Swarm Intell (2018) 12:23–51
https://doi.org/10.1007/s11721-017-0142-9

Reinforcement learning in a continuum of agents

Adrian Šošić[1] · Abdelhak M. Zoubir[1] ·
Heinz Koeppl[1]

Received: 10 March 2017 / Accepted: 29 September 2017 / Published online: 13 October 2017

Abstract We present a decision-making framework for modeling the collective behavior of large groups of cooperatively interacting agents based on a continuum description of the agents' joint state. The continuum model is derived from an agent-based system of locally coupled stochastic differential equations, taking into account that each agent in the group is only partially informed about the global system state. The usefulness of the proposed framework is twofold: (i) for multi-agent scenarios, it provides a computational approach to handling large-scale distributed decision-making problems and learning decentralized control policies. (ii) For single-agent systems, it offers an alternative approximation scheme for evaluating expectations of state distributions. We demonstrate our framework on a variant of the Kuramoto model using a variety of distributed control tasks, such as positioning and aggregation. As part of our experiments, we compare the effectiveness of the controllers learned by the continuum model and agent-based systems of different sizes, and we analyze how the degree of observability in the system affects the learning process.

Keywords Reinforcement learning · Multi-agent systems · Decentralized control · Collective behavior · Swarm intelligence · Active particles · Continuum mechanics

1 Introduction

The increasing complexity of autonomous systems in recent years has lead to a growing demand for distributed learning and decision-making algorithms.[1] In fact, many of today's and future technologies are founded on the idea of distributed problem-solving

[1] Note that we use the terms *decision-making* and *control* interchangeably in this work.

✉ Adrian Šošić
adrian.sosic@spg.tu-darmstadt.de

Heinz Koeppl
heinz.koeppl@bcs.tu-darmstadt.de

[1] Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt, 64283 Darmstadt, Germany

and require the interplay of several decision-makers or *agents*. As a consequence, collective decision-making has become a major research area in artificial intelligence and modern control theory. Recently, a particular type of distributed systems has attracted increasing attention, i.e., homogeneous agent systems or *swarms*, where all agents are treated as identical copies of a single agent prototype. Prominent examples can be found, for example, in the fields of swarm robotics (Brambilla et al. 2013), sensor networks (Lesser et al. 2003), active matter (Ramaswamy 2010), self-assembly (Whitesides and Grzybowski 2002), and nanomedicine (Freitas 2005). Instead of relying on a single processing unit, these systems perform their tasks in a decentralized manner, avoiding the need for a central fusion center and the computational bottlenecks arising therewith.

While decentralized system architectures come indeed with great flexibility, their distributed nature requires that decisions be made locally, based on a possibly incomplete representation of the system's global state. This gives rise to a whole set of important questions, such as: How is the degree of observability in the system related to the feasibility of a given task (Sipper 1999; Land and Belew 1995; Crutchfield and Mitchell 1995)? How does the global system behavior depend on the number of agents involved in the decision-making process (Vicsek et al. 1995; Hayes 2002)? What is the limiting system behavior as the size of the agent population becomes large (Aumann 1964) and, in particular, how can we learn effective control policies for such large-scale regimes? Unfortunately, classical agent-based models often cannot provide answers to these questions since both their formal and computational treatment become increasingly difficult as the system size grows.

In this paper, we present an alternative approach to multi-agent control which is, in contrast to classical agent-based modeling, specifically designed for large-scale problems. Our model is based on a continuum description of a large group of cooperatively interacting agents, which we derive under the assumption that each agent in the group has access to local information only and is, hence, partially informed about the global system state. A particular focus of our work lies on the interconnection of the derived model with the principle of reinforcement learning, to open the way for self-learned distributed control. In this context, we discuss and compare the effect of different learning paradigms on the success of the collective tasks, for which the agents build their control structures from either local or global reward feedback. We demonstrate our framework on a variant of the Kuramoto model (Kuramoto 1975) using a variety of distributed control tasks, such as positioning and aggregation. As part of our experiments, we compare the effectiveness of the controllers learned by the continuum model and agent-based systems of different sizes, and we analyze how the degree of observability in the system affects the learning process.

The usefulness of the proposed framework is twofold: (i) for multi-agent scenarios, it provides a computational approach to handling large-scale distributed decision-making problems and learning decentralized control policies. In particular, it can be used to examine the limiting behavior of a swarm system as the number of agents approaches infinity, bypassing the finite-size effects inherent to agent-based models and avoiding the computational bottlenecks of large-scale agent simulations. (ii) For single-agent systems, it offers an alternative approximation scheme for evaluating expectations of state distributions, which is shown to be advantageous in stochastic control problems with high-variance returns.

1.1 Related work

There exists a large body of literature on the topics of multi-agent systems and distributed control. While the overall scope of the field has become overwhelmingly broad, Brambilla et al. (2013) provided a concise summary which focuses on the problems relevant to swarm robotics but still covers many aspects interesting from a general system engineering perspective. In fact, much of the work found there is—either directly or indirectly—concerned with the *micro–macro-link*, which refers to the question how a particular behavior at the agent level (the microscopic scale) translates into the emergent properties of the swarm (the macroscopic scale) and, vice versa, how a certain global phenomenon can be described in terms of local rules (also referred to as the *bottom-up* and *top-down* directions, respectively).

A considerable part of the works in Brambilla et al. (2013) concerned with these questions is based on *rate equation models* (see, for example, Lerman et al. 2005; Martinoli et al. 1999; Correll and Martinoli 2006) which describe the state transitions of the agents in the system on a global scale and, as such, provide the highest level of abstraction from the microscopic agent-based paradigm toward a macroscopic system description. In fact, these models completely ignore the spatial properties of a system and can, therefore, only provide a global view on the system dynamics, without modeling potentially important local effects caused by the interactions of the agents. Two alternative types of models, which explicitly capture the spatial dynamics of a system, are *continuous spatial automata* (MacLennan 1990) and *amorphous media* (Beal 2005; Abelson et al. 2000). Although these models have not emerged directly from the classical agent-based paradigm, the underlying methodology shows many parallels to the continuum concept described in our work. Coming from the side of agent-inspired modeling, there are finally the frameworks of *mean field games* (Lasry and Lions 2007; Aumann 1964) and *Brownian agents/active particles* (Schweitzer 2003) which, similar to our work, describe the spatial interactions of the agents at a macroscopic level, using a continuum description of the system dynamics. Moreover, the model in Schweitzer (2003) was extended by Hamann and Wörn (2008) to account for direct communication between the agents. While these frameworks provide elegant mathematical tools to model the collective large-scale behavior of a system, most of the work in this area focuses on analyzing the dynamics of the systems and assumes that the underlying policy of the agents is known. In contrast, our work considers the system process from an engineering perspective and explicitly addresses the question of how we can optimize the local interactions of the agents so as to promote a certain type of macroscopic system behavior. To this end, we not only introduce an explicit policy model as a new design parameter, but also particularize how this model can be trained via different feedback mechanisms from the system process.

1.2 Paper outline

Section 2 starts with a finite-size agent-based model, where we introduce all relevant system components. In Sect. 3, we derive the continuum version of this model, together with the corresponding continuum-related system quantities. In both sections, particular attention is paid to the reward mechanisms associated with the systems, and a comparison of the models is provided at the end of Sect. 3. In Sect. 4, we then consider the problem of system optimization and present a reinforcement learning scheme for approximate optimal control. Simulation results for three different problem settings are presented in Sect. 5 before we finally conclude our work in Sect. 6 and point to some future research directions.

2 The agent model

Swarming behavior is ubiquitous in both natural and technological systems. Despite the fact that two swarm systems might operate in fundamentally different domains, their underlying mechanisms are often closely related and different systems often share a similar high-level objective, such as achieving consensus across the agents (e.g., birds synchronizing their flight direction versus sensors in a network collectively estimating some environmental quantity) or the localization of a target (e.g., ant colonies localizing a food source versus chemotaxis in bacterial swarms). In the subsequent sections, we provide two alternative views on modeling the swarming behavior of such systems, where we introduce a number of system-related quantities that describe the inherent properties of the agents in the system, such as their local states, observations, and controls. While neither of the two presented system models is tailored to a specific kind of swarm system, it is oftentimes helpful, for the sake of illustration, to imagine a swarm of robotic agents, where the state variable $X_i(t)$ represents the physical location of robot i at time t and $u_i(t)$ denotes the executed motor control. It is important to keep in mind though that, in general, the state X_i will contain all quantities related to the dynamics of an agent (such as its velocity, acceleration, mass, or even internal quantities that affect the agent's behavior, see discussion on *agent types* in Sect. 6.1). Accordingly, the agents can operate in some abstract space, such as the belief space of possible environmental parameters in the example of the sensor network, or the space of possible trading strategies in a network of traders.

2.1 System dynamics

We start by considering a finite-size system of N agents. Each agent $i \in \{1, \ldots, N\}$ is associated with a *local state*, denoted $X_i(t) \in \mathcal{X} \subseteq \mathbb{R}^d$, by which we describe the agent's trajectory in our model. Herein, $t \in \mathbb{R}_{\geq 0}$ denotes time and $\mathcal{X}$ is the d-dimensional state space of the system. The collection of all agents' states is referred to as the *global state* of the system, which we represent by the time-dependent state matrix $X(t) := [X_1(t), X_2(t), \ldots, X_N(t)] \in \mathcal{X}^N \subseteq \mathbb{R}^{d \times N}$.

The interaction of the agents is modeled by a set of locally coupled stochastic differential equations. More specifically, we assume that each agent's trajectory is described by a controlled Itô diffusion (Krylov 2008),

$$\mathrm{d}X_i(t) = h\big(X_i(t), u_i(t)\big)\mathrm{d}t + \mathrm{d}W_i(t), \tag{1}$$

where $u_i(t) \in \mathcal{U} \subseteq \mathbb{R}^U$ is the *local control command* executed by agent i at time t, the function $h : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^d$ describes the agent's dynamics, and $\mathcal{U}$ denotes the set of admissible controls. Lastly, $W_i(t) \in \mathbb{R}^d$ represents a noise term, which we model as a Wiener process with independent components for each state dimension. In particular, we assume that the noise processes $\{W_i\}_{i=1}^N$ are pairwise independent across agents, so that we may write

$$\mathbb{E}\left[W_{i,m}(t)W_{j,n}(t')\right] = 2D\delta_{m,n}\delta_{i,j} \min\left(t, t'\right),$$

where δ denotes Kronecker's delta, $W_{i,m}(t)$ refers to the mth component of the noise process affecting agent i at time t, and D is a scalar constant referred to as the diffusion coefficient. The role of the noise term is twofold: on the one hand, it allows us to model systems whose state transitions are inherently random (i.e., systems with truly stochastic state dynamics); on the other hand, it can be used to summarize the effect of unmodeled system parameters (e.g., a change of an agent's state due to external forces). In both cases, the underlying assumption is that the resulting state increments can be well described by a Gaussian distribution (Billingsley 1999). Note that this assumption does not carry over to the state variable X_i itself, which will be non-Gaussian in general due to the potentially nonlinear system dynamics h.

