Mathematical Models and Methods in Applied Sciences

Model predictive control with random batch methods for a guiding problem

Dongnam Ko

Department of Mathematics, The Catholic University of Korea
Jibongro 43, Bucheon, Gyeonggido 14662, Republic of Korea
dongnamko@catholic.ac.kr

Enrique Zuazua

Chair in Applied Analysis, Alexander von Humboldt-Professorship, Department of Mathematics,
Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany, and
Chair of Computational Mathematics, Fundación Deusto, University of Deusto
48007 Bilbao, Basque Country, Spain, and
Departamento de Matemáticas, Universidad Autónoma de Madrid
28049 Madrid, Spain
enrique.zuazua@fau.de

Received (Day Month Year)
Revised (Day Month Year)
Communicated by (xxxxxxxxxx)

We model, simulate and control the guiding problem for a herd of evaders under the action of repulsive drivers. The problem is formulated in an optimal control framework, where the drivers (controls) aim to guide the evaders (states) to a desired region of the Euclidean space. The numerical simulation of such models quickly becomes unfeasible for a large number of interacting agents, as the number of interactions grows $O(N^2)$ for N agents. For reducing the computational cost to $O(N)$, we use the Random Batch Method (RBM), which provides a computationally feasible approximation of the dynamics. First, the considered time interval is divided into a number of subintervals. In each subinterval, the RBM randomly divides the set of particles into small subsets (batches), considering only the interactions inside each batch. Due to the averaging effect, the RBM approximation converges to the exact dynamics in the L^2-expectation norm as the length of subintervals goes to zero. For this approximated dynamics, the corresponding optimal control can be computed efficiently using a classical gradient descent. The resulting control is not optimal for the original system, but for a reduced RBM model. We therefore adopt a Model Predictive Control (MPC) strategy to handle the error in the dynamics. This leads to a semi-feedback control strategy, where the control is applied only for a short time interval to the original system, and then compute the optimal control for the next time interval with the state of the (controlled) original dynamics. Through numerical experiments we show that the combination of RBM and MPC leads to a significant reduction of the computational cost, preserving the capacity of controlling the overall dynamics.

Keywords: Agent-based models; Guiding problem; large scale complex systems; Random

Batch Method; Model Predictive Control.

AMS Subject Classification: 49M29, 90C59, 93B51

1. Introduction

Control problems for collective behavior systems have received huge attention recently,[2,4,15,25] due to the growing needs in applications. In the context of interacting particle systems, the emergence of collective behavior can be viewed as a decentralized control problem,[9,13,27] where each individual reacts to the other agents according to its own decision, but a certain desired collective motion (such as synchronization) may arise. As indicated in literature,[5] a centralized control can play as an external interference boosting the collective behavior to evoke the desired dynamics. From a practical viewpoint, the control of a complex system is commonly designed to manipulate a small portion of the particles,[7,30] or preselected agents, playing the role of leaders or informed agents.[15,33]

For example, the positions of a group of vehicles can be effectively operated by local leaders,[15] and a few drones can keep a herd of birds away at an airport.[16] One of the most relevant examples is the shepherding problem, where a few shepherd dogs are required to handle a herd of sheep. Many attempts have been made in various contexts to understand how the dogs can affect a group of sheep,[36] and steer them to the desired region.[24]

Our interest in this paper consists in modeling, deriving and simulating optimal control strategies for guiding problems, where a small number of repelling agents (drivers), which play the role of controls, have to guide a herd of flocking agents (evaders), that we interpret as states, toward a given desired area. As a basis, we follow the formulation based on the *guidance-by-repulsion*[14,23] paradigm.

The problem with one driver and one evader has been addressed in Ref. 14. Its long-time behavior and controllability properties were analyzed in Ref. 23. In that simplified setting numerical simulations show that the guiding problem is very sensitive to the motion of drivers, the optimization process being highly non-convex.

The first difficulty encountered when addressing many evaders is the computational complexity of the forward dynamics, which dangerously increases since the number of interactions between N evaders grows as $O(N^2)$. With the standard Pontryagin approach for the optimal control as in Ref. 23, the optimization algorithm needs to compute the forward dynamics iteratively, for example, computing the optimal control with 4 drivers and $N = 16$ evaders in a time-horizon $T = 40$ with time step $\Delta t = 0.01$ requires nearly two hours in a typical laptop computer with CPU i5-4258U 2.4GHz and RAM DDR3L 8GB 1600MHz, operated in Matlab.

To overcome this, there are two kinds of methods that commonly used, one is for approximating dynamics and the other is to get a control without simulating dynamics numerically. A classical treatment to approximate interacting particles is the fast multi-pole method[35] with $O(N)$ computational complexity, but these methods require the decay of the interaction with respect to the distance. Hence,

they are not efficient if the interactions are similar, for example, when the sheep are gathered in a small area. On the other hand, a direct control method such as the proportional-integral-derivative (PID) controller may not be appropriate to our nonlinear guiding problem; as described in Ref. 24, 36, there are plenty of possible control functions for similar control objectives that depend on the situation.

The main novelty of this article is a new computational technique for an approximately optimal control strategy allowing to handle guiding problems with a large number of drivers and evaders. In this paper, we suggest an algorithm (see Algorithm 1 in Section 2), based on a classical optimal control formulation, but combining critically two main ingredients: First, the Random Batch Method (RBM),[22] which provides an approximation of the dynamics of the interacting particle system at a small computational cost $O(N)$, and second, Model Predictive Control (MPC),[17] a control design methodology to stabilize the approximation error for a long-time horizon.

The RBM is an approximation method particularly suitable in the context of collective dynamics when the individuals are not distinguishable. Instead of computing the whole interactions, for a given P with $1 < P \ll N$, the RBM approximates dynamics out of $O(NP)$ interactions. More precisely, for a small duration of time, we split the set of particles into random small subsets (batches) which contain, at most, P particles. Then, one only considers the interactions within each batch, ignoring the interactions between batches. In the next time interval, to average the random effect in time, we again choose batches independently.

Therefore, from the all-to-all interacting particle system, the RBM produces a deterministic networked model which periodically switches the network structure. Thanks to the random choices, the reduced RBM model approximates the original time evolution properly based on the Law of Large Numbers. As analyzed in,[22] the squared expectation distance between original and approximated trajectories follows $O(C(T)\Delta t/P)$, where $C(T)$ is a constant growing exponentially with the final time T. The approximation error for the guiding problem is numerically simulated in Section 3.1, which draw the 95% confidence intervals from 200 independent simulations with $T = 10$ and $\Delta t = 0.01$. It shows that just one realization of the computation on the approximated trajectories could catch the density profile of the evaders quite precisely though the RBM relies on the randomness.

In this paper, we solve an optimal control problem on the simplified RBM model using standard gradient descent methods. But, of course, this does not lead to an accurate control of the original dynamics and the performance of the control depends on the approximation error which accumulates in time. To handle the error for a long-time horizon, we adopt the viewpoint of Model Predictive Control (MPC).[17,19,28] The basic idea of MPC is to observe the original (controlled) system periodically to refresh the approximate dynamics and computed controls with new data while the original system evolves in time.

The MPC works as follows. We basically solve the optimal control problem on the RBM model with an artificial short time horizon. The control that results from

this computation is then applied to the original system. This leads to a final state that differs from the one corresponding to the RBM model. This difference is due to the approximation error. Of course in practical applications unexpected noise effects can increase that gap. At the final time, we again solve the optimal control problem for the next time interval with the state from the original system. In this iterative way, one can force the control to adapt to the true dynamics of the system. A more detailed description of the RBM and MPC is presented in Section 2.

As shown in this paper, the adequate combination of the RBM and MPC may lead to a significant reduction of the computational cost preserving the efficiency of the control strategy to steer the original dynamics. In Section 3, we checked that the overall computational cost mainly follows the number of considered interactions. The RBM reduces this from $O(N(N+M))$ into $O(N(P+M))$ for N evaders and M drivers (The batch size P is commonly chosen to be 2). For instance, in one of the examples we describe in Section 3, our method shows an 84% reduction of the computational time on the problem of 36 evaders and 2 drivers with $T = 4$ and $\Delta t = 0.01$, while the running cost (performance of control) only differs by about 0.3%, compared to the standard optimal control problem.

The rest of this paper is organized as follows. In Section 2.1, we formulate the guiding problem as an optimal control one. From Section 2.2 to Section 2.4, we present the preliminaries on the RBM and MPC. Then, the detailed procedure of how to combine MPC-RBM to build our algorithm is described in Section 2.5. The simulations are presented in Section 3 to test computational costs and approximation errors. Finally, in Section 4, we discuss our results and present some final remarks and open problems arising in this field.

2. The MPC-RBM algorithm

In this section, we first present the guiding optimal control problem. Then, the combined MPC-RBM algorithm is described.

