

Supplemental Results: Examples and Evaluations

LEKAN MOLU, Microsoft Research, USA

CCS Concepts: • **Software and its engineering** → **Software libraries and repositories**; • **Applied computing** → *Physical sciences and engineering*; • **Mathematics of computing** → *Solvers*.

Additional Key Words and Phrases: partial differential equations, level sets, reachability theory

ACM Reference Format:

Lekan Molu. 2025. Supplemental Results: Examples and Evaluations. *J. ACM* 00, 0, Article 000 (2025), 26 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 EXAMPLES AND NUMERICAL EXPERIMENTS

We now present problems motivated by real-world scenarios and amend them to HJ PDE forms where their numerical solutions can be resolved with our `LevelSetPy` toolbox. The problems that we consider belong in `transport`, `differential games`, and `time-to-reach` problem classes. The library has been tested on numerous problems; however, for the sake of brevity we will only report a few results.

For the *differential games*, we do not necessarily analyze a single game, but rather a *collection/family of games*, $\Upsilon = \{\Gamma_1, \dots, \Gamma_g\}$. Each game within a differential game may be characterized as a pursuit-evasion game, Γ . Such a game terminates when *capture* occurs, that is the distance between players falls below a predetermined threshold. Each player in a game shall constitute either a pursuer (P) or an evader (E). Let the cursory reader not interpret P or E as controlling a single agent. In our various numerical experiments, we are poised with one or several pursuers (enemies) or evaders (peaceful citizens). However, when P or E governs the behavior of but one agent, these symbols will denote the agent itself. The nucleolus of our illustrative examples is to geometrically (approximately) ascertain the separation between the CZ and EZ surfaces i.e. the *barrier hypersurface*, where starting points exist for which escape occurs, capture occurs, and for which the outcome is neutral.

To address our desiderata, we must settle upon how best should P pursue E . Here, at every time instant, P possesses knowledge of his own and that of E 's position so that P knows how to regulate its various controlling variables with respect to E 's motion in an optimal fashion. The task is to assay the *game of kind* for the envelope of the capturable states i.e. we are not so much as seeking a game's outcome as we are seeking the conditions under which capture can occur. This introduces the *barrier hypersurface* which separates, in the initial conditions space, the hypersurface of capture from those of escape. In this *game of kind* postulation, all optimal strategies are not unique, but rather are a *legion*. Ergo, we are concerned with the set of initial positions on the vectogram where the capture zone (CZ) exists i.e. where game termination occurs; and the nature of escape zones (EZ) i.e. zones where termination or escape does not occur – after playing the differential game.

Author's address: Lekan Molu, lekanmolu@microsoft.com, Microsoft Research, 300 Lafayette Street, New York, NY, USA, 10012.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2025 Association for Computing Machinery.

0004-5411/2025/0-ART000 \$15.00

<https://doi.org/XXXXXXX.XXXXXXX>

The rest of this section introduces different representative examples where real-world problems are adopted and amended to HJ PDE forms and whose solution we seek to numerically recover. Space here has limited us to four illustrative examples: First, we present two rockets in a pursuit-evasion game where the goal is for the evader to guard a territory and the pursuer's goal is to penetrate the boundary of the territory being guarded. Second, we describe a double integral dynamical system: the double integral plant is a simplified abstraction of many real-world force-control system e.g. those that obey Newton's second law of motion or the torque-inertia dynamics of a body with rotary dynamics. We provide the numerical enumeration of the solution to the analytical time to reach problem. Third, we describe a collective behavior system in natural starlings and we provide a mathematical abstraction that allows the computation of the collision-avoidance *safety envelope* that may then be used in e.g. runtime assurance (RTA) safety-critical controller. Fourth, we compare the resolution of the (approximately) capturable sets in Dubins' game of two identical vehicles with the levelsets toolbox's solution. RTA controllers act intelligently as a safety system between a real-world controller and the system to be controlled by providing a state monitoring scheme useful in intervening in the real-world where vulnerabilities to danger is a constant factor to be mitigated against. All the examples presented in this section can be found in the [examples](#) folder of our online library.

1.1 Two Rockets in a Pursuit-Evasion Terminal Value Differential Game

We adopt the rocket launch problem of Dreyfus [15] and amend it to a differential game between two identical rockets, \mathbf{P} and \mathbf{E} , on an (x, z) cross-section of a Cartesian plane. We set out to compute the *usable part* of the boundary of the *approximate* terminal surface of a predefined target set over a time horizon (i.e. the target tube). The usable part entails the regions of the state-space for which the min-max operation over either strategy of \mathbf{P} and \mathbf{E} is below 0. The boundary of the usable part (BUP) constitute where the variational HJI PDE is exactly zero.

The BUP, *target tube*, or in modern parlance backward reachable tube (BRT) shall be implicitly constructed with our LevelSetPy library as the zero-level set of an implicitly defined function over the entire state space. At the zero level set, resolving the *kinematic equation* of the rockets in relative coordinates helps us understand the nature of the barrier hypersurface. Specifically, the target tube is a terminal surface that enunciates the set of initial starting points for which termination (capture or \mathbf{C}), no termination (escape or \mathbf{E}) of a game does occur, or analyzing the barrier separating \mathbf{C} or \mathbf{E} after playing the differential game.

A single rocket's motion is dictated by the following system of differential equations (under Dreyfus' assumptions):

$$\dot{x}_1 = x_3, \quad x_1(t_0) = 0; \quad (1a)$$

$$\dot{x}_2 = x_4, \quad x_2(t_0) = 0; \quad (1b)$$

$$\dot{x}_3 = a \cos u, \quad x_3(t_0) = 0; \quad (1c)$$

$$\dot{x}_4 = a \sin u - g, \quad x_4(t_0) = 0; \quad (1d)$$

where, (x_1, x_2) are respectively the horizontal and vertical range of the rocket (in feet); (x_3, x_4) are respectively the horizontal and vertical velocities of the rocket (in feet per second); and a and g are respectively the acceleration and gravitational accelerations (in feet per square second).

We now make the problem amenable to a two-player differential game. Let rockets \mathbf{P} and \mathbf{E} share identical dynamics in a general sense. The coordinates of \mathbf{P} are freely chosen; however, the coordinates of \mathbf{E} are chosen a distance ϕ away from (x, z) at the origin of the plane (as illustrated in Fig. 1) so that the \mathbf{PE} vector's inclination measured counterclockwise from the x axis is θ .