2.2 Observation model

The extent to which agent i adjusts its local state $X_i(t)$ via $u_i(t)$ in Eq. (1) depends on the states of the other agents or, more precisely, on agent i's k-dimensional *local observation* $Y_i(t) \in \mathcal{Y} \subseteq \mathbb{R}^k$ of the system state $X(t)$. Herein, $\mathcal{Y}$ is referred to as the observation space of the agents. Formally, we describe these agent-related observations through an *observation model* $\xi : \mathcal{X} \times \mathcal{X}^N \to \mathcal{Y}$, which specifies how the global agent configuration is perceived from any state in $\mathcal{X}$, i.e., $\xi(x, X(t))$ tells us how the system state $X(t)$ is perceived from position x in the state space. Consequently, ξ relates $X(t)$ to the local agent observations $\{Y_i(t)\}$; for instance, agent i's local observation $Y_i(t)$ can be accessed as

$$Y_i(t) = \xi\big(X_i(t), X(t)\big).$$

Note that a generalization to stochastic observation models is possible but not considered in this work (see also discussion on stochastic controllers in Remark 4). The reason why we adopt a two-argument notation $\xi(x, X(t))$ with an independent observer state x instead of resorting to explicit agent-based indexing will become clear when we switch to the continuum model in Sect. 3.

Since the agents in a swarm are assumed to be identical and thus interchangeable, we allow only symmetric observation models which ignore the ordering of the agents and respect their interchangeability. More specifically, we assume the observation model to be of the form

$$\xi\big(x, X(t)\big) = \frac{1}{|\mathcal{N}_{\langle x\rangle}|} \sum_{i \in \mathcal{N}_{\langle x\rangle}} g\big(x, X_i(t)\big), \tag{2}$$

where g is the *interaction potential* explained below and $\mathcal{N}_{\langle x\rangle}$ denotes the set of all agents in some *spatial neighborhood* $\mathcal{B}(x) \subseteq \mathcal{X}$ of x,

$$\mathcal{N}_{\langle x\rangle} = \big\{i : X_i(t) \in \mathcal{B}(x)\big\}.$$

Note that we keep the time dependence of this set implicit to simplify our notation. To access the *agent–neighborhood* of a specific agent i, we further introduce the short-hand notation $\mathcal{N}_i := \mathcal{N}_{\langle X_i(t)\rangle}$ and write N_i to denote its cardinality, i.e., $N_i := |\mathcal{N}_i|$.

The set $\mathcal{B}(x)$ can be thought of as the "observable region" of an agent located at x, that is, the set of observable other agents' states. Within this region, the vector-valued function $g : \mathcal{X} \times \mathcal{X} \to \mathcal{Y}$ describes the partial contribution of a specific agent to another agent's local percept of the system state. More specifically, $g(x, y)$ tells us how an individual agent at state y is perceived from state x or, equivalently, how an agent at x is influenced by an agent at y. We will see various examples of this function in Sect. 5. Because the agents compute their observations based on their local neighborhood, we naturally assume that each agent i knows its neighborhood size N_i.

Remark 1 (Partial observability) Partial observability is a natural characteristic of almost all types of distributed systems, and hence, it is an integral part of our model. Note that partial observability can be generally caused by two independent aspects, which are both explicitly reflected in our observation model: (i) the limitation of the agents' observation/interaction range through the neighborhood set $\mathcal{B}$ and (ii) the restricted information content of the observations within this neighborhood, here determined by the interaction potential g. The former is typically a direct consequence of the agents' (physically) limited observation capabilities. The latter follows from the sensing modalities used by the agents as encoded by g, where the summation in Eq. (2) arises from the inherent homogeneity assumption of our model: not

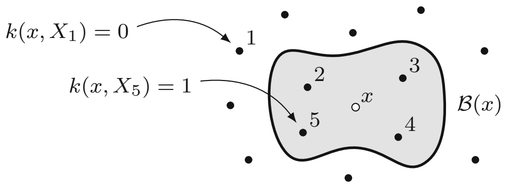

Fig. 1 Schematic illustration of the neighborhood scheme. The spatial neighborhood $\mathcal{B}(x)$ of state x is highlighted in gray. In this region, the neighborhood indicator function k returns the value 1; outside, the function value is 0. Accordingly, the agent–neighborhood at state x is $\mathcal{N}_{\langle x \rangle} = \{2, 3, 4, 5\}$

only are the agents homogeneous in their architecture, they also treat other agents in their neighborhood interchangeably (i.e., agent j located at state x has the same effect on agent i as agent k located at x). This interchangeability of agents is a necessary requirement for our continuum model in Sect. 3, where each agent will be eventually surrounded by an infinite number of other agents so that a discrimination between neighbors becomes impossible.

Remark 2 (Indicator function) Note that Eq. (2) can be written in a more explicit form, which is used for the derivation of the continuum model in Sect. 3,

$$\xi\big(x, X(t)\big) = \frac{\sum_{i=1}^{N} g\big(x, X_i(t)\big)k\big(x, X_i(t)\big)}{\sum_{j=1}^{N} k\big(x, X_j(t)\big)}. \tag{3}$$

Herein, the function $k : \mathcal{X} \times \mathcal{X} \rightarrow \{0, 1\}$ indicates whether an agent at some state y is in the neighborhood of another state x,

$$k(x, y) = \begin{cases} 1 & \text{if } y \in \mathcal{B}(x), \\ 0 & \text{otherwise.} \end{cases}$$

An illustration of the neighborhood scheme is provided in Fig. 1.

Remark 3 (Environmental information) So far, the local observations $\{Y_i(t)\}$ have been defined in terms of pairwise interactions between the agents, as indicated by the summation in Eq. (2). One way to include also environmental information (i.e., information that is independent of the states of other agents) is to extend the observation model by an additional number of dimensions which explicitly depend only on the agent's location in the state space. However, we can stick with the more compact form in Eq. (2) without loss of generality if we allow some dimensions of g to implicitly depend only on the first argument passed to the function. We will see two examples of this in Sect. 5.

2.3 System controller

Having introduced the system dynamics and the observation model, we now focus on the control signals $\{u_i(t)\}$ and define the coupling between the agents. In order to preserve the homogeneity of our system, we specify this coupling in the form of a system-wide *control policy* $\pi_\theta : \mathcal{Y} \rightarrow \mathcal{U}$,

$$u_i(t) = \pi_\theta\big(Y_i(t)\big), \tag{4}$$

which translates the agents' observations $\{Y_i(t)\}$ into local control commands $\{u_i(t)\}$. The policy is parameterized by a *control parameter* $\theta \in \Theta \subseteq \mathbb{R}^C$ with corresponding parameter space Θ. Note that Θ implicitly defines the set of admissible controls $\mathcal{U}$ through the form of π_θ. In Sect. 4, our goal will be to find an optimal θ such that the resulting system performs a given task; for now, however, we may assume that θ is fixed. Putting all ingredients together, we can write Eq. (1) in the more explicit form

$$\begin{aligned} \mathrm{d}X_i &= h\Big(X_i, \pi_\theta\big(\xi(X_i, X)\big)\Big)\mathrm{d}t + \mathrm{d}W_i \\ &= h\Bigg(X_i, \pi_\theta\Bigg(\frac{1}{N_i}\sum_{j\in\mathcal{N}_i} g(X_i, X_j)\Bigg)\Bigg)\mathrm{d}t + \mathrm{d}W_i. \end{aligned} \tag{5}$$

Remark 4 (State exploration) By restricting our model to the use of *deterministic reactive* controllers (that is, $u_i(t)$ is a function of the current observation $Y_i(t)$ only, see discussion in Sect. 6.1), we focus on settings with simple agents that do *not* explicitly face the problem of information gathering. In particular, inspired by natural swarm systems, we assume a simplistic system architecture which bypasses the requirement of long-term agent memory by replacing exploration strategies executed at the agent level with collective strategies deployed at the swarm level.[2] As we show in our results section, this architecture is, in fact, sufficient to solve a number of common swarm-related tasks. However, it should be mentioned that there exist settings in which an extension to stochastic system controllers may be necessary, for example, to model effect of spontaneous symmetry breaking in natural swarms. For the special case of linear system dynamics h, an extension to stochastic Gaussian control policies is straightforward since the stochastic component of the controller can be absorbed into the diffusion coefficient D of the noise processes $\{W_i\}$. More general cases require further generalization steps (e.g., stochastic integration with respect to Lévy processes (Karatzas 1998; Dubkov and Spagnolo 2005)), which are not considered in this work.

Remark 5 (Information processing) In the above-described model, π_θ operates directly on the raw (cumulative) sensory information acquired by the agents. The inclusion of higher-level observational features can be realized with the help of an additional preprocessing stage that first extracts this high-level information from the agent's sensory input before passing it to the controller. However, both model formulations are mathematically equivalent since any preprocessing stage can be absorbed into π_θ, so that we may as well stick with the more compact form in Eq. (5). Nevertheless, for the overall picture it can be helpful to keep this preprocessing step in mind, e.g., when the system policy operates only on a subset of the available sensory inputs (see Sect. 5.3 for an example).

2.4 Reward models

With regard to the forthcoming policy optimization step in Sect. 4, we further need to specify how to measure the performance of our system. In a reinforcement learning context, this is classically done via a *reward signal* $r(t)$ which provides an instantaneous assessment of the system's state. The overall system performance [in the reinforcement learning literature also referred to as the *value* (Sutton and Barto 1998)) can then be defined as the total reward accumulated over a certain period of time T,

$$V_\theta = \mathbb{E}\left[\int_0^{\mathrm{T}} r(t)\,\mathrm{d}t \;\middle|\; \pi_\theta\right]. \tag{6}$$

[2] Note that both these exploration types are different from the exploration in policy space, which we discuss in detail in Sect. 4.

Herein, the expectation is with respect to the random system trajectory generated under π_θ, where we assume that the initial system state is drawn from some fixed state distribution. Note that the relationship of the reward signal $r(t)$ to the system process $X(t)$ depends on the particular choice of reward model. In particular, since our system is of distributed nature, we can define the reward signal either globally or locally, as will be explained in the following two sections.

2.4.1 Global reward model

A straightforward approach to constructing a reward signal is to define a performance measure directly on the global system state,

$$r^G(t) := R^G\big(X(t)\big), \tag{7}$$

where $R^G : \mathcal{X}^N \to \mathbb{R}$ is a score function assessing the global agent configuration. For example, in a positioning task, R^G could be the (negative) sum of the agents' distances to a target location or, alternatively, in an aggregation task, it could be the (negative) average pairwise distance between the agents.

From a centralized learning perspective, the global reward signal $r^G(t)$ can be considered as a direct feedback from the system state $X(t)$ to a central critic system which dictates the system dynamics through π_θ and utilizes the provided reward information to optimize the agent behavior. In fact, if we treat our system model as a black box and ignore its internal decentralized structure, this setting can be as well interpreted as a single-agent scenario with the central critic system as the only decision-maker.