2.1. *Optimal control formulation on the guiding problem*

Let $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \mathbb{R}^{Nd}$ and $\mathbf{v} = \dot{\mathbf{x}} \in \mathbb{R}^{Nd}$ be the positions and velocities of N Newtonian particles moving in a d-dimensional space, representing the evaders. The control is indirectly introduced through M particles represented by $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_M) \in \mathbb{R}^{Md}$, called the drivers. We assume that the evaders interact with other evaders and the drivers as in the following collective behavior system:

for $t \geq 0$,

$$\begin{cases} \dot{\mathbf{x}}_i = \mathbf{v}_i, \quad i = 1, \dots, N, \\ \dot{\mathbf{v}}_i = \dfrac{1}{N-1} \displaystyle\sum_{k=1, k\neq i}^{N} a(\mathbf{x}_k - \mathbf{x}_i)(\mathbf{v}_k - \mathbf{v}_i) \\ \qquad + \dfrac{1}{N-1} \displaystyle\sum_{k=1, k\neq i}^{N} g(\mathbf{x}_k - \mathbf{x}_i)(\mathbf{x}_k - \mathbf{x}_i) \\ \qquad - \dfrac{1}{M} \displaystyle\sum_{j=1}^{M} f(\mathbf{y}_j - \mathbf{x}_i)(\mathbf{y}_j - \mathbf{x}_i), \ i = 1, \dots, N, \\ \dot{\mathbf{y}}_j = \mathbf{u}_j(t), \quad j = 1, \dots, M \\ \mathbf{x}_i(0) = \mathbf{x}_i^0, \quad \mathbf{v}_i(0) = \mathbf{v}_i^0, \quad \mathbf{y}_j(0) = \mathbf{y}_j^0. \end{cases} \tag{2.1}$$

We assume that the controls $\mathbf{u}(t) = (\mathbf{u}_1(t), \dots, \mathbf{u}_M(t)) \in \mathbb{R}^{Md}$ completely determine the dynamics of the drivers, as a means to indirectly influence the dynamics of the evaders.

The nonlinearities $a(\cdot)$, $f(\cdot)$ and $g(\cdot)$ entering in the dynamics are assumed to be smooth and positive, except for $g(\cdot)$ that may have negative values as in the context of intermolecular forces, to avoid collisions between evaders.[8,11] As a concrete example and for the purpose of developing the numerical experiments, we define $a(\cdot)$, $f(\cdot)$ and $g(\cdot)$ as follows:

$$\begin{aligned} &a(\mathbf{x}) := 1, \quad f(\mathbf{x}) := 4\exp(-8|\mathbf{x}|^2) \quad \text{and} \\ &g(\mathbf{x}) := \begin{cases} 2\left(1 - \dfrac{1}{3\sqrt{N}|\mathbf{x}|^2}\right) & \text{if} \quad \mathbf{x} \neq 0, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{2.2}$$

Note that (2.2) contains an unbounded function $g(\cdot)$, but local existence and uniqueness is guaranteed for initial data satisfying $\mathbf{x}_i^0 \neq \mathbf{x}_j^0$ for any $i \neq j$. The dynamical properties of the system (2.1) are discussed in Ref. 16 with a similar formulation of interactions. The global existence of solutions for (2.2) follows similar arguments to.[11]

The qualitative properties of the nonlinearities in (2.2) are chosen to reflect the following main features of the "drivers-evaders" interactions (see also Ref. 16, 23, 31, 36 for specific guiding problems):

- The evaders are influenced by a repulsive force $f(\cdot)$ from each driver, its strength being decreasing as the distance increases.
- Each evader has positional interactions with other evaders given by the function $g(\cdot)$, which reflects their aim to remain close together. To prevent the collisions between evaders, we also set $g(\cdot)$ to have a strong negative value when a pair of evaders is too close. Overall, all the evaders have the tendency to gather in a disk with a diameter ~ 0.5.

- The evaders also interact each other, through the term $a(\cdot)$, to align the velocities to a common value. This plays the role of friction to the mean velocity of evaders, which reduces the oscillatory behavior arising from $g(\cdot)$.

Compared to herding[24,36] and flocking problems,[12,21] the above system (2.1) contains both the velocity alignment $a(\cdot)$ and the positional potential interactions $g(\cdot)$. The combination of positional and velocities' interactions is suggested in,[16,29,37] which is used to model the dynamics of birds.

Our control objective is to guide the evaders to a desired region by the locomotion induced by the drivers in a given time horizon. In order to formulate this problem in the context of optimal control, we define the cost function as follows:

$$J(\mathbf{u}) := \int_0^T \left[\frac{\alpha_1}{N} \sum_{k=1}^{N} |\mathbf{x}_k - \mathbf{x}_f|^2 + \frac{\alpha_2}{M} \sum_{j=1}^{M} |\mathbf{u}_j|^2 + \frac{\alpha_3}{M} \sum_{j=1}^{M} |\mathbf{y}_j - \mathbf{x}_f|^2 \right] dt. \tag{2.3}$$

This functional takes account, in particular, of the running cost of the distances to the target point $\mathbf{x}_f \in \mathbb{R}^d$. The positive constants $\alpha_1, \alpha_2, \alpha_3$ allow to regulate the weight of each of the terms entering in the cost. In practice, α_1 is taken to be large compared to α_2 and α_3. Involving the running cost of the control by means of α_2 is rather standard, to avoid unfeasibly large controls. On the other hand, the term involving α_3 prevents the drivers not to go very far from the evaders. The target position $\mathbf{x}_f$ is chosen as a reference point since we expect that, tracking the evaders, the drivers will also spend most of the time near $\mathbf{x}_f$.

We expect that, minimizing this functional, the optimal control of the system will guide the evaders toward the region near $\mathbf{x}_f$ and capture them for a long time. Of course, the efficiency on doing so will depend on the length of the time-horizon and the value of the parameters α_j, $j = 1, 2, 3$.

The particular case of one driver and one evader was analyzed in Ref. 14, 23. In that case, it was proved that there exist a final time T and a control function $\mathbf{u}_1 \in L^\infty((0,T), \mathbb{R}^d)$ satisfying $\mathbf{x}_1(T) = \mathbf{x}_f$. But, of course, achieving such controllability results is much harder for multiple drivers and evaders. This is a very interesting open analytical problem, out of the scope of the present article.

To analyze a large number of particles (evaders), one of the well-known approximation methods is the mean-field limit.[18,21] It considers the distribution of particles instead of the whole trajectories, in the form of kinetic (or transport) equations. Hence, the evaders need to be indistinguishable; the interactions in (2.1) only depend on the relative positions and velocities, not the index i or k. In addition, if the interactions are bounded and smooth, then the mean-field limit strategy can be applied,[5,6,31] This mean-field approach cannot be applied when the interactions are not identical. In that case, one may rather employ a graph limit model as in Ref. 3.

In this paper, we study the particle description of the guiding problem due to its simplicity and generality. Our method, combining MPC and RBM, can be easily generalized to the mean-field control problems and other formulations since the

corresponding dynamics can be approximated by the particle model through the characteristic equations or the space discretization.

2.2. *The RBM approximation for the forward dynamics*

We use the RBM in Ref. 22 to reduce the computational cost to simulate the time evolution of the interacting particle system (2.1).

We proceed as follows.

- The control time interval $[0, T]$ is fixed. We fix a short duration of time $\Delta t > 0$ and the discrete times $t_n := n\Delta t$.
- Then, for each interval $[t_n, t_{n+1}]$, we independently choose random batches: a partition of the index set $\{1, \dots, N\}$ as $\mathcal{C}_n^1, \mathcal{C}_n^2, \cdots, \mathcal{C}_n^{N/P}$, where each batch $\mathcal{C}_n^m$ has P particles. If P is not a divisor of N, then we have $\lfloor N/P \rfloor$ batches of size P and one additional batch containing the remainders. For simplicity, we assume that P divides N.
- In this way, each evader belongs to one and only one batch in each subinterval $[t_n, t_{n+1}]$. We build a reduced dynamics which consists in considering only the interactions inside each batch. Then, the number of interactions between the evaders decreases to $O(NP)$ from the all-to-all number of interactions $O(N^2)$.

The resulting dynamics is of switching nature. In each time sub-interval, the RBM rearranges the interaction network randomly. For each index i identifying an evader, we denote $\mathcal{C}_n^{p(i)}$ as the batch that, in the time interval $[t_n, t_{n+1}]$, contains this evader. Then, the approximated dynamics on the ith evader ($\mathbf{x}_i^R$ and $\mathbf{v}_i^R$, $i = 1, \dots, N$) can be formulated as follows: for $t \in [t_n, t_{n+1}]$,

$$\begin{cases} \dot{\mathbf{x}}_i^R = \mathbf{v}_i^R, \quad i = 1, \dots, N, \\ \dot{\mathbf{v}}_i^R = \dfrac{1}{P-1} \displaystyle\sum_{k \in \mathcal{C}_n^{p(i)} \setminus \{i\}} a(\mathbf{x}_k^R - \mathbf{x}_i^R)(\mathbf{v}_k^R - \mathbf{v}_i^R) \\ \qquad + \dfrac{1}{P-1} \displaystyle\sum_{k \in \mathcal{C}_n^{p(i)} \setminus \{i\}} g(\mathbf{x}_k^R - \mathbf{x}_i^R)(\mathbf{x}_k^R - \mathbf{x}_i^R) \\ \qquad - \dfrac{1}{M} \displaystyle\sum_{j=1}^{M} f(\mathbf{y}_j - \mathbf{x}_i^R)(\mathbf{y}_j - \mathbf{x}_i^R), \qquad i = 1, \dots, N, \\ \dot{\mathbf{y}}_j = \mathbf{u}_j(t), \qquad j = 1, \dots, M, \\ \mathbf{x}_i^R(0) = \mathbf{x}_i^0, \quad \mathbf{v}_i^R(0) = \mathbf{v}_i^0, \quad \mathbf{y}_j(0) = \mathbf{y}_j^0, \ i = 1, \dots, N, \ j = 1, \dots, M. \end{cases} \tag{2.4}$$

In (2.4), the sum of interactions is averaged by $1/(P-1)$ (instead of $1/(N-1)$) since there are $P-1$ interacting evaders in each batch $\mathcal{C}_n^{p(i)}$. From the viewpoint of statistics, this is a kind of sample mean of the interactions from one batch. The total average (population mean) of the interactions is used in the original system (2.1).

