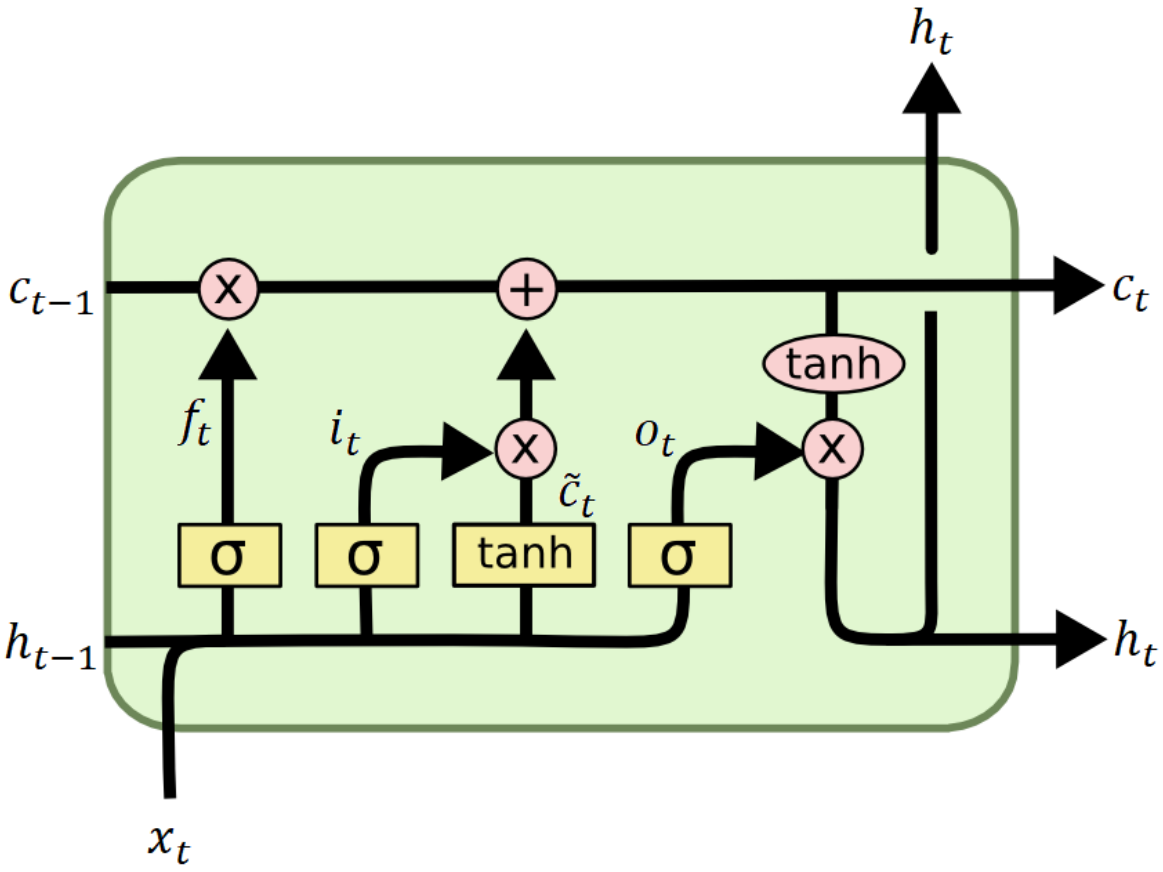


Figure 3.1: Visualization of the signals in an LSTM cell. The multiplication operator × is applied on an elementwise basis, whereas σ and $\tanh$ refer to the logistic function and hyperbolic tangent respectively. (This figure is adapted from [25].)

store the values of intermediate calculations. These latter symbols are not chosen randomly, because they represent a characteristic of the LSTM cell, namely its gates. The symbols f_t, i_t, and o_t represent the forget gate, input gate and output gate respectively. Keeping this in mind helps in understanding the equations for the LSTM cell, which are then given by:

$$f_t = \sigma_l(W_f x_t + U_f h_{t-1} + b_f) \tag{3.9}$$

$$i_t = \sigma_l(W_i x_t + U_i h_{t-1} + b_i) \tag{3.10}$$

$$\tilde{c}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \tag{3.11}$$

$$o_t = \sigma_l(W_o x_t + U_o h_{t-1} + b_o) \tag{3.12}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{3.13}$$

$$h_t = \sigma_h(c_t) \circ o_t \tag{3.14}$$

Here, the W and U are weight matrices for the input and recurrent connections respectively, whereas the b are bias vectors (confusion with the transition concentration parameters in active inference, which use the same symbol, can be prevented by looking at the subscrips). Moreover, σ_l denotes the logistic function, whereas σ_h denotes the hyperbolic tangent. Furthermore, $\circ$ represents the Hadamard product, i.e. an elementwise multiplication.

However, before these equations can be used in the first place, a translation has to be made from the variables that are used during the updating of the generative model in active inference to the signals that are used in an LSTM cell. For this translation, the following choices have been made:

- The concentration parameter matrix b is used as the cell state. Since the cell state represents the memory of an LSTM cell, it is rational to use the concentration parameter matrix as this signal, as the concentration parameter matrix keeps track of how often what transitions have occurred in the past.
- Similarly, the initial concentration parameter matrix (as set by the designer) is given as the initial cell state (i.e. the signal c_{t-1} at trial 1). Likewise, a version of this matrix normalized by the hyperbolic tangent is given as the initial hidden state (i.e. the signal h_{t-1} at trial 1), since the hidden state can never exceed this value when following the normal flow of signals in figure 3.1.
- The experience gained at the current trial, $\mathscr{b}$ (see equation 2.31), is used as the input, since this is the new information to be added every iteration.
- The weight matrices are set to identity matrices, whereas the bias vectors are set to zero vectors. The reason for this is that in supervised learning, the network is given input with known labels, and as such is able to modify the weight matrices and bias vectors to attain optimal performance. In active inference however, the data is not labeled, and as such, updating the weights and biases in this way is not possible.

Algorithm 2 shows the pseudocode for implementing the LSTM cell update scheme applied to the transition model in active inference.

Algorithm 2 Implementation of the LSTM cell update scheme in active inference.

initialize b, ...
$cellState \leftarrow b$
$hiddenState \leftarrow \tanh(cellState)$
for $nTrials$ **do**
    procedure ActiveInference(b, ...)
        ...
        return $trialExperience$, ...
    end procedure
    $input \leftarrow trialExperience$
    procedure updateSchemeLstm($input, cellState, hiddenState$)
        $forgetGate \leftarrow \text{sigmoid}(input + hiddenState)$
        $inputGate \leftarrow \text{sigmoid}(input + hiddenState)$
        $cellStateTilde \leftarrow \tanh(input + hiddenState)$
        $outputGate \leftarrow \text{sigmoid}(input + hiddenState)$
        $cellState \leftarrow cellState .* forgetGate + inputGate .* cellStateTilde$
        $hiddenState \leftarrow \tanh(cellState) .* outputGate$
        return $cellState, hiddenState$
    end procedure
    $b \leftarrow cellState$
end for

3.3.3. Rolling summation

The final approach taken here for implementing forgetting in active inference is to use a rolling summation instead of a total summation for the accumulation of the concentration parameters. Similar to a rolling average, a rolling summation takes into account only the most recent experience gained in the form of the concentration parameters. How many trials the agent takes into consideration depends on the width of the window w.

With the way the concentration parameters construct the agent's generative model, there is one caveat. If the agent would use exclusively the concentration parameters of the past w trials, in the case that the agent did not execute certain actions in certain states during the time window, the agent will not have concentration parameters for those state transitions. The remedy for this problem chosen here, is to complement the concentration parameters with the priors (i.e. the initial concentration parameters) provided by the designer. This means that in the absence of experience about certain state transitions

in the last w trials, the agent will be reset to its original parameters instead of having to restart without concentration parameters at all. Therefore, at any given trial k, the agent will use only the concentration parameters that were accumulated during trials $k-w$ to $k-1$, in addition to the initial concentration parameters, to construct its transition model:

$$b_k = b_{initial} + \sum_{\kappa=k-w}^{k-1} \mathscr{b}_\kappa \tag{3.15}$$

$$\mathscr{b}_\kappa = 0, \quad \forall \kappa \leq 0 \tag{3.16}$$

Equation 3.16 is a constraint to define that there are no concentration parameters for trials before the first trial, because there does not exist a zero or negative numbered trial. This definition is needed, because the summation index $\kappa = k - w$ in equation 3.15 is nonpositive during the first w trials.

The advantage this rolling summation updating scheme is expected to have over the other update methods, is that the agent is able to rapidly forget all of the experience that it gained w trials before the current trial, instead of having that experience slowly decay. This trait is especially desired in environments that evolve in quick succession.

Algorithm 3 shows the pseudocode for implementing the rolling summation update scheme applied to the transition model in active inference.

Algorithm 3 Implementation of the rolling summation update scheme in active inference.

initialize $b, windowSize, \ldots$
$bInitial \leftarrow b$
$summationWindow \leftarrow \text{zeros}(\text{size}(b, \dim 1), \text{size}(b, \dim 2), \text{size}(b, \dim 3), windowSize)$
for $nTrials$ **do**
    procedure ActiveInference(b, ...)
        …
        return $trialExperience, \ldots$
    end procedure
    procedure updateSchemeRollingSummation($trialExperience, summationWindow$)
        for $i = 1, \ldots, windowSize - 1$ **do**
            $summationWindow[\text{all}, \text{all}, \text{all}, i] \leftarrow summationWindow[\text{all}, \text{all}, \text{all}, i+1]$
        end for
        $summationWindow[\text{all}, \text{all}, \text{all}, windowSize] \leftarrow trialExperience$
        return $summationWindow$
    end procedure
    $b \leftarrow bInitial + \text{sum}(summationWindow, \dim 4)$
end for

4

Test methods

The preceding two chapters explained the theory behind the fundamental working principles of discrete state-space active inference, and introduced the notion of nonstationarity in active inference along with possible methods to cope with nonstationarity respectively. This chapter will put that theoretical knowledge into practice by testing the performance of those methods. First, the problem setup that will be used for testing the performance of a discrete state-space active inference agent in a nonstationary environment is described, after which the way nonstationarity is modeled in this setup is illustrated. Hereupon, the elements that influence the performance of the agent are specified, and the chapter is concluded by defining the metrics that will be used to quantify this performance.

4.1. Problem setup

The problem to be solved by the active inference agent in this thesis is a simulated navigation task, in which the agent has to traverse a maze-like gridworld, trying to find the shortest path from start to goal. For this task, there exists an $M \times N$ gridworld, which is composed of cells of three distinct types:

- Open cells. These are the cells that the agent can freely traverse from any other open cell.
- Starting cell. This is the cell from which the agent will start at the first trial of the task. It is also the cell from which the agent will start the next trial after it has reached the goal cell during the previous trial. Apart from these properties, the starting cell has the same properties as a regular open cell.
- Goal cell. This is the cell the agent has to reach during the task. When the agent has done so, it will remain in the goal cell until the end of the trial, after which it will return to the starting cell for the beginning of the next trial.

The composition of the gridworld does not change during a task. The space enclosing the perimeter of the gridworld will act as a barricade, thereby preventing the agent from exiting the gridworld. Figure 4.1 shows an example of a gridworld, containing each of the cell types described above.

4.1.1. States, outcomes and actions

Since an active inference agent needs a state-space representation of the world in order to interact with it, first the 2d position coordinates of the cells comprising the gridworld have to be converted to states. As the 2d positions of the cells in the gridworld are essentially the subscripts of a 2d matrix, a state-space representation of the gridworld can simply be obtained by converting the subscript notation of the cells in the gridworld matrix to an index notation. For a 4×4 gridworld, this results in a 1d state vector of length $4 \times 4 = 16$, in which each different state represents a different grid cell. In this state vector, the first element refers to the cell in the upper left corner of the gridworld. The numbering continues by moving down along the rows in the left most column until the final row in this column has been reached. Hereafter, the process repeats for the next columns, until the cell in the lower right corner of the gridworld has been reached.

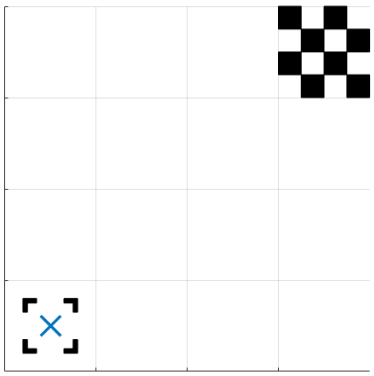


Figure 4.1: Example of a gridworld used in the active inference tasks. This figure shows each type of cell described at the start of this section in a 4×4 gridworld. White squares represent open cells, the starting cell is marked with four corner symbols, and the goal cell is marked with a checkered pattern. Additionally, everything outside of the grid is inaccessible. Finally, during the simulation, the current position of the agent would be marked with an X.

For the outcomes in this task, the choice has been made to have a one-to-one mapping from states to outcomes. In other words, in this task the agent is capable of observing the states directly. This choice was substantiated by the fact that the emphasis in this task is not on perception: rather it is on action selection and learning.

Regarding action selection, the agent has 4 options of possible actions at its disposal. These options are to move to the west, east, north, and south respectively:

$$\mathcal{U} = \{west, east, north, south\} \tag{4.1}$$

Whether the agent actually arrives at its cell of choice depends on the transition dynamics of the world. For example, when the agents chooses $u = west$, but there is no west-neighboring cell (i.e. the gridworld boundary is located to the west of the current cell), the agent will remain at the same position.

4.1.2. Policies, time horizons and trials

The agent has to take multiple consecutive actions to arrive at the goal state. The agent will look forward a certain amount of steps into the future to determine the best sequence of actions, or policy. This window which the agent uses to look forward a number of steps into the future is referred to as the time horizon [18]. How many actions are needed to reach the goal state depends on the distance between the goal state and the current state. However, there is a limiting factor in setting the time horizon of a policy. This limiting factor is easily exposed with the following example. Since the agent has the choice of 4 possible actions at any state, when the agent uses a time horizon of 2, it has to consider $4^2 = 16$ different policies. Using a time horizon of 3, the agent already has to consider $4^3 = 64$ different policies. As can be seen from this example, the number of policies to consider grows exponentially with the size of the time horizon. As such, the computation time for evaluating these policies grows exponentially as well. Clocking the computation times for different time horizons has shown that for a single trial, the computation time goes from the order of milliseconds into the order of seconds when the horizon increases from 3 to 4, and goes from the order of seconds into the order of minutes when the horizon increases from 7 to 8.

A time horizon of 4 would entail that the agent will look forward 4 steps into the future before selecting its action to take in the current state. However, this does not necessarily mean that the agent will be able to reach the goal state within those 4 steps. Thus, depending on the layout of the maze, the agent might need multiple trials to reach the goal state from the starting state. As such, if the agent was not able to reach the goal state in the previous trial, it will start the next trial from the state it ended the previous trial and not from the designated starting state. This is different from other work on similar tasks, in which the agent was always able to reach its goal within one trial [12, 13, 31].

4.2. Nonstationarity

With the setup described in the previous section, it is now possible to add nonstationarity to the equation. As mentioned in chapter 3, nonstationarity can have different manifestations in active inference, but the nonstationarity in this task is embodied exclusively in the transition process. For this task, the nonstationarity in the transition process is modeled as an external force acting on the agent that will be present for the whole duration of the task. Crucially, the properties of this force will change over time. Intuitively, this force can be interpreted as a wind, varying in magnitude and direction over the trials of the task:

$$\begin{bmatrix} W_x \\ W_y \end{bmatrix} = \begin{bmatrix} |W| \cos(\theta) \\ |W| \sin(\theta) \end{bmatrix} \tag{4.2}$$

As such, the nonstationary force perturbation will be referred to as the wind in what follows.

4.2.1. Cyclostationarity

In chapter 3, it was mentioned that this research will focus on cyclostationarity. In this subsection, it will be described how this type of nonstationarity is implemented in the force perturbation that changes over time.

When the force perturbation is modeled as a cyclostationary process, the wind magnitude is set at the start of the first trial and remains unchanged for the whole task. Additionally, the wind direction is set at the start of the first trial, but changes by $\Delta\theta$ every $1/f_{change}$ trials:

$$\theta = \theta_{initial} + \left(\lceil k \cdot f_{change} \rceil - 1\right) \cdot \Delta\theta \tag{4.3}$$

where k is the number of the current trial. For this type of task, the change in frequency of the wind direction, the initial wind direction and the angle of change in wind direction are tunable parameters. These, along with other tunable parameters and the assignment of their values are further discussed in the next section.

4.2.2. Calculating the transition probabilities

When the magnitude and direction of the wind are known for the trial, the x- and y-components of the wind can be calculated with equation 4.2. Hereafter, the only step remaining in defining the transition process is to map the state transition probabilities given the selected action and wind perturbation.

To make the analysis of the results in the upcoming chapter not unnecessarily complex, for the experiments that follow, it has been chosen to keep the magnitude of the wind constant at $|W| = 1$ and only change the direction of the wind. When the wind direction changes, it will always be by an angle of $\pi/2$. This way, at any given time, the wind direction can simply be expressed by one of the four cardinal directions: north, east, south or west. This allows for an easy mapping of the state transitions given the selected action of the agent and the current wind direction, since the possible actions now correspond to the wind directions. Table 4.1 gives a general mapping of the state transition probabilities.

4.3. Experiments

The experiments to be conducted will have the agent performing in a gridworld, while varying certain conditions of the tasks. The conditions that will be varied are the layout of the maze, the nonstationarity type in the force perturbation and the update method of the generative model. In the end, the performance of the agent will be compared between these conditions. Besides the task conditions, there exists a set of tunable parameters that will influence the performance of the agent as well. To ensure fair comparison between conditions, the values for the tunable parameters will be identical when comparing different conditions.

During the simulation of a task, the conditions are set and will not change. That is, changes in conditions only happen between tasks, not within tasks. To prevent misunderstandings, table 4.2 provides the temporal hierarchy of the experiments, showing what happens at every level of the timescale.

4.3.1. Conditions

The agent will have to solve the task in different layouts of the maze, prone to force perturbations that will take on different nonstationarity types, and using different update methods of the generative model.

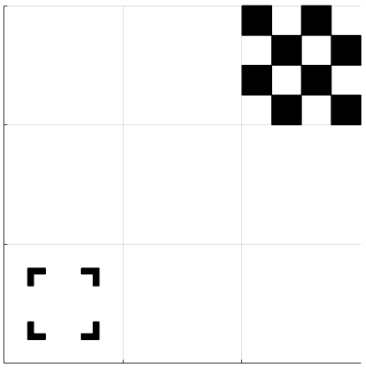
Table 4.1: General mapping of the state transition probabilities in a cyclostationary environment.

		Probability of the agent:		
		Moving in the direction of the selected action*	Moving in the direction perpendicular to the selected action, parallel to the wind*	Staying in place
The agent selects the action to:	The same direction as the wind	1	0	0
	The opposite direction of the wind	0.5	0	0.5
	A direction perpendicular to the wind	0.5	0.5	0

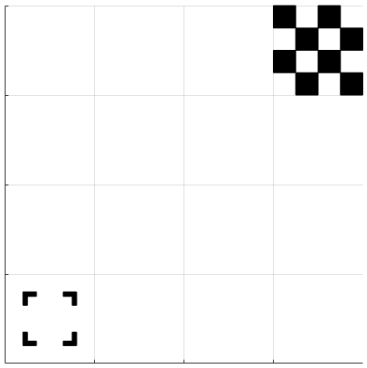
*The probability of moving in this direction applies only to the case that the cell in this direction is accessible (i.e. it is not to the outside of the gridworld). In the case that this cell is inaccessible, the agent will stay in place. For example, consider the gridworld in figure 4.1 with the agent in the bottom left cell: now, when the wind is facing to the west and the agent takes the action to the north, there is 50% probability that the agent will move to the north and a 50% probability that the agent will stay in place (since it cannot move any further to the west).

Table 4.2: Temporal hierarchy of the experiments.

Level	Description
Episode	An episode is a sequence of trials. Experience from one episode is not carried over to another episode. The purpose of having multiple episodes is just to average the results for a specific combination of conditions.
Trial	A trial is a sequence of timesteps. At each trial, the agent selects a policy to follow, performs state inference and updates its generative model. For the experiments in this research, the nonstationarity in the transition model only presents itself at this level.
Timestep	A timestep is the smallest level of the timescale in a discrete state-space active inference task. At each timestep, the agent finds itself in a specific state, observes a specific outcome and (with the exception of the final timestep in a trial) takes a specific action.



(a) Layout of map1, a 3×3 gridworld.



(b) Layout of map2, a 4×4 gridworld.

Figure 4.2: Layouts of the different maps. Refer to figure 4.1 for a description of the shapes that comprise the gridworld. At first glance, there does not seem to be an optimal path in these maps, as multiple shortest paths exist between start and goal. This however, is only true when the state transitions are deterministic in nature. For example, consider in map1 a wind that faces in the eastern direction. In this case, it is better to select a policy in which the agent first moves in the northern direction and only then in the eastern direction. This is because the agent can get stuck along the eastern edge of the map, since there is a chance that the east facing wind will push the agent against that edge of the map.

Gridworld layouts
Regarding the layout of the gridworlds, the agent will be tested in gridworlds of different sizes, to determine the influence of the size of the state-space. The layouts of the gridworlds are kept simple to obtain general conclusions about the performance of the agents. Figure 4.2 shows a map for each of the different gridworld layouts. To simplify the cross-referencing later in the next chapters, each map is given its own tag, following the simple naming convention 'map#', where # is a number.

Nonstationarity types
Regarding the nonstationarity type, the agent will be tested in environments that make use of one of the two following types of transition processes:

- A deterministic stationary transition process. In this case, the agent will always end up in the same state when taking a specific action in a specific state, regardless of the time this action was taken. Generally, this means that when the agent decides to take an action to move in a certain direction, the agent will always move in that direction. These probabilities for transitioning will remain unchanged for the whole episode.
- A cyclostationary transition process. In this case, the initial transition probabilities conform to the dynamics of a nondeterministic stationary process. That is, the agent will not always ends up in the same state when taking a specific action in a specific state. For example, when the wind is facing to the west and the agent decides to take the action to move to the north, there is a possibility that the agent moves either to the west or to the north. Moreover, between trials within the episode, the dynamics also change cyclically over time as mentioned in section 4.2. The frequency of the change in dynamics is a tunable parameter.

Just like the maps, the nonstationarity types will be given their own tag for future referencing. The two nonstationarity types will be given the tags 'stationary' and 'cyclostationary' respectively.

Update methods
To discover whether the performance can be improved from the standard implementation of discrete state-space active inference, the agent will be tested while using one of the different update methods for its generative model that were mentioned in section 3.3:

- The standard updating scheme used in discrete state-space active inference. Being the default method, incapable of forgetting, this is the method the other methods have to outperform.

- The updating scheme with the added forget rate parameter, in which case the forget rate parameter is set to a constant value. The exact value of this constant is a tunable parameter.
- The updating scheme that translates the updating of an LSTM cell to the updating of the generative model in discrete state space active inference.
- The updating scheme using a rolling summation instead of a total summation for the accumulation of the concentration parameters. The rolling summation uses a window width that determines how many trials are taken into account when accumulating concentration parameters. The value of the window width is a tunable parameter.