2.4.2 Local reward model

The computation of the global reward signal as defined in Eq. (7) certainly requires complete knowledge of the global system state $X(t)$ which, in a distributed system with many agents, may not be available at any point in the system. To model this scenario, we thus consider the setting where each agent i is instead provided its own local reward signal $r_i(t)$,

$$r_i(t) := R^L\big(Y_i(t)\big) = R^L\big(\xi(X_i(t), X(t))\big). \tag{8}$$

Herein, $R^L : \mathcal{Y} \to \mathbb{R}$ is a local score function used by the agents to assess their local state configuration captured in the form of the local observations $\{Y_i(t)\}$. Assuming that the reward signals $\{r_i(t)\}$ can be accessed by the critic, we can define our performance measure indirectly via the local agent rewards,

$$r^L(t) := \frac{1}{N}\sum_{i=1}^{N} r_i(t). \tag{9}$$

From a learning perspective, we can think of this scenario as a setting where each agent reports its local reward—in this context to be interpreted as a summary description of its local state—to the central critic system, whose goal it is to optimize the system behavior based on the reported local evidence.

Remark 6 (Global critic, local actors) For both described reward mechanisms, learning takes place centrally at the critic system, irrespective of whether the reward is computed globally or locally. Yet, it is important to stress that, once the learning phase is over, the agents no longer rely on any central information source, and hence, they can act autonomously without further instructions from the critic. The learning architecture thus resembles an actor-critic network (Grondman et al. 2012) in which the global critic system serves as a fusion center during the training period but where the trained actor (i.e., the agents' policy) relies on purely local information (see Hüttenrauch 2017 for a learning strategy which exploits this special structure). Note that one can also think of extensions of this information processing pipeline which do not involve a fusion center at all (see discussion in Sect. 6.1).

Remark 7 (Global versus local reward) The motivation for using a local feedback signal is at the heart of multi-agent learning and distributed system design, where global state information is typically rare. In fact, most distributed systems do not involve any form of centralized information processing, but they are operated purely locally from the agent level. Natural systems (such as bird flocks, ant colonies) are the primary example. For these systems, it is often not obvious *how* a global reward model needs to be designed in order to induce a certain type of system behavior (Šošić et al. 2017). For instance, one can easily think of ways to model the reward mechanisms of an individual bird (c.f. attraction–alignment–repulsion models (Couzin et al. 2002)), but it is comparably hard to describe the objective of the flock as a whole. In such cases, local reward models lend themselves as natural replacements for a global model to provide the necessary feedback to the critic system. However, due to the partial observability at the agent level, the fused reward signal $r^L(t)$ will generally contain less information about the system state $X(t)$ as compared to a globally computed signal. Consequently, it is more challenging for the critic to learn a performant policy (in terms of the global system behavior) when it has access to $r^L(t)$ only. Nevertheless, depending on the global system objective (i.e., the form of the R^G) and the amount of information carried by the local rewards $\{r_i(t)\}$, the global signal $r^G(t)$ may be sufficiently—or even fully—recovered from the local signals. We will see an example of this in Sect. 5. Note that this relationship holds trivially when R^G is itself composed of local rewards, i.e., when

$$R^G\big(X(t)\big) = \frac{1}{N}\sum_{i=1}^{N} R^L\big(\xi(X_i(t), X(t))\big). \tag{10}$$

2.5 Value estimation

In order to assess a given system policy (and to compute the policy gradient in Sect. 4), we need to estimate its value. Writing Eq. (6) in a more explicit form, we obtain[3]

$$\begin{aligned} V_\theta &= \int_\Omega \int_0^\mathrm{T} R^G\big(X(t;\chi)\big)\,\mathrm{d}t\,P(\mathrm{d}\chi) \\ &= \int_{\mathcal{X}^N} \int_0^\mathrm{T} R^G(x)p(x,t)\,\mathrm{d}t\,\mathrm{d}x, \end{aligned} \tag{11}$$

where Ω and P denote the underlying sample space and probability measure of the state process X, and $p(\cdot\,,t)$ is the corresponding marginal probability density at time t. The notation $X(\cdot\,;\chi)$ refers to a particular sample path of the system, that is, a specific system trajectory generated under the given system dynamics. To keep the notation simple, we omit the system policy π_θ in the above equations and implicitly assume that X is generated under π_θ according to Eq. (5).

[3] Note that the value in Eq. (11) is based on a global definition of reward. We can easily switch to a "local" (i.e., agent-based) value computation by choosing R^G as in Eq. (10), which is in accordance with the definition of *private value* in Šošić et al. (2017).

Having the form of an expectation, V_θ can be approximated by Monte Carlo integration,

$$\hat{V}_\theta^{\{S\}} = \frac{1}{S} \sum_{s=1}^{S} \int_0^{\mathrm{T}} R^G \left(X \left(t; \chi^{\{s\}} \right) \right) \mathrm{d}t, \tag{12}$$

where $\chi^{\{s\}} \sim P$, and S is a specified number of system roll-outs. The remaining integral can be replaced by a discrete-time approximation, as will be explained in Sect. 4.

2.6 The agent model in the Markov decision process context

The described agent-based system can be interpreted as a continuous-time version of the swarMDP model presented by Šošić et al. (2017) for the special case where the global transition dynamics of the swarMDP decouple over the agents. The swarMDP constitutes a sub-class of the more general dec-POMDP model class (Bernstein et al. 2002) with a homogeneous agent architecture. Figure 2 depicts the basic structure of the system in the form of a Bayesian network, reflecting the dependencies between all important variables. Note that, in the figure, we switch to subscript notation for the time variable to highlight the discrete-time nature of this model.

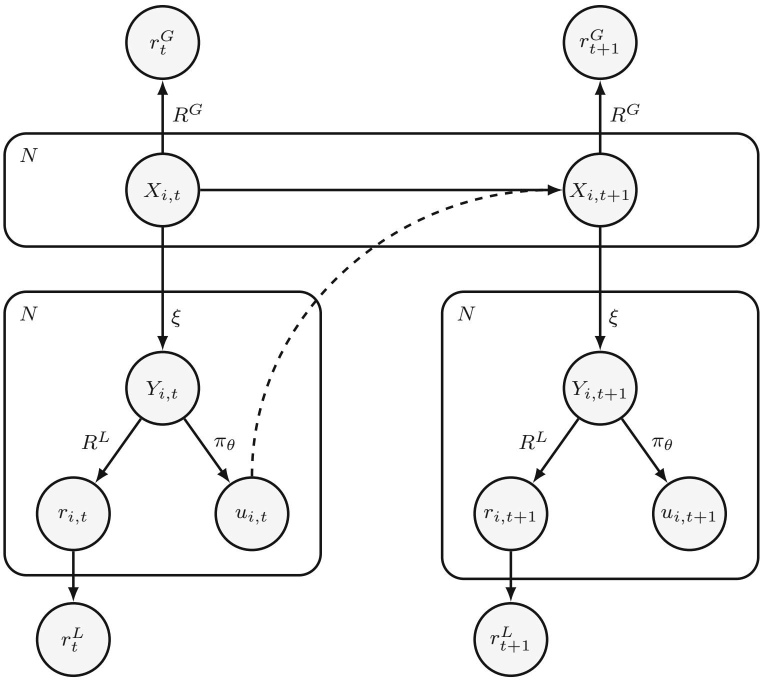

Fig. 2 Graphical model of the discrete-time N-agent system, illustrated as a swarMDP Bayesian network (Šošić et al. 2017) whose transition dynamics decouple over agents. The local observation $Y_{i,t}$ of agent i at time t is determined via the observation model ξ, based on the local states $\{X_{j,t} : j \in \mathcal{N}_i\}$ of other agents in its neighborhood. The agent then transitions to a next state $X_{i,t+1}$ by executing a control command $u_{i,t}$ which is chosen according to the control policy π_θ (the dashed arrow indicates connections between variables belonging to the same agent). At each time instant, a local reward r_t^L and a global reward r_t^G can be computed via the functions R^L and R^G, based on the agents' local observations and the global system state, respectively

3 The continuum model

While the basic model equations (5) are valid for arbitrary system sizes, the computational complexity of the agent-based model grows quadratically with the number of agents in the system, due to the involved pairwise interactions. In this section, we seek for an alternative model description which can be used in cases where the number of agents is large, i.e., when $N \to \infty$. To this end, we derive a continuum model from the agent system as a computational tool which allows us to approximate the behavior of the collective in large-scale regimes. Along these lines, we provide the corresponding reward mechanisms which are required to formalize the learning problem in the continuum limit.

3.1 The continuum equation

In order to describe the system dynamics for large agent numbers, it is convenient to switch to a more compact system representation which avoids enumerating each agent individually. The reason is that, as $N \to \infty$, the individual contribution of a specific agent to the global system behavior becomes negligible; hence, a continuum-based system model, which describes the dynamics of the collective on a macroscopic scale, offers a more efficient representation of the system's state. In fact, in a continuum of agents, there is no meaning to the notion of an individual agent. Accordingly, we will drop the concept of using agent-specific quantities—such as states $\{X_i(t)\}$ and observations $\{Y_i(t)\}$—to describe our system and replace these objects with corresponding continuum-based quantities. To this end, we consider the global *N-agent density*[4] as described by Dean (1996),

$$\rho^{(N)}(x,t) := \frac{1}{N}\sum_{i=1}^{N} \rho_i(x,t), \tag{13}$$

which is defined in terms of the following single-agent density functions,

$$\rho_i(x,t) := \delta\big(x - X_i(t)\big), \tag{14}$$

where we write $\delta(\cdot)$ to denote Dirac's delta function. Note that the state matrix $X(t)$ and the density $\rho^{(N)}(x,t)$ can be considered as equivalent descriptions of the system state since we generally ignore the ordering of the agents.

In the limit $N \to \infty$, the temporal evolution of this quantity, which we in this context refer to as the *continuum density*, is described by the following convection–diffusion equation,

$$\frac{\partial \rho(x,t)}{\partial t} = -\nabla \cdot \Big[\rho(x,t) h\big(x, \bar{u}(x,t)\big)\Big] + D\nabla^2 \rho(x,t), \tag{15}$$

where the first component on the right-hand side replaces the drift term in Eq. (5) and the second component models the cumulative effect of the agent-dependent noise processes. In this equation, the newly introduced *control field*

$$\bar{u}(x,t) := \pi_\theta\big(\bar{y}(x,t)\big)$$

and the underlying *observation field*

$$\bar{y}(x,t) := \frac{\int_{\mathcal{X}} \rho(y,t) g(x,y) k(x,y)\,\mathrm{d}y}{\int_{\mathcal{X}} \rho(y',t) k(x,y')\,\mathrm{d}y'}$$

[4] This function is not to be confused with the probability density function of a single agent's state (see Sect. 3.4) which, in contrast to the object defined here, is a deterministic quantity.

take over the roles of the agent-specific control and observation signals $\{u_i(t)\}$ and $\{Y_i(t)\}$, respectively. A detailed derivation is provided in Appendix.

In Fig. 3, we provide illustrations of the N-agent density and the continuum density based on the example of the Kuramoto model (see Sect. 5). Note that the two concepts are closely related to the *microscopic density* and the *spatiotemporal field* found in Schweitzer (2003).

3.2 Reward models

In the following two paragraphs, we show how the reward models discussed in Sect. 2.4 translate to the continuum case.