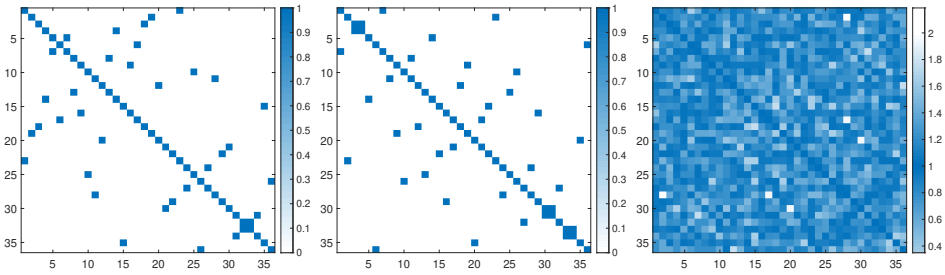

Fig. 1: Graphical representation of the adjacency matrix $A(t) = A_{ij}(t)$ for 36 evaders with $P = 2$ in the time intervals $[0, \Delta t]$ (left) and $[\Delta t, 2\Delta t]$ (middle). The dotted points illustrate the interactions within the different batches, where the diagonal line is colored for reference. The right figure represents the averaged matrix over 400 random batches. The reduced model in each time subinterval has a sparse network, while the original system consists of all-to-all interactions.

From the initial data $\mathbf{x}_i^0$ and $\mathbf{v}_i^0$, the RBM computes (2.4) with the batches $\mathcal{C}_0^1, \mathcal{C}_0^2, \dots, \mathcal{C}_0^{N/P}$ over $t \in [0, t_1] = [t_0, t_1]$. Then, the final data $\mathbf{x}_i^R(t_1)$ and $\mathbf{v}_i^R(t_1)$ are taken as the initial data for the next time interval $[t_1, t_2]$ on which the batches $\mathcal{C}_1^1, \mathcal{C}_1^2, \dots, \mathcal{C}_1^{N/P}$ are used. Note that the system (2.4) has a switching nature since the batches are defined differently on each interval $[t_n, t_{n+1}]$. Following the iterative calculations on n, we can get the approximated time evolution in $[0, T]$.

The reduced system has a network structure which switches the connectivity at each time t_n. In each time subinterval, $[t_n, t_{n+1}]$ the adjacency matrix is as follows,

$$
\begin{aligned}
A_{ij}^n &:= \begin{cases} 1 & \text{if } \ i \neq j \text{ and in the same set among } \mathcal{C}_n^1, \dots, \mathcal{C}_n^{N/P}, \\ 0 & \text{otherwise,} \end{cases} \\
A_{ij}(t) &:= A_{ij}^n \quad \text{for } \ t \in [t_n, t_{n+1}),
\end{aligned}
$$

which is a piecewise constant matrix. With this notation, the system (2.4) can be understood as a deterministic switching network system for $t \in [0, T]$,

$$
\begin{cases}
\dot{\mathbf{x}}_i^R = \mathbf{v}_i^R, \qquad i = 1, \dots, N, \\
\dot{\mathbf{v}}_i^R = \dfrac{1}{P-1} \displaystyle\sum_{k=1}^{N} A_{ki}(t) a(\mathbf{x}_k^R - \mathbf{x}_i^R)(\mathbf{v}_k^R - \mathbf{v}_i^R) \\
\qquad\quad + \dfrac{1}{P-1} \displaystyle\sum_{k=1}^{N} A_{ki}(t) g(\mathbf{x}_k^R - \mathbf{x}_i^R)(\mathbf{x}_k^R - \mathbf{x}_i^R) \\
\qquad\quad - \dfrac{1}{M} \displaystyle\sum_{j=1}^{M} f(\mathbf{y}_j - \mathbf{x}_i^R)(\mathbf{y}_j - \mathbf{x}_i^R), \qquad i = 1, \dots, N, \\
\dot{\mathbf{y}}_j = \mathbf{u}_j(t), \qquad j = 1, \dots, M, \\
\mathbf{x}_i^R(0) = \mathbf{x}_i^0, \ \mathbf{v}_i^R(0) = \mathbf{v}_i^0, \ \mathbf{y}_j(0) = \mathbf{y}_j^0, \quad i = 1, \dots, N, \ j = 1, \dots, M.
\end{cases}
\tag{2.5}
$$

This system is well-posed[22] so that the unique solution exists as a Lipschitz function. The velocity $\mathbf{v}^R$ is not C^1 in general due to the discontinuities of $\dot{\mathbf{v}}^R$ at each t_n.

The reduced system (2.5), including both the interactions among evaders and with drivers, involves a total number of $O(N(P+M))$ interactions while, in the original dynamics (2.1), the total number of interactions is $O(N(N+M))$ per time step.

Fig. 1 shows the adjacency matrix in (2.5) among 36 evaders, out of a choice of random batches of size $P = 2$ (left and middle). The network structure changes at each time step, while the number of interactions is fixed to $N(P-1)$. When operating the numerical simulation with $T = 4$ and $\Delta t = 0.01$, the connectivity along time $[0, 4]$ is averaged 400 times (right). As Δt decreases to zero, it converges to the matrix of ones except for the diagonal elements, which is the adjacency matrix of the original system (2.1).

Indeed, in Ref. 22, the error analysis of the RBM is developed for a collective behavior model in L^2-expectation sense. In detail, the expected squared distance follows,

$$\sup_{0\le t\le T} \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}|\mathbf{x}_i^R(t)-\mathbf{x}_i(t)|^2\right] \lesssim \exp(CT)\Delta t\left(\frac{1}{P-1}-\frac{1}{N-1}\right), \tag{2.6}$$

where C is a constant related to the supremum of interactions which depends on the functions a, g, f and initial data. The exponentially growing term $\exp(CT)$ can be eliminated when the system has suitable contraction properties,[22] which does not hold for our approximated system (2.4). The uniform-in-time error analysis is still open for general collective behavior systems though in the numerical simulations[10,20] the RBM shows a successful guess on the long-time behavior.

Note that our goal is to find an approximated control, similar to the optimal control for the original system (2.1) with the cost function J in (2.3). Our strategy is based on computing the optimal control for the approximated system (2.4) with the corresponding cost function J^R:

$$J^R(\mathbf{u}) := \int_0^T \left[\frac{\alpha_1}{N}\sum_{k=1}^{N}|\mathbf{x}_k^R-\mathbf{x}_f|^2+\frac{\alpha_2}{M}\sum_{j=1}^{M}|\mathbf{u}_j|^2+\frac{\alpha_3}{M}\sum_{j=1}^{M}|\mathbf{y}_j-\mathbf{x}_f|^2\right]dt. \tag{2.7}$$

2.3. *The RBM for the optimal control problem*

In order to find the optimal control of the reduced problem, we need to compute the gradient of the cost function J^R in (2.7) on the reduced RBM model (2.4). The gradient can be derived, following the Pontryagin maximum principle,[32,38] from the time evolution in the adjoint system of (2.4).

First, we consider the adjoint of the original system (2.1). From the Pontryagin

maximum principle, if we denote the controlled system (2.1) and the cost (2.3) as

$$\begin{cases} \dot{\mathbf{x}} = F^{\mathbf{x}}(\mathbf{x},\mathbf{v},\mathbf{y}), \\ \dot{\mathbf{v}} = F^{\mathbf{v}}(\mathbf{x},\mathbf{v},\mathbf{y}), \\ \dot{\mathbf{y}} = F^{\mathbf{y}}(\mathbf{x},\mathbf{v},\mathbf{y}), \end{cases} \quad \text{and} \quad J = \int_0^T L(\mathbf{x},\mathbf{v},\mathbf{y})dt, \tag{2.8}$$

then its formal adjoint system can be described by the backward equations for $t \in [0,T]$,

$$\begin{cases} -\dot{\mathbf{p}}^{\mathsf{T}} = \mathbf{p}^{\mathsf{T}}\nabla_{\mathbf{x}}F^{\mathbf{x}} + \mathbf{q}^{\mathsf{T}}\nabla_{\mathbf{x}}F^{\mathbf{v}} + \mathbf{r}^{\mathsf{T}}\nabla_{\mathbf{x}}F^{\mathbf{y}} + \nabla_{\mathbf{x}}L, \\ -\dot{\mathbf{q}}^{\mathsf{T}} = \mathbf{p}^{\mathsf{T}}\nabla_{\mathbf{v}}F^{\mathbf{x}} + \mathbf{q}^{\mathsf{T}}\nabla_{\mathbf{v}}F^{\mathbf{v}} + \mathbf{r}^{\mathsf{T}}\nabla_{\mathbf{v}}F^{\mathbf{y}} + \nabla_{\mathbf{v}}L, \\ -\dot{\mathbf{r}}^{\mathsf{T}} = \mathbf{p}^{\mathsf{T}}\nabla_{\mathbf{y}}F^{\mathbf{x}} + \mathbf{q}^{\mathsf{T}}\nabla_{\mathbf{y}}F^{\mathbf{v}} + \mathbf{r}^{\mathsf{T}}\nabla_{\mathbf{y}}F^{\mathbf{y}} + \nabla_{\mathbf{y}}L, \\ \mathbf{p}^{\mathsf{T}}(T) = 0,\ \mathbf{q}^{\mathsf{T}}(T) = 0,\ \mathbf{r}^{\mathsf{T}}(T) = 0, \end{cases} \tag{2.9}$$

where the variables $\mathbf{p}(t) \in \mathbb{R}^{Nd}$, $\mathbf{q}(t) \in \mathbb{R}^{Nd}$ and $\mathbf{r}(t) \in \mathbb{R}^{Md}$ are the adjoint states of $\mathbf{x}$, $\mathbf{v}$ and $\mathbf{y}$, respectively, and $\mathbf{p}^{\mathsf{T}}$ denotes the transpose of $\mathbf{p}$.