Again, each of the update methods is given their own tags, which are 'standard', 'constant', 'lstm' and 'rolling' respectively.

4.3.2. Tunable parameters

Besides the previously mentioned task conditions, there are other elements that will influence the performance of the agent, even when the former task conditions remain unchanged. The following list shows the parameters that can be tuned. For the results given in the first two sections of the next chapter, these parameters have been set to constant values, for which the argumentation will be given below.

- Utility, U. As mentioned in chapter 2, the utility determines the agent's prior preferences over outcomes. Since every type of outcome is given its own utility value, these values can influence for example to what extent the agent prioritizes reaching its goal over exploring other outcomes. For the experiments that follow, the utility of observations has been set as follows:

$$U(o) = \begin{cases} 10, & o = goal \\ -5, & o = start \\ 0, & \text{otherwise} \end{cases}$$

  In principle, the exact values are not as important as long as the more preferable outcomes are given higher values than less preferable outcomes.

- Initial concentration for the transition model, $b_{initial}$. A higher value for the initial concentration parameters of the transition model means that the agent will have a harder time to adapt to dynamics that do not correspond to these concentration parameters. For the experiments that follow, the agent is provided nonzero initial concentration parameters for state transitions that would happen when the transition process would be deterministic. For example, when in a certain state the agent would take an action to the west, the only nonzero concentration parameter for that action in that state would be the state transition to the west. The values for the nonzero initial concentration parameters are set to 8, a value commonly used in literature [18].

- Length of the policies. The policy length determines how many consecutive actions the agent takes per trial. A longer policy length means in theory that the agent will have an easier time reaching goals that are more distant. However, this also comes at the cost of a longer computation time. For the experiments that follow, a policy length of 4 is utilized. This is chosen because the computation time does not increase significantly when changing the length from 3 to 4 (order of milliseconds per trial), but does increase significantly when changing the length from 4 to 5 (order of multiple seconds per trial).

- Frequency of change in dynamics, f_{change}. The agent needs time to learn and adapt to the transition dynamics. As such, changing the dynamics of the task too frequently (expressed in the inverse of the number of trials) generally means that the agent will have decreased performance. For the experiments that follow, $f_{change} = 1/16$.

- Initial wind direction, $\theta_{initial}$. Since the wind direction changes cyclically, the initial wind direction is of less importance. For these tasks, $\theta_{initial} = 0$ (to the east).

- Angle of change in wind direction, $\Delta\theta$. To expose the agent to the wind an equal amount of trials in all four cardinal directions, $\Delta\theta = \pi/2$ clockwise. This means that the wind will follow the cycle $east \rightarrow south \rightarrow west \rightarrow north \rightarrow east$.
- Constant forget rate, α_F. The value the forget rate parameter takes when the update method is set to 'constant'. For the experiments that follow, $\alpha_F = 0.1$. The discussion in chapter 6 debates the modification of this value.
- Window width, w. The value of the window width (expressed in the number of trials) when the update method is set to 'rolling'. For the experiments that follow, $w = 8$.

The final section of the next chapter will also provide tuning results for when these parameters take different values. Considering the large number of tuning parameters, it is practically not feasible to test the influence of all of these and simultaneously repeat the tasks for a sufficient amount of episodes. Therefore, a choice has to be made for selecting those tuning parameters for which the influence on the performance is the most relevant. The chosen parameters that will be tested are the length of the policies and the window width.

4.4. Performance metrics

To realize a quantifiable evaluation of performance in general, suitable metrics have to be defined. For this kind of task, it is chosen to evaluate the performance of the agents based on the number of steps the agent needs to reach the goal state from the starting state per successful arrival at the goal state. As such, low values for the number of steps are an indication of good performance. The median, along with the first and third quartile will be the statistical values used to report the number of steps. These are used as opposed to the mean and standard deviation, considering that the distribution of the number of steps is highly skewed for each of the methods, as will be seen in the next chapter. Besides comparing the statistical values between the different methods, the differences in performance between the standard and newly suggested update methods need to be assessed for significance, in order to draw substantiated conclusions. Significance will be tested by using the Wilcoxon rank sum test (also known as the Mann–Whitney U test) [33], as this method is applicable when dealing with non-normally distributed unpaired datasets.

5

Results

This chapter will give the quantified results of the experiments provided by the performance metrics described at the end of the previous chapter. Both of the maps introduced in the previous chapter will be given its own section, which includes one table containing two subtables, each corresponding to one of the nonstationarity types. The first column in each individual table indicates each of the update methods described in the previous chapters. The second column quantifies the performance by providing the steps to the goal for each update method. The steps to goal are given in the format: *median*, [*first quartile*, *third quartile*]. The third column gives the sample size of the data for the corresponding update method. The sample size corresponds to how often the agent was able to reach the goal using a specific update method. The final column gives the p-values to indicate statistical significance of the results. The p-values are the result of applying the Wilcoxon rank sum test between the corresponding update method and the standard update method. Since p-values are not defined when comparing identical statistical quantities, this explains the blank entries that are present in the p-value column. The results in the tables are accompanied by the boxplots of the same results.

For both maps, for each combination of nonstationarity type and update method, the agent is exposed to the environment for 8 episodes, consisting each of 128 trials. Therefore, data is obtained for 1024 trials in total per map per nonstationarity type per update method. An important reminder is that experience gained from one episode (one sequence of 128 trials) is not transferred to another episode.

Besides the tables and boxplots, for the results that have shown to be statistically significant, a visualization of the results per nonstationarity type and update method is given by providing a heatmap displaying the locations the agent has visited the most. Additionally, a visualization of the transition concentration parameter matrices at the end of the final trial is given for each update method, allowing insight of how these matrices were updated.

Finally, at the end of this chapter, tuning results for both maps will be provided, which will summarize the influence of changing a selection of tunable parameters described in subsection 4.3.2.

5.1. Results map1

From the results in table 5.1, it can be seen that for a deterministic stationary transition process, the agent is able to reach the goal in every trial and does so in the least amount of steps, regardless of the update method it is using. However, when the transition process is not deterministic stationary in nature, the performance decreases significantly. For a cyclostationary transition process, an agent using the standard update method is roughly 3 times less likely to reach the goal (as can be seen from the sample sizes in table 5.1), and often needs more than twice as many steps to reach its goal (as can seen from figure 5.1). Figure 5.2 zooms in on the distribution of the steps to the goal for the standard active inference agent in the cyclostationary environment. Thus, going by the results for map1, with active inference as it currently is, performance in cyclostationary environments is suboptimal and should be improved upon.

When comparing the results in table 5.1 and figure 5.1 of the standard update method with those of the constant update method, it can be seen that the performance in cyclostationary environments is not improved when using the update scheme with an added forget rate parameter set to a constant value.

Table 5.1: Results for map1 after simulating 8 episodes of 128 trials, with in subtable (a) a deterministic stationary transition process, and (b) a cyclostationary transition process.

(a) Using a deterministic stationary transition process.

Update method	Steps to goal	Sample size	p-value
Standard	4, [4, 4]	1024	-
Constant	4, [4, 4]	1024	-
LSTM	4, [4, 4]	1024	-
Rolling	4, [4, 4]	1024	-

(b) Using a cyclostationary transition process.

Update method	Steps to goal	Sample size	p-value
Standard	6, [4, 11.75]	383	-
Constant	5, [4, 11]	376	0.5476
LSTM	5, [4, 14]	329	0.5645
Rolling	5, [4, 9]	475	1.8268×10^{-4}

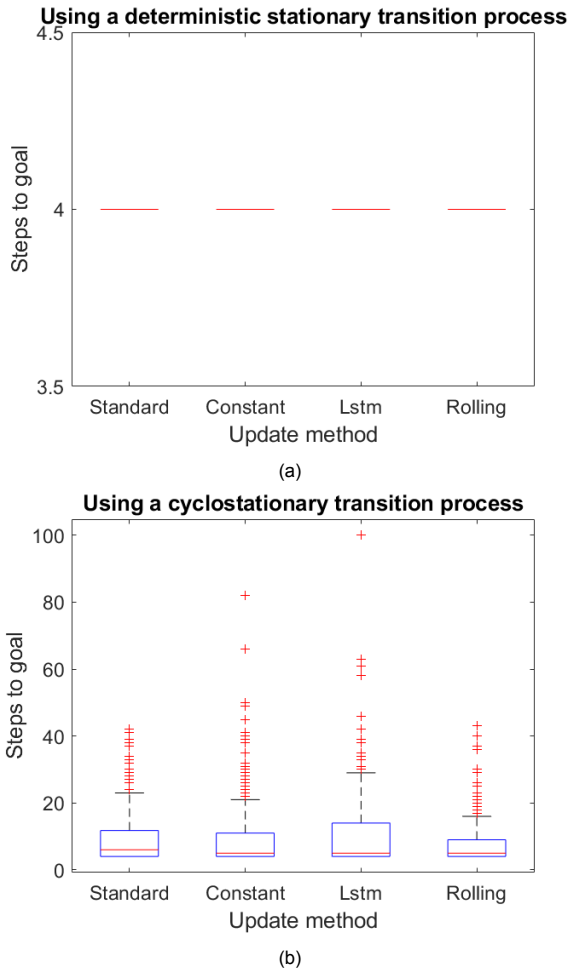


Figure 5.1: Boxplot showing the results for map1, with in subfigure (a) the results for a deterministic stationary transition process, and in (b) the results for a cyclostationary transition process.

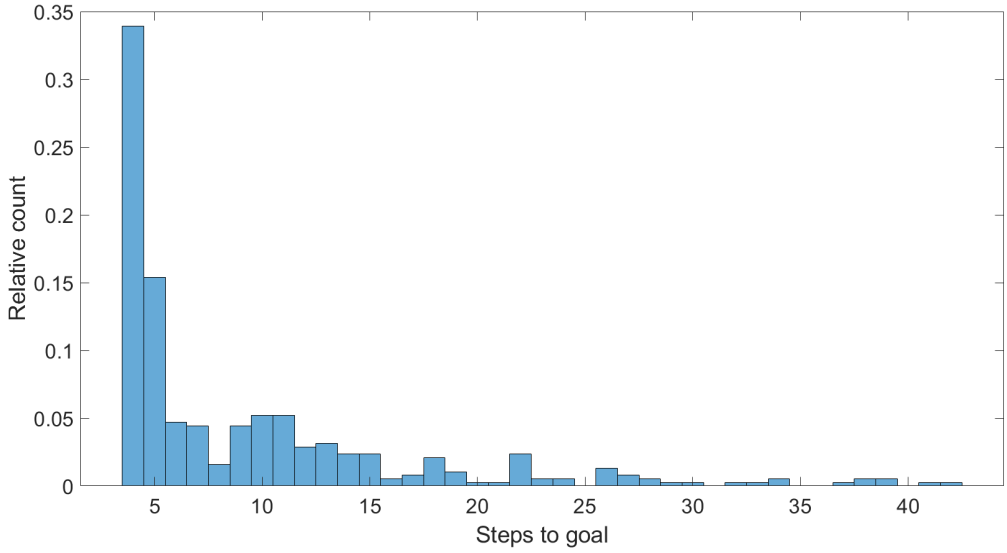


Figure 5.2: Histogram showing the relative count of the number of steps the standard active inference agent needs to reach the goal in the cyclostationary environment in map1. As can be seen from this histogram, the distribution of the steps to goal is highly skewed. This is caused by the fact that, as the number of timesteps in the task increases, it will be less likely that the agent has been influenced by the wind perturbation at every timestep up till that timestep. The distribution of the steps to goal for the other update methods take on a similar shape, as can also be inferred from the boxplots in this chapter.

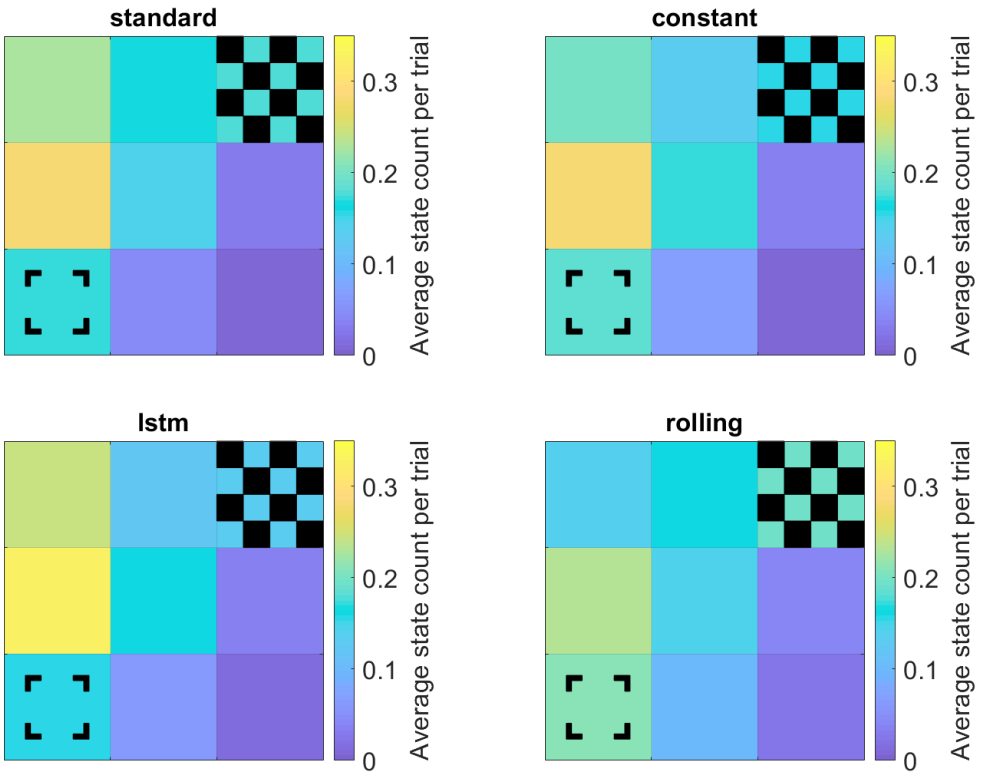
In fact, it even seems to be the case that using a constant value for the forget rate parameter decreases performance. Similarly, the use of the LSTM update method also brings a decrease in performance, even more so than with the constant update method.

From the remainder of table 5.1 and figure 5.1, it can be seen that the performance in cyclostationary environments actually can be improved: the rolling update method bring about an increase in performance. An agent using the rolling updating scheme is roughly 25% more likely to reach the goal than an agent using the standard updating scheme, and when reaching the goal, the agent also does so in less steps. By looking at the p-values in table 5.1, it can also be seen that the improvement the rolling update method gives, is actually statistically significant.

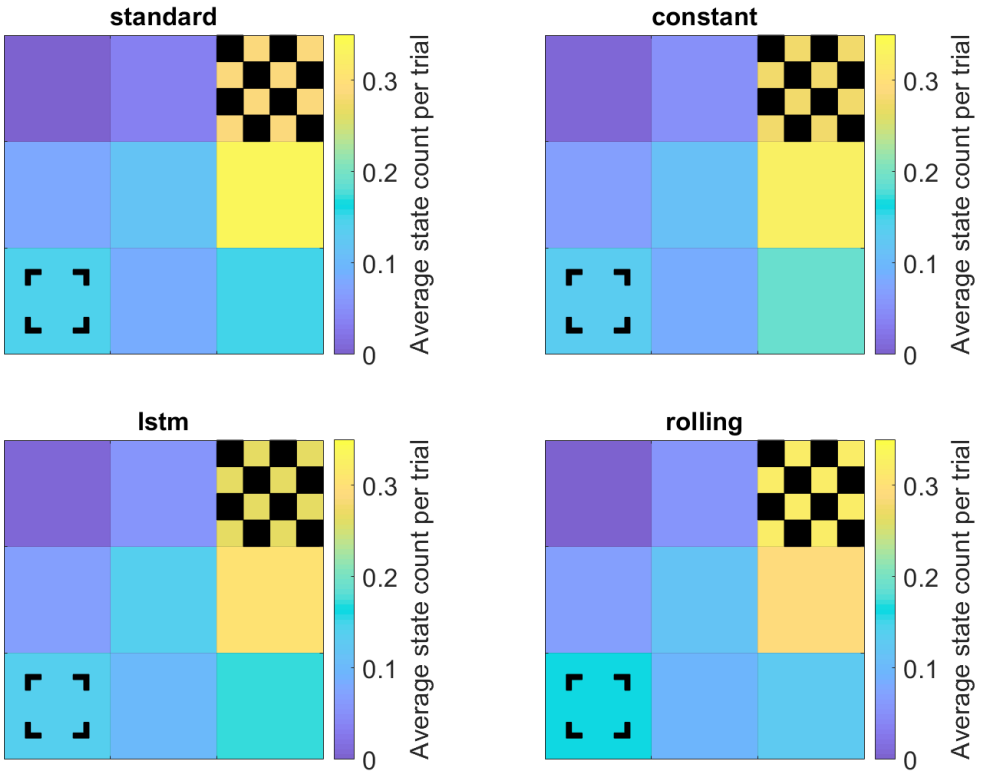
To zoom in on the comparison between the update methods in the cyclostationary environment, figure 5.3 shows a heatmap of map1 displaying the average state count per trial for an agent using the standard, constant, LSTM and rolling update methods for the four different wind directions. The heatmaps seem mostly similar, but a striking difference can be seen between the heatmaps of the standard and rolling update methods when the wind is facing north: the agent actively avoids the most northwest cell when the agent uses the rolling update method, which is beneficial in preventing to get stuck along the edge of the map. This phenomenon is also present in the easternmost cell when the wind is facing east, although somewhat less striking.

Differences in behavior are further supported by two additional statistics presented in table 5.2. These are the 'bumped-into-wall ratio', that is the amount of times the agent bumped into the perimeter divided by the total amount of trials; and the 'best-first-action ratio', that is the number of times the agent has taken the best action in the starting state divided by the total number of times the agent has been in the starting state. This best action is the action that does not increase the agent's chances of bumping into a wall before it reaches the goal, and depends on the wind direction (for example, taking the action to the east when the wind is facing to the north). As can be seen from the table, the agent using the rolling update method is less likely to bump into a wall, and more likely to take the best action in the starting state. The information in this table, along with the heatmaps, contribute to evidence that an agent using the rolling update method plans to navigate through the gridworld by using different paths than an agent using the standard update method.

Finally, to show that there actually exists a difference in the updating of the generative model between the different methods, figure 5.4 shows heatmaps of the concentration parameters at the end of the final trial (128) for each of the update methods in the cyclostationary environment. These heatmaps

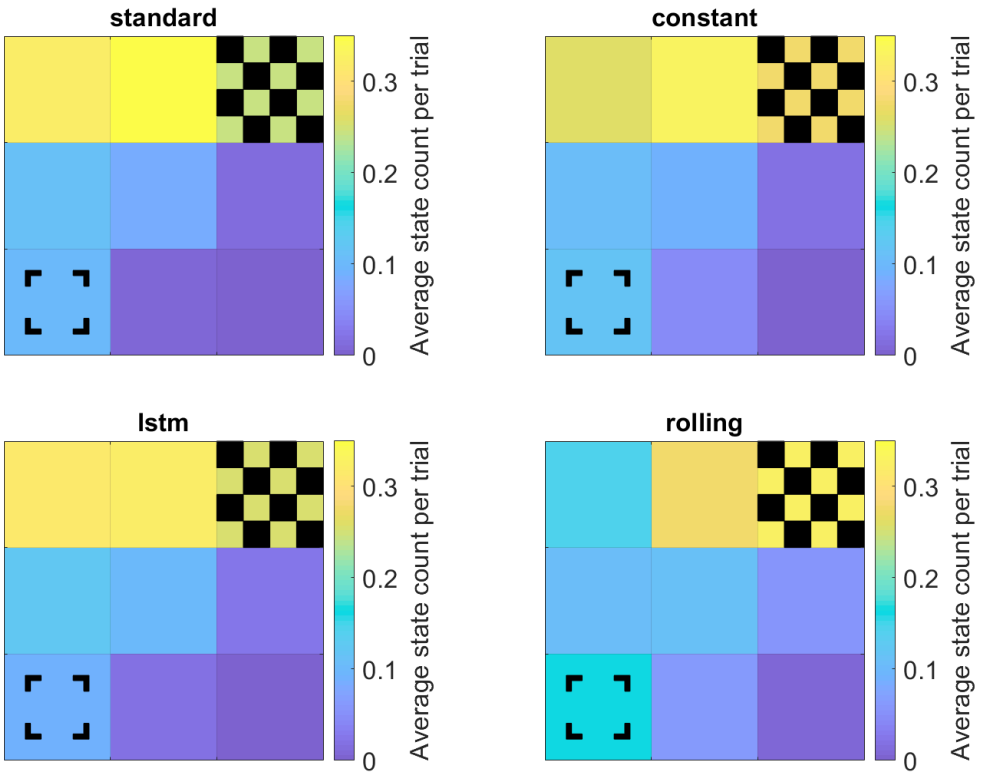


(a) When the wind is facing west.

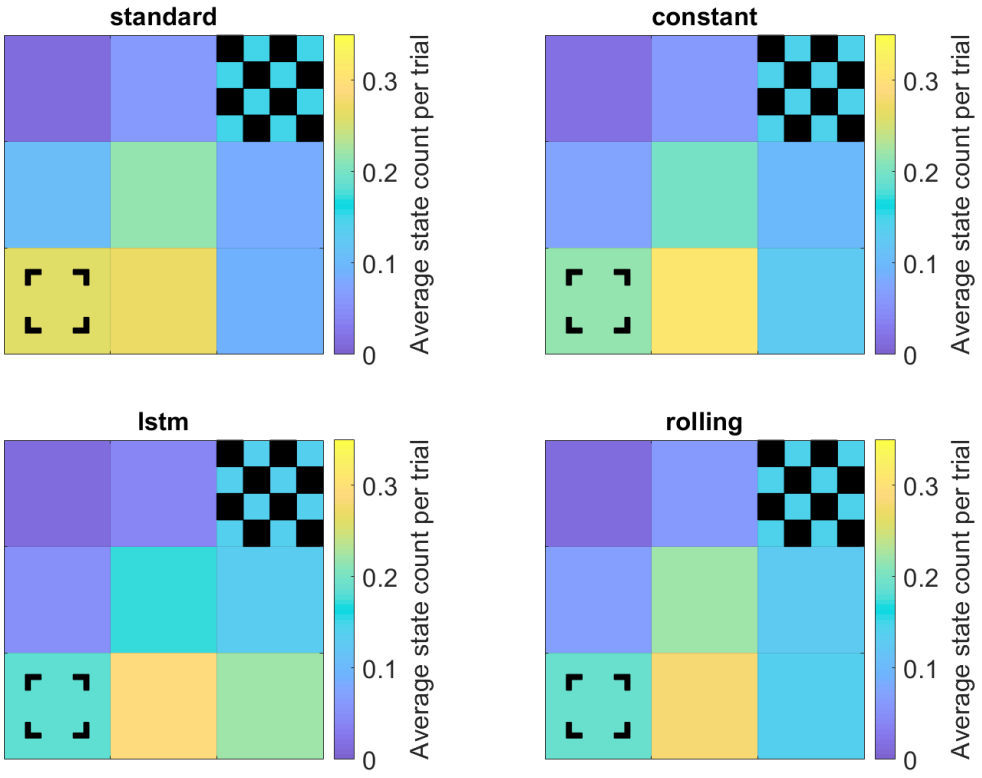


(b) When the wind is facing east.

