Model Predictive Control with Random Batch Methods for a Guiding Problem

Enrique Zuazua

FAU - AvH

Joint work with Dongnam Ko

30 April 2020

Table of Contents

Motivation: Shepherd dogs and sheep

The number of individuals is small, yet the interaction dynamics and control strategies is complex

Herding Problem: a few shepherd dogs are required to steer a herd of sheep.

Figure: Picture of "Border Collie" and "Sheep" [from Wikipedia]

The system consists of small repelling agents (drivers) and a lot of escaping agents (evaders):

1. Guide the evaders in the right direction.
2. Capture the evaders into a desired area.

Motivation: "Guidance by repulsion" model

R. Escobedo, A. Ibañez and E.Zuazua, Optimal strategies for driving a mobile agent in a "guidance by repulsion" model, Communications in Nonlinear Science and Numerical Simulation, 39 (2016), 58-72.

[R. Escobedo, A. Ibañez, E. Zuazua, 2016] suggested a **guidance by repulsion** model based on the two-agents framework: *the driver*, which tries to drive *the evader*.

- The evader is influenced by a **repulsive force from each driver**, its strength being decreasing as the distance increases.
- The objective of the problem is to **simulate optimal locomotion of drivers** to guide the evaders to the desired region.

From this, it is natural to consider multi-evader problem. From the motivation from sheep, we assume the interactions between evaders.

- Each evader has **interactions with other evaders** to remain close together on the positions and velocities.

Guiding a large number of evaders

Let $\mathbf{x}_i$, $\mathbf{v}_i$ be the **position and velocity** of the ith evader ($i = 1, \dots, N$) in $2D$ space and $\mathbf{y}_j$ is the position of the jth driver ($j = 1, \dots, M$). Then, the dynamics can be given by interactions among individuals: for $t \geq 0$,

$$\begin{cases} \dot{\mathbf{x}}_i = \mathbf{v}_i, \quad i = 1, \dots, N, \\ \dot{\mathbf{v}}_i = \dfrac{1}{N-1} \displaystyle\sum_{k=1, k\neq i}^{N} a(\mathbf{x}_k - \mathbf{x}_i)(\mathbf{v}_k - \mathbf{v}_i) \qquad \leftarrow \text{velocity alignment} \\ \qquad + \dfrac{1}{N-1} \displaystyle\sum_{k=1, k\neq i}^{N} g(\mathbf{x}_k - \mathbf{x}_i)(\mathbf{x}_k - \mathbf{x}_i) \qquad \leftarrow \text{position flocking} \\ \qquad - \dfrac{1}{M} \displaystyle\sum_{j=1}^{M} f(\mathbf{y}_j - \mathbf{x}_i)(\mathbf{y}_j - \mathbf{x}_i), \quad i = 1, \dots, N, \leftarrow \text{evading drivers} \\ \dot{\mathbf{y}}_j = \mathbf{u}_j(t), \quad j = 1, \dots, M \qquad \leftarrow \text{drivers are directly controlled} \\ \mathbf{x}_i(0) = \mathbf{x}_i^0, \quad \mathbf{v}_i(0) = \mathbf{v}_i^0, \quad \mathbf{y}_j(0) = \mathbf{y}_j^0. \end{cases}$$

Studies on the herding problem

Similar consideration have been addressed with repulsive interactions in control theory:

- Problems on gathering sheep:
  Well-posedness of optimal control problems [Burger, Pinnau, Roth, Totzeck, Tse, 2016]
  and its simulations [Pinnau, Totzeck, 2018].
- Repelling birds from the airport: [Gade, Paranjape, Chung, 2015],
- Modeling hunting strategies:
  [Muro, Escobedo, Spector, Coppinger, 2011 and 2014],

Question: How we can **efficiently simulate the locomotion of drivers** controlling a lot of evaders?