In our case, the adjoint system (2.9) of the original dynamics (2.1) would take the following form in the dual variables $(\mathbf{p},\mathbf{q},\mathbf{r})$ corresponding to $(\mathbf{x},\mathbf{v},\mathbf{y})$:

$$\begin{cases} -\dot{\mathbf{p}}_i^{\mathsf{T}} = \dfrac{1}{N-1}\displaystyle\sum_{k=1,k\neq i}^{N} \mathbf{q}_k^{\mathsf{T}}\Big[\nabla_{\mathbf{x}_i} a(\mathbf{x}_k-\mathbf{x}_i)(\mathbf{v}_k-\mathbf{v}_i)) + \nabla_{\mathbf{x}_i}(g(\mathbf{x}_k-\mathbf{x}_i)(\mathbf{x}_k-\mathbf{x}_i))\Big] \\ \qquad\qquad -\dfrac{1}{M}\displaystyle\sum_{j=1}^{M} \mathbf{q}_i^{\mathsf{T}}\nabla_{\mathbf{x}_i}(f(\mathbf{y}_j-\mathbf{x}_i)(\mathbf{y}_j-\mathbf{x}_i)) + \dfrac{2\alpha_1}{N}(\mathbf{x}_i-\mathbf{x}_f), \qquad i=1,\dots,N, \\ -\dot{\mathbf{q}}_i^{\mathsf{T}} = \mathbf{p}_i^{\mathsf{T}} + \dfrac{1}{N-1}\displaystyle\sum_{k=1,k\neq i}^{N} \mathbf{q}_k^{\mathsf{T}} a(\mathbf{x}_i-\mathbf{x}_k), \qquad i=1,\dots,N, \\ -\dot{\mathbf{r}}_j^{\mathsf{T}} = -\dfrac{1}{M}\displaystyle\sum_{k=1}^{N} \mathbf{q}_k^{\mathsf{T}}\nabla_{\mathbf{y}_j}(f(\mathbf{y}_j-\mathbf{x}_i)(\mathbf{y}_j-\mathbf{x}_i)) + \dfrac{2\alpha_3}{M}(\mathbf{y}_j-\mathbf{x}_f), \qquad j=1,\dots,M, \\ \mathbf{p}_i^{\mathsf{T}}(T) = 0,\ \mathbf{q}_i^{\mathsf{T}}(T) = 0,\ \mathbf{r}_j^{\mathsf{T}}(T) = 0, \qquad i=1,\dots,N, \quad j=1,\dots,M. \end{cases}$$

Finally, the total derivative of the cost function J in (2.8) with respect to the control $\mathbf{u}(t)$ is

$$\nabla_{\mathbf{u}} J = \nabla_{\mathbf{u}}[\mathbf{p}\cdot F^{\mathbf{x}} + \mathbf{q}\cdot F^{\mathbf{v}} + \mathbf{r}\cdot F^{\mathbf{y}} + L] = \mathbf{r} + \frac{\alpha_2}{M}\mathbf{u}. \tag{2.10}$$

Therefore, the implementation of the gradient descent method for the original system (2.1) would lead to an iterative approximation of the optimal control as below

$$\mathbf{u}^{k+1} := \mathbf{u}^k - \alpha\nabla_{\mathbf{u}} J(\mathbf{u}^k), \quad k \geq 0$$

out of an initial guess $\mathbf{u}^0$ (that we could take to be $\mathbf{u}^0 = 0$) and with $\alpha > 0$ small enough.

But, as we have described in the introduction, this algorithm is computationally expensive since, in each step of the gradient descent iteration, it requires to solve

the full state equation and adjoint system taking account of all interactions. We now present the adaptation of the gradient descent to the RBM reduced model.

We start from the RBM approximation (2.5) in Section 2.2. The corresponding adjoint system is computed piecewise in each of the time subintervals $[t_n, t_{n+1}]$, reducing the interaction terms to those corresponding to each batch. The adjoint dynamics of the reduced system for $(\mathbf{p}^R, \mathbf{q}^R, \mathbf{r}^R)$ then reads as follows:

$$\begin{cases} -(\dot{\mathbf{p}}_i^R)^\intercal = \dfrac{1}{P-1}\displaystyle\sum_{k=1}^{N} A_{ki}(t)(\mathbf{q}_k^R)^\intercal \Big[\nabla_{\mathbf{x}_i^R}(a(\mathbf{x}_k^R - \mathbf{x}_i^R))(\mathbf{v}_k^R - \mathbf{v}_i^R)\Big] \\ \qquad\quad + \dfrac{1}{P-1}\displaystyle\sum_{k=1}^{N} A_{ki}(t)(\mathbf{q}_k^R)^\intercal \Big[\nabla_{\mathbf{x}_i^R}(g(\mathbf{x}_k^R - \mathbf{x}_i^R)(\mathbf{x}_k^R - \mathbf{x}_i^R))\Big] \\ \qquad\quad - \dfrac{1}{M}\displaystyle\sum_{j=1}^{M}(\mathbf{q}_i^R)^\intercal \nabla_{\mathbf{x}_i^R}(f(\mathbf{y}_j - \mathbf{x}_i^R)(\mathbf{y}_j - \mathbf{x}_i^R)) + \dfrac{2\alpha_1}{N}(\mathbf{x}_i^R - \mathbf{x}_f),\ i = 1, \dots, N, \\ -(\dot{\mathbf{q}}_i^R)^\intercal = (\mathbf{p}_i^R)^\intercal + \dfrac{1}{P-1}\displaystyle\sum_{k=1}^{N} A_{ki}(t)(\mathbf{q}_k^R)^\intercal a(\mathbf{x}_i^R - \mathbf{x}_k^R),\ i = 1, \dots, N, \\ -(\dot{\mathbf{r}}_j^R)^\intercal = -\dfrac{1}{M}\displaystyle\sum_{k=1}^{N}(\mathbf{q}_k^R)^\intercal \nabla_{\mathbf{y}_j}(f(\mathbf{y}_j - \mathbf{x}_i^R)(\mathbf{y}_j - \mathbf{x}_i^R)) + \dfrac{2\alpha_3}{M}(\mathbf{y}_j - \mathbf{x}_f),\ j = 1, \dots, M, \\ (\mathbf{p}_i^R)^\intercal(T) = 0,\ (\mathbf{q}_i^R)^\intercal(T) = 0,\ (\mathbf{r}_j^R)^\intercal(T) = 0, \quad i = 1, \dots, N, \quad j = 1, \dots, M. \end{cases} \tag{2.11}$$

Remark 2.1. Note that the adjoint system of the reduced RBM model coincides with the reduced RBM model of the complete adjoint system with the same adjacency matrix $A(t)$. Hence, the reduced adjoint system (2.11) can be derived symbolically from (2.5).

As in the forward dynamics, the RBM reduces the computational cost to $O(N(P + M))$ in the computation of the adjoint system. The gradient descent iteration can then be computed similarly to (2.10) in order to find the optimal control of the reduced model (2.4). From the reduced cost function J^R in (2.7), we use the following iterative scheme:

$$\mathbf{u}^{k+1} := \mathbf{u}^k - \alpha \nabla_{\mathbf{u}} J^R(\mathbf{u}^k), \quad k \geq 0. \tag{2.12}$$

As we mentioned in (2.6), the approximation error accumulates in time, in the worst case, exponentially on T following the Gronwall's inequality. Hence, the resulting control may not suffice to guide the original system (2.1) in a long-time horizon. The MPC procedure is now presented to deal with the model approximation error by observing the original controlled system from time to time.

2.4. *The MPC procedure for the approximated model*

MPC is aimed to adapt the control obtained for the reduced dynamics (2.5) to the full system (2.1) in an iterative manner.

First, we numerically compute the approximated control by minimizing J^R in (2.7) over an auxiliary time interval $[0, \widehat{T}] \subset [0, T]$. Moreover, we consider here a shorter time interval $[0, \tau] \subset [0, \widehat{T}]$ and apply the computed control to the original system (2.1) only for $[0, \tau]$. The time parameters τ and $\widehat{T}$ need to be chosen a priori. Then, the final state at time $t = \tau$ is used as the initial data of the reduced dynamics (2.5) for the next interval $[\tau, \tau + \widehat{T}]$. We again apply the control computed on $[\tau, \tau + \widehat{T}]$ only during the time interval $[\tau, 2\tau]$. If the final time $\tau + \widehat{T}$ is larger than the original time horizon T, then we can cut down the time interval to $[\tau, T]$ or may still use $[\tau, \tau + \widehat{T}]$ for convenience. In this manner, the MPC strategy updates the state after each time interval of length τ, and the error from the RBM is not accumulated. This iterative process is carried out until the final time T.

A proper choice of τ may depend on the error from the RBM. If the reduced model gets more accurate, then we may choose a bigger τ so that the computation becomes cheaper. On the other hand, $\widehat{T}$ is rather affected by the nature of the system (2.1). If $\widehat{T}$ is too short, the computed control would significantly differ from the optimal control for the whole time interval $[0, T]$. A conservative choice is $\widehat{T} = T$, however, in many cases smaller $\widehat{T}$ is enough since τ needs to be much shorter. By setting $\widehat{T}$ properly, MPC can also handle the infinite horizon control problem ($T = \infty$).

