5

Interacting with any system

Concept. To design and build a software which can interact with stochastic processes of any kind, either manually through user input, or automatically by introducing a 'policy'. The mathematical formalism and software that we introduce here will serve as a common language and interface for any simulation studies into manipulating real world phenomena, and will enable the learning of control algorithms in later chapters of this book. We will implement this new interaction software as an extension to the stochadex package. For the mathematically-inclined, this chapter will cover how interactions are structured in theory by adding some new concepts to the stochadex formalism and illustrating with some simple examples. For the programmers, the public Git repository for the code described in this chapter can be found here: https://github.com/umbralcalc/stochadex.

5.1 Formalising general interactions

Let's start by considering how we might adapt the mathematical formalism we have been using so far to be able to take actions which can manipulate the state at each timestep. Using the mathematical notation that we inherited from the stochadex, we may extend the formula for updating the state history matrix $X_{0:\mathtt{t}} \to X_{0:\mathtt{t}+1}$ to include a new layer of possible interactions which is facilitated by a new vector-valued 'take action' function $G_{\mathtt{t}}$. In doing so we shall be defining the domain of an acting entity in the stochastic process environment — which we shall hereafter refer to as simply the 'agent'.

During a timestep over which actions are performed by the agent, the stochadex state update formula can be extended to include interactions by composition with the original state update function like so

$$X^i_{\mathtt{t}+1} = G^i_{\mathtt{t}+1}[F_{\mathtt{t}+1}(X_{0:\mathtt{t}}, z, \mathtt{t}), A_{\mathtt{t}+1}] = \mathcal{F}^i_{\mathtt{t}+1}(X_{0:\mathtt{t}}, z, A_{\mathtt{t}+1}, \mathtt{t})\,, \tag{5.1}$$

where we have also introduced the concept of the 'actions' performed $A_{\mathtt{t}+1}$ on the system; some vector of parameters which define what actions are taken at timestep $\mathtt{t}+1$. The code for the new iteration formula would look something like Fig. 5.1.

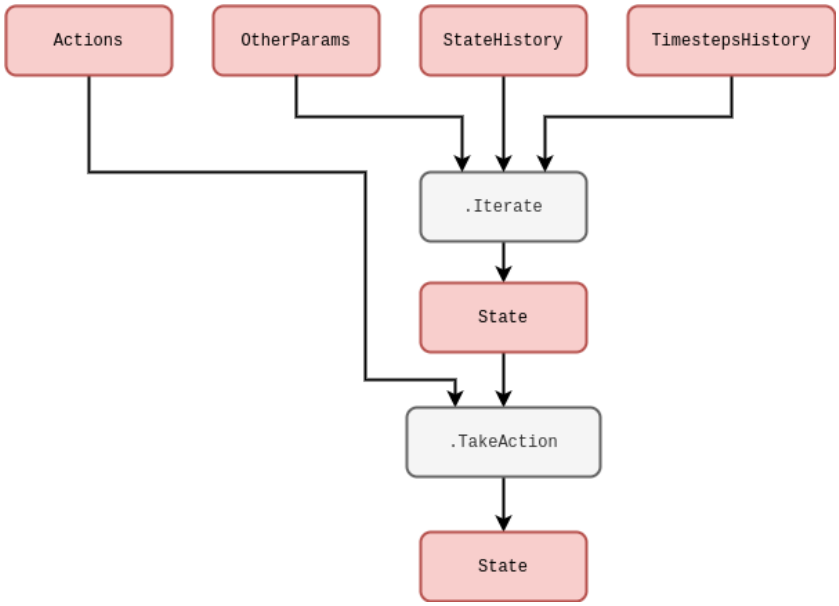

Figure 5.1: Code schematic of Eq. (5.1).

So far, Eq. (5.1) on its own will allow the agent to take actions that are scheduled up front through some fixed process or perhaps through user interaction via a game interface. So what's next? In order to start creating algorithms which will act on the system state for us, we need to develop a formalism which 'closes the loop' by feeding information back from the stochastic process to the agent's decision-making algorithm.

If we use $A_{0:\mathsf{t}+1}$ a referring to the matrix of historically-taken actions which up to time $\mathsf{t}+1$, we can build up a more generalised, non-Markovian picture of automated interactions with the system which matches the notation we are already using for $X_{0:\mathsf{t}+1}$. Let us now define a Non-Markovian Decision Process (NMDP) as a probabilistic model which draws an actions matrix $A_{0:\mathsf{t}+1} = A$ from a 'policy' distribution $\Pi_{(\mathsf{t}+1)\mathsf{t}}(A|X,\theta)$ given $X_{0:\mathsf{t}} = X$ and a new vector of parameters which fully specify the automated interations. Using the probabilistic notation from the previous part of the book, the joint probability that $X_{0:\mathsf{t}+1} = X$ and $A_{0:\mathsf{t}+1} = A$ at time $\mathsf{t}+1$ is

$$P_{\mathsf{t}+1}(X, A|z,\theta) = P_{\mathsf{t}}(X'|z,\theta)\,\Pi_{(\mathsf{t}+1)\mathsf{t}}(A|X',\theta)P_{(\mathsf{t}+1)\mathsf{t}}(x|X',z,A)\,, \tag{5.2}$$

where we recall that $P_{(\mathsf{t}+1)\mathsf{t}}(x|X',z,A)$ is the conditional probability of $X_{\mathsf{t}+1} = x$ given $X_{0:\mathsf{t}} = X'$ and z that we have encountered before, but it now requires $A_{0:\mathsf{t}+1} = A$ as another given input. We have illustrated Eq. (5.1) and how it relates to the policy distribution of Eq. (5.2) with a new graph representation in Fig. 5.2.