3.2.1 Global reward model

For the centralized setting, the reward mechanism can be defined in a similar way as for the agent-based model, with the subtle difference that the reward is no longer a function of the state matrix $X(t)$ but becomes a functional of the continuum density $\rho(x,t)$,

$$r^G(t) = R^G\big(\rho(\cdot\,,t)\big), \tag{16}$$

where $R^G : \mathcal{M} \to \mathbb{R}$, and $\mathcal{M}$ is the set of density functions on $\mathcal{X}$,

$$\mathcal{M} = \left\{ f : f(x) \geq 0\ \forall x \in \mathcal{X}, \int_{\mathcal{X}} f(x)\,\mathrm{d}x = 1 \right\}.$$

While the mathematical forms of the continuum-based and the agent-based reward model differ, both models are conceptually equivalent because $\rho(x,t)$ effectively takes over the role of the state matrix $X(t)$ in the continuum. In that sense, $r^G(t)$ remains a performance measure defined on the global system state.

3.2.2 Local reward model

For the local setting, we derive the corresponding reward model directly from the agent-based framework. Starting from Eqs. (8) and (9), we write the fused reward signal as

$$r^L(t) = \frac{1}{N}\sum_{i=1}^{N} R^L\big(\xi(X_i(t), X(t))\big).$$

As in the derivation of the continuum equation (see Appendix), we express this quantity via the N-agent density $\rho^{(N)}(x,t)$,

$$r^L(t) = \int_{\mathcal{X}} \rho^{(N)}(x,t) R^L\big(\xi(x, X(t))\big)\,\mathrm{d}x,$$

and make use of the observation field $\bar{y}^{(N)}(x,t)$ to simplify the notation,

$$r^L(t) = \int_{\mathcal{X}}^{(N)} (x,t) R^L\big(\bar{y}^{(N)}(x,t)\big) x.$$

Analogous to the control field $\bar{u}^{(N)}(x,t)$, we now introduce a reward field $\bar{r}^{(N)}(x,t)$ which provides the reward of a (hypothetical) agent coupled to the system at state x,

$$\bar{r}^{(N)}(x,t) := R^L\big(\bar{y}^{(N)}(x,t)\big).$$

With this definition at hand, we finally get

$$r^L(t) = \int_{\mathcal{X}} \rho(x,t)\bar{r}(x,t)\,\mathrm{d}x, \tag{17}$$

where we replaced all finite-size quantities with their continuum equivalents.

3.3 Value estimation

In Sect. 2.5, we saw that the value of the agent-based system can be estimated via Monte Carlo integration, based on a finite number of independent system roll-outs. For the continuum model, whose system behavior is uniquely described by the deterministic continuum density $\rho(x,t)$, the estimation scheme is fundamentally different. Here, it is sufficient to consider a single roll-out per initial system state, requiring no repetitions of the process,

$$V_\theta^\rho = \int_0^{\mathrm{T}} R^G\big(\rho(\cdot\,,t)\big)\mathrm{d}t. \tag{18}$$

In contrast to the Monte Carlo estimator in Eq. (12), the estimation accuracy is hence not limited by the number of experiments but by the accuracy of the chosen representation for the continuum density ρ.

3.4 Continuum-based modeling versus agent-based modeling

Having introduced two alternative views on the same problem, we would like to wrap up the previous sections by highlighting some of the main differences between the two modeling approaches and pointing out some important implications.

3.4.1 Single-agent systems and decoupled system dynamics

So far, we have considered the continuum framework under the implicit assumption that the agents in the system interact cooperatively with each other in a joint state space. Interestingly, a continuum formulation is not only useful for modeling cooperative decision-making problems but also in the special case of single-agent systems or—equivalently—for decoupled homogeneous systems where no interaction occurs. In such a decoupled system, the simulation of the N-agent state $X(t)$ corresponds to simulating N independent roll-outs of the unique underlying single-agent system. Hence, in this context the continuum density becomes equivalent to the probability density function of that single agent's state (see Eq. (11)), i.e., it follows that

$$\rho(x,t) = p(x,t), \tag{19}$$

and the continuum equation (15) reduces to the Fokker–Planck equation (Risken et al. 1996) that describes the state evolution of the agent. This equivalence holds because, mathematically, there is no difference in modeling the probability of state occurrence of a single agent or modeling the concentration profile of infinitely many identical decoupled agents. Changing our viewpoint, however, allows us to switch from the sample-based estimator in Eq. (12) to the continuum computation in Eq. (18) which, by modeling the agent distribution (19) directly, realizes an average over *all* sample paths of the system. More precisely, for $S \to \infty$ it holds that

$$\hat{V}_\theta^{\{S\}} \xrightarrow{a.s.} V_\theta^\rho,$$

by the law of large numbers (see Eqs. 12, 18). Due to this virtually infinite sample size, a continuum-based system model can be advantageous in scenarios with high-variance returns as, for example, in problems with sparse or fragmented reward structures (see Sect. 5.4 for an example). This argument holds true not only for the system value but for the computation of arbitrary expectations with respect to the agent's state distribution.

3.4.2 Cooperative multi-agent systems

In the case of cooperative systems with coupled agents, the continuum approach offers an additional benefit. Due to the pairwise interactions of the agents, the global behavior of these systems will, in general, depend on the number of agents involved in the decision-making process. With the need to simulate these interactions, an agent-based approach to approximating the limiting system behavior can result in a prohibitively large computational burden. As $N \to \infty$, the continuum approach not only provides a more efficient approximation to this limit but also a more accurate one in the sense that it bypasses finite-size effects which would arise in any agent-based simulation. Consequently, the continuum formulation provides an elegant alternative to classical agent-based computation for both, single-agent and cooperative systems. However, being a deterministic system model, the continuum model cannot be used to describe the erratic behavior of an individual agent at the microscopic scale and, consequently, for small system sizes the predicted system behavior may differ significantly from that of the corresponding agent model (Houchmandzadeh and Vallade 2015; Ohkubo et al. 2008). If the focus lies on analyzing the small-scale dynamics of a system, the agent-based approach is thus to be preferred over a continuum approximation.

3.4.3 Computational cost

From a computational point of view, the two modeling approaches have complementary strengths and weaknesses: the feasibility of conducting an agent-based simulation largely depends on the size of the system (see previous paragraph), while for the continuum framework the computation limits are set by the dimensionality of the system state space which dictates the complexity of the required continuum representation. A promising future research direction is, therefore, to consider hybrid approaches that combine the merits of both methods (see discussion in Sect. 6.1).

4 Reinforcement learning

In this section, we present a simple reinforcement learning strategy to optimize the behavior of our system. Herein, we restrict ourselves to policy gradient methods (Deisenroth et al. 2013), which provide a model-free learning paradigm to perform the system optimization directly in the parameter space of the policy.

While our model is defined in terms of stochastic differential equations and, as such, operates in continuous time, we conduct the policy optimization on a discretized time scale, with regard to the forthcoming numerical simulation of the system on a finite time grid. To this end, we introduce the random variable $\mathcal{R}$, henceforth referred to as the sample return, which measures the (local or global) reward signal $r(t)$ at regular time intervals $\Delta := \frac{\mathrm{T}}{M}$,

$$\mathcal{R} := \sum_{k=0}^{M} r(k\Delta), \tag{20}$$

for some $M \in \mathbb{N}$. Note that the distribution of $\mathcal{R}$, which we denote as $p_{\mathcal{R}}(\mathcal{R} \mid \theta)$, depends on the control parameter θ due to the inherent dependence of $r(t)$ on the system policy. Based on the sample return, we define our objective function J_θ as

$$J_\theta := \int_{-\infty}^{\infty} p_{\mathcal{R}}(\tau \mid \theta)\, \tau\tau. \tag{21}$$

Up to a rescaling with Δ, J_θ can be regarded as a discrete-time approximation of the continuous-time value in Eq. (6) and may, for small enough Δ, be used as a proxy. Hence, an immediate learning objective would be to optimize J_θ with respect to θ. However, such an approach is unfavorable for the following reason. Assuming that the system policy π_θ is parameterized as a deterministic function of the agent observation (see Sect. 2.3), computing the policy gradient requires a model of the underlying return-generating mechanism in order to compute the involved function derivatives (Sehnke et al. 2010). Although in principle such a model could be derived from the system equation (5), this approach would clearly destroy the black box character (and hence the general applicability) of the learning scheme.

An apparent solution to the problem seems to be the replacement of the deterministic system policy π_θ with a stochastic one, which not only resolves the above-mentioned problem but which is also advantageous for the purpose of exploration (see Remark 4). However, using a stochastic policy artificially injects noise into the system which, in turn, increases the variance of the resulting gradient estimator. The effect is further amplified by the discrete-time simulation of the model since the number of random local controls fed into the system process increases with the temporal resolution $\frac{1}{\Delta}$. In the extreme case, this can degenerate the gradient estimate up to a point where learning becomes impossible (Munos 2006).

Algorithm 1: PARAMETER EXPLORING POLICY GRADIENT

```
Input:   return-generating process $p_{\mathcal{R}}(\tau \mid \theta)$, learning rate $\alpha$
Output:  control parameter $\theta$
1: initialize exploration parameters $\omega$ randomly
   while stopping criterion not fulfilled
       for $q = 1, 2, ..., Q$
2:         sample control parameter: $\theta^{\{q\}} \sim p(\theta \mid \omega) = \prod_{c=1}^{C} \mathcal{N}(\theta_c \mid \omega_c, \sigma^2)$
           for $s = 1, 2, ..., S$
3:             sample return value from the process: $\mathcal{R}^{\{q,s\}} \sim p_{\mathcal{R}}(\tau \mid \theta^{\{q\}})$
           end for
4:         estimate value for $\theta^{\{q\}}$: $\mathcal{R}^{\{q\}} \leftarrow \frac{1}{S}\sum_{s=1}^{S} \mathcal{R}^{\{q,s\}}$
       end for
5:     compute baseline: $b \leftarrow \frac{1}{Q}\sum_{q=1}^{Q} \mathcal{R}^{\{q\}}$
6:     estimate policy gradient: $\delta\omega \leftarrow \frac{1}{Q\sigma^2}\sum_{q=1}^{Q}(\theta^{\{q\}} - \omega)(\mathcal{R}^{\{q\}} - b)$
7:     update exploration parameters: $\omega \leftarrow \omega + \alpha \cdot \frac{\delta\omega}{\|\delta\omega\|_2}$
   end while
8: return $\theta \leftarrow \omega$
```

A principled approach to handling this dilemma is provided by the *parameter exploring policy gradient* (PEPG) (Sehnke et al. 2010). The algorithm makes use of an explicit upper level *exploration policy* $p(\theta \mid \omega)$, which is parameterized by an exploration parameter ω and defined on the control parameter space Θ itself. Accordingly, the objective function (21) is expanded to an upper level criterion.

$$\mathcal{J}_\omega := \int_\Theta p(\theta \mid \omega) J_\theta \, \mathrm{d}\theta,$$

which measures the expected system performance for a given exploration parameter ω. To find an optimal value for ω, we apply a normalized gradient ascent procedure,

$$\omega \leftarrow \omega + \alpha z^{-1} \frac{\partial \mathcal{J}_\omega}{\partial \omega},$$

with $z = \| \frac{\partial \mathcal{J}_\omega}{\partial \omega} \|_2$ and some fixed learning rate $\alpha \in (0, \infty)$. Herein, the gradient is given as

$$\begin{aligned} \frac{\partial \mathcal{J}_\omega}{\partial \omega} &= \int_\Theta p(\theta \mid \omega) \frac{\partial}{\partial \omega} \log p(\theta \mid \omega) J_\theta \, \mathrm{d}\theta \\ &= \mathbb{E}\left[\frac{\partial}{\partial \omega} \log p(\theta \mid \omega) J_\theta \right]. \end{aligned} \tag{22}$$

Having the form of an expectation, it can be estimated using Monte Carlo integration,

$$\frac{\partial \mathcal{J}_\omega}{\partial \omega} \approx \frac{1}{Q} \sum_{q=1}^{Q} \frac{\partial}{\partial \omega} \log p(\theta^{\{q\}} \mid \omega) \big[\mathcal{R}^{\{q\}} - b \big],$$

where we first sample the control parameters, $\theta^{\{q\}} \sim p(\theta \mid \omega)$, and keep them fixed while rolling out the system, $\mathcal{R}^{\{q,s\}} \sim p_\mathcal{R}(\tau \mid \theta^{\{q\}})$. An estimate for the value $J_{\theta^{\{q\}}}$ is then obtained as $\mathcal{R}^{\{q\}} := \frac{1}{S} \sum_{s=1}^{S} \mathcal{R}^{\{q,s\}}$, from which we subtract the baseline $b := \frac{1}{Q} \sum_{q=1}^{Q} \mathcal{R}^{\{q\}}$ to reduce the variance of the estimator (Deisenroth et al. 2013). Based on this two-step approach, the exploration process is effectively shifted from the lower to the upper level and is, hence, completely decoupled from the system process, which avoids the gradient degeneracy problem mentioned earlier.