Table of Contents

Optimal Control Problem

To obtain proper control functions, $\mathbf{u}_j(t)$, $j = 1, \dots, M$, $t \in [0, T]$, a natural strategy is to find the **optimal control** which minimizes a cost function J:

$$\mathbf{u}^* := \operatorname{argmin}_{\mathbf{u}} J(\mathbf{u}).$$

Since our objective is to guide the evaders to the desired area, we may set the cost $J(\mathbf{u}(\cdot))$ as, for a given target point $\mathbf{x}_f \in \mathbb{R}^2$,

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

We expect that, by minimizing the time integration on $|\mathbf{x}_k - \mathbf{x}_f|^2$, the optimal control will guide the evaders to $\mathbf{x}_f$.

Simulation on the optimal control problem

The following figure shows the optimal control problem with 36 evaders and 2 drivers toward the target $(0.5, 0.5)$ in the time horizon $[0, 4]$.

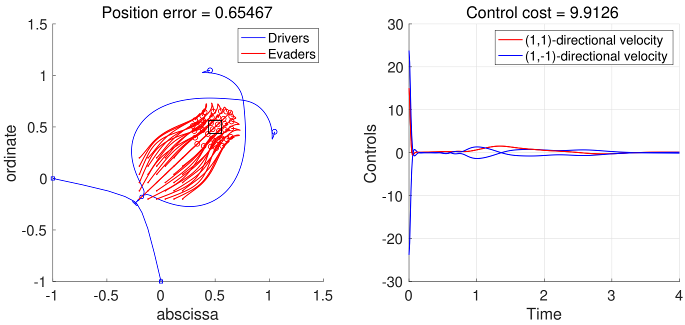


Figure: Controlled trajectories and the control functions from the optimal control problem.

The drivers started from $(0, -1)$ and $(-1, 0)$ while the evaders are initially gathered near $(0, 0)$.

Gradient descent method

The **gradient descent method** is widely used to obtain the optimal control.

This optimization algorithm searches the optimal control from an iterative approximation of the optimal control,

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

our of an initial guess $\mathbf{u}^0$ (for example, $\mathbf{u}^0(t) = 0$ for all $t \in [0, T]$) with $\alpha > 0$ small enough.

Hence, we need to calculate the gradient of the cost $\nabla_{\mathbf{u}} J(\mathbf{u}^k)$ iteratively on each step of optimization algorithm.

Gradient of the cost function

The gradient of the cost function, $\nabla_{\mathbf{u}} J(\mathbf{u}^k)$, can be obtained from the controlled dynamics with $\mathbf{u}^k$ from the Pontryagin Maximal Principle. From the optimal control problem

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

we define the backward dynamics of adjoint variables $\mathbf{p}(t), \mathbf{q}(t), \mathbf{r}(t)$, $t \in [0, T]$,

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

Then, the gradient of the cost is given by the derivative of Hamiltonian,

$$\nabla_{\mathbf{u}} J = \nabla_{\mathbf{u}} [\mathbf{p} \cdot F^{\mathbf{x}} + \mathbf{q} \cdot F^{\mathbf{v}} + \mathbf{r} \cdot F^{\mathbf{y}} + L] = \mathbf{r} + (\alpha_2 / M) \mathbf{u}. \qquad (1)$$

Implementation of Gradient Descent

Algorithm 1 Pseudocode for the Optimal Control Problem

```
1: The step size $\alpha > 0$ and tolerance $\varepsilon > 0$ are given.
2: function OCP($(\mathbf{x}^0, \mathbf{v}^0, \mathbf{y}^0), \mathbf{u}_0(t), [0, T]$)
3:     Define the cost function $J$ over $[0, T]$.
4:     Initialize the control, $\mathbf{u}(t) = \mathbf{u}_0(t)$.
5:     while $\|D_\mathbf{u} J\| < \varepsilon$ (or any stopping criteria) do
6:         Compute the controlled trajectories $(\mathbf{x}(t), \mathbf{v}(t), \mathbf{y}(t))$ from the initial data $(\mathbf{x}^0, \mathbf{v}^0, \mathbf{y}^0)$.
7:         Compute the adjoint time evolution $(\mathbf{p}(t), \mathbf{q}(t), \mathbf{r}(t))$.
8:         Calculate the gradient $D_\mathbf{u} J(t) = \mathbf{r}(t) + (\alpha_2/M)\mathbf{u}(t)$.
9:         Update $\mathbf{u}(t) = \mathbf{u}(t) - \alpha D_\mathbf{u} J(t)$.
10:    end while
11:    return $\mathbf{u}(t)$.
12: end function
```