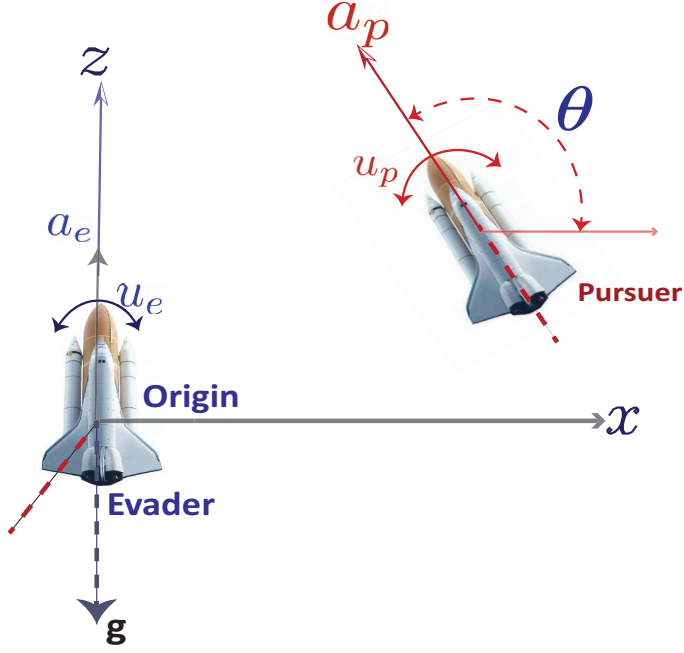


Fig. 1. Motion of two rockets on a Cartesian xz -plane with a thrust inclination in relative coordinates given by $\theta := u_p - u_e$.

Being a free endpoint problem, let the states of P and E be denoted by (x_p, x_e) . Furthermore, let the rockets be driven by their thrusts, denoted by (u_p, u_e) for P and E respectively (see Figure 1). Fix the range of the rockets so that what is left of the motion of either P or E 's is restricted to orientation on the (x, z) plane as illustrated in Fig. 1. Whence, the relevant *kinematic equations* (KE) from equation (1) are

$$\dot{x}_{2e} = x_{4e}; \quad \dot{x}_{2p} = x_{4p}, \quad (2a)$$

$$\dot{x}_{4e} = a \sin u_e - g; \quad \dot{x}_{4p} = a \sin u_p - g \quad (2b)$$

where a and g are respectively the acceleration and gravitational accelerations (in feet per square second)¹.

We want to determine the outcome of a simulated game between the two agents over a time interval. In the process of this protracted simulation, the nature of the barrier surface (henceforth called the backward reachable tube [37] or BRT²) will change.

Our desideratum is determining if capture can be achieved at all in a “yes-or-no” fashion. Therefore, we pose the game over a finite range over outcomes so that the game at hand assumes Isaac’s [26] description of a *game of kind*. P can achieve as much proximity to a given *target set* as much as possible while E is set up to protect the *target set*. For example, one may take P as seeking to penetrate a (closed) territory (called *target*) under guard by player E ; and P 's goal may be to maximize the time of play so as to penetrate the barrier surface of the target. E seeks to protect a given target's surface. As long as E remains within this *backward reachable tube* (or BRT), P

¹We set $a = 1 \text{ ft/sec}^2$ and $g = 32 \text{ ft/sec}^2$ in our simulation.

²It is called backward because the game is simulated backward in time.

cannot cause damage or exercise an action of deleterious consequence on, say, the territory being guarded by E .

Setting up E to maximize the payoff quantity (4) with the largest possible margin or at least frustrate the efforts of P with minimal collateral damage while the pursuer minimizes the payoff quantity constitutes a terminal value *optimal* differential game: there is no optimal pursuit without an optimal evasion since P and E are both executing motions as they see fit within the problem parameters.

Therefore, we rewrite (1) with P 's motion relative to E 's along the (x, z) plane so that the relative orientation as shown in Fig. 1 is $\theta = u_p - u_e$ – this shall serve as the control input. Following the conventions in Fig. 1, the game's relative equations of motion in *reduced space* [26, §2.2] i.e. is $\mathbf{x} = (x, z, \theta)$ where $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $(x, z) \in \mathbb{R}^2$ are

$$\dot{\mathbf{x}} = \begin{cases} \dot{x} &= a_p \cos \theta + u_e x, \\ \dot{z} &= a_p \sin \theta + a_e + u_e x - g, \\ \dot{\theta} &= u_p - u_e. \end{cases} \quad (3)$$

The boundary of the *usable part* of the origin-centered circle of radius ϕ (we have set $\phi = 1.5$ feet in our evaluations) is $\|\mathbf{PE}\|_2$ so that

$$\phi^2 = x^2 + z^2. \quad (4)$$

All capture points are specified by the variational HJ PDE [37]:

$$\frac{\partial \phi}{\partial t}(\mathbf{x}, t) + \min \left[0, \mathbf{H}(\mathbf{x}, \frac{\partial \phi(\mathbf{x}, t)}{\partial \mathbf{x}}) \right] \leq 0, \quad (5)$$

with Hamiltonian given by

$$\mathbf{H}(\mathbf{x}, p) = - \max_{u_e \in [\underline{u}_e, \bar{u}_e]} \min_{u_p \in [\underline{u}_p, \bar{u}_p]} \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix} \begin{bmatrix} a_p \cos \theta + u_e x \\ a_p \sin \theta + a_e + u_p x - g \\ u_p - u_e \end{bmatrix} \quad (6)$$

Here, the co-states p is defined with a strict corresponding property, and $[\underline{u}_e, \bar{u}_e]$ denotes the extremals that the evader must choose as input in response to the extremal controls that the pursuer plays i.e. $[\underline{u}_p, \bar{u}_p]$.