Once an optimal value ω^* is found, the final low-level parameter θ^* can be set, for example, to the mean value,

$$\theta^* := \int_\Theta p(\theta \mid \omega^*) \, \theta \, \mathrm{d}\theta.$$

For our experiments in the subsequent sections, we use a factored Gaussian exploration policy, where the exploration parameters $\{\omega_c\}$ are used to parameterize the mean values of the respective low-level control parameters $\{\theta_c\}$,

$$p(\theta \mid \omega) = \prod_{c=1}^{C} \mathcal{N}(\theta_c \mid \omega_c, \sigma^2).$$

For this class of exploration policy, the gradient of the log density in Eq. (22) takes the form

$$\frac{\partial}{\partial \omega} \log p(\theta \mid \omega) = \frac{\theta - \omega}{\sigma^2},$$

which results in the learning scheme outlined in Algorithm 1.

5 Simulation results

To demonstrate our framework, we consider a system of interacting agents on a circle. The system is based on the well-known model by Kuramoto (1975) (see Fig. 3), which has been used extensively in the past to study the synchronization behavior of groups of coupled

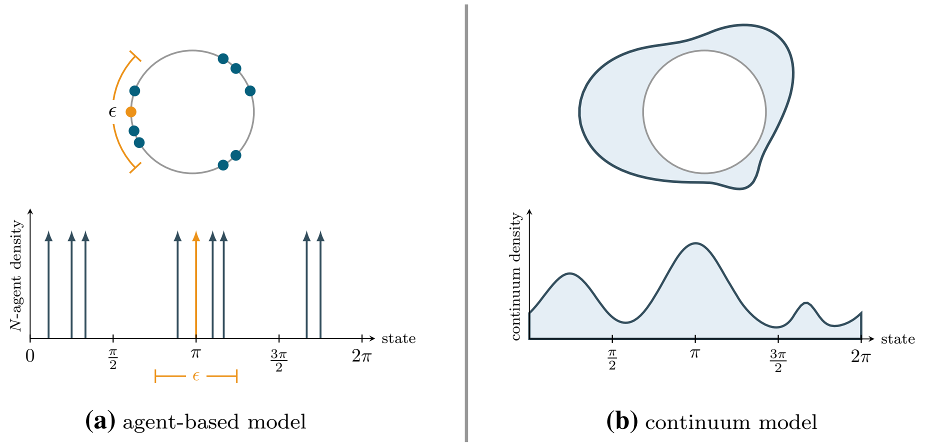

Fig. 3 Illustration of the agent-based Kuramoto model (left) and the continuum version (right). Top: Distribution of the agents/the continuum density on the ring. Bottom: Corresponding N-agent density/continuum density on the state space

oscillators. The basic Kuramoto dynamics are described by the following set of differential equations,

$$\mathrm{d}X_i(t) = \left[\varphi_i + \frac{K}{N}\sum_{j=1}^{N} \sin\left(X_j(t) - X_i(t)\right)\right]\mathrm{d}t + \mathrm{d}W_i(t),$$

where φ_i is the natural frequency of oscillator i and K represents the coupling strength of the system.

Comparing these dynamics to Eq. (5), we observe that the Kuramoto system can be recovered as a special case of our model by assuming state-independent agent dynamics, i.e., $h(x, u) = u$, setting the pairwise interaction potential to $g(x, y) = \sin(y - x)$, coupling the agents globally, i.e., $\mathcal{B}(x) = \mathcal{X}$, and applying a linear control policy, $\pi_\theta(Y) = \theta \cdot Y$. In this context, the control parameter θ takes over the role of the coupling constant K. However, our model is more general than the classical Kuramoto model in the sense that we are not restricted to linear control policies and sinusoidal observations, but we are free to utilize nonlinear system controllers that operate on arbitrary observational features. This gives rise to a broader class of system behaviors, as will be shown in the subsequent sections.

5.1 Simulation setup

For all our experiments, we use the basic simulation setup listed below, if not explicitly stated otherwise. The simulation parameters are hand-selected and adapted to the considered scenario, e.g., to ensure numerical stability of the continuum approximation (see, for example, Schweitzer (2003)) and to provide a sufficient number of learning iterations for the algorithm to converge. Note that all experiments except one (which is part of Sect. 5.4) are conducted using continuum-based simulations. Nevertheless, we will conveniently specify all model-related objects in terms of agent-based quantities; the corresponding continuum model follows accordingly, as described in Sect. 3. In this spirit, we will generally speak of "the agents" when we discuss our results.

System dynamics The continuum dynamics in Eq. (15) are approximated numerically using an explicit finite difference method with a temporal resolution of $\Delta t = 0.01$ and a spatial resolution of $\Delta x = \frac{2\pi}{100}$, resulting in a discretization of the state space $\mathcal{X} = [0, 2\pi)$ into 100 bins. The continuum density ρ is initialized uniformly with small perturbations to break the initial state symmetry. Further, we use a diffusion coefficient of $D = 0.1$ and a maximum simulation time of $\mathrm{T} = 5$.

Learning For the exploration policy of the policy gradient algorithm (Algorithm 1), we use a standard deviation of $\sigma = 0.2$. The gradient is estimated based on $Q = 10$ parameter samples, where we threshold all parameters to the range $[-1, 1]$ after the update to bound the agents' local control signals. For each sample, we perform a single system roll-out ($S = 1$) in order to realize a fair comparison of both model types.[5] During the learning period, the parameters are updated 50 times using a learning rate of $\alpha = 0.2$. Note, however, that the final system performance could be achieved much earlier in most cases.

5.2 Aggregation

As an introductory example, we consider a simple aggregation task in which we teach the agents to accumulate as fast and close as possible. For this purpose, we adopt the original Kuramoto dynamics, i.e., $g(x, y) = \sin(y - x)$ and $\mathcal{B}(x) = \mathcal{X}$, which we extend with a nonlinear system controller π_θ,

$$\mathrm{d}X_i(t) = \pi_\theta\left(\frac{1}{N}\sum_{j=1}^{N} \sin\left(X_j(t) - X_i(t)\right)\right)\mathrm{d}t + \mathrm{d}W_i(t).$$

The controller π_θ is parameterized in the form of a radial basis function (RBF) network whose component amplitudes are controlled by the entries of θ,

$$\pi_\theta(Y) = \sum_{c=1}^{C} \theta_c \cdot \exp\left\{-\left(\frac{Y - q_c}{\gamma}\right)^2\right\}.$$

Herein, $\{q_c\}_{c=1}^{C}$ are the mode locations of the RBF network which we place on a regular grid on the agents' observation space $\mathcal{Y} = [-1, 1]$ (see left part of Fig. 4). Note that the finiteness of $\mathcal{Y}$ arises from the sinusoidal shape of the applied interaction potential g. For the experiment, we use a total of $C = 10$ basis functions whose width is chosen according to the grid spacing, i.e., $\gamma = \frac{2\pi}{C-1}$. During learning, desired system behavior is rewarded by a global reward signal $r^G(t)$ which, in this experiment, takes the form of the system's order parameter (Kuramoto 1975),

$$r^G(t) = \frac{1}{N}\left|\sum_{j=1}^{N} \exp\left(iX_j(t)\right)\right|.$$

Note that i refers to the imaginary unit in the above equation. Figure 4 shows the outcome of 100 Monte Carlo runs of the experiment. The left subfigure depicts the learned control parameters and the resulting system policy, and the right subfigure illustrates the induced behavior. The results demonstrate that the algorithm is able to reliably learn a system policy that solves the aggregation task.

[5] Recall that the continuum model requires only one system roll-out (see Sect. 3.3).

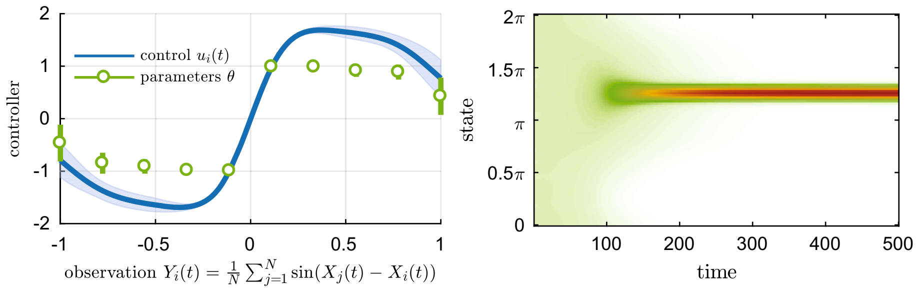

Fig. 4 Results for the aggregation task in Sect. 5.2, based on 100 Monte Carlo runs. Left: Learned system controller. The green circles and bars indicate, respectively, the mean values and standard deviations of the learned control parameters used for the RBF network. The blue curve denotes the average control policy π_θ defined by these parameters, again shown with the corresponding standard deviation. Positive control values let the executing agent drift toward larger angles, and negative values cause a drift in the opposite direction. Right: Example system trajectory generated from a random, approximately uniform initial agent density using the mean control parameters from the left subfigure. The trajectory corresponds to a temporal roll-out of the continuum density as illustrated in Fig. 3. Red color indicates high density values

5.3 Partial observability

In Sect. 2.4.1 we saw that from a centralized perspective, the learning problem can be effectively treated as a single-agent control problem. Yet, the internal distributed character of the system remains and the degree of observability crucially affects the flow of information in the system. The goal of this section is to demonstrate how different observation modalities influence the learning process of the system. To this end, we re-run the experiment from the previous section in a partially observable environment by restricting the interaction of the agents to a limited range $\epsilon \in (0, \pi]$,

$$\mathcal{B}(x) = \left\{y \in \mathcal{X} : |\angle(x, y)| \leq \epsilon\right\},$$

where $|\angle(x, y)|$ denotes the absolute angular distance between the states x and y. Note that the maximum value of $\epsilon = \pi$ exactly recovers the setup in Sect. 5.2. Again, we utilize the global order parameter $r^G(t)$ to train the system but use an extended simulation period of $\mathrm{T} = 20$ to account for the increased difficulty of the task. The result is depicted by the solid red line in Fig. 5 which displays the final reward (i.e., the final order parameter) of the system for different interaction ranges. The plot shows that the system performance quickly breaks down in small-range interaction regimes. This is also reflected in the right subfigure which unveils the inability of the system to learn a functioning control policy for small values of ϵ. The obtained result is not surprising as, for small ϵ, the information available to the agents is not sufficient to solve the aggregation task reliably: due to the limited observation range, it is impossible for the agents to decide in which direction to move in order to form a single global aggregation instead of multiple local ones.