Difficulties

- Note that, in gradient descent methods, we **need to compute the controlled dynamics** and its adjoint at each step of optimization.
- The problem is that, in order to compute the derivatives at a fixed time, we need to consider $N - 1$ interacting evaders for each evader. Hence, it takes $O(N^2)$ **computational complexity** for N evaders.
- This is same for the controlled dynamics and its adjoint system. **Finding the optimal control becomes unfeasible** as N increases.
- We want to construct an **approximative model** for the dynamics to reduced computational cost.

Table of Contents

Approximative dynamics: Random Batch Methods (RBM)

The random batch methods (RBM) [Shi Jin et al., 2020, JCP] is an approximation method for large systems of particles (evaders), particularly suitable when the particles are homogeneous.

For example, we need the following positional interactions in the computation of $\dot{\mathbf{v}}_i$:

$$\frac{1}{N-1}\sum_{k=1,k\neq i}^{N} g(\mathbf{x}_k-\mathbf{x}_i)(\mathbf{x}_k-\mathbf{x}_i) \qquad \leftarrow \text{position flocking}$$

The interactions are averaged from the pairs of evaders, only depending on the relative position.

Instead of computing $N-1$ interactions, for a given $1 < P < N$, the RBM approximates dynamics out of $P-1$ **interactions on each evader**.

Random Batch Methods (RBM)

More precisely, for a small duration of time Δt, 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**.

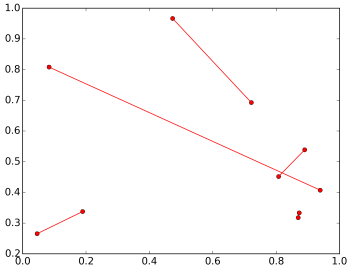


Figure: Grouping 10 particles into pairs (batches with size $P = 2$).

This choice of grouping needs to be **independent for each time** intervals, namely, $[t_n, t_{n+1}]$ from $t_n = n\Delta t$, $n \geq 0$.

Random Batch Methods (RBM)

The connectivity at fixed time can be represented by the following adjacency matrix:

$$A_{ij}^n := \begin{cases} 1 & \text{if } \; i,j \text{ are in the same batch,} \\ 0 & \text{otherwise,} \end{cases}$$

$$A_{ij}(t) := A_{ij}^n \quad \text{for} \quad t \in [t_n, t_{n+1}),$$

Hence, this is a switching network system for $t \in [0, T]$.

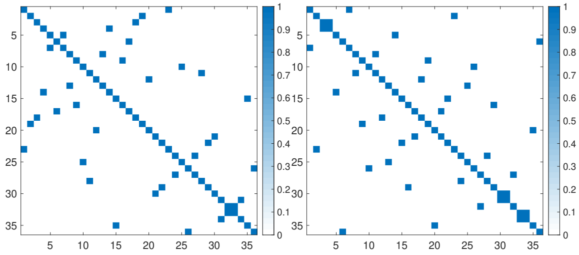


Figure: Network representations on two different connectivities, A_{ij}^0 and A_{ij}^1.