We must consider the possibilities of behavior by either agent in an all-encompassing fashion in order to know what an outcome may be in the future should either agent execute different controls. Rather than resort to analytical *geometric reasoning*, we may analyze this *game* via a principled numerical simulation. This is what we present next. From (6), set $\underline{u}_e = \underline{u} \triangleq -1$ and $\bar{u}_p = \bar{u} \triangleq +1$ so that

$$\begin{aligned} \mathbf{H}(\mathbf{x}, p) &= - \max_{u_e \in [\underline{u}_e, \bar{u}_e]} \min_{u_p \in [\underline{u}_p, \bar{u}_p]} \left[p_1(a_p \cos \theta + u_e x) + p_2(a_p \sin \theta + a_e + u_p x - g) + p_3(u_p - u_e) \right], \\ &= -a_p p_1 \cos \theta - a_p p_2 \sin \theta - a_p p_2 + g p_2 - \max_{u_e \in [\underline{u}_e, \bar{u}_e]} \min_{u_p \in [\underline{u}_p, \bar{u}_p]} (p_1 u_e + p_2 u_p x + p_3(u_p - u_e)), \\ &= -a_p p_1 \cos \theta - a_p p_2 \sin \theta - a_p p_2 + g p_2 - \bar{u} |p_1 x + p_3| + \underline{u} |p_2 x + p_3|, \\ &\triangleq -a_p p_1 \cos \theta - p_2(g - a - a \sin \theta) - \bar{u} |p_1 x + p_3| + \underline{u} |p_2 x + p_3|, \end{aligned} \quad (7)$$

where the last line in (7) follows from setting $a_e = a_p \triangleq a$.

For the target set being guarded by E , we choose an implicit representation with a cylindrical mesh on a three-dimensional grid as our representation. The grid's nodes are uniformly spaced apart at a resolution of 100 points per dimension over the interval $[-64, 64]$. In numerically solving for the Hamiltonian (7), a TVD-RK discretization scheme [42] based on fluxes is used in choosing

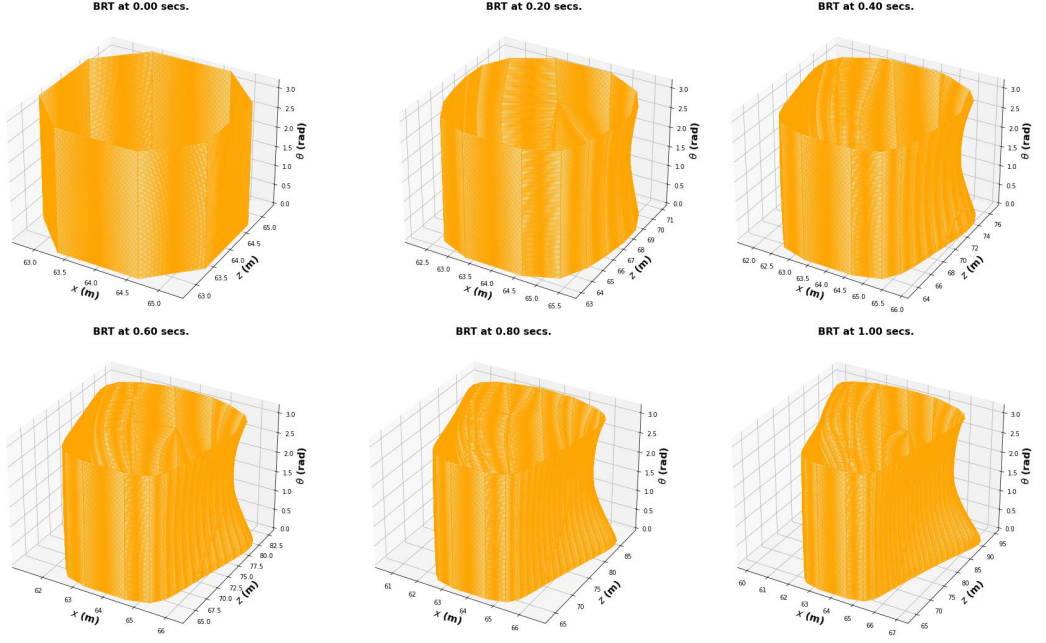


Fig. 2. (Left to Right): Backward reachable tubes (capture surfaces) for the rocket system (cf. Fig. 1) optimized for the paths of slowest-quickest descent in equation (6) at various time steps during the differential game. In all, the BRTs were computed using the method outlined in [9, 36, 39]. We set $a_e = a_p = 1 \text{ ft/sec}^2$ and $g = 32 \text{ ft/sec}^2$ as in Dreyfus' original example.

smooth nonoscillatory results. Denote by (x, y, z) a generic point in \mathbb{R}^3 so that given mesh sizes $\Delta x, \Delta y, \Delta z, \Delta t > 0$, letters u, v, w represent functions on the x, y, z lattice: $\Delta = \{(x_i, y_j, z_k) : i, j, k \in \mathbb{Z}\}$. We define the numerical monotone flux $\hat{H}(x, p)$, of $H(x, p)$ as

$$\begin{aligned} \hat{H}(u^+, u^-, v^+, v^-, w^+, w^-) &= H\left(\frac{u^+ + u^-}{2}, \frac{v^+ + v^-}{2}, \frac{w^+ + w^-}{2}\right), \\ &= -\frac{1}{2} \left[\alpha_x^{(i)j} (u^+ - u^-) + \alpha_y^{(i)j} (v^+ - v^-) + \alpha_z^{(i)j} (w^+ - w^-) \right], \end{aligned} \quad (8)$$

where

$$\alpha_x = \max_{\substack{a \leq u \leq b \\ c \leq v \leq d \\ e \leq w \leq f}} |H_u(\cdot)|, \quad \alpha_y = \max_{\substack{a \leq u \leq b \\ c \leq v \leq d \\ e \leq w \leq f}} |H_v(\cdot)|, \quad \text{and} \quad \alpha_z = \max_{\substack{a \leq u \leq b \\ c \leq v \leq d \\ e \leq w \leq f}} |H_w(\cdot)| \quad (9)$$

are the dissipation coefficients, controlling the level of numerical viscosity in order to realize a stable solution that is physically realistic [11]. Here, the subscripts of H are the partial derivatives w.r.t the subscript variable, and the flux, $\hat{H}(\cdot)$ is monotone for $a \leq u^\pm \leq b, c \leq v^\pm \leq d, e \leq w^\pm \leq f$. It is easy to verify from (7) that

$$\alpha_x = |a \cos \theta| + u|x|, \quad \alpha_y = |g - a - a \sin \theta| + \underline{u}|x|, \quad \text{and} \quad \alpha_z = |\bar{u}| + \underline{u}. \quad (10)$$

```

1 finite_diff_data = {"innerFunc": termLaxFriedrichs,
2   "innerData": {"grid": g, "hamFunc": rocket_rel.ham,
3   "partialFunc": rocket_rel.dissipation,
4   "dissFunc": artificialDissipationGLF,
5   "CoStateCalc": upwindFirstENO2},
6   "positive": True} // direction of approx. growth

```

Listing 1. HJ ENO2 computational scheme for the rockets.