An interesting question is, therefore, whether the agents can develop a functioning strategy if we provide them with additional information. To verify this hypothesis, we equip the agents with additional observation capabilities and let them further sense the relative number of agents in their vicinity, resulting in the following two-dimensional observation,

$$Y_i(t) = \begin{bmatrix} \frac{1}{N_i}\sum_{j\in\mathcal{N}_i}\sin\left(X_j(t) - X_i(t)\right) \\ \frac{N_i}{N} \end{bmatrix}. \tag{23}$$

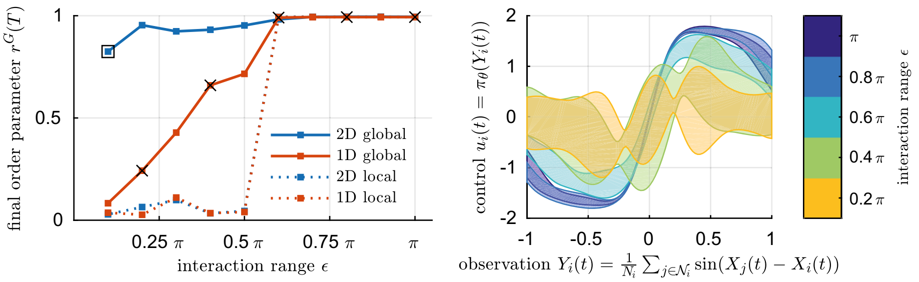

Fig. 5 Results for the partially observable setting in Sect. 5.3, based on 20 Monte Carlo runs. Left: Aggregation performance of the system for different interaction ranges ϵ, measured in terms of the system's final order parameter. Legend: global/local indicates the type of reward signal used during training. 1D refers to the sinusoidal observation model, and 2D refers to the augmented model where the agents can additionally observe the relative agent mass in their vicinity. For large ϵ the state alignment is always successful, whereas for small ϵ a functioning aggregation strategy is only found if the system is trained with global reward information. Right: Standard deviation intervals of the learned control policies (centered around their mean values) for the 1D global setting. The colored areas are equivalent to the blue area shown in the left part of Fig. 4, but correspond to different interaction ranges ϵ, as marked by the black crosses in the left subfigure. For small ϵ, the system is unable to learn a functioning aggregation strategy (reflected by the order parameter shown on the left). For $\epsilon = \pi$, the result from Fig. 4 is recovered

Note that the above model is not directly consistent with Eq. (2); nonetheless, the observation can be computed locally by agent i from its observed features $\xi\big(X_i(t), X(t)\big)$ if we assume that the agents are aware of the total number of agents N in the system (see Remark 5). The latter assumption is fulfilled if we treat N as an observable environmental feature,

$$g(x, y) = \big[\sin(y - x), N\big]^\top.$$

Since the resulting observation space is now two dimensional, i.e., $\mathcal{Y} = [-1, 1] \times [0, 1]$, we use a bivariate RBF network to parameterize the system policy,

$$\pi_\theta(Y) = \sum_{c=1}^{C} \theta_c \cdot \prod_{k=1}^{2} \exp\left\{-\left(\frac{Y_k - q_{c,k}}{\gamma_k}\right)^2\right\},$$

where the subscript k is used to index the observation dimension. As before, we place the modes $\{q_c\}$ of the basis functions on a regular grid on the observation space by discretizing the first dimension into 10 center positions and the second dimension into 4 positions, resulting in a total of $C = 40$ parameters to be learned.

By exploiting the additional local state information, the agents are now able to solve the aggregation task for arbitrary interaction ranges, as is indicated by the blue solid line in Fig. 5. It is particularly interesting to inspect closer the learning progress and the final aggregation strategy found by the algorithm (see Fig. 6). At first, the policy performs no better than the one learned on the one-dimensional observation model as the controller only manages to aggregate the agents locally. However, after a few iterations, the algorithm learns to exploit the additional state information contained in the neighborhood size and assigns different drift velocities to agent groups of different sizes to finally form a rotating group of agents which collects all smaller agent aggregations. This strategy is then optimized toward the end of the learning period. It is worth mentioning that a similar swarm behavior was discovered for an agent-based system in Hamann (2014) using an evolutionary algorithm in the context of a very different learning problem, where the goal of the agents was to predict their local observations one step ahead.

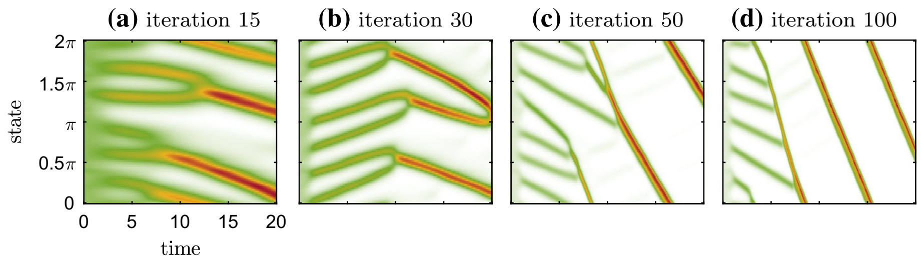

Fig. 6 Learning progress of the system described in Sect. 5.3 when trained with global reward information, using the two-dimensional observation model in Eq. (23) and an interaction range of $\epsilon = \frac{\pi}{10}$ (the setting is indicated by the black square in Fig. 5). Red color indicates high continuum densities. The controller learns an aggregation strategy which exploits the local agent mass to collect all agents by utilizing different drift velocities (Color figure online)

As a final variant of the experiment, we replace the global reward signal $r^G(t)$ with the fused signal $r^L(t)$, assuming that the central critic system has no longer access to the global system state. For this purpose, we let each agent k compute its own "local order parameter"

$$r_k(t) = \frac{1}{|\mathcal{N}_k|}\left|\sum_{j\in\mathcal{N}_k} \exp\left\{i\left(X_j(t) - X_k(t)\right)\right\}\right|,$$

which measures the local alignment of the agent relative to its neighbors. Accordingly, we extend the interaction potential g by a dimension of the form $\exp\{i(y-x)\}$ to provide the agents with the necessary state information. Note that, for $\epsilon = \pi$, the locally computed reward $r^L(t)$ reported to the critic coincides with the global signal $r^G(t)$ that would be obtained in a centralized setting with known global system state (as was the case in Sect. 5.2).

With the entire reward information being computed locally at the agent level, we compare the resulting system performance to the centralized setting, again by measuring the global order parameter of the system (see dashed lines in Fig. 5). The result reveals that, for small ϵ, the aggregation now fails again, even for the augmented observation model in Eq. (23) which additionally contains the relative agent number information. This time, however, the reason for the failure is fundamentally different and can be traced back to the learning phase—the local state information provided to the agents is, in fact, sufficient to solve the problem, as we have seen just before. The problem is rather that the critic cannot tell a locally aggregated system state from a globally aggregated one based on the reported local reward signal because both system states result in similar reward signatures (as measured locally). Consequently, the learning algorithm is unable to develop a functioning strategy that favors one of these two system states.

The result gives rise to the following interesting conclusion: while the local state information of the agents can be sufficient for *executing* a certain task, it might not be sufficient for *learning* the task. Yet, the example also demonstrates that a global system goal can be still achievable through local interaction if the behavior is learned under global supervision. This underpins the idea of guided learning by Hüttenrauch (2017) and motivates the use of a centralized feedback architecture to acquire new skills for a task whose execution can be finally conducted in a decentralized manner. Note, however, that such a learning paradigm not only requires the existence of a central critic system (see discussion in Sect. 6.1) but also a communication channel between the critic and the agents, which is not necessarily available. Hence, in many scenarios, the use of a local reward mechanism will often be the only viable option.

5.4 Aggregation and localization

In our third experiment, we consider a combination of the aggregation and the localization problem, where we teach the agents to collectively approach a target position. Also, we investigate how a finite-size agent system performs compared to the continuum model.

The basic simulation setup is the one described in Sect. 5.2, but we replace the initial state distribution with a Gaussian mixture consisting of two balanced mixture components at 0.3π and 0.7π. In addition to the aggregation reward, we pay a target reward at the end of each training episode that is proportional to the agent mass contained in the state discretization bin located at $\frac{3\pi}{2}$, i.e., in the interval $[\frac{3\pi}{2} - \Delta x, \frac{3\pi}{2})$. The relative weight of this reward compared to the aggregation reward is chosen as 50:1 ($\widehat{=} \frac{\pi}{\Delta x} : 1$), meaning that the paid reward for the case that all agent mass is located inside the target region at the end of an episode is fifty times higher than the reward paid for perfect state synchronization throughout the whole episode. While this ratio was indeed hand-selected, we observed that the following results are largely insensitive to the specific choice of weights in a wide range. In order to be able to locate the newly introduced target reward, we let the agent additionally sense their absolute position in space,

$$g(x, y) = \left[\, \sin(y - x), x \,\right]^\top .$$

Accordingly, we use a two-dimensional RBF representation for the policy based on 6×6 grid positions (as indicated by the black crosses in Fig. 7), resulting in a total of $C = 36$ control parameters.

The rationale behind the described setup is the following: while each agent can collect a high reward by approaching the target region along the shortest path (i.e., independently of the other agents), a higher total reward can be achieved if all agents move coherently and additionally exploit the aggregation reward. However, the task is constructed in such a way that the shortest paths differ for both initialized agent groups, and hence, the agents need to break the symmetry of the initial state in order to decide for a common movement direction. As it turns out, both optimal strategies can be observed in our simulations (see Fig. 7) and, by deactivating either of the two rewards, we can teach the system to focus on only one of the two tasks.

5.4.1 Continuum model versus agent model

As explained in Sect. 3.3, a continuum formulation can be advantageous in situations where the variance of the measured return is high (e.g., when the environment contains a localized reward as in the last example) so that we would need a high number of agent-based system roll-outs to reliably estimate the policy gradient. To demonstrate this effect, we compare the learned behavior of the continuum model with that of the corresponding agent-based system for different agent numbers N. Figure 8 shows the final agent distributions of all systems for their learned policies, averaged over 100 Monte Carlo runs. As expected, for small system sizes the agent-based systems mainly focus on the easier-to-detect aggregation reward and the agents are disctracted from the target region, losing much of the achievable reward. In contrast to this, the continuum system learns the optimal strategy with a success rate of almost 100%, due to its more reliable gradient estimation.