Random Batch Methods (RBM)

Hence, the interaction term in $\dot{\mathbf{v}}_i$,

$$\frac{1}{N-1}\sum_{k=1,k\neq i}^{N} g(\mathbf{x}_k-\mathbf{x}_i)(\mathbf{x}_k-\mathbf{x}_i)$$

is approximated by

$$\frac{1}{P-1}\sum_{k=1,k\neq i}^{N} A_{ij}(t)g(\mathbf{x}_k-\mathbf{x}_i)(\mathbf{x}_k-\mathbf{x}_i).$$

For example, if we do RBM approximation for $T=4$ and $\Delta t = 0.01$, then $A_{ij}(t)$ is changed 400 times to average random effect for $t\in[0,4]$.

Note that the number of computation is reduced to $O(N(P+M))$ from $O(N(N+M))$ in the original system.

Implementation of Random Batch Methods

Algorithm 2 Approximated controlled dynamics from the RBM

```
1:  The time discretization $\Delta t > 0$ and batch size $P > 1$ are given.
2:  function RBM-State($(\mathbf{x}^0, \mathbf{v}^0, \mathbf{y}^0), \mathbf{u}(t), [0, T]$)
3:      Set a random seed for the choices of batches.
4:      for $n$ from 0 to $[T/\Delta t]$ do
5:          Divide $\{1, 2, \ldots, N\}$ into random batches with size $P$.
6:          for $i = 1, \ldots, N$ do
7:              Compute $\mathbf{x}_i(t_{n+1})$ and $\mathbf{v}_i(t_{n+1})$ from $\mathbf{x}_k(t_n)$ and $\mathbf{v}_k(t_n)$, the
    states of $k$th evaders in the same batch as $i$th evader, and $\mathbf{y}(t_n)$.
8:          end for
9:          Compute $\mathbf{y}(t_{n+1})$ from $\mathbf{y}(t_n)$ and $\mathbf{u}(t_n)$.
10:     end for
11:     return $\mathbf{x}(t_n)$, $\mathbf{v}(t_n)$ and $\mathbf{y}(t_n)$ for all $n$.
12: end function
```

Simulations on the RBM

The approximation error from the RBM is still open, not estimated rigorously in general systems. In numerical simulations, RBM properly approximates the distribution of evaders, while tracking each particle is difficult.

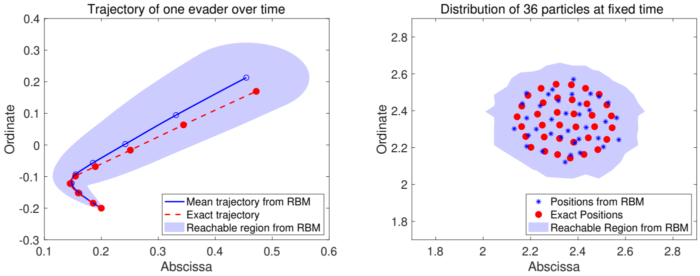


Figure: RBM approximation on trajectories of the evaders, one evader along time $t \in [0, 4]$ (left) and the whole evaders at a fixed time $t = 10$ (right). The colored region is drawn from 200 RBM approximations. The guiding problem is simulated with constant controls pushing evaders in the northeast direction.

Simulations on the RBM

For $N = 36$, $M = 2$ and $P = 2$, the computation on the approximated time evolution ($t \in [0, 10]$, $\Delta t = 0.01$) takes 3.87 milliseconds, which is nearly 10% of the time from the complete system, 37.72 milliseconds.

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: The computation time (in milliseconds) to calculate the controlled trajectories of $t \in [0, 10]$. The number of interactions is also denoted to compare with the time. The standard Euler forward method is used and averaged for 1000 simulations.

RBM for the adjoint system

Note that, on the RBM approximated dynamics, the Pontryagin maximal principle can be applied to deduce the adjoint system and the gradient of the cost.

Since the adjoint system is also described by the connectivity matrix $A(t)$, the computational complexity is also $O(N(P+M))$.

In conclusion, we can find the **optimal control of the RBM reduced model with $O(N(P+M))$ computational cost**. This is an approximative control for the original system.