The Hamiltonian, upwinding scheme, flux dissipation method, and the overapproximation parameter for the essentially non-oscillatory polynomial interpolatory data used in geometrically reasoning about the *target set* is set up as seen in Listing 1. The data structure `finite_diff_data` contains all the routines needed for adding dynamics to the original implicit surface representation of $v(x, t)$. The monotone spatial upwinding scheme used (here `termLaxFriedrichs`) is passed into the `innerFunc` query field. The explicit form of the Hamiltonian (see (7)) is passed to the `hamFunc` query field and the grid described in the foregoing is passed to the `grid` field. We adopted a second-order accurate upwinding scheme together with the Lax–Friedrichs approximator. To indicate that we intend to overapproximate the value function, we specify a `True` parameter for the `positive` query field.

Safety is engendered by having the evader respond optimally to the pursuer at various times during the game. We are thus interested in the entire safety set over the time interval of play (i.e. the safety tube). The backward reachable tube (BRT) [37], under the control strategies of P or E , is a part of the phase space that constitutes $\Omega \times T$. We would like the BRT to cover as much of the entire phase space as possible. Thus, we *overapproximate* it. Using our GPU-accelerated levelset toolbox, we compute the *overapproximated* BRT of the game over a time span of $[-2.5, 0]$ seconds over 11 global optimization time steps. The BRTs at representative time steps in the optimization procedure is depicted in Fig. 2.

The initial value function (leftmost inset of Fig. 2) is represented as a (closed) dynamic implicit surface over all point sets in the state space (using a signed distance function) for a coordinate-aligned cylinder whose vertical axes runs parallel to the orientation of the rockets depicted in Fig. 1. This closed and bounded assumption of the target set is a prerequisite of the backward reachable analysis (see [37]). It allows us to include all limiting velocities. The two middle capture surfaces indicate the evolution of the capture surface (here the zero levelset) of the target set upon the optimal response of the evader to the pursuer. We reach convergence at the eleventh global optimization timestep (rightmost inset of Fig. 2).

Reachability theory thus affords us an ability to numerically reason about the behavior of these two rockets aforesaid in a principled manner. To do this, we have passed relevant parameters to the package as shown in Listing 1 and run a CFL constrained optimization scheme for a finite number of global optimization timesteps. It is global because internally, there are other local spatial and temporal finite differencing scheme that occurs “under the hood”.

1.2 Time Optimal Control: The Double Integral Plant

Here, we analyze a time-optimal control problem to determine what admissible control³ can “transport” the system under consideration to a desired “origin” in the shortest possible time. We consider the double integral plant [5, 46] as an illustrative example of our objective, which is to compute the points in the state space that can reach the origin in *finite-time* under the influence of a time-optimal controller.

We shall leverage standard necessary conditions from the principle of optimality [4] to obtain a time-optimal feedback control design; introduce the notion of isochrones and switching surfaces;

³A control law is admissible when its range belongs in the admissible input set where it is bounded.

and discuss the analytic and approximate solutions (with our library) to the time-optimal control problem for a double integrator. We shall conclude the section by comparing the analytic and the overapproximated numerical solution (using the LevelSetPy toolbox) to the time to reach the origin problem.

1.2.1 Dynamics and Problem Setup. The double integrator is controllable, so that open-loop strategies may be employed in driving specific states to the origin in finite time [46]. The plant has the following second-order dynamics

$$\ddot{\mathbf{x}}(t) = \mathbf{u}(t) \quad (11)$$

and admits bounded control signals $|\mathbf{u}(t)| \leq 1$ for all time t . After a change of variables, we have the following system of first-order differential equations

$$\begin{aligned} \dot{\mathbf{x}}_1(t) &= \mathbf{x}_2(t), \\ \dot{\mathbf{x}}_2(t) &= \mathbf{u}(t), \quad |\mathbf{u}(t)| \leq 1. \end{aligned} \quad (12)$$

The *reachability problem* that we consider is to address the question of what states can reach a certain point (here, the origin) in a transient manner. That is, we would like to find point sets on the state space, at a particular time step, such that we can bring the system to the equilibrium, $(0, 0)$.

1.2.2 Time-optimal control scheme. This is an H -minimal control problem whereupon we must find the control law that minimizes the Hamiltonian

$$H(\mathbf{x}, p) = p_1 \dot{\mathbf{x}}_1 + p_2 \dot{\mathbf{x}}_2. \quad (13)$$

The necessary optimality condition stipulates that the minimizing control law be

$$\mathbf{u}(t) = -\text{sign}(p_2(t)) \triangleq \pm 1. \quad (14)$$

For the co-states in question, suppose that their initial values (for constants k_1 and k_2) are $p_1(t_0) = k_1$ and $p_2(t_0) = k_2$, only four candidates can serve as time-optimal control sequences i.e. $\{[+1], [-1], [+1, -1], [-1, +1]\}$. On a finite time interval, $t \in [t_0, t_f]$, the time-optimal $\mathbf{u}(t)$ is a constant $k \equiv \pm 1$ so that for initial conditions $\mathbf{x}_1(t_0) = \xi_1$ and $\mathbf{x}_2(t_0) = \xi_2$, it can be verified that the state trajectories obey the relation

$$\mathbf{x}_1(t) = \xi_1 + \frac{1}{2}k(\mathbf{x}_2^2 - \xi_2^2), \text{ for } t = k(\mathbf{x}_2(t) - \xi_2). \quad (15)$$

The trajectories of (15) traced out over a finite time horizon $t = [-1, 1]$ with *piecewise constant control laws*, $u = \pm 1$ on a state space and under the control laws $\mathbf{u}(t) = \pm 1$ is depicted in Fig. 3. Curves with arrows that point upwards denote trajectories under the control law $\mathbf{u} = +1$; call these trajectories γ_+ ; while the trajectories marked by dashed arrows pointing downward on the curves were executed under $\mathbf{u} = -1$; call these trajectories γ_- .