For additional clarity, let's take a moment to think about what $\Pi_{(\mathsf{t}+1)\mathsf{t}}(A|X,\theta)$ represents and how generally descriptive it can be. If an agent is acting under and entirely deterministic policy, then the policy distribution may be simplified to a direct function mapping which is parameterised by θ. At the other extreme, the distribution may also describe a fully stochastic policy where actions are drawn randomly in time. If we combine this consideration of noise with the observation that

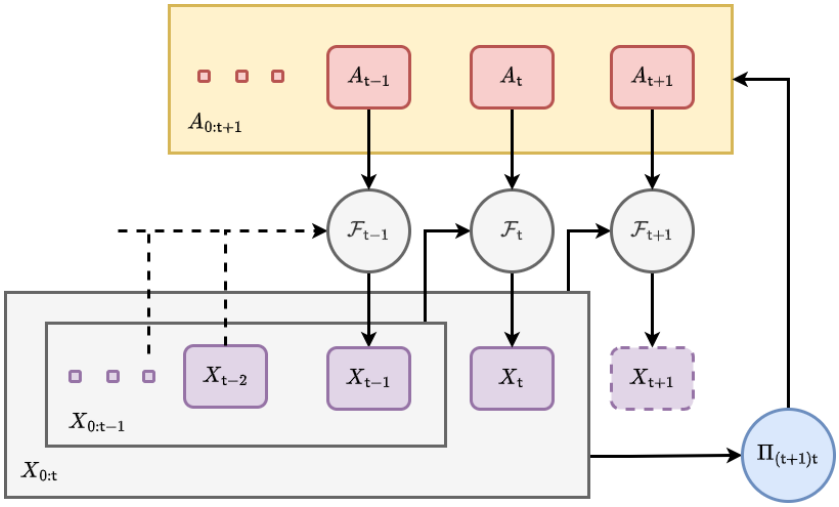

Figure 5.2: Graph representation of Eq. (5.1) with the policy distribution of Eq. (5.2).

policies described by a distribution $\Pi_{(t+1)t}(A|X,\theta)$ permit a memory of past actions and states, it's easy to see that this structure can be used in a wide variety of different use cases.

By marginalising over Eq. (5.2) we find an updated probabilistic iteration formula for the stochastic process state which now takes the influence of agent actions into account

$$P_{t+1}(X|z,\theta) = \int_{\Xi_{t+1}} \mathrm{d}A\, P_t(X'|z,\theta)\, \Pi_{(t+1)t}(A|X',\theta) P_{(t+1)t}(x|X',z,A)\,. \tag{5.3}$$

This relationship will be very useful in the last part of this book when we begin to look at optimising control algorithms.

What are the main categories of action which are possible in the rows of A? Since the NMDP described by $\Pi_{(t+1)t}(A|X',\theta)$ is just another form of stochastic process, the main categories of action will fall into the same as those we covered in defining the stochadex formalism. The first, and perhaps most obvious, category would probably where the actions are defined in a continuous space and are continuously applied on every timestep. Some examples of these 'continuously-acting' decision processes include controlling the temperature of chemical reactions [1] (such as those in a brewery), spacecraft control [2] and guidance systems, as well as the driving of autonomous vehicles [3]. Within a kind of subset of the continuously-acting category; we can also find the event-based acting decision processes (where actions are not necessarily taken every timestep), e.g. controlling traffic through signal timings [4], managing disease spread through treatment intervals [5] and automated trading on stock markets [6].

Many of the examples we have given above have continuous action spaces, but we might also consider classes of decision processes where actions are defined discretely. Examples of these include the famous multi-armed bandit problem [7] (like choosing between website layouts for E-commerce [8]), managing a sports team through player substitutions, sensor measurement scheduling [9] and the sequential design prioritisation of large-scale scientific experiments [10].

Bibliography

[1] C. Beeler, S. G. Subramanian, K. Sprague, N. Chatti, C. Bellinger, M. Shahen et al., *Chemgymrl: An interactive framework for reinforcement learning for digital chemistry*, *arXiv preprint arXiv:2305.14177* (2023) .

[2] M. Tipaldi, R. Iervolino and P. R. Massenio, *Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges*, *Annual Reviews in Control* (2022) .

[3] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani et al., *Deep reinforcement learning for autonomous driving: A survey*, *IEEE Transactions on Intelligent Transportation Systems* **23** (2021) 4909–4926.

[4] D. Garg, M. Chli and G. Vogiatzis, *Deep reinforcement learning for autonomous traffic light control*, in *2018 3rd IEEE international conference on intelligent transportation engineering (ICITE)*, pp. 214–218, IEEE, 2018.

[5] A. Q. Ohi, M. Mridha, M. M. Monowar and M. A. Hamid, *Exploring optimal control of epidemic spread using reinforcement learning*, *Scientific reports* **10** (2020) 22106.

[6] T. L. Meng and M. Khushi, *Reinforcement learning in financial markets*, *Data* **4** (2019) 110.

[7] J. Gittins, K. Glazebrook and R. Weber, *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[8] Y. Liu and L. Li, *A map of bandits for e-commerce*, *arXiv preprint arXiv:2107.00680* (2021) .

[9] A. S. Leong, A. Ramaswamy, D. E. Quevedo, H. Karl and L. Shi, *Deep reinforcement learning for wireless sensor scheduling in cyber–physical systems*, *Automatica* **113** (2020) 108759.

[10] T. Blau, E. V. Bonilla, I. Chades and A. Dezfouli, *Optimizing sequential experimental design with deep reinforcement learning*, in *International Conference on Machine Learning*, pp. 2107–2128, PMLR, 2022.