The time parameters τ and $\widehat{T}$ critically affect the performance of the MPC strategy though there is no general argument to determine them. In the numerical simulations of Section 3, we set $T = 4$, $\tau = 1.5$ and $\widehat{T} = 3$.

The error induced by the MPC strategy has been studied for linear problems,[26,34] however, it is difficult to extend to nonlinear systems as the guiding problem (2.1). The analysis of the reliability of the MPC-RBM strategy with error bounds is an interesting and challenging open problem. In the absence of analytical results on the MPC-RBM algorithm, we present several computational experiments that confirm the numerical efficiency of the method later in Section 3.

2.5. *Implementation of the MPC-RBM algorithm*

We now summarize the discussion from Section 2.1 to Section 2.4 on the RBM-MPC algorithm:

(1) The objective of the algorithm is to control the system (2.1) with a small computational cost, solving the optimal control problem for the cost $J^R(\mathbf{u})$ associated with the functional (2.3) in a (possibly long) time-interval $[0, T]$. This is done by combining MPC with the RBM approximated dynamics.
(2) We set a short time length $\tau > 0$ for the control time and a long time length $\widehat{T} \leq T$ for the predictive time. Then, we define the discrete times $\tau_m := m\tau$, $m \geq 0$. For each m, we iteratively solve the optimal control problems on each predictive time interval as follows.
(3) Since the original dynamics is autonomous, the time-interval $[\tau_m, \tau_m + \widehat{T}]$ can be shifted to $[0, \widehat{T}]$. The RBM model (2.5) is implemented, combined

with the Euler forward time-discretization. Then, the RBM control is computed by a gradient descent method (2.12) employing the corresponding RBM adjoint system (2.11).
(4) This control is implemented in the original system up to time τ_1.
(5) This strategy is iteratively applied in the intervals $[\tau_m, \tau_m+\widehat{T}]$ for the whole time interval $[0, T]$.

This Algorithm 1 has the following features:

(1) The cost function J^R monotonically decreases along the iteration of the gradient method and the algorithm converges to a local minimum. This is different from other stochastic approaches that guarantee the convergence only in expectation.
(2) The resulting control is not the (local) minimizer of the cost J for the original dynamics. But it gives an approximation, which is the minimizer of the cost J^R for the approximated dynamics.
(3) The total computational effort is reduced by a factor $O(N(P + M))/O(N(N + M))$, N being the dimension of the original system and P the size of the batches.
(4) The process of MPC acts in a semi-feedback way, periodically checking the current state of the original system to update the control.

3. Simulations on the MPC-RBM algorithm

We consider the guiding problem with 36 evaders ($N = 36$) and 2 drivers ($M = 2$) in the two-dimensional space ($d = 2$). The target is chosen to be $\mathbf{x}_f = (0.5, 0.5)$ with the final time $T = 4$ and the time step $\Delta t = 0.01$ in the implementation of the RBM. Initially the evaders are uniformly distributed in $[-0.2, 0.2]^2$ with zero velocities, and the drivers start from two points $(-1, 0)$ and $(0, -1)$.

The cost function is given by (2.3) and (2.7) with the regularization coefficients $\alpha_1 = 1$ and $\alpha_2 = \alpha_3 = 10^{-4}$ as follows:

$$J(\mathbf{u}) := \int_0^T \left[\frac{1}{N} \sum_{k=1}^{N} |\mathbf{x}_k - \mathbf{x}_f|^2 + \frac{10^{-4}}{M} \sum_{j=1}^{M} |\mathbf{u}_j|^2 + \frac{10^{-4}}{M} \sum_{j=1}^{M} |\mathbf{y}_j - \mathbf{x}_f|^2 \right] dt.$$

The numerical simulations are operated in Matlab with a laptop consisting of CPU i5-4258U 2.4GHz and RAM DDR3L 8GB 1600MHz. The symbolic calculations on gradients and adjoints are implemented with CasADi.[1] The functions on the forward and adjoint dynamics (*RBM-STATE* and *RBM-COSTATE* in Algorithm 1) are also implemented as CasADi symbolic functions for a fast calculation in a pre-calculated form. The random batches are chosen with the *randperm* function in Matlab.

Algorithm 1 MPC-RBM algorithm for (2.1)

P, τ, Δt and $\widehat{T}$ are given.
function RBM-State($\mathbf{x}^0, \mathbf{v}^0, \mathbf{y}^0, \mathbf{u}(t)$)
  Fix a random seed for the choices of batches.
  for n from 0 to $[\widehat{T}/\Delta t]$ **do**
    Divide $\{1, 2, \ldots, N\}$ into random batches with size P.
    for each batch **do**
      Update $\mathbf{x}_i^R$ and $\mathbf{v}_i^R$ by solving the reduced model (2.5) from $t = n\Delta t$ to $t = (n+1)\Delta t$.
    end for
    Update $\mathbf{y}_i$ from $t = n\Delta t$ to $t = (n+1)\Delta t$.
  end for
  return $\mathbf{x}^R(t)$, $\mathbf{v}^R(t)$ and $\mathbf{y}(t)$.
end function

function RBM-Costate($\mathbf{p}^0, \mathbf{q}^0, \mathbf{r}^0, \mathbf{x}(t), \mathbf{v}(t), \mathbf{y}(t)$))
  Fix a random seed, the same one from RBM-State in reverse order.
  for n from $[\widehat{T}/\widehat{\tau}]$ to 0 **do**
    Divide $\{1, 2, \ldots, N\}$ into random batches with size P.
    for each batch **do**
      Update $\mathbf{p}_i^R$ and $\mathbf{q}_i^R$ (the adjoint of $\mathbf{x}_i^R$ and $\mathbf{v}_i^R$) by solving the adjoint of the reduced model (2.5) from $t = (n+1)\Delta t$ to $t = n\Delta t$.
    end for
    Update $\mathbf{r}_i^R$ (the adjoint of $\mathbf{y}_i$) from $t = (n+1)\Delta t$ to $t = n\Delta t$.
  end for
  return $\mathbf{p}^R(t)$, $\mathbf{q}^R(t)$ and $\mathbf{r}^R(t)$.
end function

function OCP($\mathbf{x}^0, \mathbf{v}^0, \mathbf{y}^0, \mathbf{u}_0(t)$))
  Define the cost function J^R with (2.7) over $[0, \widehat{T}]$.
  Initialize the control $\mathbf{u}(t)$ with a guess $\mathbf{u}_0(t)$.
  while $\|D_{\mathbf{u}} J^R\| < \varepsilon$ (or any stopping criteria) **do**
    Operate RBM-State and RBM-Costate.
    Calculate the gradient $D_{\mathbf{u}} J^R$ of the cost J^R.
    Update $\mathbf{u}(t)$ using $D_{\mathbf{u}} J^R$.
  end while
  return $\mathbf{u}(t)$.
end function

procedure MPC-RBM algorithm
  Set the initial data $\mathbf{x}^0, \mathbf{v}^0, \mathbf{y}^0$ for the system (2.1).
  Give an initial guess on control $\mathbf{u}_0(t)$ for $t \in [0, T]$.
  Let $\tau_m := m\tau$ for $m = 0, 1, \ldots, [T/\tau] + 1$.
  for m from 0 to $[T/\tau]$ **do**
    Operate the function OCP with initial data $\mathbf{x}(\tau_m), \mathbf{v}(\tau_m), \mathbf{y}(\tau_m)$ and $\mathbf{u}_0(t)$ to get the optimal control $\mathbf{u}(t)$ for $t \in [\tau_m, \tau_m + \widehat{T}]$.
    Process the original system (2.1) with the control $\mathbf{u}(t)$ for $t \in [\tau_m, \tau_{m+1}]$.
  end for
  return the trajectories and control over $t \in [0, T]$.
end procedure

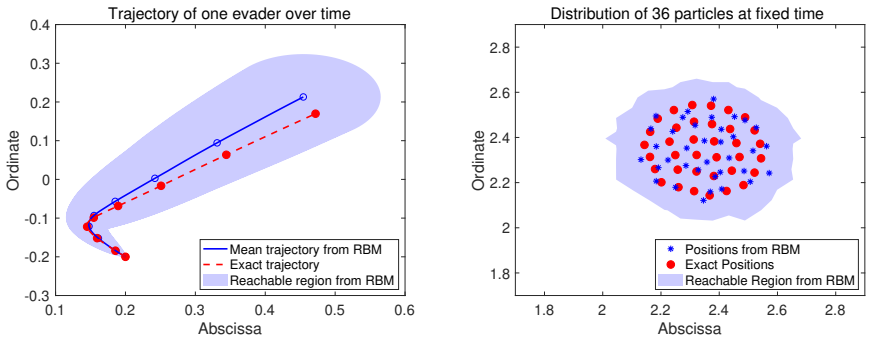

Fig. 2: Simulation of the evaders' trajectories using the RBM ($P = 2$). *Left*: the blue line shows the mean trajectory of one evader starting from $(-0.2, 0.2)$ for $t \in [0, 4]$, which is averaged among 200 RBM approximations. The red line represents the trajectory from (2.1). The reachable region is drawn with 95% reliability from the standard deviation and the 2D normal distribution. *Right*: The blue marks show the final positions of the 36 evaders from the reduced RBM model at $t = 10$, while the red marks are the final positions from the original system. The reachable region draws the area containing more than one particle from 200 RBM approximations, which represents 99% credible region.