1.2.3 Analytic Time to Reach the Origin. The *time to reach the origin* $(0, 0)$ from any other pair (x_1, x_2) on the state plane of Fig. 3 in the *shortest possible time* is our approximation problem. This minimum time admits an analytical solution given by [1]

$$t^*(x_1, x_2) = \begin{cases} x_2 + \sqrt{4x_1 + 2x_2^2} & \text{if } x_1 > \frac{1}{2}x_2|x_2| \\ -x_2 + \sqrt{-4x_1 + 2x_2^2} & \text{if } x_1 < -\frac{1}{2}x_2|x_2| \\ |x_2| & \text{if } x_1 = \frac{1}{2}x_2|x_2|. \end{cases} \quad (16)$$

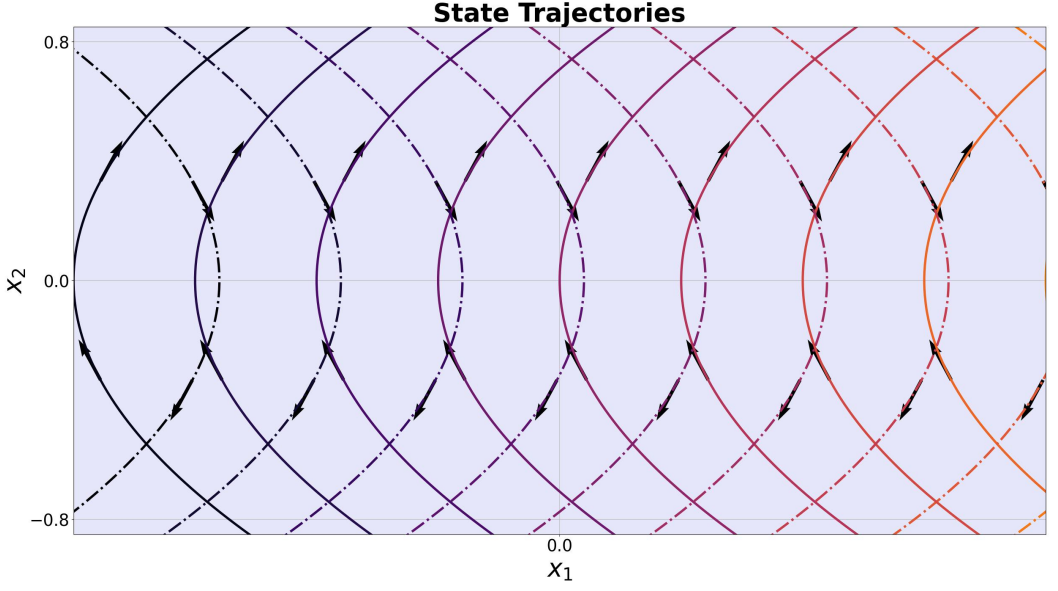


Fig. 3. State trajectories of the double integral plant. The solid curves are trajectories generated for $u = +1$ while the dashed curves are trajectories for $u = -1$.

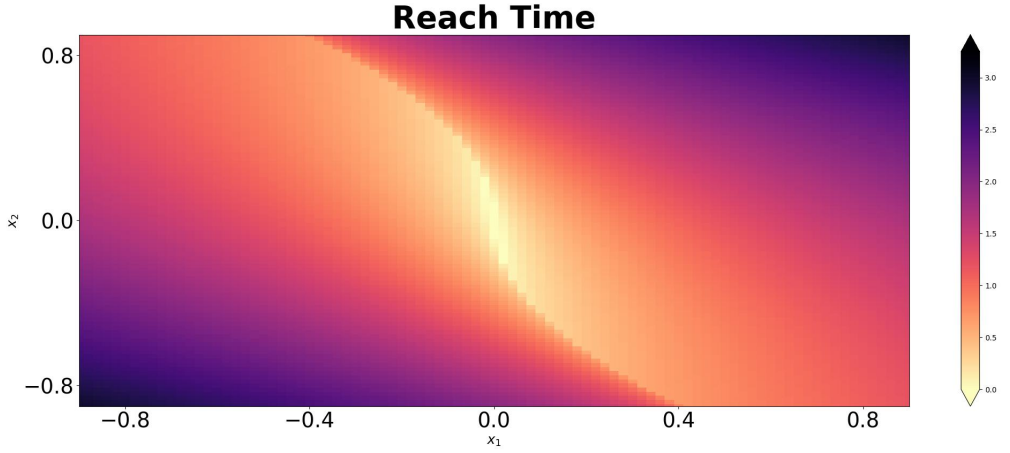


Fig. 4. Analytical time to reach the origin on the state grid, $(\mathbb{R} \times \mathbb{R})$; the switching curve, $\gamma = \gamma_- \cup \gamma_+$, passes through states on $(0, 0)$.

The geometry (phase portrait) of (16) is shown in Fig. 4. Let us define γ_+ as the locus of all points (x_1, x_2) which can be forced to the origin by $u = +1$ i.e.

$$\gamma_+ = \{(x_1, x_2) : x_1 = \frac{1}{2}x_2^2; x_2 \leq 0\} \quad (17)$$

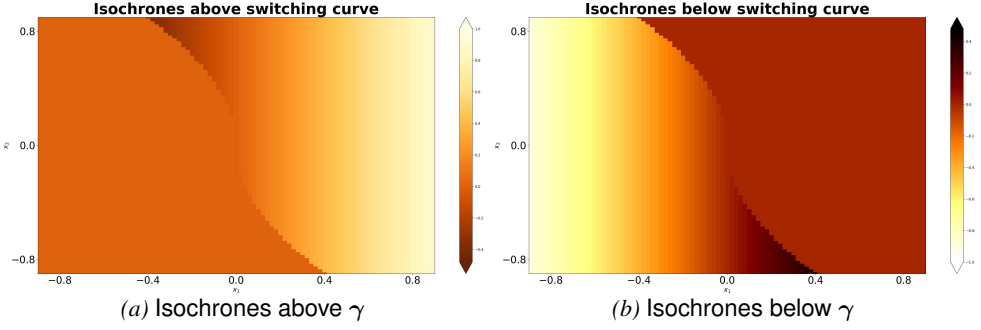


Fig. 5. (a) Isochrones for states above the switching curve, (b) states below the switching curve.