Figure 5.3: Heatmap of map1 displaying the average state count per trial for an agent. Each subfigure shows the results for one of the four wind directions, and within each subfigure, the results of the four update methods are visualized.



(c) When the wind is facing north.



(d) When the wind is facing south.

Figure 5.3: Heatmap of map1 displaying the average state count per trial for an agent. Each subfigure shows the results for one of the four wind directions, and within each subfigure, the results of the four update methods are visualized.

Table 5.2: Additional statistics for map1 using a cyclostationary transition process. More details on the definition of these two statistics are given in the final paragraph of section 5.1.

Update method	Bumped-into-wall ratio	Best-first-action ratio
Standard	1.0957	0.5349
Constant	1.0537	0.5863
LSTM	1.1689	0.5430
Rolling	0.8525	0.6647

are the average of the 8 simulated episodes. As is expected, it can be seen that the concentration parameters for the standard update method have the highest values, since agents using the standard update method do not forget part of their concentration parameters. Also as expected is that the constant update method overall has low values for the concentration parameters. This phenomenon is also present for the LSTM update method. The values for the rolling update method are somewhere between the high values of the standard update method and the low values of the constant and LSTM update methods. However, unlike the standard update method, the rolling update method generally does not have multiple nonzero transition concentration parameters for one particular state (i.e. the concentration parameter matrices mostly have only one brightly colored cell in each column), and additionally still has relatively high values for these nonzero concentration parameters, unlike the constant and LSTM update methods. For a more comprehensive comparison of the concentration parameter matrices in time, refer to the appendix for a visualization of the evolution of the concentration parameters.

5.2. Results map2

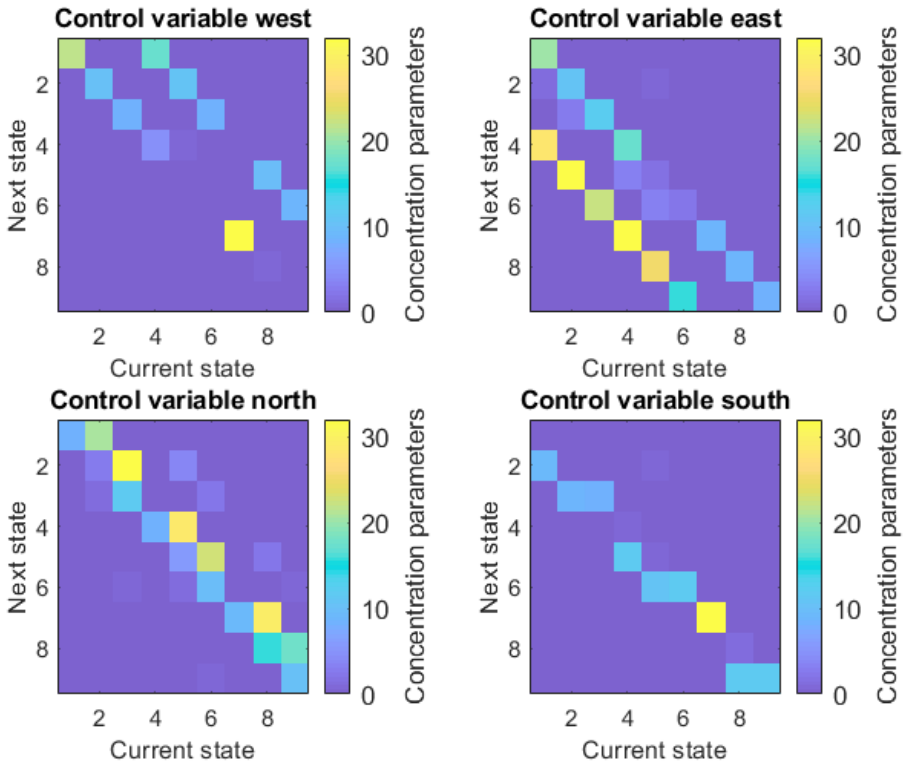
In contrast to map1, in map2 it is impossible to reach the goal state from the starting state in one trial, since the shortest path from start to goal is larger than the policy horizon the agent utilizes (6 vs 4 respectively). This means that the agent requires at least two trials to reach the goal. Going by the results in table 5.3, when dealing with a deterministic stationary transition process, the agent using the standard update method is always able to reach the goal in the minimum amount of trials possible (2), as is evident from the sample size $(1024/2 = 512)$. Compared to map1 though, the agent does not always reach the goal in the least amount of steps (6). However, different than in map1, the agent can afford to not take the shortest path and still reach the goal in the minimum amount of trials.

The agent using the rolling update methods performs very similarly to the agent using the standard update method. When the transition process is stationary, the agent using the rolling update method takes a fraction of steps more to reach the goal, and given the relatively large sample size, these results are even statistically significant. However, one can argue the practical significance of this difference, considering the effect size is very small. The boxplots in figure 5.5 even show identical results for the standard and rolling update methods.

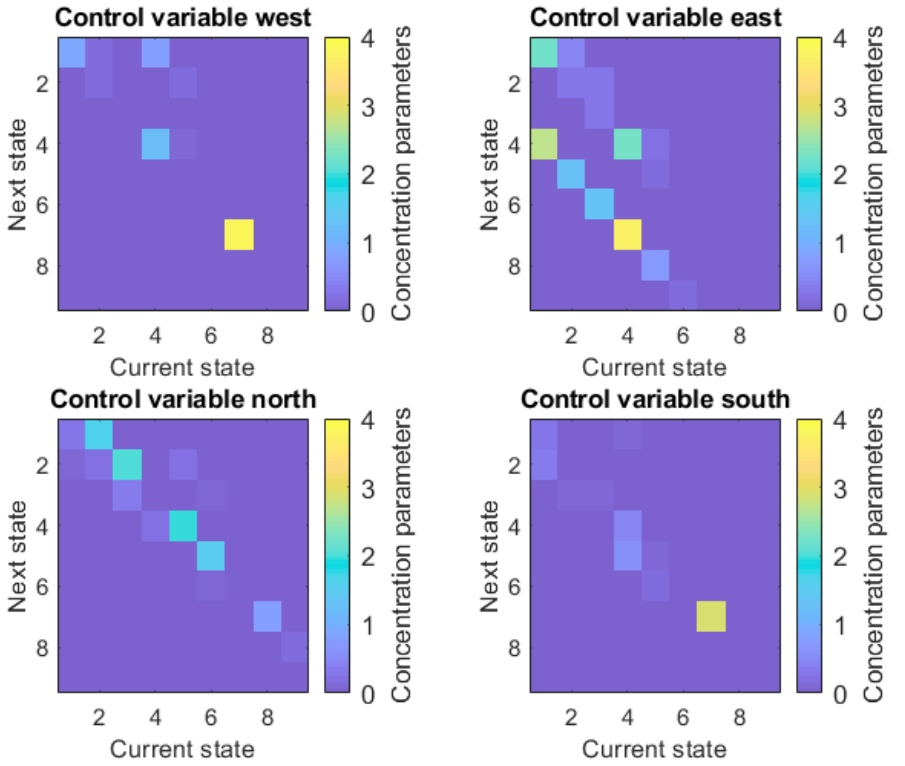
More noticable is the decrease in performance for the constant and LSTM update methods. This was already assumed from the results for map1, but the effect size for the constant update method was small and the results for both the constant and LSTM update methods were not statistically significant. Here, the effect size is noticable for both methods and the differences are statistically significant. Moreover, in contrast to map1, for map2 this decline is already present when the transition process is just stationary in nature.

Similar to map1, the performance decreases considerably when the transition process is not deterministic stationary in nature. For a cyclostationary transition process, an agent using the standard update method is roughly 2 times less likely to reach the goal, and often needs twice or more as many steps to reach its goal. Thus, there is room for improvement in the case of a cyclostationary transition process for map2 as well.

However unlike map1, going by the results from the remainder of table 5.3, it cannot be said that using another update scheme for the generative model improves performance in the case of cyclostationarity. The results show a slight increase in performance when using the rolling update method, but this improvement is not statistically significant, as is seen from the high p-values. On the other hand, it is reinforced that the constant and LSTM update methods actually decrease performance.

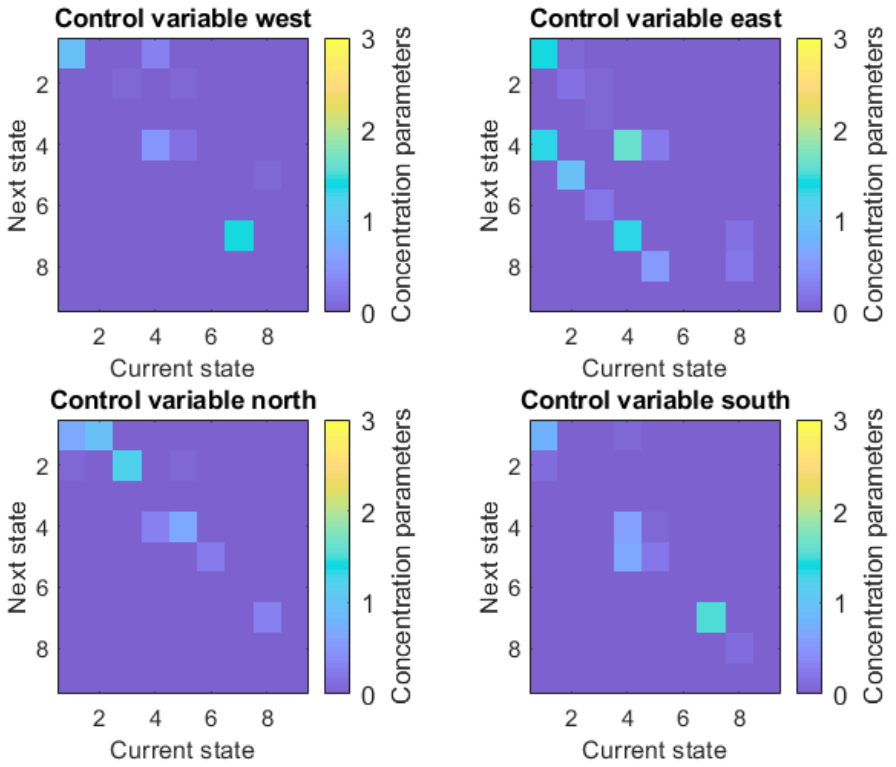


(a) For the standard update method.

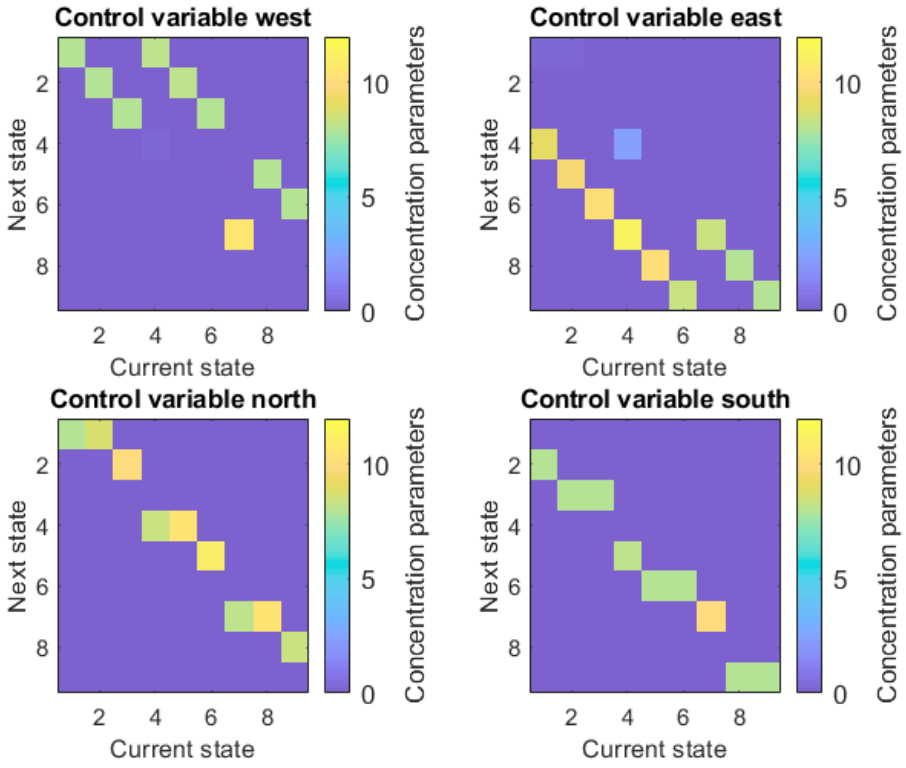


(b) For the constant update method.

Figure 5.4: Heatmap of the transition concentration parameters at the end of trial 128 for map1. Each subfigure shows to results for one of the four update methods, and within each subfigure, the concentration parameters for each control variable are visualized.



(c) For the LSTM update method.



(d) For the rolling update method.

Figure 5.4: Heatmap of the transition concentration parameters at the end of trial 128 for map1. Each subfigure shows to results for one of the four update methods, and within each subfigure, the concentration parameters for each control variable are visualized.

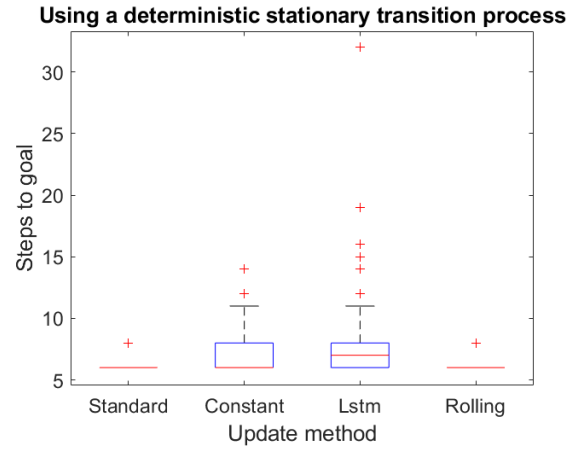
Table 5.3: Results for map2 after simulating 8 episodes of 128 trials, with in subtable (a) a deterministic stationary transition process, and (b) a cyclostationary transition process.

(a) Using a deterministic stationary transition process.

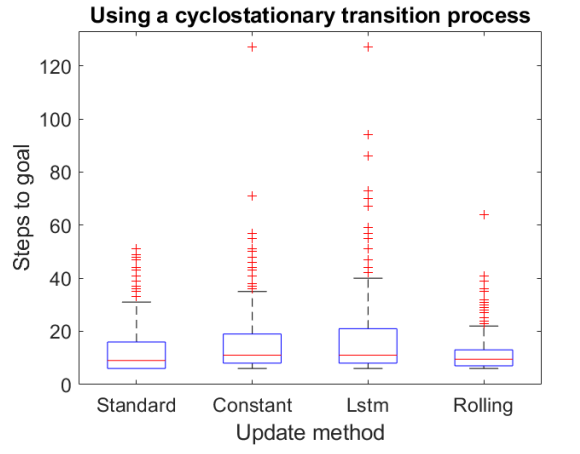
Update method	Steps to goal	Sample size	p-value
Standard	6, [6, 6]	512	-
Constant	6, [6, 8]	500	1.9876×10^{-20}
LSTM	7, [6, 8]	452	4.2978×10^{-51}
Rolling	6, [6, 6]	512	0.0046

(b) Using a cyclostationary transition process.

Update method	Steps to goal	Sample size	p-value
Standard	9, [6, 16]	293	-
Constant	11, [8, 19]	226	2.5730×10^{-5}
LSTM	11, [8, 21]	190	1.9477×10^{-6}
Rolling	9.5, [7, 13]	306	0.4210



(a)



(b)

Figure 5.5: Boxplot showing the results for map2, with in subfigure (a) the results for a deterministic stationary transition process, and in (b) the results for a cyclostationary transition process.

5.3. Tuning results

This final section of chapter 5 will provide tuning results obtained by repeating the experiments conducted, but now with different values for the tunable parameters. As mentioned in subsection 4.3.2, the tunable parameters that will be tested are the length of the policies, which will be set to 3 (instead of 4); and the window width, for which values of 4 and 16 will be tested (instead of 8).

5.3.1. Policy length

Despite the fact that map1 and map2 are very similar, a drastic difference in performance was observed between the results from the previous two sections. This has to do with the reachability of the goal: in map1 the goal can be reached in one trial, whereas in map2 this can only be done in a minimum of two trials. Therefore, changing the policy length can give more insight in the performance.

Ideally, it is desired to obtain results about the performance when the agent is able to reach the goal in one trial for map2 as well. However, this would require a policy length of 6, considering that the shortest path from start to goal equals 6 cells. Ultimately, experiments using a policy length of 6 have not been performed: considering the fact that the computation time per trial increases exponentially when increasing the policy length, it would be practically infeasible to obtain enough data.

Even though the influence of a larger policy length will not be tested here, testing the influence of a smaller policy length is still possible. Therefore, for the following experiments, the length of the policies is set to 3 instead of 4. This means that the agent is now unable to reach the goal in one trial for map1. Also for map2, the agent now must take one of the multiple shortest paths in order to reach the goal in two trials. Table 5.4 shows the results for map1, whereas table 5.5 shows the results for map2.

When looking at table 5.4, the results are mostly similar to the main results of map2 instead of map1, keeping in mind that it is now impossible for the agent to reach the goal in one trial. The constant and LSTM update methods still perform the worst, and the rolling update method gives the best performance, although the difference with the standard update method is small.

The results in table 5.5 are also similar to the main results of map2. The constant and LSTM update methods perform the worst, whereas the rolling update method performs only somewhat better than the standard update method. An important observation here is that, even though the policy length is decreased by 1 compared to the main results, the agent does not need more steps to reach the goal for the standard and rolling update methods. The reason for this is that the agent now does not have an extra step to afford per trial. For the main results of map2 this was the case, and this allowed the agent to make a step not in the direction towards the goal, while still being able to reach the goal in the minimum amount of trials (2). For the results here, the agent must take all steps in the direction towards the goal to be able to reach the goal in the minimum amount of trials.

5.3.2. Window width

Since the rolling update scheme performed best overall when going by the previous results, with the following experiments, the influence of the window width of the rolling update scheme will be tested. The influence of both a smaller window width ($w = 4$) and a larger window width ($w = 16$) will be investigated. Table 5.6 shows the results of the experiments with the agents using the modified window widths for map1, whereas table 5.7 shows those for map2.

As can be seen from table 5.6, regardless of the size the window width, the agent using the rolling update scheme always significantly outperforms the agent using the standard update scheme. However, differences do exist within the results of the rolling update scheme as well. When the agent uses a window width that is twice as large, the agent requires almost one extra step (an increase of approximately 10%) on average to reach its goal. Conversely, when the agent uses a window width that is twice as small, no notable difference in the performance can be observed.

Just like was the case for the main results in section 5.2, the results between the different update methods for map2 are largely similar. Using the rolling update method, regardless of the window width seems to result in the best performance, but the difference with the standard update method is still not statistically significant, as is seen from table 5.7.

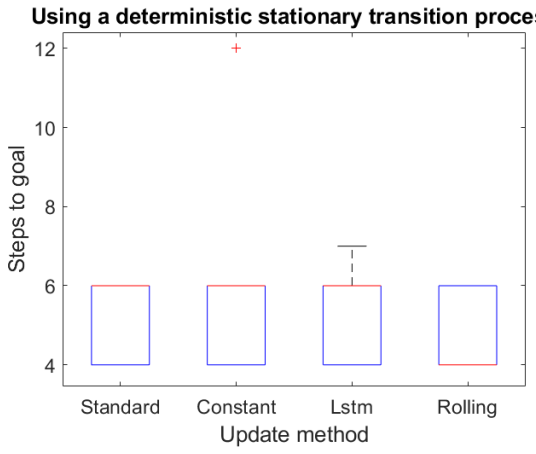
Table 5.4: Results for map1 after simulating 8 episodes of 128 trials with the policy length set to 3. Subtable (a) shows the results for a deterministic stationary transition process, and (b) the results for a cyclostationary transition process.

(a) Using a deterministic stationary transition process.

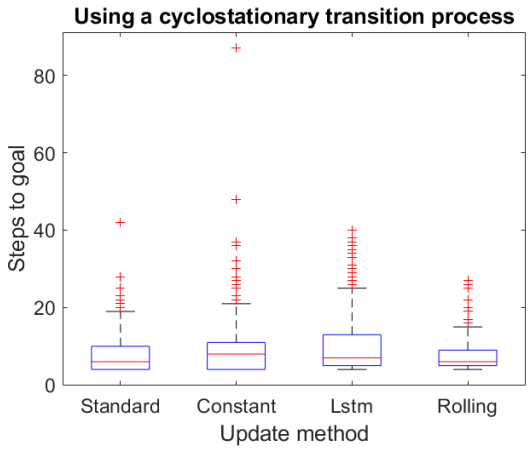
Update method	Steps to goal	Sample size	p-value
Standard	6, [4, 6]	512	-
Constant	6, [4, 6]	508	0.1937
LSTM	6, [4, 6]	509	0.6715
Rolling	4, [4, 6]	512	1.0657×10^{-7}

(b) Using a cyclostationary transition process.

Update method	Steps to goal	Sample size	p-value
Standard	6, [4, 10]	328	-
Constant	8, [4, 11]	270	0.0052
LSTM	7, [5, 13]	261	0.0013
Rolling	6, [5, 9]	350	0.9148



(a)



(b)

Figure 5.6: Boxplot showing the results for map1, with in subfigure (a) the results for a deterministic stationary transition process, and in (b) the results for a cyclostationary transition process, when the policy length is set to 3.

Table 5.5: Results for map2 after simulating 8 episodes of 128 trials with the policy length set to 3. Subtable (a) shows the results for a deterministic stationary transition process, and (b) the results for a cyclostationary transition process.

(a) Using a deterministic stationary transition process.

Update method	Steps to goal	Sample size	p-value
Standard	6, [6, 6]	500	-
Constant	6, [6, 8]	361	2.8894×10^{-40}
LSTM	6, [6, 9]	351	4.4699×10^{-50}
Rolling	6, [6, 6]	502	0.9880

(b) Using a cyclostationary transition process.