Time-evolution	Computation time	(Ratio)	Interactions	(Ratio)
Full system	37.7220	(9.759)	8136	(10.27)
RBM ($P = 2$)	3.8654	(1.000)	792	(1.000)
RBM ($P = 4$)	6.6993	(1.733)	1224	(1.545)
RBM ($P = 6$)	8.9043	(2.304)	1656	(2.091)
RBM ($P = 9$)	11.3447	(2.935)	2304	(2.909)
RBM ($P = 18$)	19.4503	(5.032)	4248	(5.364)

Table 1: Computation time (in milliseconds) to calculate the forward dynamics (controlled trajectories) for $t \in [0, 10]$. For each reduced RBM model, the number of calculated interactions (per time step) is also described to compare with the original dynamics. The standard Euler forward method is used and averaged for 1000 simulations.

3.1. *Simulations on RBM for the controlled dynamics*

First, we explore the performance of the RBM for various values of P in the controlled dynamics, to discuss its time efficiency and approximation error.

Fig. 2 shows the positions of evaders simulated along time (left) and at fixed time (right). The trajectories are calculated with the original system and the reduced

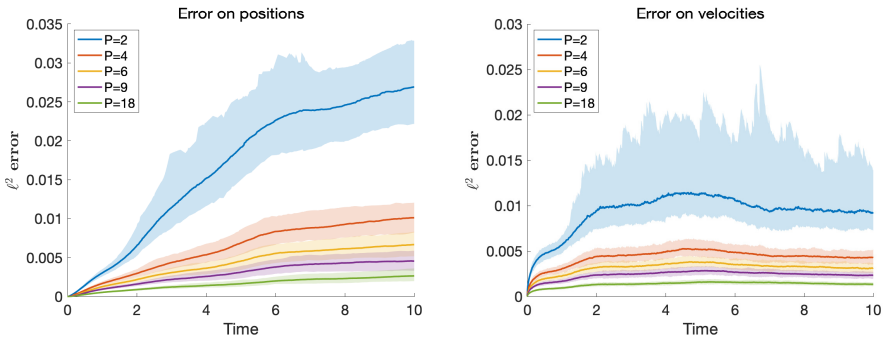

Fig. 3: The comparison between the dynamics from the RBM approximations and the exact system over time $t \in [0, 10]$. *Left*: the ℓ^2-errors on positions at each time with 200 independent RBM approximations for various P. The trajectory with median error is drawn in a thick line. The colored region shows the confidence interval for each time with 95% reliability. *Right*: the corresponding ℓ^2-errors for velocities.

RBM model with the same control functions. The controls are given by $\mathbf{u}_1(t) = (0.2, 0.02)$ and $\mathbf{u}_2(t) = (0.02, 0.2)$ for $t \in [0, 10]$, where the drivers push the evaders toward the northeast direction. Note that the motion of the drivers is not optimized in any sense but follows a constant function toward the direction of the sheep. This simulation then shows the error between the original and approximated trajectories from the same control functions.

The left figure shows the trajectory of one evader for $t \in [0, 4]$ starting from the point $(-0.2, 0.2)$. The blue colored reachable region is the confidence region with radius 1.73σ (95% reliability from the 2D normal distribution) from the standard deviation σ at each time. In the simulation, the radius of the reachable region at $t = 4$ is nearly 0.1, which is too rough compared to the distance between nearby evaders. However, according to the right figure, the blue markers (evaders from the reduced model) have a similar distribution to the red dots (evaders from the original system) at fixed time, $t = 10$. In particular, the reduced model produces good approximations for the mean position and the diameter of the herd, which are critical in the guiding problem.

On the other hand, Fig. 3 shows the approximation errors while the system evolves in time $t \in [0, 10]$. For each batch size P, 200 independent RBM approximations are numerically simulated to present the averaged ℓ^2 errors on positions,

$$(\text{Error on positions})(t) := \frac{1}{N}\sqrt{\sum_{i=1}^{N} |\mathbf{x}_i^R(t) - \mathbf{x}_i(t)|^2}, \quad N = 36,$$

and also for velocities. Among these 200 approximations, a representative one is

Adaptive GD	GD iter.	EV comp.	Cost J	Comp. time	(Ratio)
Full system	3123	9415	0.6646	767.96	(8.004)
RBM ($P = 2$)	2721	7567	0.6665	95.95	(1.000)
RBM ($P = 4$)	3054	9208	0.6651	164.02	(1.710)
RBM ($P = 6$)	3027	9127	0.6651	214.05	(2.231)
RBM ($P = 9$)	2604	7858	0.6648	229.57	(2.393)
RBM ($P = 18$)	2654	8008	0.6648	392.71	(4.093)

Table 2: The computation time (Comp. time) in the adaptive gradient descent algorithm for the guiding problem. The stopping criterion is set with tolerance 10^{-6} on the cost J, which tests the ratio between the difference of the cost and the previous cost. GD iterations (GD iter.) are the number of iterations in gradient descent, and EV computations (EV comp.) are the number of calculations on the time evolution for the adaptive step size.

drawn in a thick colored line, which has the median value of the time-integrated errors:

$$\int_0^T \frac{1}{N}\sqrt{\sum_{i=1}^N |\mathbf{x}_i^R(t) - \mathbf{x}_i(t)|^2}dt.$$

The colored region represents the confidence interval at each time, showing the other RBM approximations except for 10 trajectories (hence, 95% reliability) with 5 biggest and 5 smallest errors. Note that the errors on velocities are bounded and the errors on positions grow linearly on time t, not exponentially. We can also observe that the errors are reduced and more stable when P gets bigger.

In Table 1, the computation time for the evolution is described with different values of P, averaged over 1000 simulations. For N evaders and M drivers in a d-dimensional space, the number of interactions in the derivatives $(\dot{\mathbf{x}}^R, \dot{\mathbf{v}}^R, \dot{\mathbf{y}})$ can be estimated by $Nd(1 + Md + P(d + 1))$. Therefore, from the full system to the RBM one with $P = 2$, the computational cost is divided, roughly, by a factor of 10 (8136 in the original system and 792 in the reduced model). Note that the computational time is nearly proportional to the number of interactions in Table 1.

3.2. *Simulations on RBM for the optimal controls*

We now compute the optimal controls both for the original system and the reduced RBM model. As described above, we fix the choice of random batches during the optimization process.

In order to find the optimal control, here we use the gradient descent with an adaptive step size. The initial guess on the control is the constant functions used in the simulations of Fig. 2. The step size α in (2.12) is initially 0.1. If the cost $J^R(\mathbf{u}^{k+1})$ is not smaller than $J^R(\mathbf{u}^k)$, we try the half step size for (2.12) and

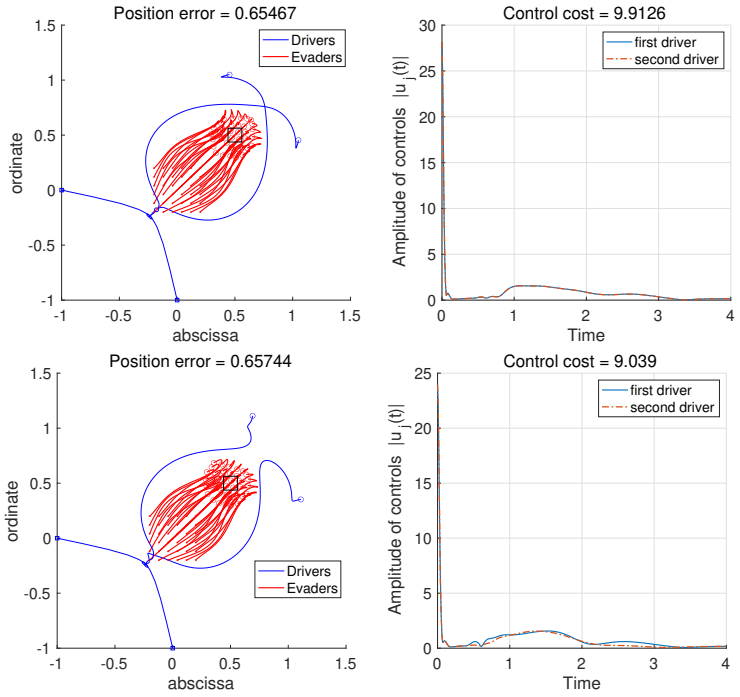

Fig. 4: Controlled trajectories generated by the optimal controls calculated with the original system (above) and the RBM model with $P = 2$ (below). The trajectories are simulated from the original system. *Left*: the trajectories of the drivers and evaders for $t \in [0, 4]$. Red lines are for the evaders and blue lines are for the drivers. Initially the evaders are pushed in the northeast direction. The position error denotes the time integration of the averaged squared distance from the evaders to the target point. *Right*: the strength of control functions along time. The control cost is the sum of time integrations, one on the averaged squared norm of the controls and the other on the averaged squared distance from the drivers to the target.

compute $\mathbf{u}^{k+1}$ again until $J^R(\mathbf{u}^{k+1}) < J^R(\mathbf{u}^k)$. The algorithm will be terminated if the step size α is less than 10^{-15} but still $J^R(\mathbf{u}^{k+1}) \geq J^R(\mathbf{u}^k)$. For the next iteration of the gradient descent, we double the step size of the previous iteration, to avoid redundant iterations and get a faster convergence to a local minimum. This adaptive method guarantees the monotonicity of the approximated cost J^R during the optimization process.

Table 2 shows the computation time to find the optimal controls with different values of P. Note that the optimization steps are not much different in all cases, hence, the computation time follows similar ratios to Table 1.