and let γ_- be the locus of all points (x_1, x_2) which can be forced to the origin by $u = -1$ i.e.

$$\gamma_- = \{(x_1, x_2) : x_1 = \frac{1}{2}x_2^2; x_2 \geq 0\}. \quad (18)$$

The confluence of the locus of points on γ_+ and γ_- is the *switching curve*, depicted in bright orange in Fig. 4, is

$$\gamma \triangleq \gamma_+ \cup \gamma_- = \left\{ (x_1, x_2) : x_1 = \frac{1}{2}x_2|x_2| \right\}. \quad (19)$$

The *unique* time-optimal control law, u^* , that solves this problem can be determined to be

$$\begin{aligned} u^* &= u^*(x_1, x_2) = +1 \quad \forall (x_1, x_2) \in \gamma_+ \cup \mathbb{R}_+, \\ u^* &= u^*(x_1, x_2) = -1 \quad \forall (x_1, x_2) \in \gamma_- \cup \mathbb{R}_-, \\ u^* &= -\text{sgn}\{x_2\} \quad \forall (x_1, x_2) \in \gamma. \end{aligned} \quad (20)$$

The minimum cost for this problem is equivalent to the minimum time for states (x_1, x_2) to reach the origin $(0, 0)$. This is given as

$$\Phi^*(x, t) \triangleq t^*(x_1, x_2) \quad (21)$$

with the associated terminal value

$$-\frac{\partial \Phi^*(x, t)}{\partial t} = H\left(t, x, \frac{\partial \Phi^*(x, t)}{\partial t}, u\right) \Bigg|_{\substack{x=x^* \\ u=u^*}} \quad \text{with } H(t; x, u, p_1, p_2) = x_2(t)p_1(t) + u(t)p_2(t) \quad (22)$$

and

$$p_1 = \frac{\partial t^*}{\partial x_1}, \quad p_2 = \frac{\partial t^*}{\partial x_2}. \quad (23)$$

The HJ equation is given by

$$\begin{aligned} \frac{\partial \Phi^*}{\partial t} + x_2 \frac{\partial \Phi^*}{\partial x_1} - \frac{\partial \Phi^*}{\partial x_2} &= 0 & \text{if } x_1 > -\frac{1}{2}x_2|x_2|, \\ \frac{\partial \Phi^*}{\partial t} + x_2 \frac{\partial \Phi^*}{\partial x_1} + \frac{\partial \Phi^*}{\partial x_2} &= 0 & \text{if } x_1 < -\frac{1}{2}x_2|x_2|, \\ \frac{\partial \Phi^*}{\partial t} + x_2 \frac{\partial \Phi^*}{\partial x_1} - \text{sgn}\{x_2\} \frac{\partial \Phi^*}{\partial x_2} &= 0 & \text{if } x_1 = -\frac{1}{2}x_2|x_2|. \end{aligned} \quad (24)$$

The set of states (x_1, x_2) that can be forced to reach the origin in the same minimum time $t^* \equiv \Phi^*$ are the system's *isochrones* which are illustrated in Fig. 5. A point (x_1, x_2) on the state grid belongs to the set of states $S(t^*)$ from which it can be forced to the origin $(0, 0)$ in the same minimum time t^* . We call the set $S(t^*)$ the minimum isochrone. These are the isochrones of the system – akin to the isochrone map used in geography, hydrology, and transportation planning for depicting areas of equal travel time to a goal state. The level sets of (24) correspond to the *isochrones* of the system as illustrated in Fig. 5.

1.2.4 Approximate Time to Reach the Origin. We compare the analytical solution to the *time to reach (TTR) the origin* problem (see Fig. 5) against the approximated TTR solution using a dynamic implicit surface representation of the approximate value function. An ellipsoid with a radius of 1.0 along its major axis was chosen to represent the initial time to reach interface (see Fig. 6a, right inset). We then choose a controller with values ± 1 depending on which side of the switch surface Fig. 5 we are on in generating the system's phase portrait illustrated in Fig. 3.

The closed-form solution to the time-to-reach the origin problem on a 2-D grid with x/y axis limits $[-1, 1], [-1, 1]$ is shown in the left inset of Fig. 3a. We set out to investigate the result of adding dynamics (with levelsets) to the elliptic implicit representation of the analytical TTR and evaluate the efficacy of our computational scheme. We proceed as shown in Listing 3.

```
1 finite_diff_data = {"innerFunc": termLaxFriedrichs,
2   "innerData": {"grid": g, "hamFunc": dint.ham,
3   "partialFunc": dint.dissipation,
4   "dissFunc": artificialDissipationGLF,
5   "CoStateCalc": upwindFirstENO2},
6   "positive": False} // direction of approx. growth
```

Listing 2. Overapproximation setup for the double integrator TTR problem.

As a custom, a separate class (see `DoubleIntegrator` in the folder `DynamicalSystems`) holds all the dynamics (cref. equations 11 and 12), switching surface (cref. equations 17, 18, and 19), Hamiltonian (cref. equation 13), dissipation, and costates (cref. 23) of the double integrator plant. Over a twenty-step timespan ranging from 0 to 20, we integrate the right-hand-side of (24) forward in time by the Courant-Friedrichs-Lewy constrained second-order accurate integrator i.e. `odeCFL2` in our library:

```
1 t, y, ~ = odeCFL2(termRestrictUpdate, t_span, y0, options, finite_diff_data)
```

Listing 3. Overapproximation setup for the double integrator TTR problem.

where `y0` is the initial elliptic function that represents Φ in (24), `options` are the set of integration parameters such as the number of actual timesteps to take in the adaptive integration scheme, the maximum step size and so on. The routine `termRestrictUpdate` restricts the sign of the update of the HJ approximation by either increasing or decreasing the levelset.

A side-by-side comparison of the level sets is shown in Fig. 6a. The approximation to the isochrones by our integration scheme is an overapproximation of the analytical TTR problem. This is illustrated in the right inset of Fig. 6a. Because we are not concerned with the safe set (unlike the example in 1.1), we do not overapproximate the time-to-reach solution. On the overall, we obtain similar isochrones to the analytical result, hence validating our hypothesis.