Implementation of RBM on the adjoint system

Algorithm 3 Approximated adjoint dynamics from the RBM

```
1: The time discretization $\Delta t > 0$ and batch size $P > 1$ are given.
2: function RBM-AdjointState($(\mathbf{p}^0, \mathbf{q}^0, \mathbf{r}^0), (\mathbf{x}(t), \mathbf{v}(t), \mathbf{y}(t)), [0, T]$)
3:     Set a random seed, the same one as in forward dynamics, RBM-State().
4:     for $n$ from $[T/\Delta t]$ to 0 do
5:         Divide $\{1, 2, \dots, N\}$ into random batches with size $P$.
6:         for $i = 1, \dots, N$ do
7:             Compute $\mathbf{p}_i(t_n)$ and $\mathbf{q}_i(t_n)$ from $\mathbf{p}_k(t_{n+1})$ and $\mathbf{q}_k(t_{n+1})$ of $k$th evaders in the same batch as $i$th evader, and $(\mathbf{x}(t_{n+1}), \mathbf{v}(t_{n+1}), \mathbf{y}(t_{n+1}))$.
8:         end for
9:         Compute $\mathbf{r}(t_{n+1})$ from $\mathbf{r}(t_n)$ and $(\mathbf{x}(t_{n+1}), \mathbf{v}(t_{n+1}), \mathbf{y}(t_{n+1}))$.
10:    end for
11:    return $\mathbf{p}(t_n)$, $\mathbf{q}(t_n)$ and $\mathbf{r}(t_n)$ for all $n$.
12: end function
```

Simulation on the optimal control from the RBM

The following figure shows the optimal control from the RBM reduced model, applied to the original dynamics.

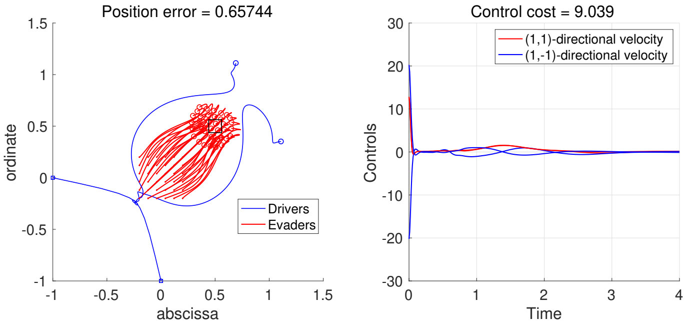


Figure: Controlled trajectories and the control functions from the RBM.

The performance of the control $J(\mathbf{u})$ is similar to the optimal control (from 0.6646 to 0.6665), while the computation time is reduced from 767.96 to 95.95 seconds.

Table of Contents

Error accumulates along time

Note that the approximation error, as in other numerical methods, grows while the system evolves.

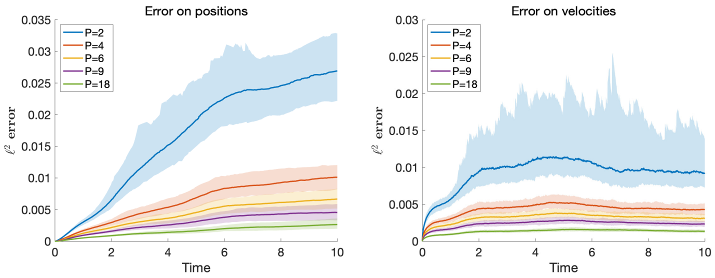


Figure: The errors on the positions (left) and velocities (right) in ℓ^2-norm. The colored region is drawn from 200 RBM approximations.

This may lead to the failure of control objectives, when the error is unfortunately large in the long-time horizon.

Model Predictive Control (MPC)

To overcome this difficulty, we adopt the process of **Model Predictive Control (MPC)**.