It is worth mentioning that the learned continuum policy can be readily applied to the agent-based systems since both model formulations are compatible with each other. When equipped with this policy, the agent-based systems indeed achieve the same performance level as the continuum model (red line in Fig. 8). Example trajectories for a two-agent system are

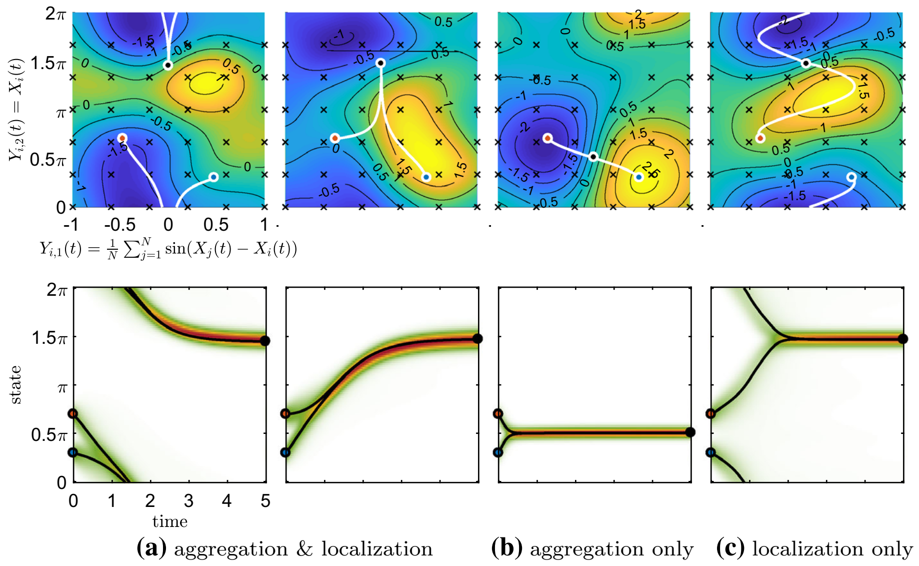

Fig. 7 Simulation results for the aggregation–localization task in Sect. 5.4. Top row: Example policies found for different reward settings. The black crosses mark the center positions of the RBF network. Note that the second dimension (vertical axis), which denotes the position of the agent, is treated cyclically in the parameterization. Bottom row: Resulting system trajectories. Red color indicates high density values. In addition to the continuum density, we further show the trajectories of a noise-free two-agent system executing the learned continuum policy, initialized at the density modes (black lines). These trajectories are projected onto the agents' observation space in the top row (white lines) (Color figure online)

shown by the black and white lines in Fig. 7. For a real-world decision-making problem, this motivates the idea of learning the system policy off-line using a continuum model, and to deploy the learned policy to the real-world system of agents to solve the collective task.

6 Discussion and outlook

6.1 Discussion

We have presented a continuum framework for modeling the collective behavior of large groups of interacting agents in a reinforcement learning context. The proposed framework offers a computational strategy for handling large-scale distributed control problems and, as such, provides an alternative to classical agent-based modeling. Simulation results on a Kuramoto-type of system demonstrated the capabilities of our framework for a number of distributed control tasks. Most notably, our results provide empirical evidence that the framework is able to learn better performing system policies than obtained through an equivalent agent-trained model. Since, apart from modeling multi-agent problems, the presented methodology is also useful for single-agent control (see Sect. 3.4), we expect that our findings can be reproduced in a variety of other settings.

While the obtained results are promising, the presented experiments are preliminary and should be understood as a proof of concept insofar as we used a simple target system with a one-dimensional state space for demonstration purposes. The same methodology applies, of

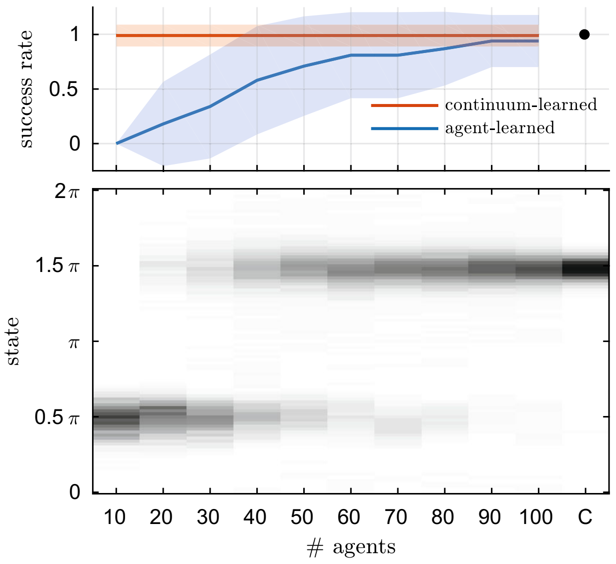

Fig. 8 Performance comparison of the system policies learned by the continuum model and by agent models of different sizes, based on 100 Monte Carlo runs of the aggregation–localization experiment in Sect. 5.4. Both figures share the same abscissa, which indicates the size of the test system (where "C" stands for the continuum model). Top: Success rate (and standard deviation) of learning a policy which drives the system to the target region located at $\frac{3\pi}{2}$. The color of the line indicates whether the system policy was learned using the continuum model or an agent-based system. In the latter case, the training system was of the same size as the test system, indicated by the abscissa value. The test system which corresponds to the black dot was trained using the continuum model. Bottom: Corresponding final state distributions. The shown values correspond to the blue line in the top figure (agent-learned policies) and to the black dot (continuum-learned policy)

course, to higher-dimensional problems, but a direct application to systems with more than a few state dimensions quickly becomes intractable due to the simplistic state discretization approach followed in this paper. At this point, one needs to resort to alternative approximation schemes with better scaling properties. In particular, we would like to point out two promising directions for future research.

The first is to use hybrid approaches that combine the strengths of continuum-based and agent-based simulation. In such an approach, most state dimensions would be treated using an agent-based model while only a few dimensions, which require a fine-grained resolution of states, would be described as a continuum (c.f. Rao–Blackwellized particle filters (Doucet et al. 2000)). This way, we can get the best of both worlds, i.e., the state-dimensional scalability of the agent model and the compactness of the continuum framework. The second approach is to use a more sophisticated and scalable approximation of the continuum density itself, in order to avoid the costly discretization of the system state space in the first place. One possibility is, for example, to resort to spectral approximation methods. Here, the continuum density would be expanded using a small set of basis functions with time-dependent coefficients, yielding a parametric representation of the system state. The coefficients can then be computed using an ordinary differential equation solver (Fornberg and Flyer 2015).

Finally, we would like to discuss two learning-related shortcomings of the current continuum formulation. The first one is the use of purely reactive control policies, i.e., policies which compute the control signal based only on the current observation of an agent. For complex collective tasks, it might be necessary to include (at least parts of) the agents' histories into the decision-making process in order to account for the non-Markovianity of the local observation signals (Kaelbling et al. 1998). In theory, such an extension can be formulated for the continuum model by expanding the observation field $\bar{y}(x,t)$ to a more general field of observation histories; however, such an extension goes beyond the scope of this paper and we leave it for future work.

The second shortcoming is the requirement of a centralized learning architecture. The main reason for using this architecture was that it preserved the homogeneity of our system during the learning phase, i.e., the key property which formed the basis for our continuum formulation. However, we also saw in Sect. 5.3 that global state information, which is only available in a centralized architecture, is generally superior to local state information in terms of the achievable system performance—even when this information is exploited only during the learning phase. Unfortunately, in many applications we will not have the freedom to exploit such a centralized learning architecture but must fall back on purely distributed learning paradigms where the agents share their experiences locally. A promising approach in this direction is to utilize diffusion-based learning strategies to spread the gathered local experience of the agents throughout the network (c.f. Macua et al. 2015). Unfortunately, any local update of the policy in such a learning scheme would inevitably destroy the homogeneity of the system. A potential solution is here to treat the control parameter explicitly as part of an agent's local state, which is tantamount to equipping the agents with an *internal state variable*. As final remark in this context, it should be pointed out that, while the homogeneity *within* an agent population is indeed a critical requirement, the continuum formulation naturally extends to systems which contain different *types* of populations, where each population itself is homogeneous [e.g., predator–prey systems (Ermentrout and Edelstein-Keshet 1993)].

6.2 Outlook

Apart from the general use of our framework in the context of large-scale system modeling, we see at least two immediate applications related to learning and inference in swarm systems which we would like to mention here.

The first is found in the field of inverse reinforcement learning, where the goal is to infer the reward function of a system from the observed system behavior. The solution to this problem usually involves finding a solution for the corresponding forward reinforcement learning problem itself (Michini and How 2012), which either requires a model of the underlying system or, at least, a simulator. Since our model is directly compatible to the swarm model presented in Šošić et al. (2017), it is possible to use the continuum description as a proxy for the actual swarm model in this forward step, in order to exploit the computational and learning-related benefits described in this paper.

Finally, we see a promising application of our framework in combination with multi-agent learning approaches that make explicit use of global state information, such as the guided learning approach described by Hüttenrauch (2017). Representing the global system state in some ordered object like the state matrix $X(t)$ always comes with the drawback that the inherent symmetries of the swarm system, like its permutation invariance, are ignored in large part. This renders the learning process highly inefficient because the algorithm will not be able to exploit any of these symmetries. The continuum representation, which provides a permutation-invariant description of the system state, offers an elegant and natural solution to this problem.

Acknowledgements H. Koeppl gratefully acknowledges support from the German Research Foundation (DFG) within the Collaborative Research Center (CRC) 1053 - MAKI.

Appendix: Derivation of the continuum equation

In the following, we show how the continuum equation (15) can be derived from the agent-based system of stochastic differential equations (1) as the number of agents in the system approaches infinity. Herein, we follow the derivation in Dean (1996), which we extend with the necessary control-related objects.