1.3 Reach-Avoid Games: Flocks within Starling Murmurations

Here, we will borrow inspiration from natural swarms, particularly the murmuration [21] of European starlings – the *sturnus vulgaris* – in our problem construction and solution concept. We are concerned with reach-avoid games in multiagent systems, whereupon agents must safely navigate a phase space (e.g. in achieving an attitude convergence goal), whilst avoiding collision with one another

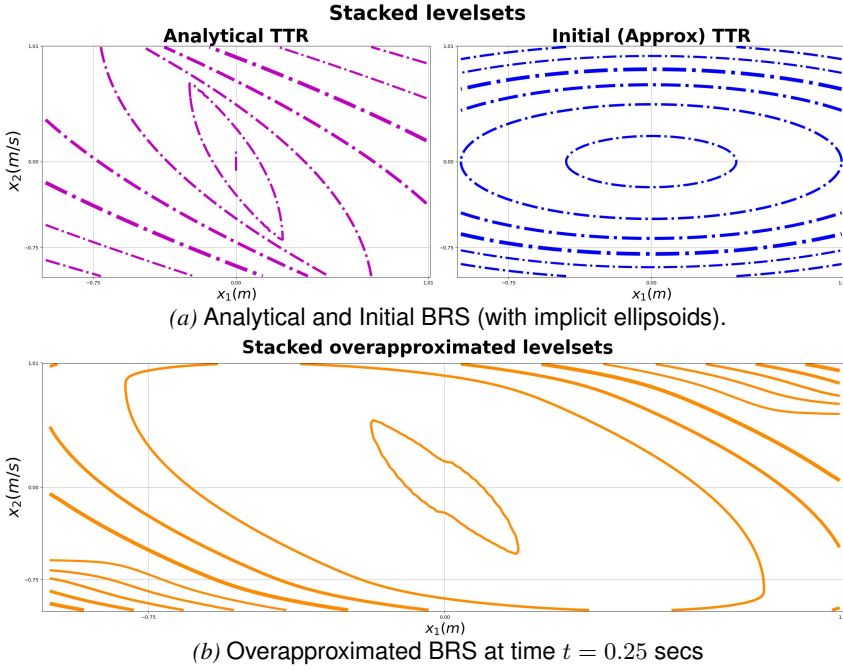


Fig. 6. Time to reach the origin at different integration time steps. Top-left: Closed-form Solution to the time to reach the origin problem. Top-right: Implicit representation of the initial TTR solution. Bottom: Lax-Friedrichs Approximation to the TTR the origin problem.

and capture by an external predator. A natural environment where this problem occurs is in the murmuration of European Starlings. The problem that we study is of importance in multiagent systems such as the safe control of quadcopters, safe interaction among distributed agents on a computing network where local nearest neighbor rules apply. In what follows, we formulate the problem mathematically and pose the collision avoidance for a local set of birds as a reachable differential game.

1.3.1 Problem Description. Consider a group of starlings moving on a space-time continuum $\Omega \times T^4$ as illustrated in Fig. 7. Recent field studies [8] suggest that emergent collective motion observed among these birds is as a result of local nearest neighbor interactions among separable subsets of bird groups on $\Omega \times T$. There is evidence with justifiable confidence [27] suggesting that when density varies among the birds (henceforth called agents), the relationship among agents in local groups is not determined by the metric distance among nearest neighbors but rather by a topological notion of distance (defined as the number of intermediate birds between one agent from another [2]).

Starlings exhibit complex formation patterns that are effective in avoiding capture – mostly by peregrine Falcons in the wild [20]. We will leverage this notion of topological distance between agents in developing a target or safe set [24] for a subset of agents within the murmuration. If we can compute this safe set, it can serve as e.g. a safety filter for multiagent systems (to be controlled) and their actual controller whilst respecting state constraints [24] and the control constraints (admissible inputs, dynamic update frequency of input control laws e.t.c.). An illustration of the problem

⁴Here Ω is the open set that contains all states of the birds and T is the length of time over the real line.

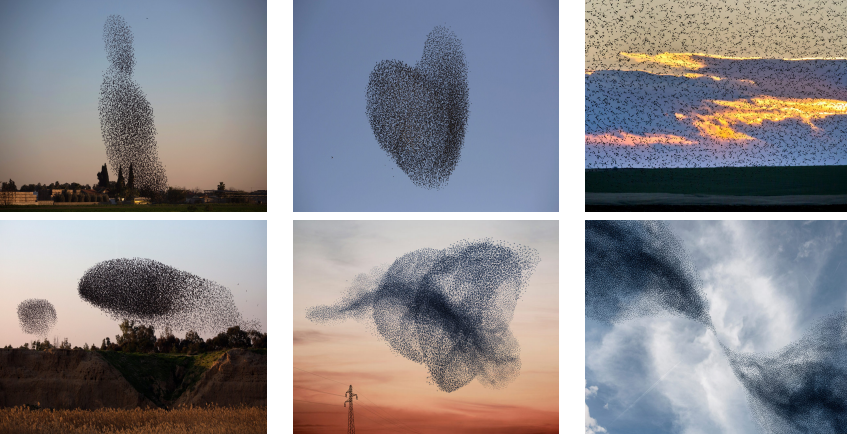


Fig. 7. Starlings murmurations. From the top-left and clockwise. (i) A starlings flock rises into the air, in a dense structure (Reuters/Amir Cohen). (ii) Starlings migrating over an Israeli village (AP Photo/Oded Balilty). (iii) Starlings feeding on laid seeds in the ground in Romania. (iv) Two flocks of migrating starlings (Menahem Kahana/AFP/Getty Images). (v) A concentric conical formation of starlings (Courtesy of The Gathering Site.). (vi) Splitting and joining of a flock of starlings.

setup is illustrated in Fig. 7. For a comprehensive understanding of the intuition that guides our mathematical formulation here, we refer agents to the works of Ballerini et al. [2], Cavagna et al. [8], and Cardaliaguet [7].