Update method	Steps to goal	Sample size	p-value
Standard	9, [7, 13]	242	-
Constant	10, [7, 18]	172	0.0213
LSTM	11, [7, 19.25]	149	3.5120×10^{-4}
Rolling	9, [6.25, 13]	255	0.4482

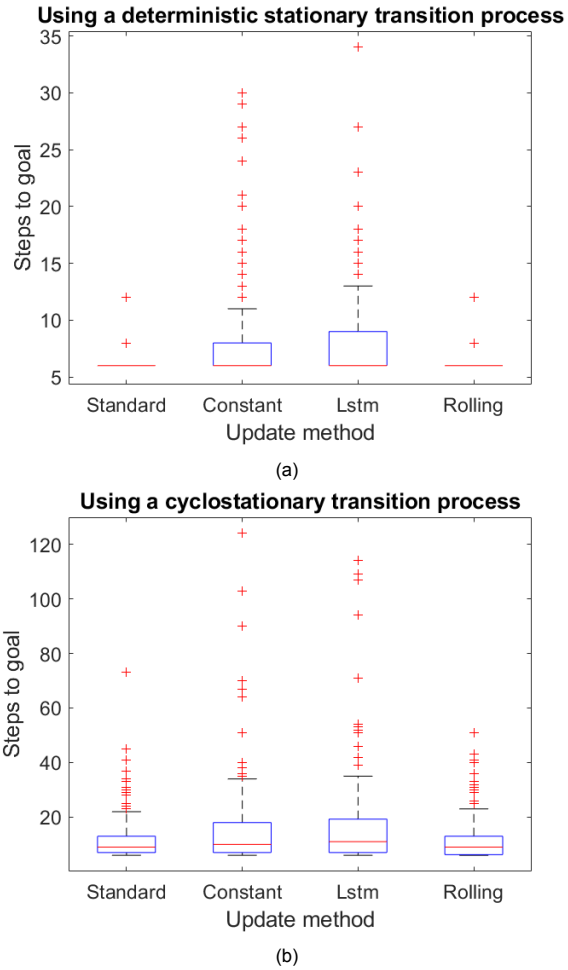


Figure 5.7: Boxplot showing the results for map2, with in subfigure (a) the results for a deterministic stationary transition process, and in (b) the results for a cyclostationary transition process, when the policy length is set to 3.

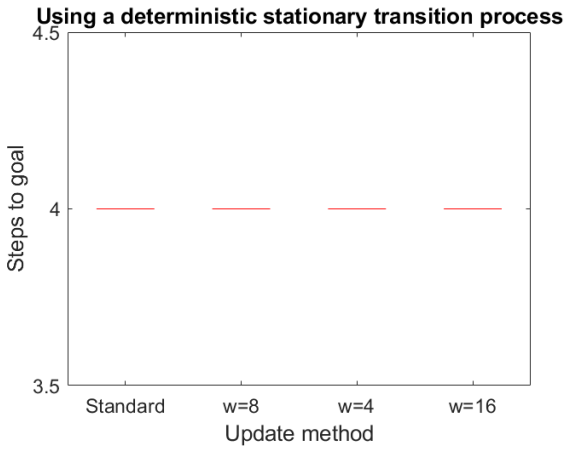
Table 5.6: Results for map1 after simulating 8 episodes of 128 trials when determining the influence of the window width of the rolling update method. Subtable (a) shows the results for a deterministic stationary transition process, and (b) the results for a cyclostationary transition process. The results of the standard and rolling update methods are the same results from section 5.1 and serve only as a comparison.

(a) Using a deterministic stationary transition process.

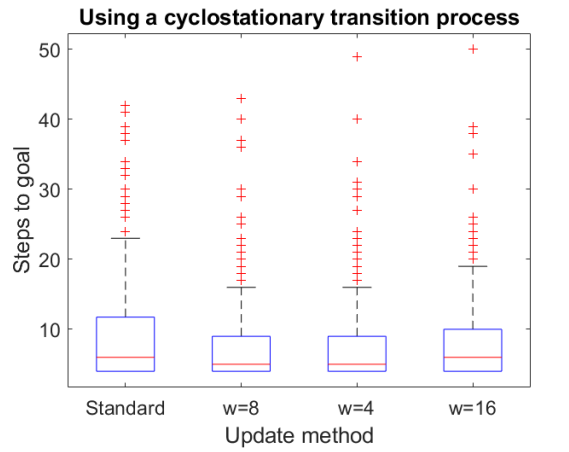
Update method	Steps to goal	Sample size	p-value
Standard	4, [4, 4]	1024	-
Rolling, $w = 8$	4, [4, 4]	1024	-
Rolling, $w = 4$	4, [4, 4]	1024	-
Rolling, $w = 16$	4, [4, 4]	1024	-

(b) Using a cyclostationary transition process.

Update method	Steps to goal	Sample size	p-value
Standard	6, [4, 11.75]	383	-
Rolling, $w = 8$	5, [4, 9]	475	1.8268×10^{-4}
Rolling, $w = 4$	5, [4, 9]	479	0.0027
Rolling, $w = 16$	6, [4, 10]	434	0.0858



(a)



(b)

Figure 5.8: Boxplot showing the results for map1, with in subfigure (a) the results for a deterministic stationary transition process, and in (b) the results for a cyclostationary transition process, when the window width of the rolling update method is altered.

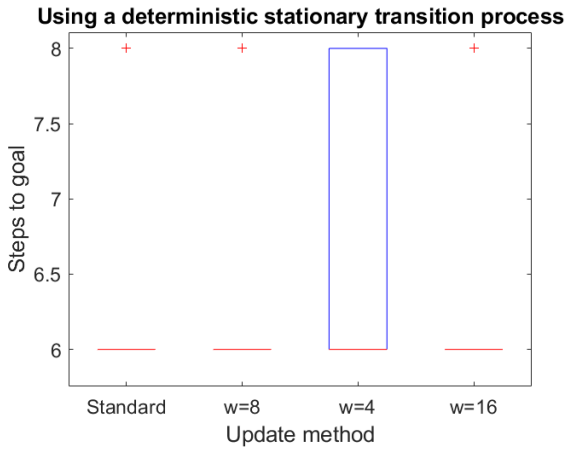
Table 5.7: Results for map2 after simulating 8 episodes of 128 trials when determining the influence of the window width of the rolling update method. Subtable (a) shows the results for a deterministic stationary transition process, and (b) the results for a cyclostationary transition process. The results of the standard and rolling update methods are the same results from section 5.2 and serve only as a comparison.

(a) Using a deterministic stationary transition process.

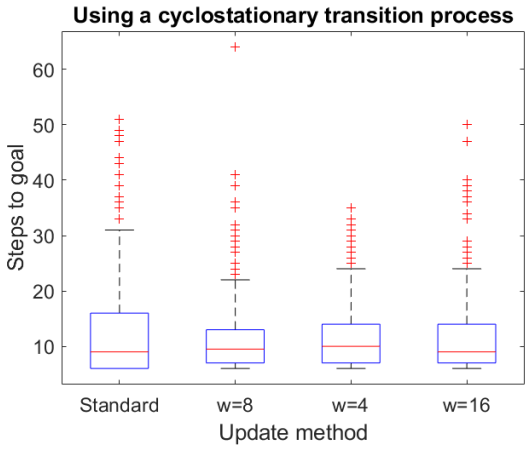
Update method	Steps to goal	Sample size	p-value
Standard	6, [6, 6]	512	-
Rolling, $w = 8$	6, [6, 6]	512	0.0046
Rolling, $w = 4$	6, [6, 8]	512	2.3627×10^{-13}
Rolling, $w = 16$	6, [6, 6]	512	2.5355×10^{-4}

(b) Using a cyclostationary transition process.

Update method	Steps to goal	Sample size	p-value
Standard	9, [6, 16]	293	-
Rolling, $w = 8$	9.5, [7, 13]	306	0.4210
Rolling, $w = 4$	10, [7, 14]	317	0.2821
Rolling, $w = 16$	9, [7, 14]	300	0.3223



(a)



(b)

Figure 5.9: Boxplot showing the results for map2, with in subfigure (a) the results for a deterministic stationary transition process, and in (b) the results for a cyclostationary transition process, when the window width of the rolling update method is altered.

6

Discussion

This chapter will go deeper into the performance of the different methods, first and foremost by giving a thorough analysis of the results obtained in the previous chapter. The final two sections in this chapter will also discuss potential improvements for the methods proposed in this research, by discussing possible additions to the proposed methods and alternatives to these methods respectively.

6.1. Analysis of the results

6.1.1. How changes in the transition model influence the agent's performance

As is evident from the results, the way how the transition model gets updated has a large influence on the performance of the agent. The performance of the agent is directly related to the policy selection of the agent, since this is the way the agent can exercise influence on the environment. Recalling from section 2.5, the agent's policy selection is a function of the expected free energy (equation 2.19):

$$P(\pi) = \pi_0 = \sigma(-\gamma \cdot G(\pi)) \tag{6.1}$$

But the agent's transition model does not appear in the expected free energy equation (equation 2.17):

$$G(\pi, \tau) = \mathbf{o}_\tau^\pi \cdot (\hat{\mathbf{o}}_\tau^\pi - U) + \mathbf{s}_\tau^\pi \cdot H \tag{6.2}$$

To answer the question of how the transition model influences the agent's policy selection, equation 6.2 has to be further analyzed. From this equation can be seen the expected outcomes under policies $\mathbf{o}_\tau^\pi$. Since the likelihood matrix $\mathbf{A}$ is a mapping from states to outcomes, as explained in section 2.4, these expected outcomes are obtained by multiplying the likelihood matrix $\mathbf{A}$ by the expected states under policies $\mathbf{s}_\tau^\pi$:

$$\mathbf{o}_\tau^\pi = \mathbf{A}\mathbf{s}_\tau^\pi \tag{6.3}$$

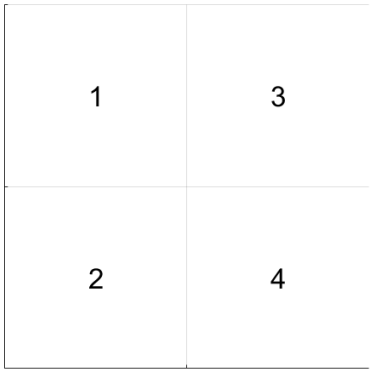
On their turn, these expected states under policies at a certain timestep $\mathbf{s}_\tau^\pi$ are obtained by multiplying the transition matrix $\mathbf{B}$ by the expected states under policies at the previous timestep $\mathbf{s}_{\tau-1}^\pi$, which is where the transition matrix comes into play:

$$\mathbf{s}_\tau^\pi = \mathbf{B}_{\tau-1}^\pi \mathbf{s}_{\tau-1}^\pi \tag{6.4}$$

Thus, even though the transition model does not appear explicitly in the expected free energy equation 6.2, the expected free energy is still an indirect function of the transition model, as becomes evident by substituting equations 6.3 and 6.4 into equation 6.2:

$$\begin{aligned} G(\pi, \tau) &= \mathbf{o}_\tau^\pi \cdot (\hat{\mathbf{o}}_\tau^\pi - U) + \mathbf{s}_\tau^\pi \cdot H \\ &= \mathbf{A}\mathbf{s}_\tau^\pi \cdot (\ln(\mathbf{A}\mathbf{s}_\tau^\pi) - U) + \mathbf{s}_\tau^\pi \cdot H \\ &= \mathbf{A}\mathbf{B}_{\tau-1}^\pi \mathbf{s}_{\tau-1}^\pi \cdot (\ln(\mathbf{A}\mathbf{B}_{\tau-1}^\pi \mathbf{s}_{\tau-1}^\pi) - U) + \mathbf{B}_{\tau-1}^\pi \mathbf{s}_{\tau-1}^\pi \cdot H \end{aligned} \tag{6.5}$$

The example in figure 6.1 then shows how having a better model of the transition process leads to the selection of different policies.



(a) Map of the environment.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 \\ -5 \\ 10 \\ 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

(b) Likelihood matrix, utility vector and initial state vector, which are identical for both agents.

$$B(east) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \qquad B(north) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{aligned} G(ee) = {} & AB(east)\mathbf{s}_1^\pi \cdot (\ln(AB(east)\mathbf{s}_1^\pi) - U) + B(east)\mathbf{s}_1^\pi \cdot H \\ & + AB(east)\mathbf{s}_2^\pi \cdot (\ln(AB(east)\mathbf{s}_2^\pi) - U) + B(east)\mathbf{s}_2^\pi \cdot H = 0 \end{aligned}$$

$$\begin{aligned} G(en) = {} & AB(east)\mathbf{s}_1^\pi \cdot (\ln(AB(east)\mathbf{s}_1^\pi) - U) + B(east)\mathbf{s}_1^\pi \cdot H \\ & + AB(north)\mathbf{s}_2^\pi \cdot (\ln(AB(north)\mathbf{s}_2^\pi) - U) + B(north)\mathbf{s}_2^\pi \cdot H = -10 \end{aligned}$$

$$\begin{aligned} G(ne) = {} & AB(north)\mathbf{s}_1^\pi \cdot (\ln(AB(north)\mathbf{s}_1^\pi) - U) + B(north)\mathbf{s}_1^\pi \cdot H \\ & + AB(east)\mathbf{s}_2^\pi \cdot (\ln(AB(east)\mathbf{s}_2^\pi) - U) + B(east)\mathbf{s}_2^\pi \cdot H = -10 \end{aligned}$$

$$\begin{aligned} G(nn) = {} & AB(north)\mathbf{s}_1^\pi \cdot (\ln(AB(north)\mathbf{s}_1^\pi) - U) + B(north)\mathbf{s}_1^\pi \cdot H \\ & + AB(north)\mathbf{s}_2^\pi \cdot (\ln(AB(north)\mathbf{s}_2^\pi) - U) + B(north)\mathbf{s}_2^\pi \cdot H = 0 \end{aligned}$$

$$\pi_0 = \sigma\left(-1 \cdot \begin{bmatrix} 0; & -10; & -10; & 0 \end{bmatrix}\right) = \begin{bmatrix} 0; & 0.5; & 0.5; & 0.5 \end{bmatrix}$$

(c) The agent that has a model of the world in which no wind perturbation exists.

$$B(east) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \qquad B(north) = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.5 & 0 & 1 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \end{bmatrix}$$

$$G(ee) = 0$$

$$G(en) = -5.6931$$

$$G(ne) = -6.3863$$

$$G(nn) = -6.7329$$

$$\pi_0 = \sigma\left(-1 \cdot \begin{bmatrix} 0; & -5.6931; & -6.3863; & -6.7329 \end{bmatrix}\right) = \begin{bmatrix} 0.0006; & 0.1715; & 0.3429; & 0.4850 \end{bmatrix}$$

(d) The agent that has a model of the world in which the wind faces to the east.

Figure 6.1: Worked out example of how having a better model of the transition process leads to the selection of different policies. Consider the scenario in which the agent is perturbed by a wind that faces to the east. For this example two agents are compared: (1) an agent that has a model of the world in which no wind perturbation exists (incorrect), and (2) an agent that has a model of the world in which the wind faces to the east (correct). Both agents have the same observation model, utility vector (the agent prefers to be in state 3) and initial state vector (the agent beliefs it starts in state 2). For simplicity, the agent only has two control variables at its disposal: $\mathcal{U} = \{east, north\}$. Also, the length of the policy is set to 2, i.e. the agent considers only two consecutive actions. Thus, the possible policies are ee $(east, east)$, en, ne and nn $(north, north)$. After setting the precision over policies for simplicity to 1 and calculating the expected free energy for all policies for both agents, the policy distribution π_0 then shows that the agent with the better model is more likely to move to the north first, since this way the agent is less likely to get stuck along the eastern edge of the map due to the wind perturbation.

6.1.2. Why the addition of a constant forget rate leads to a decrease in performance

The use of a constant forget rate is not effective in active inference and actually decreases performance, as is evident from from every table in the previous chapter. A constant forget rate causes the concentration parameters of state transitions that are not frequently occuring, to approach zero as the number of trials grows large. To prove this, equation 3.8 is first rewritten in a recursive manner:

$$\begin{aligned} b_k &= (1-\alpha_F)\cdot b_{k-1} + \mathscr{b}_k \\ &= \alpha_R \cdot b_{k-1} + \mathscr{b}_k \\ &= \alpha_R \cdot (\alpha_R \cdot b_{k-2} + \mathscr{b}_{k-1}) + \mathscr{b}_k \\ &= (\alpha_R)^2 \cdot b_{k-2} + \alpha_R \cdot \mathscr{b}_{k-1} + \mathscr{b}_k \\ &= (\alpha_R)^2 \cdot (\alpha_R \cdot b_{k-3} + \mathscr{b}_{k-2}) + \alpha_R \cdot \mathscr{b}_{k-1} + \mathscr{b}_k \\ &= (\alpha_R)^3 \cdot b_{k-3} + (\alpha_R)^2 \cdot \mathscr{b}_{k-2} + \alpha_R \cdot \mathscr{b}_{k-1} + \mathscr{b}_k \\ &= (\alpha_R)^\infty \cdot b_{k-\infty} + \sum_{\kappa=0}^{\infty-1} (\alpha_R)^\kappa \cdot \mathscr{b}_{k-\kappa} \\ &= \sum_{\kappa=0}^{\infty} (\alpha_R)^\kappa \cdot \mathscr{b}_{k-\kappa} \end{aligned} \tag{6.6}$$

where $\alpha_R = 1 - \alpha_F$ (which can be interpreted as the remember rate). The final expression of equation 6.6 represents a geometric series

$$\sum_{\kappa=0}^{\infty} ar^\kappa$$

with $a = \mathscr{b}_{k-\kappa}$ and $r = \alpha_R$. If $|r| < 1$ the geometric series converges:

$$\sum_{\kappa=0}^{\infty} ar^\kappa = \frac{a}{1-r} \tag{6.7}$$

Since $|\alpha_R| < 1$, this is translated to:

$$\sum_{\kappa=0}^{\infty} (\alpha_R)^\kappa \cdot \mathscr{b}_{k-\kappa} = \frac{\mathscr{b}_{k-\kappa}}{1-\alpha_R} = \frac{\mathscr{b}_{k-\kappa}}{\alpha_F} \tag{6.8}$$

The a in equation 6.7 is a constant, but since the value of the experience $\mathscr{b}_{k-\kappa}$ in equation 6.8 is dependent on the trial, the average of the experience per trial is considered. Since the experience gained per trial for the state transitions that are not frequently occuring is zero for most trials, the average experience per trial is a very small number. As such, the converged value for the concentration parameters becomes this small number scaled by the forget rate. This converged value for the concentration parameters decreases as the forget rate increases and vice versa, which is as expected.

The decay of the concentration parameters is especially problematic when the state transition probabilities are not unimodal (i.e. when selecting a specific action in a specific state does not result in a high chance of transitioning to only one other specific state). This is because, in the case that a not frequently ocurring state transition does occur, the agent will believe that its recently experienced state transition has a much higher chance of occuring than it has in reality (since the other state transitions have decayed to a low value).

It is not expected that the modification of the exact value of the forget rate will lead to an increase in performance. Increasing this value will only result in the concentration parameters to approach zero more quickly, as is evident from equation 6.8. Conversely, decreasing this value will make the impact of forgetting between two consecutive trials smaller, which means that the actual purpose of forgetting gets lost, while still contributing to the decay of concentration parameters to zero in the long run.

6.1.3. Why forgetting with the LSTM update method decreases performance

As was seen from the results in the previous chapter, forgetting by utilizing the update mechanism of an LSTM cell substantially decreases performance. Since the agent only obtains concentration parameters for the state transitions experienced during the trial, this means that per trial, for the majority of state transitions no experience will be gained. For the LSTM implementation in this work, this is equivalent to the input for the LSTM cell being zero. The following paragraph will mathematically show how the cell state will change in this case.

As was done in the experiments, all the weight and bias matrices are set to identity and zero respectively. Therefore, to improve readability, the symbols representing these matrices will be omitted in the following equations. The update equation for the cell state as given in subsection 3.3.2 can then be written as:

$$\begin{aligned} c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\ &= \sigma_l(h_{t-1}) \circ c_{t-1} + \sigma_l(h_{t-1}) \circ \sigma_h(h_{t-1}) \end{aligned} \tag{6.9}$$

Now, by writing $\sigma_h(x) = \tanh(x)$ and rewriting the logistic function as a function of the hyperbolic tangent as well,

$$\sigma_l(x) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{x}{2}\right)$$

equation 6.9 can be written as:

$$\begin{aligned} c_t &= \sigma_l(h_{t-1}) \circ c_{t-1} + \sigma_l(h_{t-1}) \circ \sigma_h(h_{t-1}) \\ &= \left(\frac{1}{2} + \frac{1}{2}\tanh\left(\frac{h_{t-1}}{2}\right)\right) \circ c_{t-1} + \left(\frac{1}{2} + \frac{1}{2}\tanh\left(\frac{h_{t-1}}{2}\right)\right) \circ \tanh(h_{t-1}) \end{aligned} \tag{6.10}$$

After substituting

$$P_{t-1} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{h_{t-1}}{2}\right)$$

and writing recursively:

$$\begin{aligned} c_t &= (P_{t-1}) \circ c_{t-1} + (P_{t-1}) \circ \tanh(h_{t-1}) \\ &= (P_{t-1}) \circ ((P_{t-2}) \circ c_{t-2} + (P_{t-2}) \circ \tanh(h_{t-2})) + (P_{t-1}) \circ \tanh(h_{t-1}) \\ &= (P_{t-1}) \circ (P_{t-2}) \circ c_{t-2} + (P_{t-1}) \circ (P_{t-2}) \circ \tanh(h_{t-2}) + (P_{t-1}) \circ \tanh(h_{t-1}) \\ &= (P_{t-1}) \circ (P_{t-2}) \circ ((P_{t-3}) \circ c_{t-3} + (P_{t-3}) \circ \tanh(h_{t-3})) + (P_{t-1}) \circ (P_{t-2}) \circ \tanh(h_{t-2}) \\ &\quad + (P_{t-1}) \circ \tanh(h_{t-1}) \\ &= (P_{t-1}) \circ (P_{t-2}) \circ (P_{t-3}) \circ c_{t-3} + (P_{t-1}) \circ (P_{t-2}) \circ (P_{t-3}) \circ \tanh(h_{t-3}) \\ &\quad + (P_{t-1}) \circ (P_{t-2}) \circ \tanh(h_{t-2}) + (P_{t-1}) \circ \tanh(h_{t-1}) \\ &= \prod_{i=1}^{\infty} P_{t-i} \circ c_{t-\infty} + \prod_{i=1}^{\infty} P_{t-i} \circ \tanh(h_{t-\infty}) + \sum_{j=1}^{\infty}\left(\prod_{i=1}^{\infty} P_{t-i} \circ \tanh\left(h_{t-j}\right)\right) \end{aligned} \tag{6.11}$$

From the fact that

$$P_{t-i} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{h_{t-i}}{2}\right) < 1$$

the infinite products in the first two terms of the final equality in equation 6.11 converge to zero. To see whether the final term in this equation converges, consider the case that P_{t-i} is not dependent on time, in which case the final term

$$\sum_{j=1}^{\infty}\left(\prod_{i=1}^{\infty} P_{t-i} \circ \tanh\left(h_{t-j}\right)\right)$$

can be rewritten as

$$\sum_{j=1}^{\infty}\left(P^j \circ \tanh\left(h_{t-j}\right)\right)$$

which is a geometric series. Even though P_{t-i} is dependent on time, the series will still converge, since it always holds that $|P_{t-i}| < 1$.