In Fig. 4, the controlled trajectories are presented with the different optimal controls from the original system and the reduced model ($P = 2$). Both trajectories

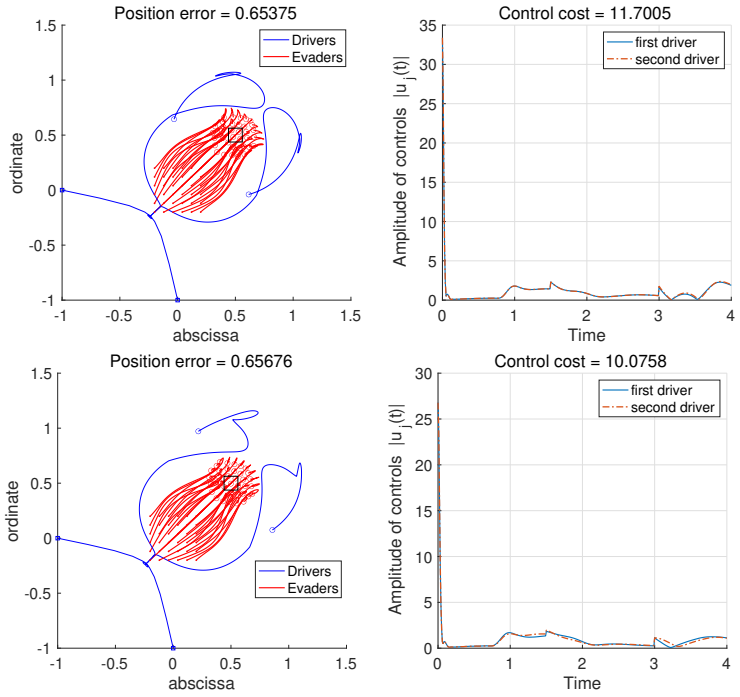

Fig. 5: The controlled trajectories from the MPC algorithms with $\tau = 1.5$ and $\widehat{T} = 3$ over the original system (above) and the reduced RBM model with $P = 2$ (below). *Left*: the trajectories of the drivers and evaders for $t \in [0, 4]$. *Right*: the strength of control functions along time.

are simulated on the system (2.1). Note that the detailed motion of drivers differs, however, the overall running costs are similar, as presented in Table 2.

3.3. *The effect of MPC in the guiding problem*

The simulations in Fig. 4 show the open-loop optimal control, which does not use the strategy of MPC. Next, we compare the MPC-RBM algorithm with the same conditions. Fig. 5 shows the simulations using the strategy of MPC, where the controls are calculated from the original system (above) and the RBM approximation with $P = 2$ (below).

For the process of MPC, we set $\tau = 1.5$ and $\widehat{T} = 3$. This implies that the controls on the intervals $[0.0, 1.5]$, $[1.5, 3.0]$ and $[3.0, 4.0]$ are calculated from the predictive time intervals $[0.0, 3.0]$, $[1.5, 4.5]$ and $[3.0, 6.0]$, respectively. For the first interval $[0, 3]$, we used the same initial guess on the control as before. However, for the next time intervals, we used the control obtained in the previous time intervals. In detail, for the second interval $[1.5, 4.5]$, we used the optimal control from the time horizon $[0.0, 3.0]$ to apply it on $[1.5, 3.0]$, and set zero values on $[3.0, 4.5]$.

	Control interval	GD iter.	Comp. time	Cost J
MPC-Full system	$[0.0, 1.5]$	2819	586.94	0.6654
(Fig. 5, above)	$[1.5, 3.0]$	1307	241.98	
(Total time: 955.81)	$[3.0, 4.0]$	7006	126.89	
MPC-RBM ($P = 2$)	$[0.0, 1.5]$	1727	47.42	0.6668
(Fig. 5, below)	$[1.5, 3.0]$	216	7.30	
(Total time: 82.96)	$[3.0, 4.0]$	1040	28.24	

Table 3: The number of iterations and computation time (in seconds) for Fig. 5 to find the optimal controls with MPC.

One of the differences between Fig. 4 and Fig. 5 is the motion of the drivers near the final time. In Fig. 5, the drivers move around the evaders to capture them near the target point $(0.5, 0.5)$. This is due to the effect of $\widehat{T}$ since the control is obtained with the dynamics on $t \in [0, 6]$, not $[0, 4]$. Therefore, in Fig. 5, the control on the time interval $[3, 4]$ is vibrating and not ignorablly small compared to the control in Fig. 4. Roughly speaking, the optimal control manipulates the evaders into a desired state until the final time $T = 4$, while the approximated control wants to keep it for a longer time, $t = 6$. Due to the unstability of the final state of evaders, the optimal control with, for example, an infinite horizon, needs to be active for the whole time.

On the other hand, the computation time for Fig. 5 is described in Table 3. The MPC algorithm formulates the optimal control problems three times, but the overall times are similar to Table 2. In particular, the simulation of the MPC algorithm takes 956 seconds with the original system and 83 seconds with the RBM, while the open-loop controls are calculated with 768 and 96 seconds, respectively.

Since it differs from case to case, it is difficult to estimate the computation time with MPC. However, note that the computational cost of the MPC-RBM algorithm is still in the order of $O(N(P + M))$, which is much less than $O(N^2)$ of the original system when N is large.

3.4. *Simulations of MPC-RBM on a noisy system*

As we presented in Section 2.4, the strategy of MPC is adopted to overcome the errors from the reduced dynamics. This effect can be seen significantly when the system has a noisy behavior. We consider a stochastic system by adding multiplicative

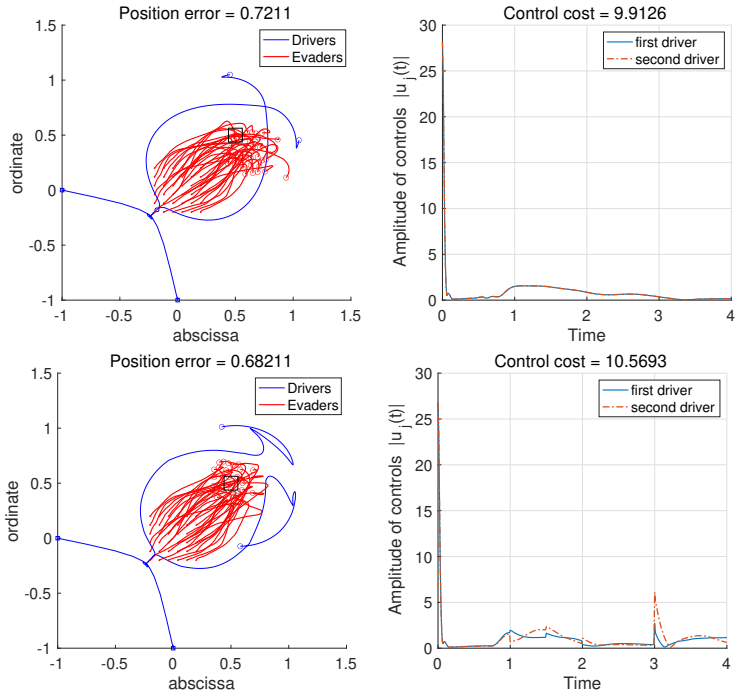

Fig. 6: The simulations on a noisy system with the open-loop control (above) and the control from the MPC-RBM algorithm with $P = 2$, $\tau = 1$ and $\widehat{T} = 3$ (below). The multiplicative noise is added on the evaders' velocities, where the evaders escape to the right direction in the case of the open-loop control. *Left*: the trajectories of the drivers and evaders for $t \in [0, 4]$. *Right*: the strength of control functions along time.

	Control interval	GD iter.	Comp. time	Cost J
MPC-RBM, noisy system	$[0.0, 1.0]$	1727	47.42	0.7040
(Fig. 6)	$[1.0, 2.0]$	429	12.27	
(Total time: 124.33)	$[2.0, 3.0]$	48	1.59	
	$[3.0, 4.0]$	2252	63.05	

Table 4: The number of iterations and computation time (in seconds) for Fig. 6 to find the optimal controls with the MPC-RBM algorithm.

noise $\mathbf{v}_i dB_t^i$ to the time derivatives of the velocities in (2.1):

$$\begin{cases} \dot{\mathbf{x}}_i = \mathbf{v}_i, \qquad i = 1, \dots, N, \\ d\mathbf{v}_i = \Bigg[\dfrac{1}{N-1} \displaystyle\sum_{k=1, k\neq i}^{N} a(\mathbf{x}_k - \mathbf{x}_i)(\mathbf{v}_k - \mathbf{v}_i) - \dfrac{1}{N-1} \sum_{k=1, k\neq i}^{N} g(\mathbf{x}_k - \mathbf{x}_i)(\mathbf{x}_k - \mathbf{x}_i) \\ \qquad\qquad - \dfrac{1}{M} \displaystyle\sum_{j=1}^{M} f(\mathbf{y}_j - \mathbf{x}_i)(\mathbf{y}_j - \mathbf{x}_i) \Bigg] dt + \sigma \mathbf{v}_i dB_t^i, \qquad i = 1, \dots, N \\ \dot{\mathbf{y}}_j = \mathbf{u}_j(t), \qquad j = 1, \dots, M, \\ \mathbf{x}_i(0) = \mathbf{x}_i^0, \quad \mathbf{v}_i(0) = \mathbf{v}_i^0, \quad \mathbf{y}_j(0) = \mathbf{y}_j^0, \qquad i = 1, \dots, N, \quad j = 1, \dots, M, \end{cases}$$

where $(B_t^1, \dots, B_t^N)$ is the N-dimensional Brownian motion and the noise strength σ is a constant, $\sigma = 0.5$. The multiplicative noise is simulated with Milstein method.

Fig. 6 shows the numerical simulations with the open-loop control (above) and the control from the MPC-RBM algorithm (below). The open-loop control is the one we calculated in Fig. 4 from the original deterministic system (2.1). The MPC-RBM algorithm is operated from the RBM reduced model (2.4) with $\tau = 1$ and $\widehat{T} = 3$ by updating the controlled trajectories of the noisy system. We set a smaller τ compared to the Fig. 5 since the difference from the controlled system and the reduced model is increased.