MPC is a control design, aimed to adapt the control obtained for the reduced dynamics to the full system in an iterative manner.

The basic idea is that, at each discrete time $\tau_m := m\tau$ for $m \geq 0$, we **update the current information of the controlled system** and calculate the optimal control again for the next time.

In this manner, the MPC strategy takes account of the gap between the trajectories of the complete and the reduced dynamics.

Model Predictive Control (MPC)

For this to be done, first, at $t = 0$, we compute the optimal control of the reduced model for an artificial time-horizon $[0, \hat{T}]$. (For example, $\hat{T} = T$). This control will be applied to the original system for $t \in [0, \tau]$.

At $t = \tau$, we again compute the optimal control for $[\tau, \tau + \hat{T}]$ with the initial data drawn from the original system. This strategy is repeated until the system evolves for $t \in [0, T]$.

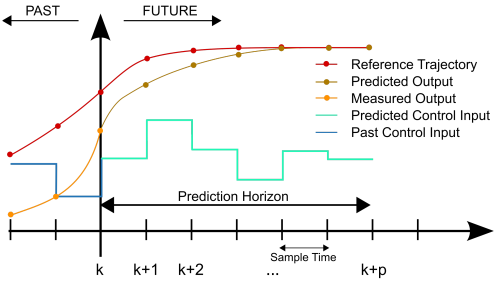


Figure: The diagram of iterative control computation in MPC [Wikipedia].

Implementation of Model Predictive Control

Algorithm 4 The procedure of MPC with receding time intervals

```
1: The control time $\tau > 0$ and predictive time $\hat{T} \geq \tau$ are given.
2: procedure MPC-RBM algorithm
3:     Set the initial data $(\mathbf{x}^0, \mathbf{v}^0, \mathbf{y}^0)$ for the contorolled system.
4:     Initialize the control function $\mathbf{u}(t)$ from $\mathbf{u}_0(t)$ for $t \in [0, T]$.
5:     Let $\tau_m := m\tau$ for $m = 0, 1, \dots, [T/\tau] + 1$.
6:     for $m$ from 0 to $[T/\tau]$ do
7:         Construct the Optimal Control Problem (OCP) for $[\tau_m, \tau_m + \hat{T}]$ with the initial data $(\mathbf{x}(\tau_m), \mathbf{v}(\tau_m), \mathbf{y}(\tau_m))$.
8:         Operate OCP with RBM to get $\mathbf{u}(t)$ for $[\tau_m, \tau_m + \hat{T}]$.
9:         Process the original system with $\mathbf{u}(t)$ for $t \in [\tau_m, \tau_{m+1}]$.
10:        Get the data $\mathbf{x}(\tau_{m+1})$, $\mathbf{v}(\tau_{m+1})$ and $\mathbf{y}(\tau_{m+1})$ for the next loop.
11:    end for
12:    return the combined control $\mathbf{u}(t)$ of $[\tau_m, \tau_{m+1}]$ for all $m$.
13: end procedure
```

Simulations of MPC on the full system

The time parameters τ and $\hat{T}$ critically affect the performance of the MPC strategy though there is no general argument to determine them.

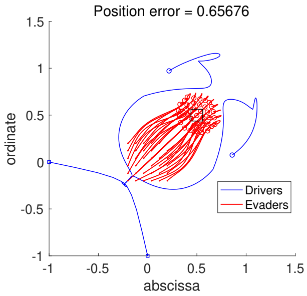
If τ is too large, then the approximation error of the RBM will critically affect the control function. Hence, we need a small τ, but the computational cost gets biggger.

On the other hand, if $\hat{T}$ is too small, then the control cannot catch the long-time behavior of the system. This may cause an oscillatory behavior of the system near the target trajectories.

For example, we set $T = 4$, $\tau = 1.5$ and $\hat{T} = 3$ in the simulations of guiding problems.

Simulations of MPC with RBM approximation

Since MPC updates the current positions (and velocities) of evaders periodically, the resulting control captures the evaders in a better way.