Thus, similar to the constant update method, the LSTM update method causes the concentration parameters of state transitions that are not frequently occurring, to converge to a small value. However, unlike the constant update method, with the LSTM update method this does not exclusively happen when the number of the trial grows large. As is seen from the multiple instances of sigmoid functions in the LSTM cell update equations, the signals in the LSTM cell tend to become normalized. As such, when larger values for the initial concentration parameters are set as the initial cell state, these values will decrease rapidly. This can be illustrated with the following example. First, the equation for the hidden state is rewritten, again by setting the input to zero, and the weight and bias matrices to identity and zero respectively:

$$\begin{aligned} h_{t-1} &= \sigma_h(c_{t-1}) \circ o_{t-1} \\ &= \sigma_h(c_{t-1}) \circ \sigma_l(h_{t-2}) \\ &= \tanh(c_{t-1}) \circ \left(\frac{1}{2} + \frac{1}{2}\tanh\left(\frac{h_{t-2}}{2}\right)\right) \end{aligned} \tag{6.12}$$

In the most extreme case of c_{t-1} and h_{t-2} both being ∞, equation 6.12 shows that h_{t-1} becomes 1. Using this result allows one to reapply equation 6.12 with a more reasonable value of $h_{t-2} = 1$. When the value for c_{t-1} remains large, it is then calculated from equation 6.12 that $h_{t-1} = 0.7311$. Subsequently applying equation 6.10 with the value of $h_{t-1} = 0.7311$, results in $c_t = 0.6750c_{t-1} + 0.4210$. This result gives that c_t will decrease in value compared to c_{t-1} unless $c_{t-1} < 1.295$ (this threshold is even lower when a less optimistic value for h_{t-2} is used). Considering that the initial nonzero concentration parameters are set to 8 for the experiments, this means that after just one trial, the concentration parameters for state transitions that are not experienced during the first trial will have dropped to at most $0.6750 \cdot 8 + 0.4210 = 5.821$, which is lower than the $(1 - 0.1) \cdot 8 = 7.2$ value for the constant update method with the constant forget rate set to 0.1. This shows that the LSTM update method is capable of forgetting much faster than the constant update method, although this way of forgetting is not beneficial to the agent's performance.

6.1.4. Possible caveats for the rolling update method

The rolling update method performs the best overall, as it substantially increases performance over the standard update method in the best case scenario, and does not decrease performance in the worst case scenario. However, it should be emphasized that the initial concentration parameters play a large role in the performance of the rolling update method. The role of the initial concentration parameters is of more importance for the rolling update method than for the standard update method, because with the rolling update method, the agent's concentration parameters of certain state transitions get reset to their initial value when the agent did not gain experience about those state transitions within its memory window, as is seen from 3.15. In tasks like the one in this research, this does not cause any problems, because the designer can easily feed the agent with neutral state transition concentration parameters, that is, how the state transitons would be in absence of nonstationarity (e.g. selecting the action to the west always results in the state transition to the west, the action to the east always results in the state transition to the east, etc.). But providing the agent with neutral concentration parameters might not always be that easy in other types of tasks.

Another point of interest, as is supported by the tuning results in section 5.3, the width of the memory window for the rolling update method can further increase or decrease the performance. Generally, increasing the window width will decrease performance, since the agent will use the concentration parameters of more trials, which means that there is a greater chance that the agent will use outdated concentration parameters. Similarly, decreasing the window width will increase performance. However, it is expected that performance will cease to increase when the window width is set too small, considering that the agent will exclusively use its initial concentration parameters as the window width approaches zero. This means that the agent will be unable to learn anything, which can result in a decrease in performance.

6.1.5. Why an increase in performance for the proposed methods is not present when the goal is not reachable in one trial

Attaining an increase in performance for bigger maps might be a problem, since the agent just starts wandering around when the distance between the start and finish is too large. A similar issue already arises when the agent is not able to reach the goal within one trial. In this case, the agent tries to move towards the center, because this leaves the agent with the most options. The agent does this, because it prioritizes keeping its options open when prior preferences are not specified [11]. This is essentially the case when the agent cannot reach the goal within the trial, because all other outcomes will have equal utility (which is equivalent to no prior preferences specified). This explains why the differences in performance between the standard and new update methods in the cyclostationary environment for map2 are not as striking as they are for map1. The same holds for the performance difference for map1 between agents using a policy length of 4 and agents using a policy length of 3.

Can longer policy lengths solve this problem? Longer policy lengths have an advantage over shorter lengths in that these allow the agent to take more distant goals into consideration. However, longer policy lengths also have an important drawback, besides larger computation times. The drawback is that within the trial that policy is followed, there is a larger chance for errors to accumulate in the predicting of the sequence of state transitions that come with that policy, because the trial/policy is longer. This is easier to expose with a real life example, like predicting the weather: with just the measurements of today, it is easier to correctly predict the weather of only tomorrow than to correctly predict the weather of the next three days. This problem is of less importance when the agent is equipped with an accurate observation model (as was the case in this work), since the agent makes intermediate observations, but this is not always the case.

A better approach would be to tackle this problem at a more fundamental level. Since the problem is fundamentally caused by the agent having equivalent prior preferences for all outcomes when the goal is not reachable, a possible solution would be to modify the prior preferences at each trial. This can for example be done by introducing subgoals for the agent.

6.2. Possible additions to forgetting in active inference

6.2.1. Implementing subgoals

As is evident from the results in the previous chapter, applying forgetting to active inference can increase performance when the goal is reachable in one trial, but does not affect performance otherwise. As mentioned in the previous section, introducing subgoals for the agent can potentially solve this problem. When applied to the type of task in this research, the subgoals would tell the agent at any trial to move to a cell that is reachable within that trial. Using subgoals would allow the agent to use smaller policy horizons as well, which is beneficial to the computation time. The use of smaller policy horizons and subgoals is not new, as this has already been implemented in maze planning tasks, albeit only in stationary environments [18]. Combining the concepts of subgoals and forgetting might make the agent able to perform in nonstationary environments with larger state-spaces.

6.2.2. Making forgetting a function of the model parameters

As is the case with the constant forget rate and the memory window, these methods implement forgetting as a function of time, independent of what happens during the interaction between agent and environment. A more refined way of forgetting would be to take this interaction into account, and determine what and how much to forget, based on the difference between the agent's existing transition model and the experience the agent gained from its most recent trial. This way, forgetting can become a function of the agent's model parameters instead of a direct function of time. Currently, this concept is already present in the LSTM update method, but as seen from the update equations for an LSTM cell, the cell state has the tendency to maintain values between 0 and 1. Moreover, because of the complexity of the LSTM update equations, the influence of the difference between the existing transition model and the experience gained from the most recent trial can be made more explicit. This can be done by utilizing a direct similarity measure between these two entities and letting the forgetting be dependent on this similarity measure.

6.2.3. Specific modifications to the methods in this research

Besides adding new components to the concept of forgetting, specific modifications to the methods proposed in this research can possibly increase performance as well:

- The performance of the constant update method can possibly be improved by adding a lower threshold for the concentration parameters. This way, the concentration parameters of infrequently occurring state transitions will not approach zero when the number of trials grows large.
- For the LSTM update method, an increase in performance might be obtained by removing the hyperbolic tangent operators that are applied to obtain the signal $\tilde{c}_t$ and the updated hidden state h_t (see figure 3.1), as this would allow for maintaining values for the cell state larger than 1. For updating the LSTM signals, these hyperbolic tangent operators are used to output values between -1 and 1, but since the implementation of the LSTM cell in this work does not need to output values in this range, these operators are not necessary. In deep learning, the removal of such a function can potentially lead to problems in calculating the gradient used for backpropagation, as the gradient might become too large or small. However, since the implementation of the LSTM cell in this work only makes use of the forward pass of the LSTM calculation process, this situation is not an issue here.
- Currently, for the rolling update method, a rectangular window is used to perform the rolling summation, which causes the agent to completely remember the experience of trials within the window and completely forget the experience of trials outside of the window. By using different types of windows, the performance of the rolling update method might be improved.

6.3. Possible alternatives to forgetting in active inference

6.3.1. Learning prior preferences

At the time of conducting the research in this thesis, research that introduces nonstationarity in discrete state-space active inference has been carried out in parallel by the active inference community [30]. This work similarly focussed on the planning and navigation in gridworlds, but the nonstationarity was manifested in a different way: instead of the transition process changing over time, in [30] the layout of the map itself changed over time (e.g. the location of the goal changed). Besides a goal state, the layouts also included a hole state which the agent had to avoid. Moreover, the way the agents cope with nonstationarity in [30] was by learning prior preferences instead of forgetting part of the concentration parameters. This way, the agent could learn whether a certain state is preferred after reaching it (technically, the agent learns the preferences over outcomes, but since there is a one-to-one mapping from states to outcomes in [30] as well, the result is the same). Because of the difference in the tasks between this research and the research in [30], a rational comparison is not practical. But this shows that different manifestations of nonstationarity might require different solutions. Nevertheless, a similar point of attention is that all the experiments in [30] were done on gridworlds only of size 3×3. This again raises the question of how the results are scaled to larger state-spaces.

6.3.2. Switching between generative models

Performance can potentially be improved by adding a layer to the generative model that represents the different variations the dynamics can take on. When this is done, the agent can switch between generative models, depending on the environmental dynamics of the world. For example, when solving the tasks in this research, the agent would use a different generative model when the wind is facing to the west than when the wind is facing to the east, north or south. An advantage of this approach is that the agent does not lose experience when the environmental dynamics have changed: rather the experience is distributed over the different generative models. A disadvantage of this approach is that it requires the designer to manually partition the variations of the dynamics the world can take on into a finite set. The problem with this is that, in addition to that it would be more computationally expensive, it requires knowledge of the designer about the real world dynamics which is not always available.

An additional issue is that, even when the designer can implement this knowledge, during runtime the agent still has to know (i.e. the designer has to modify the updating scheme so the agent can detect) when there occurs a change in the transition dynamics. This is not a trivial task, considering that the transition dynamics at any trial have the properties of a stochastic process. The agent therefore

essentially has to know, not when there occurs a change in the outcome of a stochastic process, but when there occurs a change between stochastic processes. This task is manageable if the agent is exposed to the dynamics for a sufficiently large amount of trials, but the problem is that due to the nature of nonstationarity, the dynamics might already have changed before this can happen.

A solution to the last problem, while this will not always be possible in practice, this is perhaps the most effective method for dealing with nonstationarity in general: adding a layer of observations to be able to sense the nonstationarity. In practice this would come down to providing the agent with additional sensors. For the tasks in this research, the agent would be able to directly observe the wind direction and switch between generative models accordingly.

6.3.3. Updating the preferences of outcomes, based on the model parameters

The final alternative to forgetting for dealing with nonstationarity discussed here, is to use the model parameters to update the preferences of outcomes. This might sound similar to the method discussed in subsection 6.3.1, but the principle is different. Whereas in subsection 6.3.1 the agent has to observe an outcome to update the preference of that specific outcome, the method suggested here updates the preferences of a selection of outcomes, based on multiple parameters in the transition model. This can be made more clear by translating to the tasks in this research. Consider the situation in map1 (figure 4.2a), where the agent has experienced that taking the action to the east in multiple states has resulted in moving to the north instead. The agent can then lower the preferences of all outcomes that are associated with states to the south of the northern edge of the map. Although this method sounds promising, it requires the agent to have understanding of the spatial relationship between the states in its generative model, which is something that has to be implemented externally (e.g. with a graph Laplacian).

7

Conclusion

This thesis focussed on discrete state-space active inference in nonstationary environments. More specifically, the following research questions have been addressed in this thesis:

1. *How do state-of-the-art active inference agents perform in discrete state-space nonstationary environments?*
2. *Can the performance of discrete state-space active inference agents in nonstationary environments be improved by forgetting part of their previous experiences?*
3. *What are the most effective methods for implementing forgetting in discrete state-space active inference?*

The answer to the first research question is backed up by the following findings: in a planning and navigation task, when traversing a gridworld, a standard active inference agent needs on average approximately twice as many steps to reach its goal when the transition dynamics are changed from stationary deterministic to cyclostationary. As such, an improvement in performance is needed.

To answer the second research question, it turns out that the performance of discrete state-space active inference agents in nonstationary environments can indeed be improved by forgetting part of their previous experiences. When the agent is able to reach its goal within one trial, agents capable of forgetting actually perform better than agents incapable of forgetting (i.e. using the standard update method), if provided with the correct methods of forgetting (more on this in the answering of the third research question). However, a possible increase in performance due to forgetting is not apparent in cyclostationary environments when the agent is not able to reach its goal in one trial. The reason for this is that when the agent is unable to reach its goal in one trial, it is not moved by goal-directed behavior, because all of its reachable outcomes have similar prior preferences.

For answering the third and final research question, it cannot be overemphasized that whether an actual improvement in performance is attained, is largely dependent on how forgetting is implemented. For cyclostationary environments, periodic complete unlearning of the consequences of actions in the past with the use of a memory window provides better results than continuous unlearning by using the updating mechanism of an LSTM cell applied to the updating of the generative model or using a constant forget rate in active inference. These latter two methods even adversely affect performance, as it turns out that these update methods can destabilize the agent's transition model.

As stated before, a possible increase in performance due to forgetting is not apparent in cyclostationary environments when the agent is not able to reach its goal in one trial. That being said, regardless of the reachability of the goal in the environment, in the cases when an increase in performance of agents that forget using a memory window is not apparent, an actual decrease in performance is ruled out according to the results. Moreover, agents provided with a memory window also do not perform worse than agents incapable of forgetting when the transition dynamics are simply stationary.

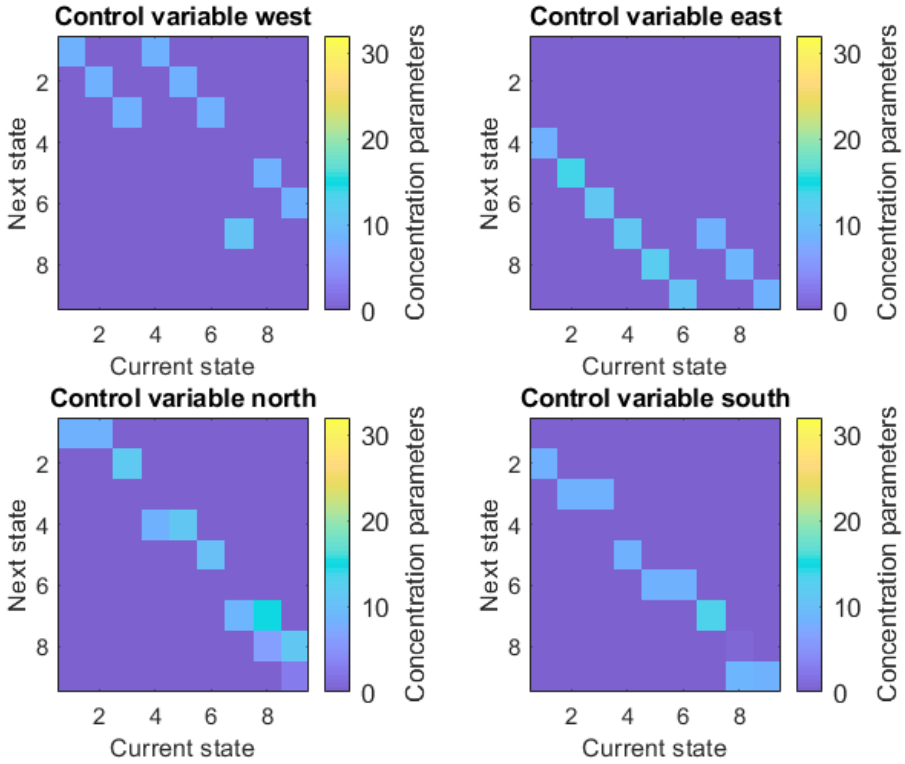
In future work, the performance of agents capable of forgetting can even be improved upon, considering that the forgetting with the memory window (the method currently showing the best results) is a direct function of time. By making the forgetting a function of the agent's model parameters, the agent can have more control of its forgetting.

Altogether, summarizing the results in this research, the active inference agent that continuously forgets by using a constant forget rate and the one that bases its forgetting on the updating mechanism of an LSTM cell unanimously perform worse than the standard active inference agent incapable of forgetting. The agent that uses a memory window to completely forget part of its previous experiences is able to outperform this agent incapable of forgetting. Therefore, the memory window update method can be considered an improvement over the standard update method for the generative model used in active inference.

A

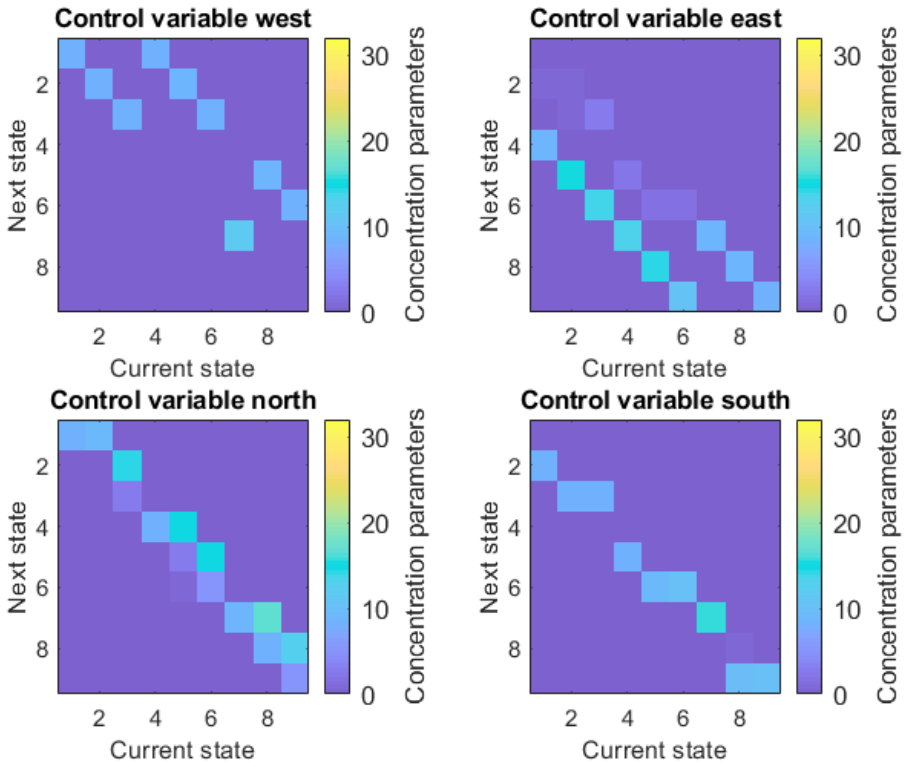
Evolution of the concentration parameters

A.1. The standard update method

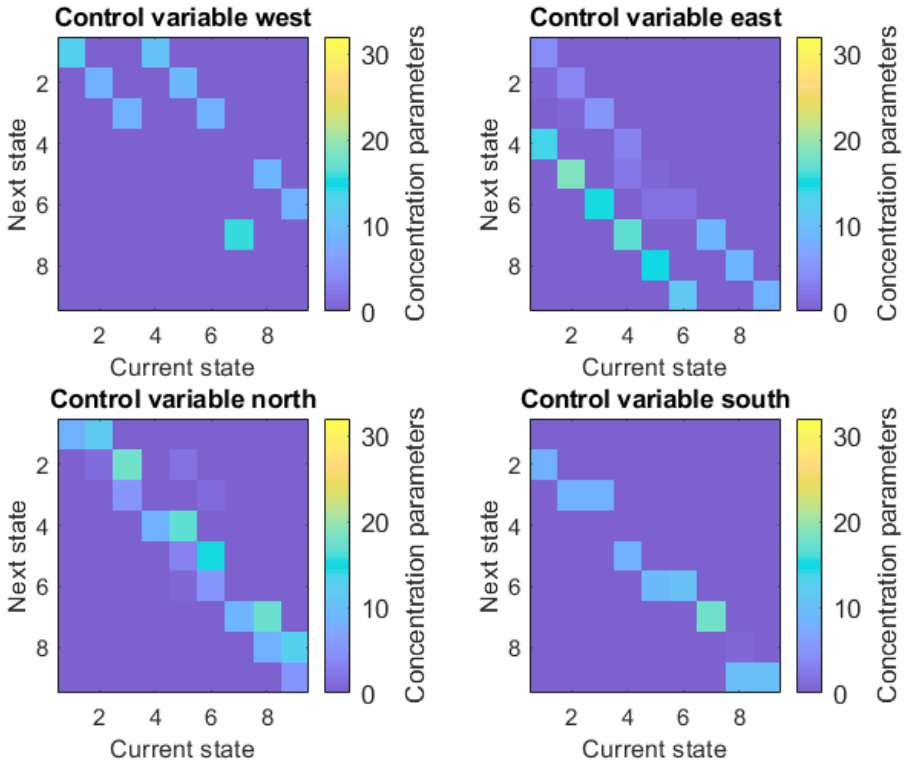


(a) At the end of trial 16.

Figure A.1: Heatmap of the transition concentration parameters for map1 for the standard update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

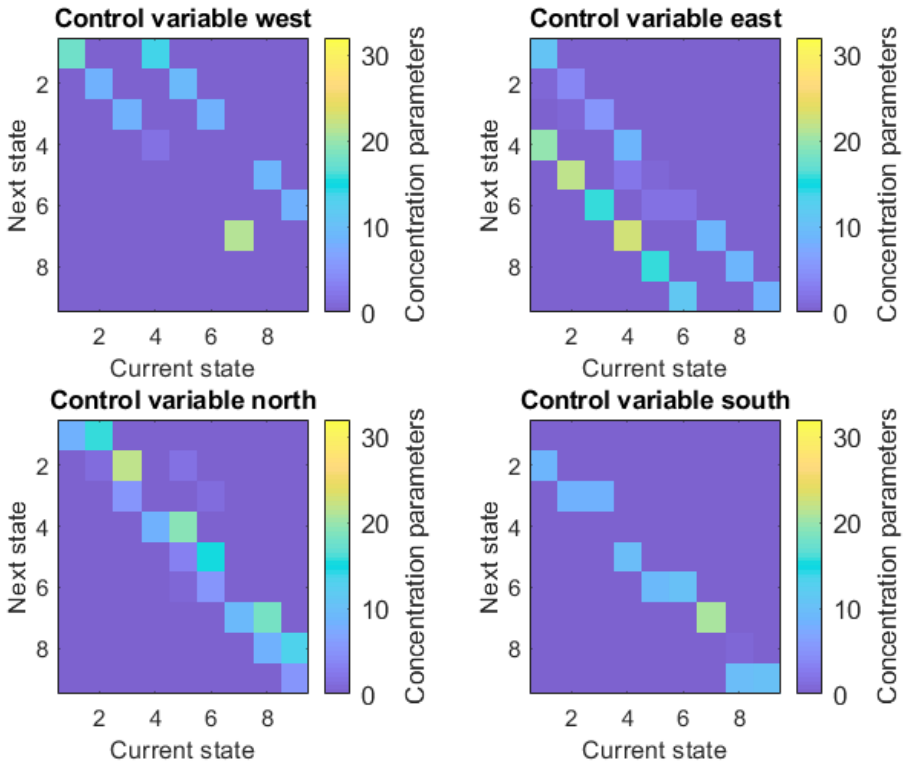


(b) At the end of trial 32.