Our goal is to find an expression for the temporal evolution of the global agent density $\rho^{(N)}(x, t)$ as $N \to \infty$. We start with an Itô expansion of the stochastic differential equation (5),

$$\begin{aligned} \mathrm{d} f\big(X_i(t)\big) = \nabla f\big(X_i(t)\big) \cdot h\Big(X_i(t), \pi_\theta\big(\xi(X_i(t), X(t))\big)\Big)\mathrm{d}t \\ +\ D\nabla^2 f\big(X_i(t)\big)\,\mathrm{d}t + \nabla f\big(X_i(t)\big) \cdot \mathrm{d}W_i(t), \end{aligned} \tag{24}$$

where $f : \mathcal{X} \to \mathbb{R}$ is a twice-differentiable test function. Using the identity

$$f\big(X_i(t)\big) = \int_{x\in\mathcal{X}} \rho_i(x, t) f(x)\,\mathrm{d}x, \tag{25}$$

which follows from the definition of the single-agent density in Eq. (14), we rewrite Eq. (24) as

$$\begin{aligned} \mathrm{d} f\big(X_i(t)\big) = \int_{x\in\mathcal{X}} \Big[\nabla f(x) \cdot h\Big(x, \pi_\theta\big(\xi(x, X(t))\big)\Big)\mathrm{d}t \\ +\ D\nabla^2 f(x)\,\mathrm{d}t + \nabla f(x) \cdot \mathrm{d}W_i(t)\Big]\rho_i(x, t)\,\mathrm{d}x. \end{aligned}$$

Next, we integrate this equation by parts, which yields

$$\begin{aligned} \mathrm{d} f\big(X_i(t)\big) = \int_{x\in\mathcal{X}} \Big\{ -\nabla \cdot \Big[\rho_i(x, t) h\Big(x, \pi_\theta\big(\xi(x, X(t))\big)\Big)\Big]\mathrm{d}t \\ +\, D\nabla^2 \rho_i(x, t)\,\mathrm{d}t - \nabla \cdot \rho_i(x, t)\,\mathrm{d}W_i(t)\Big\} f(x)\,\mathrm{d}x. \end{aligned}$$

On the other hand, identity (25) also implies that

$$\mathrm{d} f\big(X_i(t)\big) = \int_{x\in\mathcal{X}} \mathrm{d}\rho_i(x, t)\ f(x)\,\mathrm{d}x.$$

Comparing both equations, we conclude that

$$\begin{aligned} \mathrm{d}\rho_i(x, t) = -\nabla \cdot \Big[\rho_i(x, t) h\Big(x, \pi_\theta\big(\xi(x, X(t))\big)\Big)\Big]\mathrm{d}t \\ +\, D\nabla^2 \rho_i(x, t)\,\mathrm{d}t - \nabla \cdot \rho_i(x, t)\,\mathrm{d}W_i(t). \end{aligned}$$

In order to obtain an expression for the global density, we sum up all agent-based increments, which gives

$$\begin{aligned}\mathrm{d}\rho^{(N)}(x,t) &= \frac{1}{N}\sum_{i=1}^{N}\mathrm{d}\rho_i(x,t)\\ &= -\nabla\cdot\Big[\rho^{(N)}(x,t)h\big(x,\bar{u}^{(N)}(x,t)\big)\Big]\mathrm{d}t\\ &\quad + D\nabla^2\rho^{(N)}(x,t)\,\mathrm{d}t - \nabla\cdot\frac{1}{N}\sum_{i=1}^{N}\rho_i(x,t)\,\mathrm{d}W_i(t),\end{aligned} \tag{26}$$

where we introduced the finite-size control field $\bar{u}^{(N)}(x,t)$,

$$\bar{u}^{(N)}(x,t) := \pi_\theta\big(\overline{y}^{(N)}(x,t)\big), \tag{27}$$

and the underlying observation field $\overline{y}^{(N)}(x,t)$,

$$\overline{y}^{(N)}(x,t) := \xi\big(x,X(t)\big) = \frac{\int_{\mathcal{X}}\rho^{(N)}(y,t)g(x,y)k(x,y)\,\mathrm{d}y}{\int_{\mathcal{X}}\rho^{(N)}(y',t)k(x,y')\,\mathrm{d}y'}, \tag{28}$$

as replacements for the agent-based control and observation signals, $\{u_i(t)\}$ and $\{Y_i(t)\}$, respectively. Note that the latter equation follows directly from Eq. (3) using the definition of the N-agent density in Eq. (13).

As shown by Dean (1996), the cumulative influence of the agent-dependent noise terms in Eq. (26) can be described by a statistically equivalent, agent-independent field of noise processes $\overline{W}(x,t)$ with correlation function

$$\mathbb{E}\left[\overline{W}_m(x,t)\overline{W}_n\left(y,t'\right)\right] = 2D\delta_{m,n}\delta_{x,y}\min\left(t,t'\right),$$

where $\overline{W}_m(x,t)$ denotes the mth component of the field at position x and time t. Equation (26) then simplifies to

$$\begin{aligned}\mathrm{d}\rho^{(N)}(x,t) &= -\nabla\cdot\Big[\rho^{(N)}(x,t)h\big(x,\bar{u}^{(N)}(x,t)\big)\Big]\mathrm{d}t\\ &\quad + D\nabla^2\rho^{(N)}(x,t)\,\mathrm{d}t + \nabla\cdot\left[\frac{1}{N}\sqrt{\rho^{(N)}(x,t)}\,\mathrm{d}\overline{W}(x,t)\right].\end{aligned}$$

In the limit $N\to\infty$, the stochastic component of this differential equation vanishes and we obtain our final convection–diffusion dynamics for the continuum density $\rho(x,t)$,

$$\frac{\partial\rho(x,t)}{\partial t} = -\nabla\cdot\Big[\rho(x,t)h\big(x,\bar{u}(x,t)\big)\Big] + D\nabla^2\rho(x,t),$$

where the continuum control field $\bar{u}(x,t)$ and the underlying continuum observation field $\overline{y}(x,t)$ are defined as in Eqs. (27) and (28), respectively, but $\rho^{(N)}(x,t)$ is replaced by $\rho(x,t)$.

References

Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, T. F., et al. (2000). Amorphous computing. *Communications of the ACM*, *43*(5), 74–82.

Aumann, R. J. (1964). Markets with a continuum of traders. *Econometrica*, *32*(1), 39–50.

Beal, J. (2005). Programming an amorphous computational medium. In J. P Banâtre, P. Fradet, J. L. Giavitto, & O. Michel (Eds.), *Unconventional programming paradigms* (pp. 121–136). Berlin: Springer.

Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, *27*(4), 819–840.

Billingsley, P. (1999). *Convergence of probability measures*. New York: Wiley.

Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, *7*(1), 1–41.

Correll, N., & Martinoli, A. (2006). System identification of self-organizing robotic swarms. In M. Gini & R. Voyles (Eds.) *Distributed autonomous robotic systems 7* (pp. 31–40). Tokyo: Springer Japan.

Couzin, I. D., Krause, J., James, R., Ruxton, G. D., & Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, *218*(1), 1–11.

Crutchfield, J. P., & Mitchell, M. (1995). The evolution of emergent computation. *Proceedings of the National Academy of Sciences*, *92*(23), 10742–10746.

Dean, D. S. (1996). Langevin equation for the density of a system of interacting Langevin processes. *Journal of Physics A: Mathematical and General*, *29*(24), L613.

Deisenroth, M. P., Neumann, G., & Peters, J. (2013). A survey on policy search for robotics. *Foundations and Trends in Robotics*, *2*(1–2), 1–142.

Doucet, A., Godsill, S., & Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, *10*(3), 197–208.

Dubkov, A., & Spagnolo, B. (2005). Generalized Wiener process and Kolmogorov's equation for diffusion induced by non-Gaussian noise source. *Fluctuation and Noise Letters*, *5*(02), L267–L274.

Ermentrout, G. B., & Edelstein-Keshet, L. (1993). Cellular automata approaches to biological modeling. *Journal of Theoretical Biology*, *160*(1), 97–133.

Fornberg, B., & Flyer, N. (2015). Solving PDEs with radial basis functions. *Acta Numerica*, *24*, 215–258.

Freitas, R. A. (2005). Current status of nanomedicine and medical nanorobotics. *Journal of Computational and Theoretical Nanoscience*, *2*(1), 1–25.

Grondman, I., Busoniu, L., Lopes, G. A. D., & Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(6), 1291–1307.

Hamann, H. (2014). Evolution of collective behaviors by minimizing surprise. In *Proceedings of the 14th international conference on the synthesis and simulation of living systems* (pp. 344–351). MIT Press.

Hamann, H., & Wörn, H. (2008). A framework of space–time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, *2*(2), 209–239.

Hayes, A. T. (2002). How many robots? Group size and efficiency in collective search tasks. In H. Asama, T. Arai, T. Fukuda, & T. Hasegawa (Eds.), *Distributed autonomous robotic systems 5* (pp. 289–298). Tokyo: Springer Japan.

Houchmandzadeh, B., & Vallade, M. (2015). Exact results for a noise-induced bistable system. *Physical Review E*, *91*(2), 022115.

Hüttenrauch, M., Šošić, A., & Neumann, G. (2017). Guided deep reinforcement learning for swarm systems. In *AAMAS workshop on autonomous robots and multirobot systems*. arXiv:1709.06011.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1), 99–134.

Karatzas, I., & Shreve, S. (1998). *Brownian motion and stochastic calculus*. Berlin: Springer Science & Business Media.

Krylov, N. V. (2008). *Controlled diffusion processes*. Berlin: Springer Science & Business Media.

Kuramoto, Y. (1975). Self-entrainment of a population of coupled non-linear oscillators. In *International symposium on mathematical problems in theoretical physics* (pp. 420–422). Springer.

Land, M., & Belew, R. K. (1995). No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, *74*(25), 5148.

Lasry, J.-M., & Lions, P.-L. (2007). Mean field games. *Japanese Journal of Mathematics*, *2*(1), 229–260.

Lerman, K., Martinoli, A., & Galstyan, A. (2005). A review of probabilistic macroscopic models for swarm robotic systems. In *Swarm robotics: SAB 2004 international workshop* (pp. 143–152). Berlin: Springer.

Lesser, V., Ortiz, C. L., & Tambe, M. (2003). *Distributed sensor networks: A multiagent perspective*. Berlin: Springer Science & Business Media.

MacLennan, B. J. (1990). *Continuous spatial automata*. Technical report, University of Tennessee, Computer Science Department.

Macua, S. V., Chen, J., Zazo, S., & Sayed, A. H. (2015). Distributed policy evaluation under multiple behavior strategies. *IEEE Transactions on Automatic Control*, *60*(5), 1260–1274.

Martinoli, A., Ijspeert, A. J., & Mondada, F. (1999). Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, *29*(1), 51–63.

Michini, B., & How, J. P. (2012). Bayesian nonparametric inverse reinforcement learning. In P. A. Flach, T. De Bie, & N. Cristianini (Eds.), *Machine learning and knowledge discovery in databases* (pp. 148–163). Berlin: Springer.
Munos, R. (2006). Policy gradient in continuous time. *Journal of Machine Learning Research*, *7*, 771–791.
Ohkubo, J., Shnerb, N., & Kessler, D. A. (2008). Transition phenomena induced by internal noise and quasi-absorbing state. *Journal of the Physical Society of Japan*, *77*(4), 044002.
Ramaswamy, S. (2010). The mechanics and statistics of active matter. *Annual Review of Condensed Matter Physics*, *1*(1), 323–345.
Risken, H. (1996). Fokker–Planck equation. In H. Haken (Ed.) *The Fokker–Planck equation* (pp. 63–95). Berlin, Heidelberg: Springer.
Schweitzer, F. (2003). *Brownian agents and active particles: Collective dynamics in the natural and social sciences*. Berlin, Heidelberg: Springer.
Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., & Schmidhuber, J. (2010). Parameter-exploring policy gradients. *Neural Networks*, *23*(4), 551–559.
Sipper, M. (1999). The emergence of cellular computing. *Computer*, *32*(7), 18–26.
Šošić, A., KhudaBukhsh, W. R., Zoubir, A. M., Koeppl, H. (2017). Inverse reinforcement learning in swarm systems. In *Proceedings of the 16th international conference on autonomous agents and multiagent systems* (pp. 1413–1421). International Foundation for Autonomous Agents and Multiagent Systems.
Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge: MIT Press.
Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., & Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, *75*(6), 1226–1229.
Whitesides, G. M., & Grzybowski, B. (2002). Self-assembly at all scales. *Science*, *295*(5564), 2418–2421.