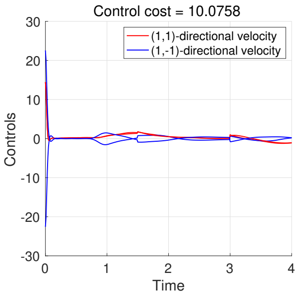


Figure: Controlled trajectories and the control functions from the process of MPC with the RBM reduced model.

We can also observe that the drivers go around the evaders near the final time, being ready to guide them even after $t = 4$, from the effect of $\hat{T}$.

The effect of MPC on a noisy system (without MPC)

The correction of the control can be seen significantly when the system has a noisy behavior. For example, we may consider the situation that the evaders suddenly move to the right direction.

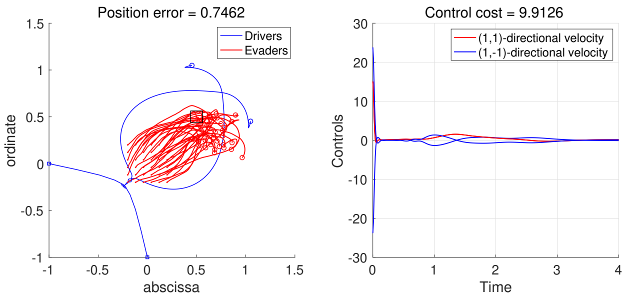


Figure: Controlled trajectories from the Optimal control problem, when the evaders's velocities are affected by random noise.

The optimal control computed a priori cannot react on such situation.

The effect of MPC on a noisy system (with MPC)

The strategy of MPC-RBM catches the error on the position at $t = \tau$, and modifies the control for the next time to minimize the cost function.

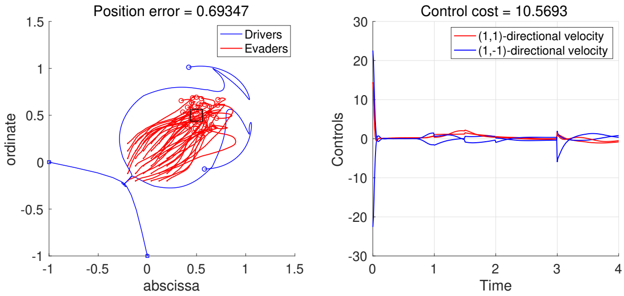


Figure: Controlled trajectories from the MPC-RBM ($\tau = 1$ and $\hat{T} = 3$), when the evaders's velocities are affected by random noise.

Table of Contents

Summary

- The algorithm combines two arguments, MPC and RBM, in order to compute a reliable control strategy in a short time.
- RBM reduces the computation cost on the forward and adjoint dynamics, from the order of $O(N^2)$ to $O(NP)$.
- MPC makes the approximative control adapt to the original system from an iterative computation.
- From a simulation with 36 evaders and 2 drivers, the computation cost is reduced to 16%, while the performance of control J differs only about 0.5%.

Future directions

In nonlinear systems, **the effect of approximation error is not rigorously studied yet**. This needs to be analyzed in the following different levels:

- On the approximation error of the RBM in general complex systems.
- On the performances of control functions with respect to the errors in the model.
- On the effect of MPC in the controlled dynamics.

The error estimates of RBM has been done contracting systems [Jin, Li, Liu, 2020, JCP], though numerical simulations show good performances [Carrillo, Jin, Li, Zhu, 2019], [Ha, Jin, Kim, 2019].

The analysis of MPC has been focused on the linear systems, for instance, [Mhaskar, 2006], [Prett, Garcia, 1987].

Future directions

Though we only suggested 2D problems, the MPC-RBM algorithm and the formulation of guiding problem is not restricted to a specific case.

Figure: *Left*: Herding sheep with drone [Dailypost], *Right*: the sheep on non-flat hills [Scottish Farmer]