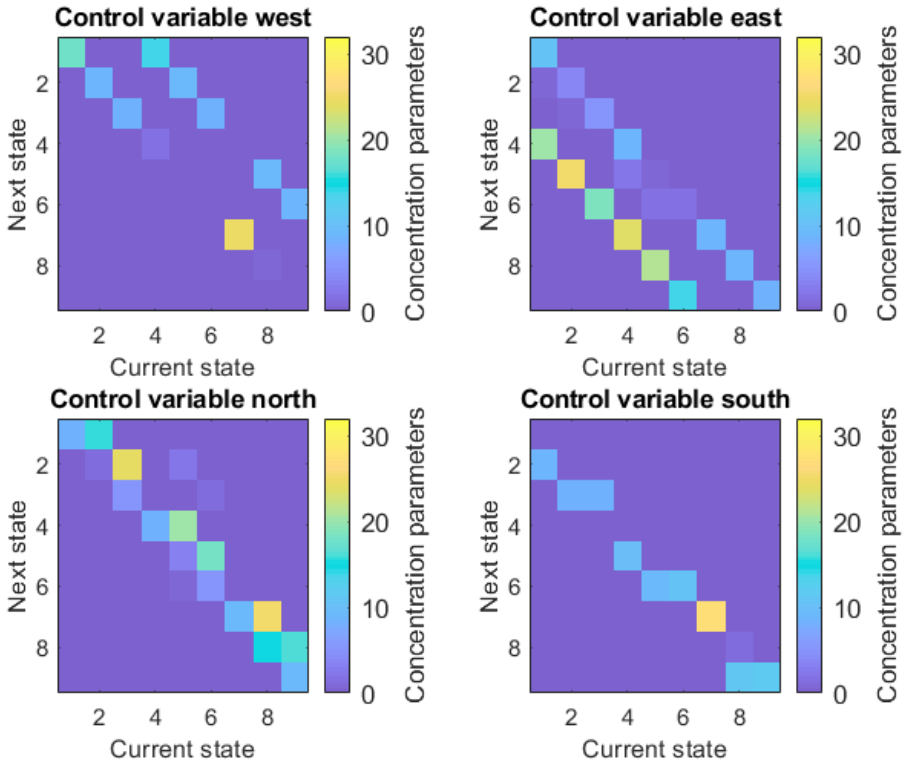


(c) At the end of trial 48.

Figure A.1: Heatmap of the transition concentration parameters for map1 for the standard update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

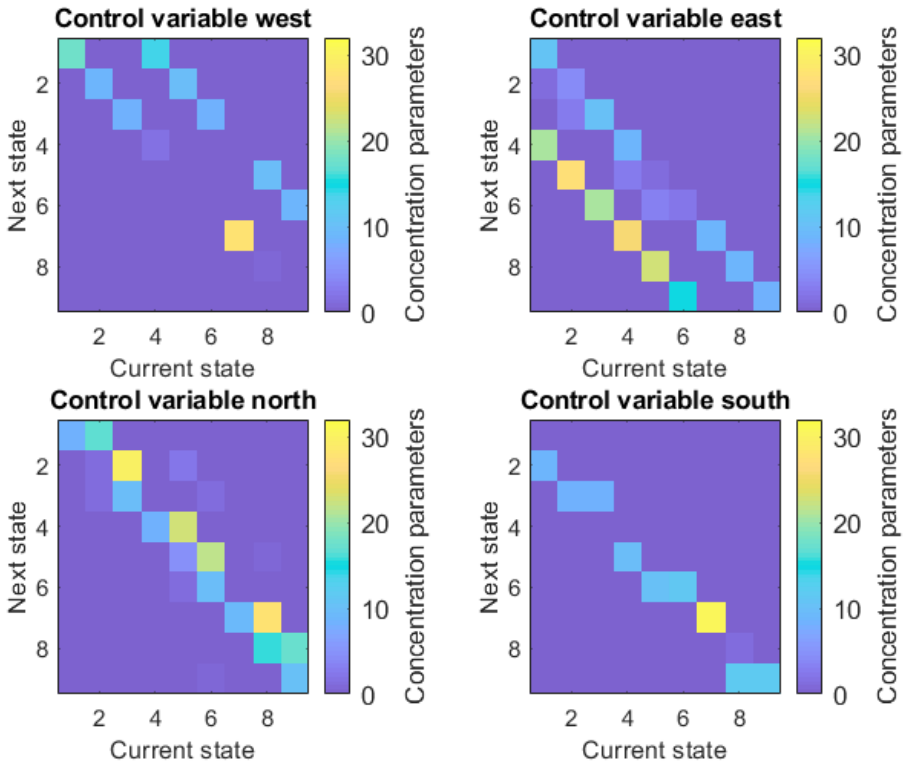


(d) At the end of trial 64.

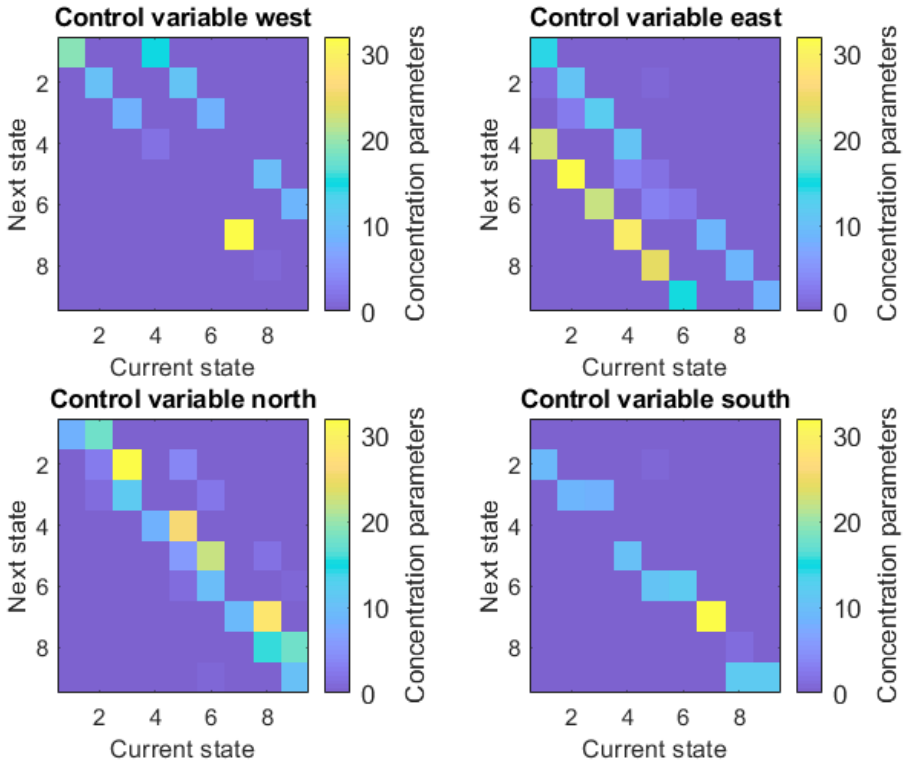


(e) At the end of trial 80.

Figure A.1: Heatmap of the transition concentration parameters for map1 for the standard update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

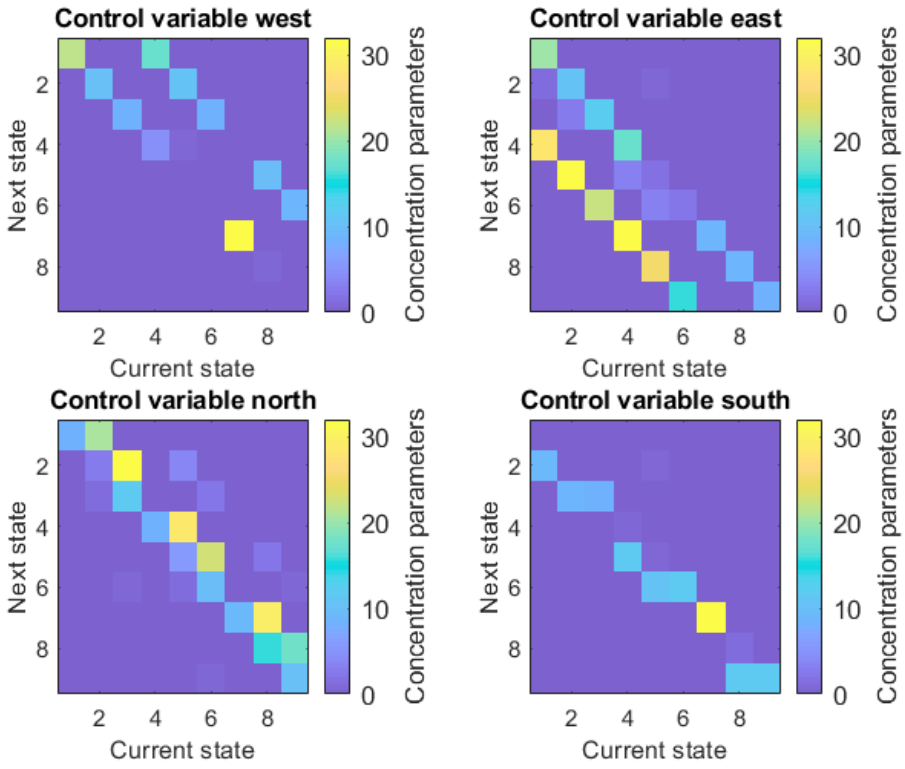


(f) At the end of trial 96.



(g) At the end of trial 112.

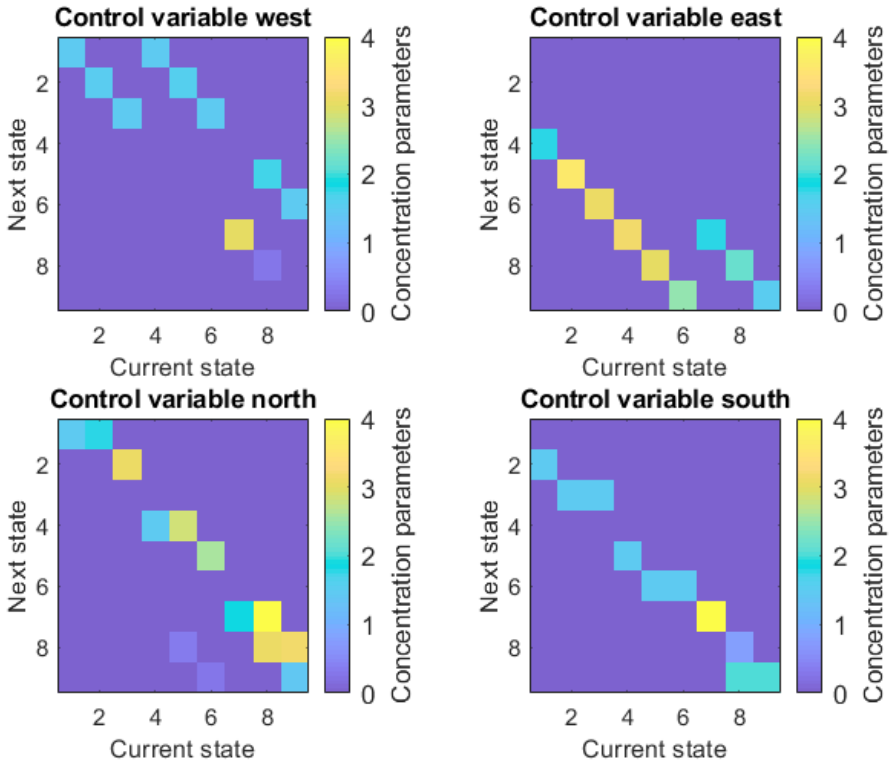
Figure A.1: Heatmap of the transition concentration parameters for map1 for the standard update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.



(h) At the end of trial 128.

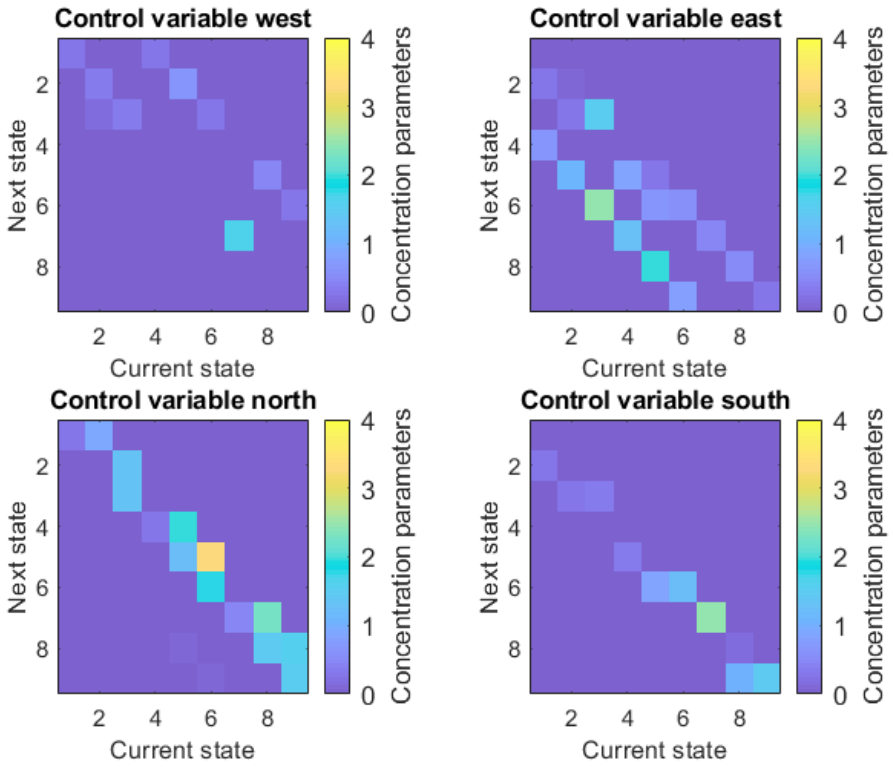
Figure A.1: Heatmap of the transition concentration parameters for map1 for the standard update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

A.2. The constant update method

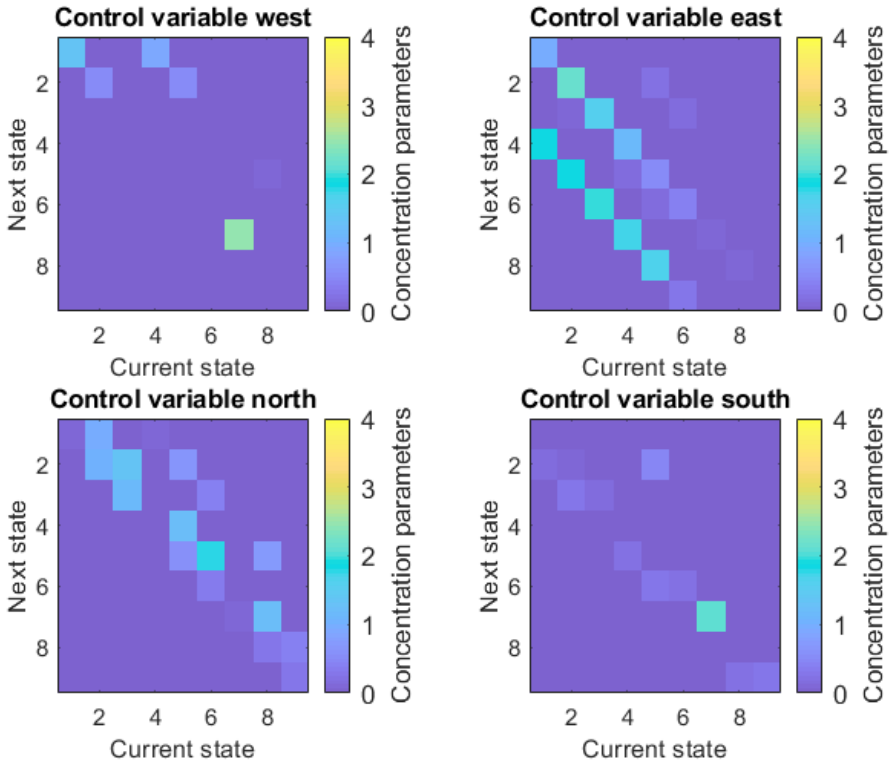


(a) At the end of trial 16.

Figure A.2: Heatmap of the transition concentration parameters for map1 for the constant update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

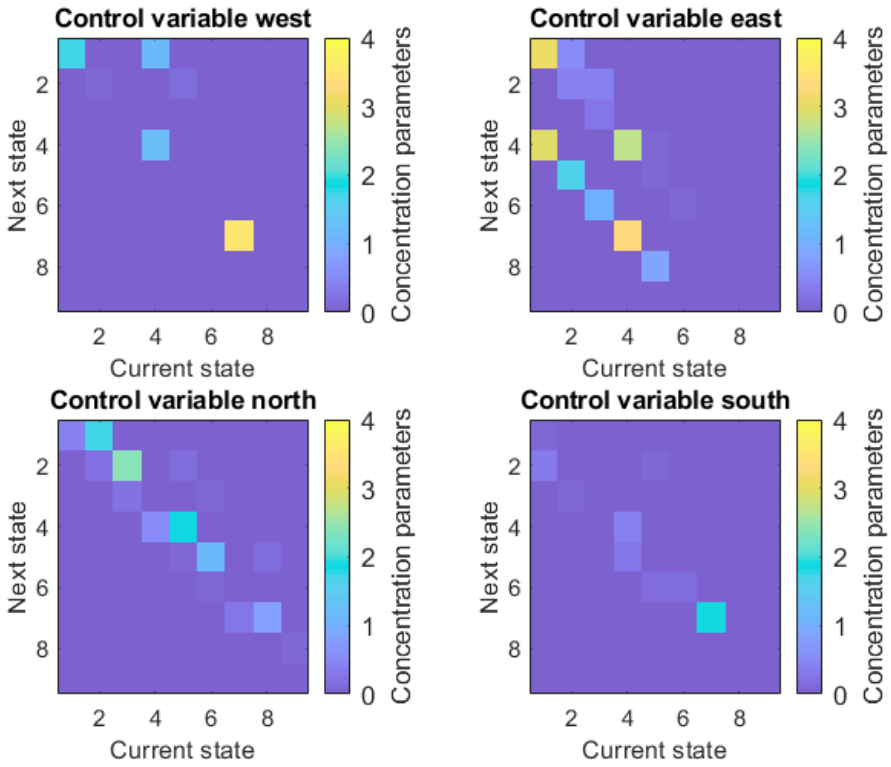


(b) At the end of trial 32.

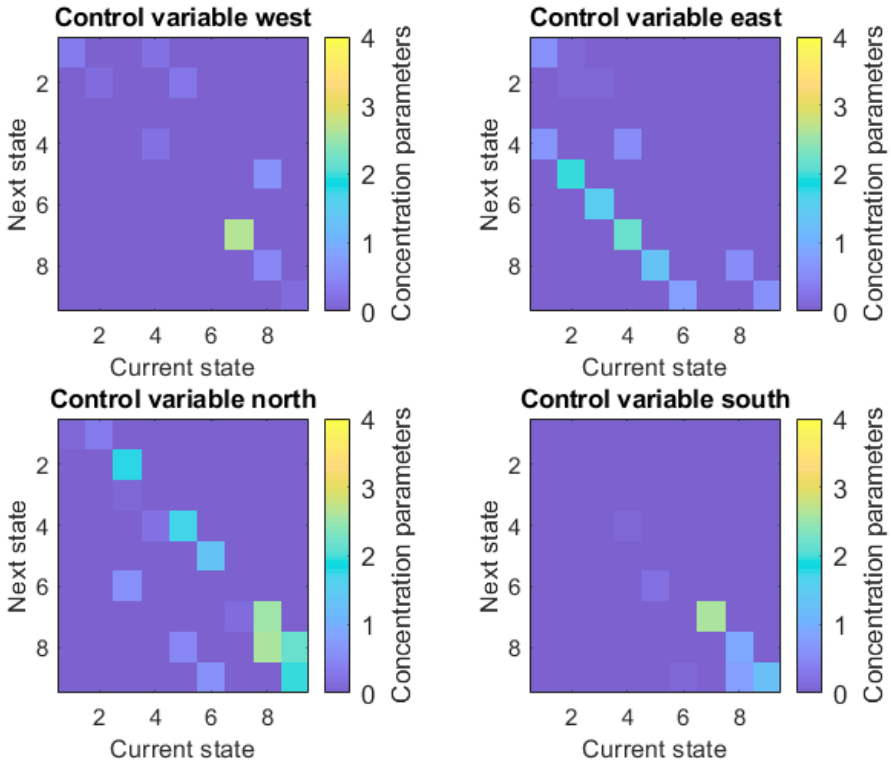


(c) At the end of trial 48.

Figure A.2: Heatmap of the transition concentration parameters for map1 for the constant update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

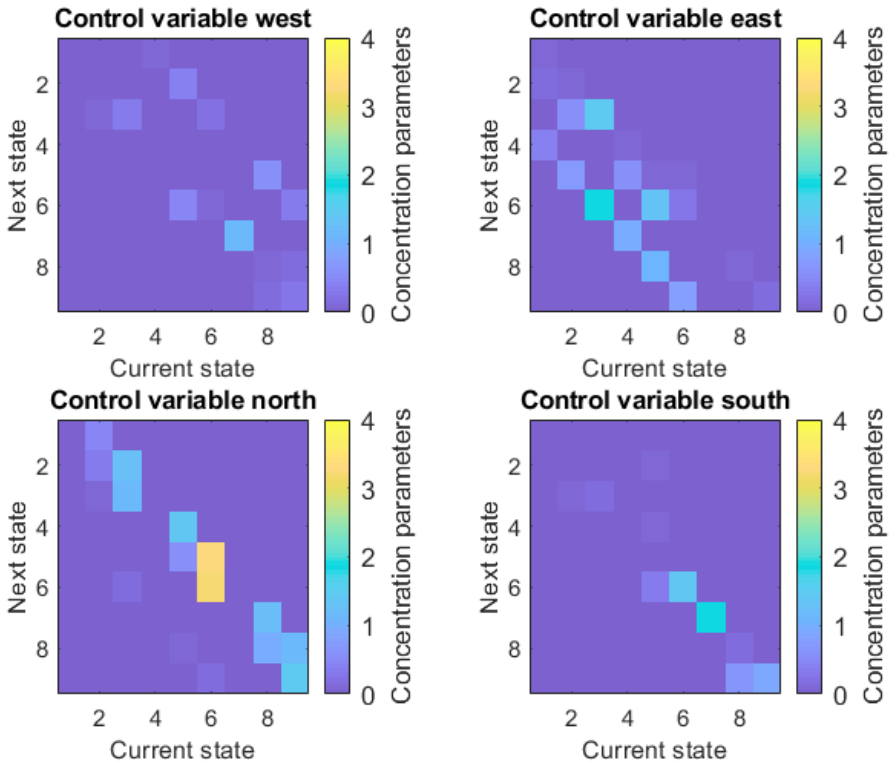


(d) At the end of trial 64.