In the open-loop control, the evaders escape to an unexpected direction near the final time. Using the MPC-RBM algorithm, the drivers effectively surround the evaders with the information at $t = 1$, 2, and 3. The computation time and the cost functions are described in Table 4, where the cost function is reduced to 0.7040 from the cost of the open-loop control, 0.7561.

4. Conclusion and final remarks

In this paper, we combine the model predictive control (MPC) with the random batch methods (RBM) to get a reliable control strategy for the guiding problem with a large number of evaders in a short computation time. The suggested algorithm finds the optimal control on a reduced model from the RBM, as a predictive model in the process of MPC.

The RBM simplifies the full system through the random sampling of the interactions. With a given batch size P with $1 < P < N$, the computation cost on the dynamics is reduced to the order of $O(N(P + M))$ from $O(N(N + M))$ for the problem of N evaders and M drivers. Hence, in the MPC-RBM algorithm, the overall computation time is significantly reduced when there are plenty of evaders.

Though our focus is on a specific situation, the MPC-RBM algorithm works on a general interacting particle system. For example, the guiding problem can be simulated in a three-dimensional space or with a restriction on control. However, the error analysis of the algorithm is still open and leads to several interesting questions on the guiding problem. First, the error analysis of Ref. 22 only suggests the rough estimate in (2.6) that grows at most exponentially on the final time. Since Fig. 3 shows a linear growth, the approximation error for the guiding system (2.1) may be smaller due to the properties of the collective behavior dynamics. Moreover, the performance of the approximated control looks better than the approximation error by comparing Fig. 3 and Fig. 4. This seems also related to the fact that, in Fig. 2, the RBM approximates the density profile much better than one particle trajectory. The error analysis on the MPC-RBM algorithm will be left as a future work.

Acknowledgment

The authors would like to thank Dr. Umberto Biccari and Dr. Daniel Veldmann for their helpful comments on this work. This project has received funding from the

European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 694126-DyCon). The work of D. Ko was supported by the Catholic University of Korea, Research Fund, 2021, and by National Research Foundation of Korea (NRF-2021R1G1A1008559). The work of E. Zuazua has been funded by the Alexander von Humboldt-Professorship program, the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No.765579-ConFlex, grant MTM2017-92996-C2-1-R COSNET of MINECO (Spain), ELKARTEK project KK-2020/00091 CONVADP of the Basque Government, AFOSR Grant FA9550-18-1-0242, and Transregio 154 Project *Mathematical Modelling, Simulation and Optimization using the Example of Gas Networks* of the German DFG

References

1. J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, Casadi: a software framework for nonlinear optimization and optimal control, *Math. Program. Comput.* **11** (2019) 1–36.
2. R. Bailo, M. Bongini, J. A. Carrillo and D. Kalise, Optimal consensus control of the Cucker-Smale model, *IFAC-PapersOnLine* **51** (2018) 1–6.
3. U. Biccari, D. Ko and E. Zuazua, Dynamics and control for multi-agent networked systems: A finite-difference approach, *Math. Models Methods Appl. Sci.* **29** (2019) 755–790.
4. M. Bongini, M. Fornasier, O. Junge and B. Scharf, Sparse control of alignment models in high dimension, *Netw. Heterog. Media* **10** (2015) 647–697.
5. M. Bongini, M. Fornasier, F. Rossi and F. Solombrino, Mean-Field Pontryagin Maximum Principle, *J. Optim. Theory Appl.* **175** (2017) 1–38.
6. M. Burger, R. Pinnau, A. Roth, C. Totzeck and O. Tse, Controlling a self-organizing system of individuals guided by a few external agents – particle description and mean-field limit, *arXiv:1610.01325 [math]* .
7. M. Caponigro, M. Fornasier, B. Piccoli and E. Trélat, Sparse stabilization and optimal control of the Cucker-Smale model, *Math. Control. Relat. Fields* **3** (2013) 447–466.
8. J. A. Carrillo, Y.-P. Choi, P. B. Mucha and J. Peszek, Sharp conditions to avoid collisions in singular Cucker–Smale interactions, *Nonlinear Anal. Real World Appl.* **37** (2017) 317–328.
9. J. A. Carrillo, Y.-P. Choi, C. Totzeck and O. Tse, An analytical framework for consensus-based global optimization method, *Math. Models Methods Appl. Sci.* **28** (2018) 1037–1066.
10. J. A. Carrillo, S. Jin, L. Li and Y. Zhu, A consensus-based global optimization method for high dimensional machine learning problems, *ESAIM Control Optim. Calc. Var.* **27** (2021) S5.
11. F. Cucker and J.-G. Dong, Avoiding Collisions in Flocks, *IEEE Trans. Automat. Contr.* **55** (2010) 1238–1243.
12. F. Cucker and S. Smale, Emergent Behavior in Flocks, *IEEE Trans. Automat. Contr.* **52** (2007) 852–862.
13. F. Dorfler and F. Bullo, Synchronization and transient stability in power networks and nonuniform kuramoto oscillators, *SIAM J. Control. Optim.* **50** (2012) 1616–1642.
14. R. Escobedo, A. Ibañez and E. Zuazua, Optimal strategies for driving a mobile agent in a "guidance by repulsion" model, *Commun. Nonlinear. Sci.* **39** (2016) 58–72.

15. J. A. Fax and R. M. Murray, Information flow and cooperative control of vehicle formations, *IEEE Trans. Automat. Contr.* **49** (2004) 1465–1476.
16. S. Gade, A. A. Paranjape and S.-J. Chung, Herding a Flock of Birds Approaching an Airport Using an Unmanned Aerial Vehicle, in *AIAA Guidance, Navigation, and Control Conference* (American Institute of Aeronautics and Astronautics, Kissimmee, Florida, 2015).
17. C. E. García, D. M. Prett and M. Morari, Model predictive control: Theory and practice—A survey, *Automatica* **25** (1989) 335–348.
18. F. Golse, The mean-field limit for the dynamics of large particle systems, *Journ. Équ. Dériv. Partielles* (2003) 1–47.
19. L. Grüne and J. Pannek, Nonlinear model predictive control, in *Nonlinear Model Predictive Control* (Springer, 2017), pp. 45–69.
20. S.-Y. Ha, S. Jin and D. Kim, Convergence of a first-order consensus-based global optimization algorithm, *Math. Models Methods Appl. Sci.* To appear.
21. S.-Y. Ha and J.-G. Liu, A simple proof of the cucker-smale flocking dynamics and mean-field limit, *Commun. Math. Sci.* **7** (2009) 297–325.
22. S. Jin, L. Li and J.-G. Liu, Random Batch Methods (RBM) for interacting particle systems, *J. Comput. Phys.* **400** (2020) 108877.
23. D. Ko and E. Zuazua, Asymptotic behavior and control of a "guidance by repulsion" model, *Math. Models Methods Appl. Sci.* **30** (2020) 765–804.
24. J. M. Lien, O. B. Bayazit, R. T. Sowell, S. Rodriguez and N. M. Amato, Shepherding behaviors, in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004* (IEEE, New Orleans, LA, USA, 2004), pp. 4159–4164 Vol.4.
25. J. Ma and E. M.-K. Lai, Finite-time flocking control of a swarm of cucker-smale agents with collision avoidance, in *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)* (IEEE, Auckland, 2017), pp. 1–6.
26. P. Mhaskar, Robust model predictive control design for fault-tolerant control of process systems, *Ind. Eng. Chem. Res.* **45** (2006) 8565–8574.
27. S. Motsch and E. Tadmor, A New Model for Self-organized Dynamics and Its Flocking Behavior, *J. Stat. Phys.* **144** (2011) 923–947.
28. M. Nikolaou, Model predictive controllers: A critical synthesis of theory and industrial needs, *Adv. Chem. Eng.* **26** (2001) 131–204.
29. J. Park, H. J. Kim and S.-Y. Ha, Cucker-Smale Flocking With Inter-Particle Bonding Forces, *IEEE Trans. Automat. Contr.* **55** (2010) 2617–2623.
30. B. Piccoli, N. P. Duteil and E. Trélat, Sparse Control of Hegselmann–Krause Models: Black Hole and Declustering, *SIAM J. Control. Optim.* **57** (2019) 2628–2659.
31. R. Pinnau and C. Totzeck, Interacting particles and optimization, *PAMM* **18** (2018) e201800182.
32. L. S. Pontryagin, *Mathematical theory of optimal processes* (Routledge, 2018).
33. M. Porfiri and M. Di Bernardo, Criteria for global pinning-controllability of complex networks, *Automatica* **44** (2008) 3100–3106.
34. D. M. Prett and C. E. Garcia, Design of robust process controllers, *IFAC Proceedings Volumes* **20** (1987) 275–280.
35. V. Rokhlin, Rapid solution of integral equations of classical potential theory, *J. Comput. Phys.* **60** (1985) 187–207.
36. D. Strömbom, R. P. Mann, A. M. Wilson, S. Hailes, A. J. Morton, D. J. T. Sumpter and A. J. King, Solving the shepherding problem: heuristics for herding autonomous, interacting agents, *J. R. Soc. Interface* **11** (2014) 20140719.
37. H. G. Tanner, A. Jadbabaie and G. J. Pappas, Flocking in Fixed and Switching

Networks, *IEEE Trans. Automat. Contr.* **52** (2007) 863–868.
38. E. Trélat, *Contrôle optimal: théorie & applications* (Vuibert Paris, 2005).