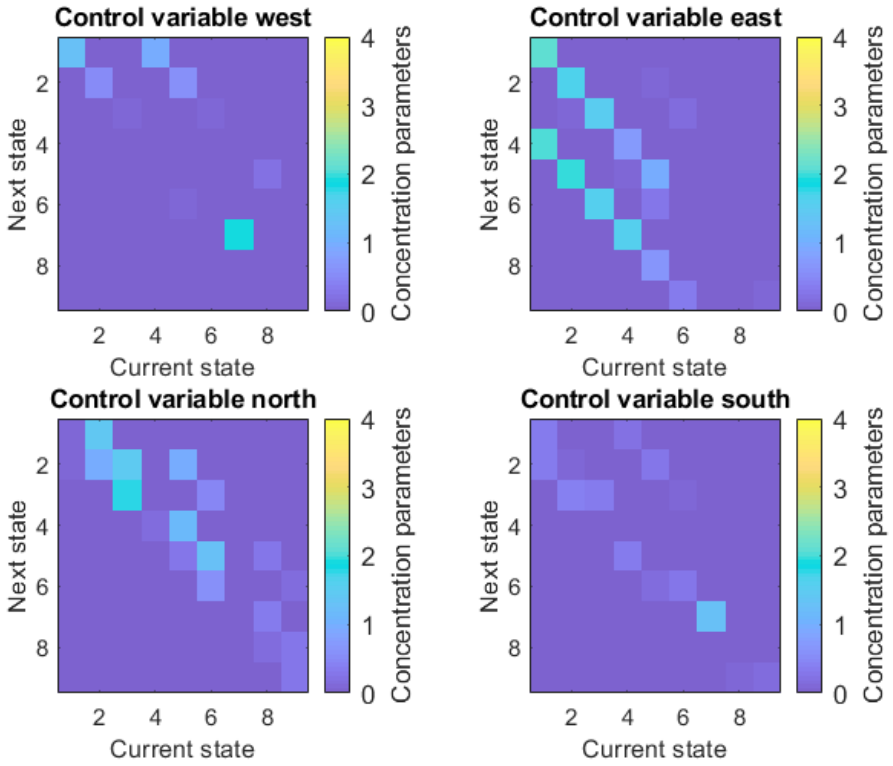


(e) At the end of trial 80.

Figure A.2: Heatmap of the transition concentration parameters for map1 for the constant update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

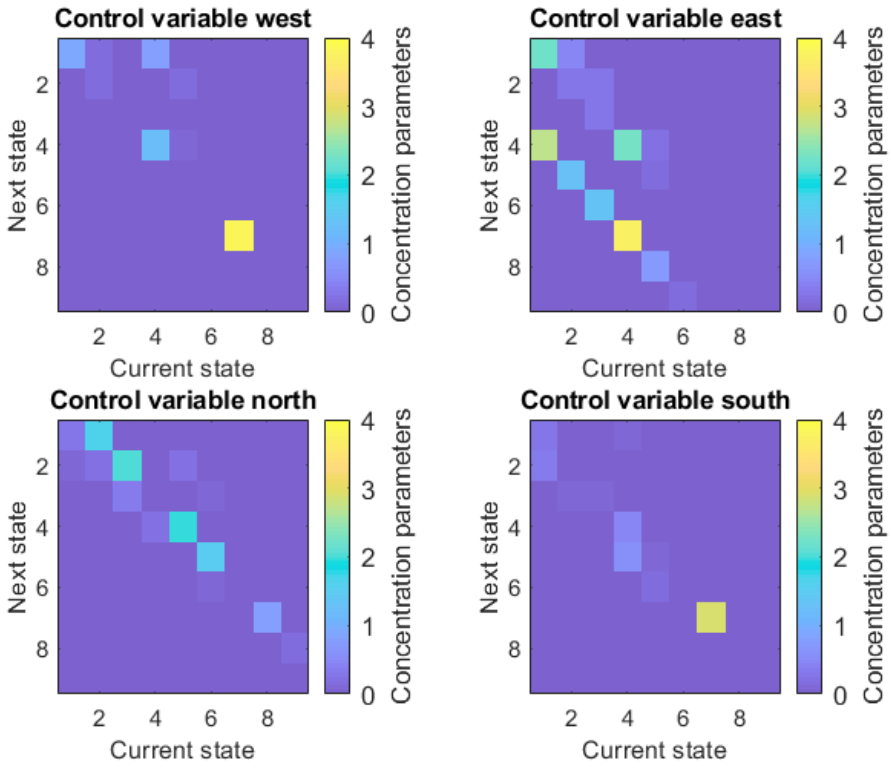


(f) At the end of trial 96.



(g) At the end of trial 112.

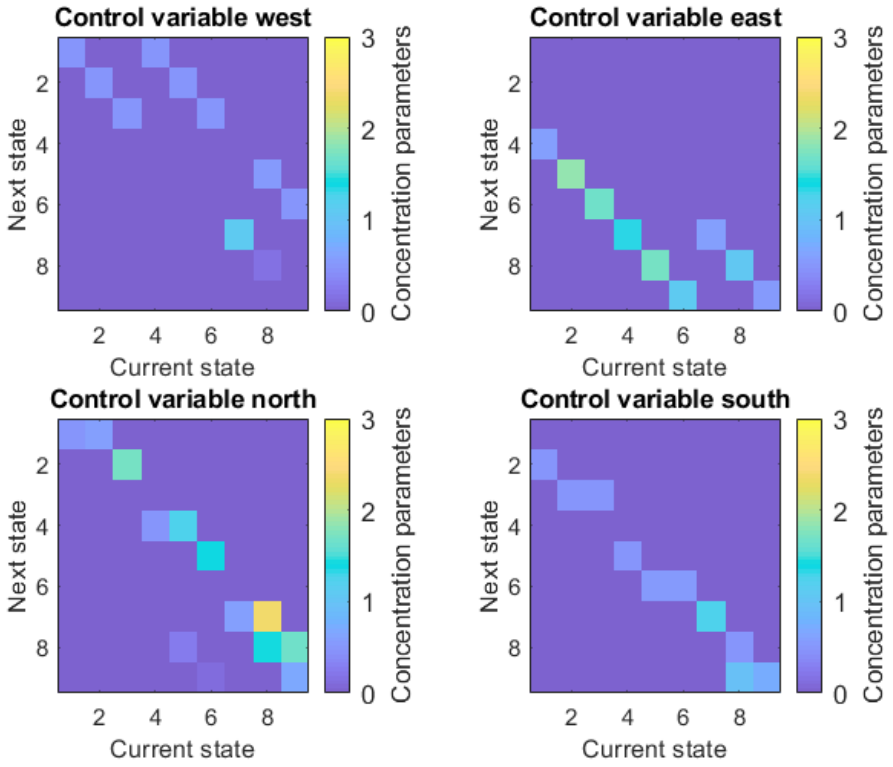
Figure A.2: Heatmap of the transition concentration parameters for map1 for the constant update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.



(h) At the end of trial 128.

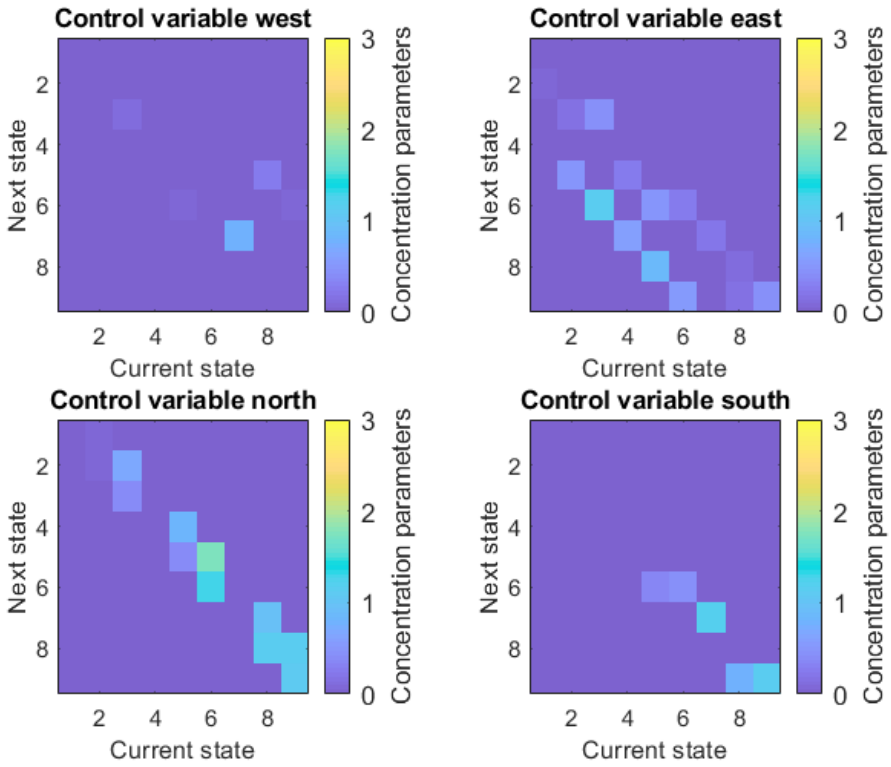
Figure A.2: Heatmap of the transition concentration parameters for map1 for the constant update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

A.3. The LSTM update method

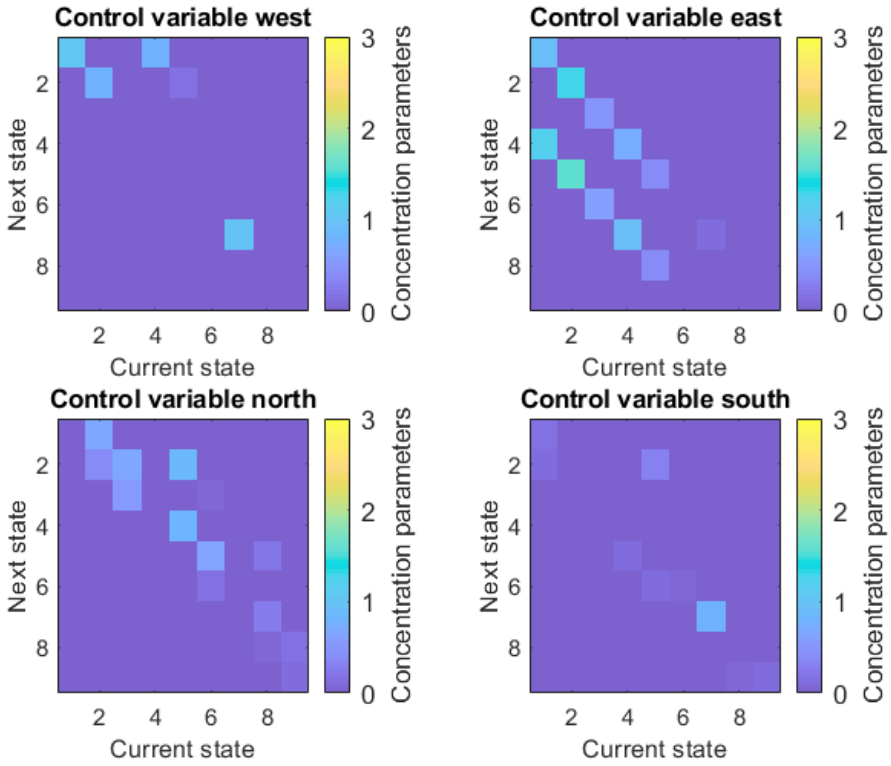


(a) At the end of trial 16.

Figure A.3: Heatmap of the transition concentration parameters for map1 for the LSTM update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

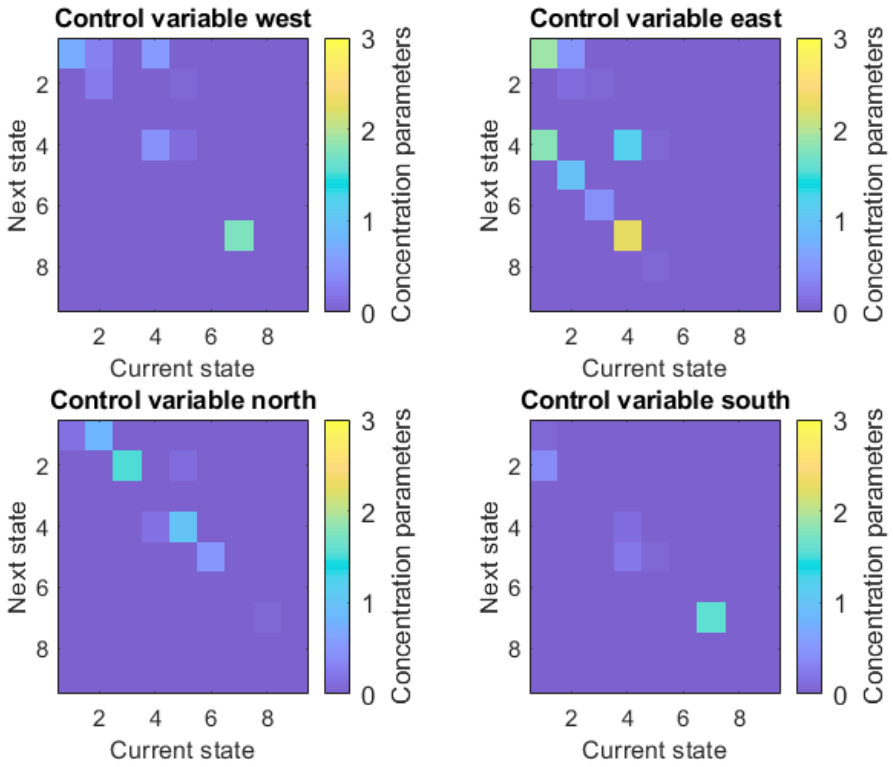


(b) At the end of trial 32.

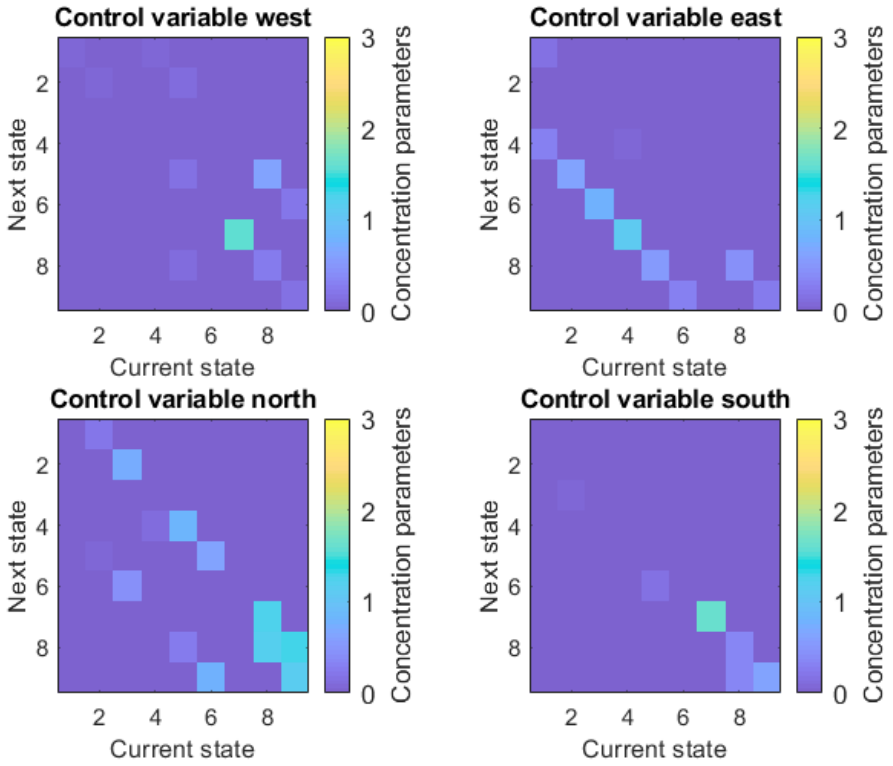


(c) At the end of trial 48.

Figure A.3: Heatmap of the transition concentration parameters for map1 for the LSTM update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

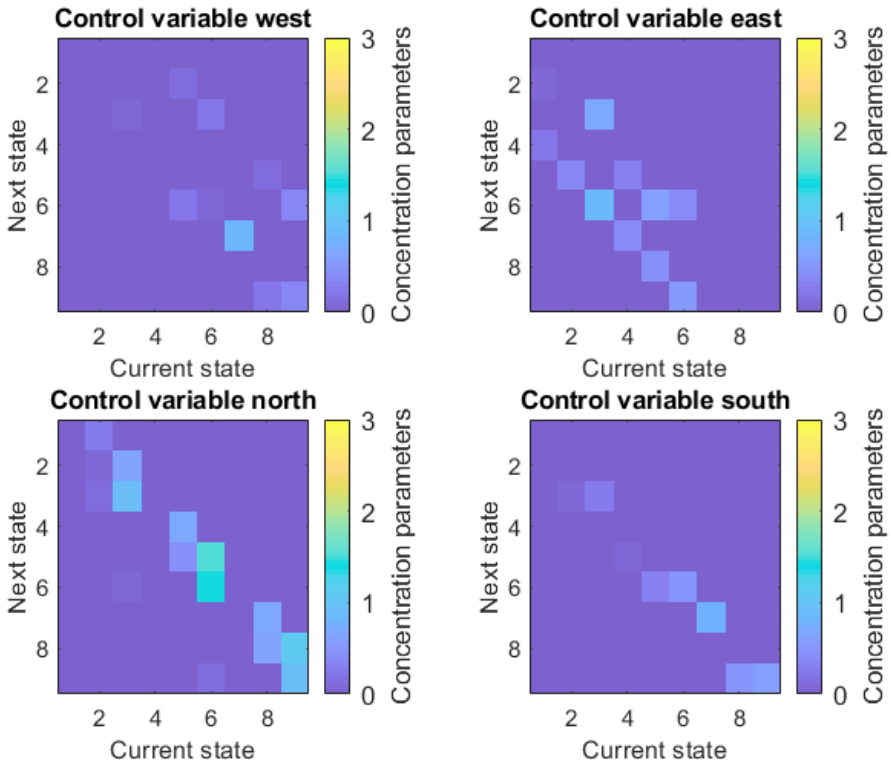


(d) At the end of trial 64.

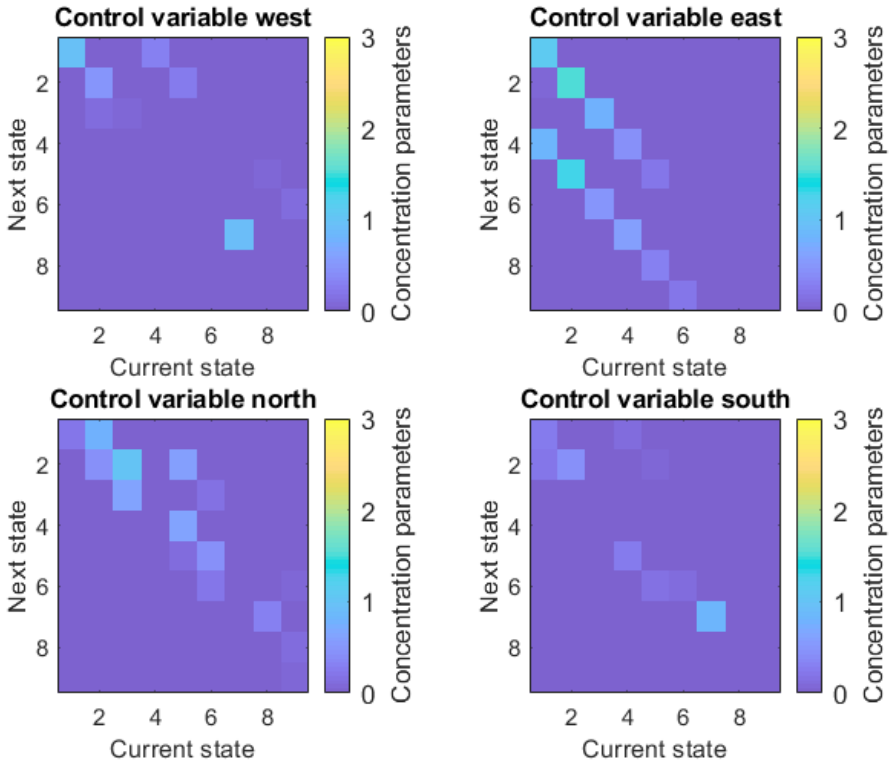


(e) At the end of trial 80.

Figure A.3: Heatmap of the transition concentration parameters for map1 for the LSTM update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

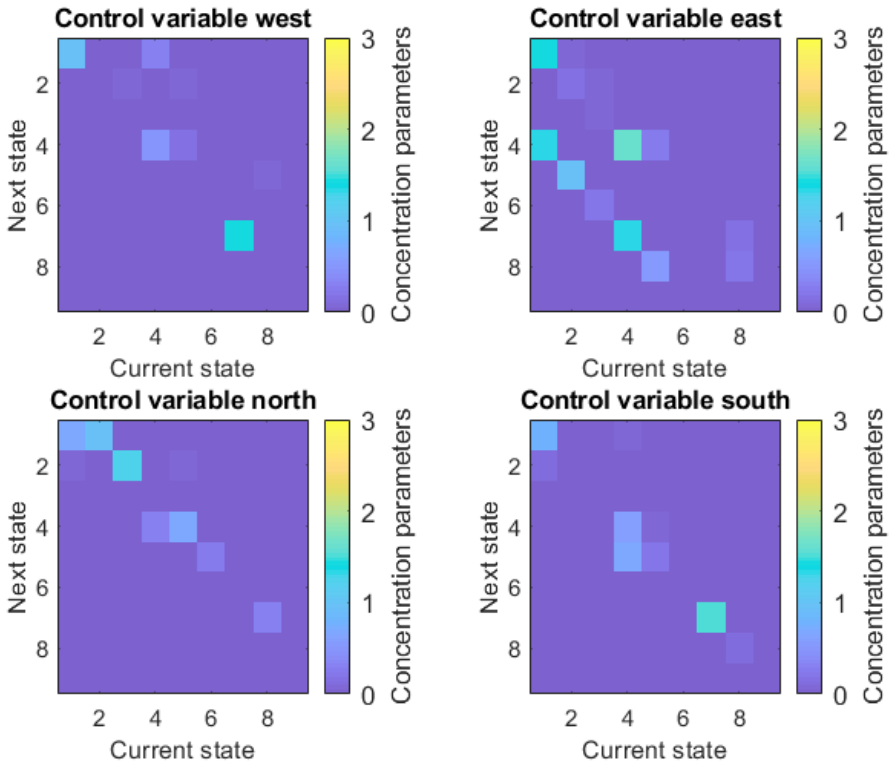


(f) At the end of trial 96.



(g) At the end of trial 112.

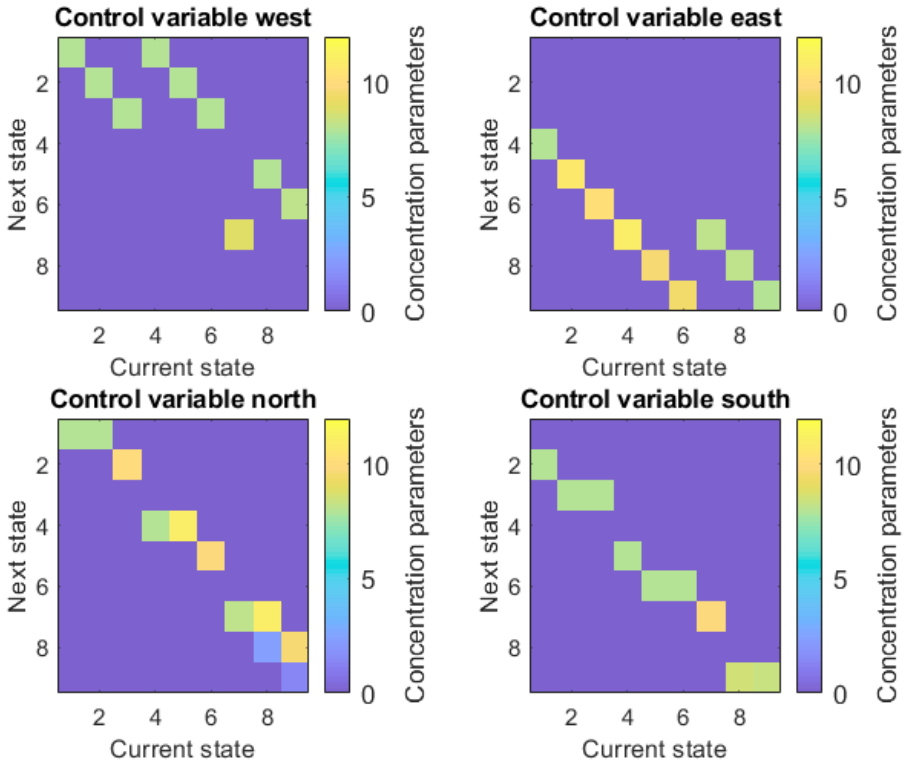
Figure A.3: Heatmap of the transition concentration parameters for map1 for the LSTM update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.



(h) At the end of trial 128.

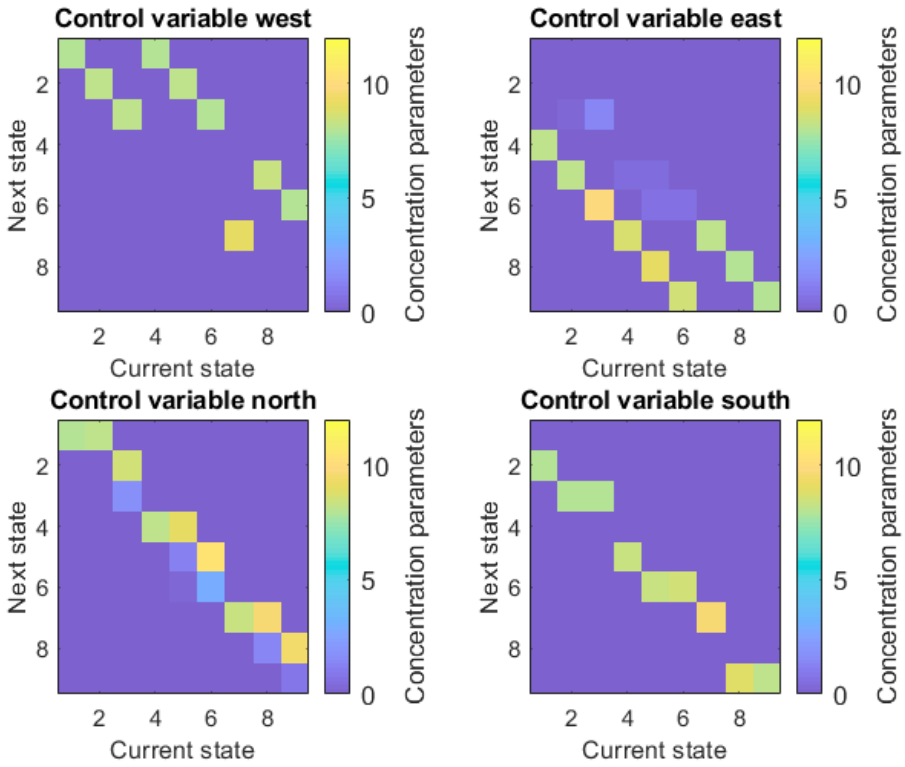
Figure A.3: Heatmap of the transition concentration parameters for map1 for the LSTM update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

A.4. The rolling update method

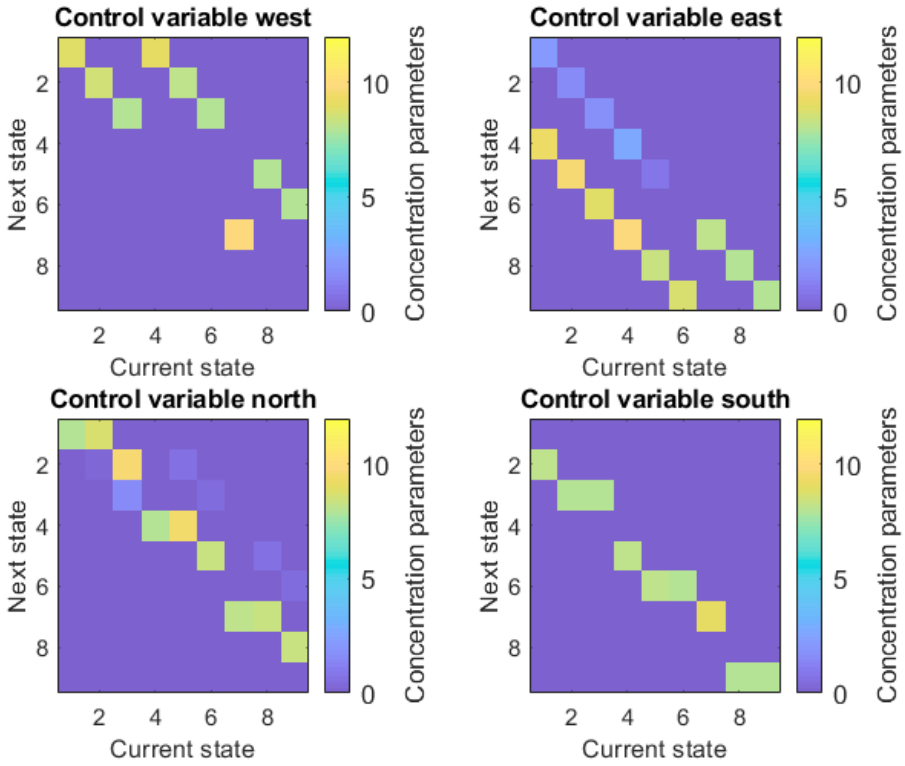


(a) At the end of trial 16.

Figure A.4: Heatmap of the transition concentration parameters for map1 for the rolling update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

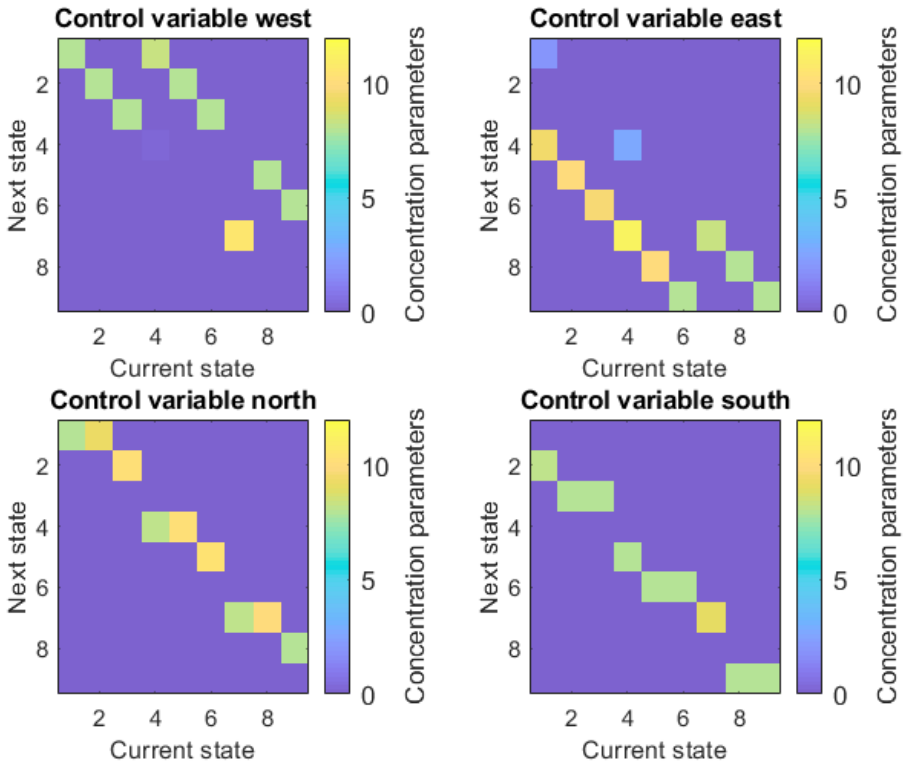


(b) At the end of trial 32.

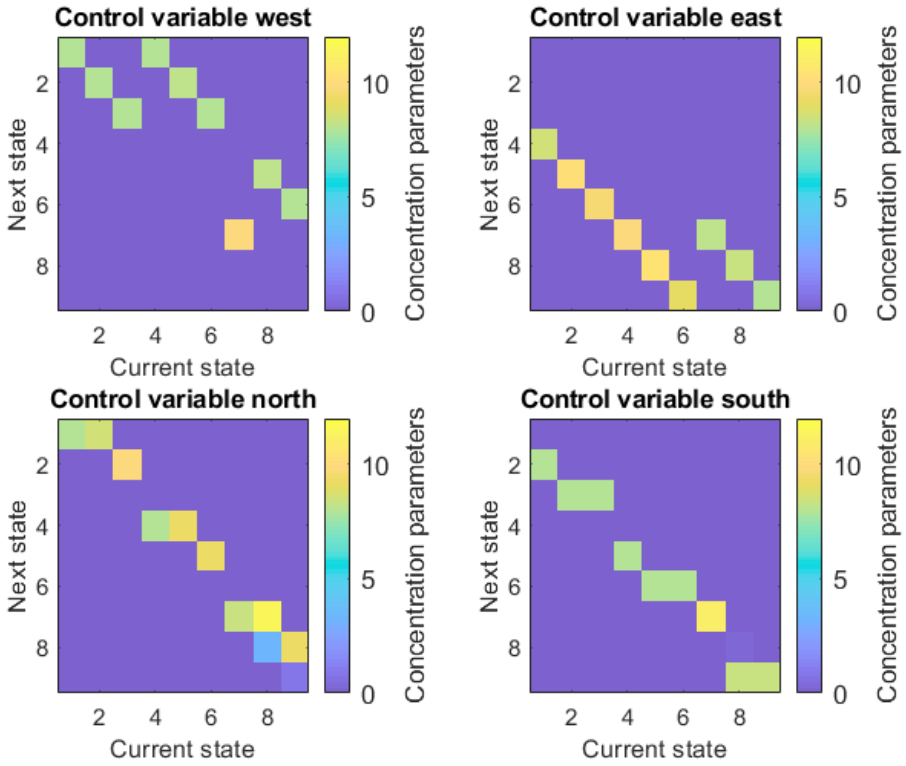


(c) At the end of trial 48.

Figure A.4: Heatmap of the transition concentration parameters for map1 for the rolling update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

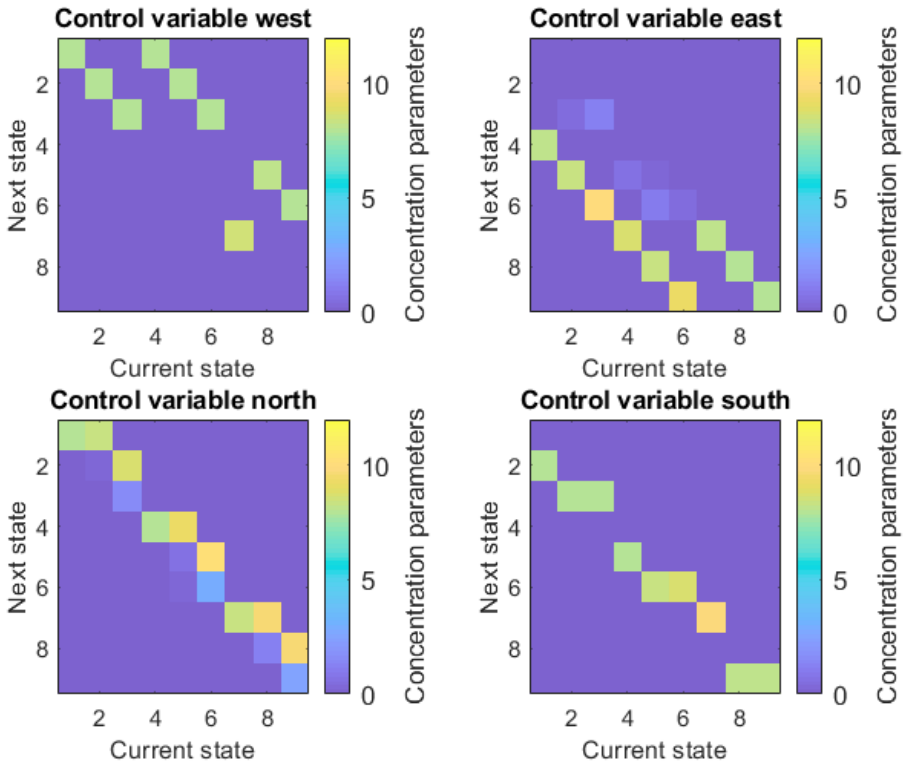


(d) At the end of trial 64.

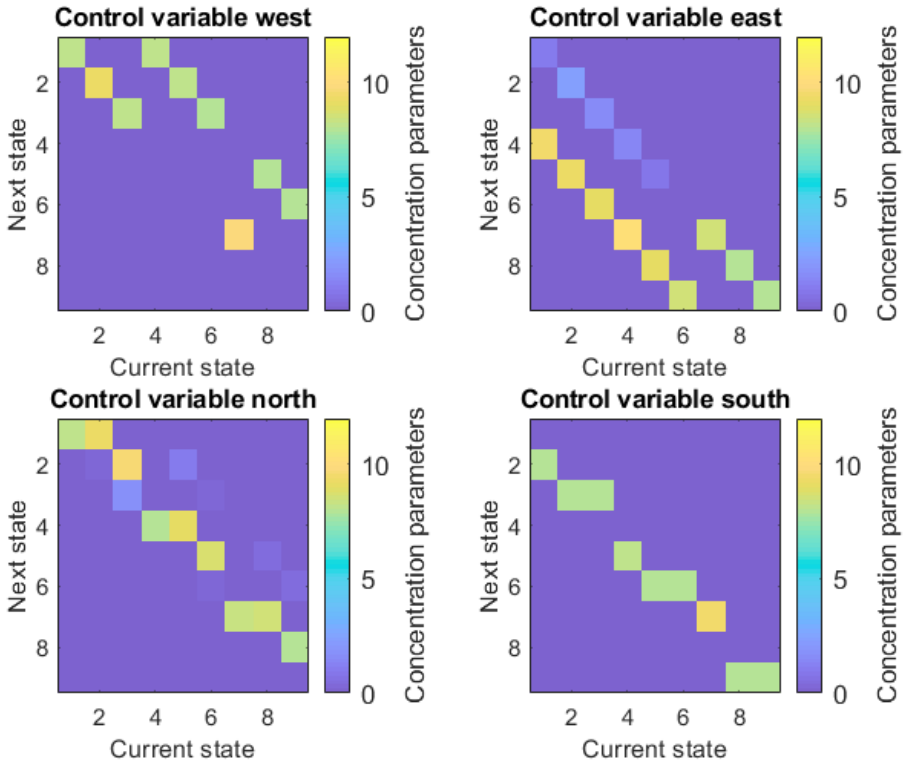


(e) At the end of trial 80.

Figure A.4: Heatmap of the transition concentration parameters for map1 for the rolling update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

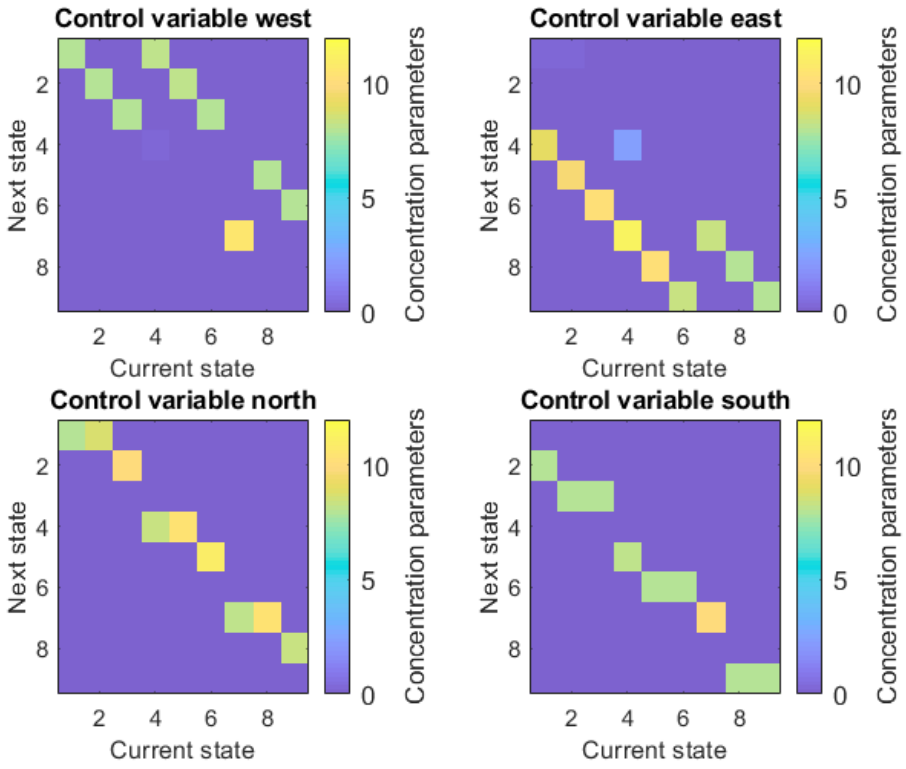


(f) At the end of trial 96.



(g) At the end of trial 112.

Figure A.4: Heatmap of the transition concentration parameters for map1 for the rolling update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.



(h) At the end of trial 128.

Figure A.4: Heatmap of the transition concentration parameters for map1 for the rolling update method. Each subfigure shows to results at the final trial before a change in transition dynamics occurs, and within each subfigure, the concentration parameters for each control variable are visualized.

Bibliography

[1] Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5): 679–684, 1957.

[2] Christopher L. Buckley, Chang Sub Kim, Simon McGregor, and Anil K. Seth. The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, 81:55–79, 2017.

[3] Samuel P. M. Choi, Dit-Yan Yeung, and Nevin L. Zhang. Hidden-mode Markov decision processes for nonstationary sequential decision making. In Ron Sun and C. Lee Giles, editors, *Sequence Learning*, pages 264–287. Springer, Berlin, Heidelberg, 2001.

[4] Lancelot Da Costa, Thomas Parr, Noor Sajid, Sebastijan Veselic, Victorita Neacsu, and Karl Friston. Active inference on discrete state-spaces: A synthesis. *Journal of Mathematical Psychology*, 99, 2020.

[5] F.M. Dekking, C. Kraaikamp, H.P. Lopuhaä, and L.E. Meester. *A Modern Introduction to Probability and Statistics: Understanding Why and How*. Springer, London, UK, 2005.

[6] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[7] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11:127–138, 2010.

[8] Karl Friston. Life as we know it. *Journal of the Royal Society Interface*, 10(86), 2013.

[9] Karl Friston, James Kilner, and Lee Harrison. A free energy principle for the brain. *Journal of Physiology - Paris*, 100(1-3):70–87, 2006.

[10] Karl Friston, Spyridon Samothrakis, and Read Montague. Active inference and agency: optimal control without cost functions. *Biological Cybernetics*, 106:523–541, 2012.

[11] Karl Friston, Philipp Schwartenbeck, Thomas FitzGerald, Michael Moutoussis, Timothy Behrens, and Raymond J. Dolan. The anatomy of choice: active inference and agency. *Frontiers in Human Neuroscience*, 7, 2013.

[12] Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, John O'Doherty, and Giovanni Pezzulo. Active inference and learning. *Neuroscience & Biobehavioral Reviews*, 68: 862–879, 2016.

[13] Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, and Giovanni Pezzulo. Active inference: A process theory. *Neural Computation*, 29(1):1–49, 2017.

[14] Karl J. Friston. Active inference and cognitive consistency. *Psychological Inquiry*, 29(2):67–73, 2018.

[15] Karl J. Friston, Jean Daunizeau, and Stefan J. Kiebel. Reinforcement learning or active inference? *PLoS ONE*, 4(7), 2009.

[16] Karl J. Friston, Richard Rosch, Thomas Parr, Cathy Price, and Howard Bowman. Deep temporal models and active inference. *Neuroscience & Biobehavioral Reviews*, 77:388–402, 2017.

[17] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: continual prediction with LSTM. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, ICANN, pages 850–855. IET, 1999.

[18] Raphael Kaplan and Karl J. Friston. Planning and navigation as active inference. *Biological Cybernetics*, 112:323–343, 2018.

[19] Kentaro Katahira. The relation between reinforcement learning parameters and the influence of reinforcement history on choice behavior. *Journal of Mathematical Psychology*, 66:59–69, 2015.

[20] Ayaka Kato and Kenji Morita. Forgetting in reinforcement learning links sustained dopamine signals to motivation. *PLoS Computational Biology*, 12(10), 2016.

[21] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[22] M. Berk Mirza, Rick A. Adams, Karl Friston, and Thomas Parr. Introducing a Bayesian model of selective attention based on active inference. *Scientific Reports*, 9, 2019.

[23] Douglas C. Montgomery, Cheryl L. Jennings, and Murat Kulahci. *Introduction to Time Series Analysis and Forecasting*. John Wiley & Sons, Incorporated, Hoboken, NJ, USA, 2015.

[24] Antonio Napolitano. *Cyclostationary Processes and Time Series: Theory, Applications, and Generalizations*. Academic Press, Amsterdam, Netherlands, 2019.

[25] Christopher Olah. Understanding LSTM networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.

[26] Alan V. Oppenheim and George C. Verghese. *Signals, Systems and Inference*. Pearson Education Limited, Harlow, England, 2017.

[27] Thomas Parr and Karl J. Friston. Uncertainty, epistemics and active inference. *Journal of the Royal Society Interface*, 14(136), 2017.

[28] Thomas Parr and Karl J. Friston. The discrete and continuous brain: From decisions to movement—and back again. *Neural Computation*, 30:2319–2347, 2018.

[29] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, USA, 2010.

[30] Noor Sajid, Philip J. Ball, Thomas Parr, and Karl J. Friston. Active inference: Demystified and compared. *Neural Computation*, 33:1–39, 2021.

[31] Philipp Schwartenbeck, Johannes Passecker, Tobias U Hauser, Thomas HB FitzGerald, Martin Kronbichler, and Karl J Friston. Computational mechanisms of curiosity and goal-directed exploration. *eLife*, 8, 2019.

[32] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, USA, 2018.

[33] Rand R. Wilcox. *Basic Statistics: Understanding Conventional Methods and Modern Insights*. Oxford University Press, New York, NY, USA, 2009.

[34] John Winn and Christopher M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6:661–694, 2005.