1.3.2 Mathematical abstractions. Individual agents self-organize into phases or regions \mathcal{S} which are in turn members of a union of multiple regions \mathcal{C} . Every $\mathcal{S} \subseteq \mathcal{C}$ and all members of \mathcal{C} are disjoint from one another i.e. $\mathcal{S}_i \cup \mathcal{S}_j = \emptyset$ for any $i \neq j$. The total number of elements in \mathcal{S} is denoted $[\mathcal{S}]$, and we denote by $\text{int } \Omega$ the interior of Ω . The closure of Ω is $\bar{\Omega}$. We let $\delta\Omega := \bar{\Omega} \setminus \text{int } \Omega$ be the boundary of Ω .

An evading agent in a region \mathcal{S}_i has a state notation \mathbf{x}_a^i (read: the state of agent a in region i). A state \mathbf{x}_a^i has linear velocity components, $\mathbf{x}_{a1}^i, \mathbf{x}_{a2}^i$, and heading $\mathbf{x}_{a3}^i := w_a^i$. When we must distinguish an agent $\mathbf{x}_a^i \in \mathcal{C}_x$ from some other agent e.g. in another multiphase \mathcal{C}_y , we shall write $^x\mathbf{x}_a^i$ and $^y\mathbf{x}_a^i$ respectively.

The set of players in a game shall be denoted by $\mathcal{N} = \{i, j, \dots\}$ with the subscript index indicating players e.g. \mathcal{N}_i for player i . The set of neighbors of player i is $\mathcal{N}(i) \subseteq \mathcal{N}$. Player i moves dynamically with a control $u_i \in \pi_i$ (i.e. u_i belongs to a policy class π_i) that is both (a) optimal with respect to its own objective \mathcal{J}_i ; and (b) optimal with respect to its neighboring players' current policy $\pi_{-i} \in \prod_{j \in \mathcal{N}, j \neq i} \pi_j$. Neighbors of agent i at time t are those which are either within, or on a circle specified by a fixed topological range, r_c . This topological range is given by the difference in the numerical label of individuals (see Definition 3), and is consistent with findings in collective swarm behaviors as it reinforces *group cohesion* [2].

The topological metric is given by the label of an agent and it quantifies the number of intermediate agents that separate two agents. This is consistent with collective animal behaviors where individuals' bookkeeping on their neighbors' positions help maintain the strength of an interaction when density varies or when they need to reorient a control input, given by the average of its own orientation and

that of its neighbors. Instead of metric distance interaction rules that make agents very vulnerable to predation [2], we resort to a topological interaction rule⁵. Let us set forth with a few definitions first.

DEFINITION 1. *Every agent within a flock has similar dynamics to that of its neighbor(s). Furthermore, agents travel at the same linear speed, v ; the angular headings, w , however, may be different between agents, seeing we are dealing with a many-bodied system. Each agent's continuous-time dynamics, $\dot{\mathbf{x}}^{(i)}(t)$, evolves as*

$$\begin{bmatrix} \dot{\mathbf{x}}_1^{(i)}(t) \\ \dot{\mathbf{x}}_2^{(i)}(t) \\ \dot{\mathbf{x}}_3^{(i)}(t) \end{bmatrix} = \begin{bmatrix} v(t) \cos \mathbf{x}_3^{(i)}(t) \\ v(t) \sin \mathbf{x}_3^{(i)}(t) \\ \langle w^{(i)}(t) \rangle_r \end{bmatrix}, \langle w^{(i)}(t) \rangle_r = \frac{1}{1 + n_i(t)} \left(w^{(i)}(t) + \sum_{j \in \mathcal{N}_i(t)} w_j(t) \right) \quad (25)$$

for agents $i = \{1, 2, 3, \dots, n_a\}$, where t is the continuous-time index, $n_i(t)$ is the number of agent i 's neighbors at time t , $\mathcal{N}_i(t)$ denotes the sets of labels of agent i 's neighbors at time t , and $\langle w^{(i)}(t) \rangle_r$ is the average orientation of agent i and its neighbors at time t . Note that for a game where all agents share the same constant linear speed and heading, (25) reduces to the dynamics of a Dubins' vehicle in absolute coordinates with $-\pi \leq w^{(i)}(t) < \pi$. The averaging over the degrees of freedom of other agents in (25) is consistent with the mean field theory, whereby the effect of all other agents on any one agent is an approximation of a single averaged influence.

DEFINITION 2 (NEIGHBORS OF AN AGENT). *We define the neighbors $\mathcal{N}_i(t)$ of agent i at time t as the set of all agents that lie within a predefined radius, r_i .*

DEFINITION 3. *We define a flock, F , consisting of agents labeled $\{1, 2, \dots, n_a\}$ as a collection of agents within a phase space $(\Omega \times T)$ such that all agents within the flock interact with their nearest neighbors in a topological sense.*

REMARK 1. *Note that for a game where all agents share the same constant linear speed and heading, (25) reduces to the dynamics of a Dubins' vehicle in absolute coordinates with $-\pi \leq w^{(i)}(t) < \pi$. The averaging over the degrees of freedom of other agents in (25) is consistent with mean field theory, whereby the effect of all other agents on any one agent is an approximation of a single averaged influence.*

DEFINITION 4 (PAYOFF OF A FLOCK). *To every flock F_j (with a finite number of agents n_a) within a murmuration, $j = \{1, 2, \dots, n_f\}$, we associate a payoff, Φ_j , that is the union of all respective agent's payoffs for expressing the outcome of a desired kinematic behavior.*

1.3.3 Flock Motion as Differential Games. We restrict our analysis to a single local flock within a murmuration. We must find a mathematical way to replicate the collision-avoidance scheme that agents execute structural homogeneity of movement in every region $\mathcal{S}_i \in \mathcal{C}$ for $i = 1, \dots, |\mathcal{C}|$ as observed in natural systems. We will locally synthesize the kinematics of agents in a manner amenable to state representation by resolving local payoff extremals, $\{\phi_1, \dots, \phi_{n_f}\}$. This is a state space partition induced by an aggregation of desired collective behavior from local flocks' values $\{\mathbf{v}_1, \dots, \mathbf{v}_{n_f}\}$. Let the cursory reader understand that we use the concept of a flock loosely. The value function could represent a palette of composed value functions whose extremals resolve local behaviors we would like to emerge over separated local regions of the state space of dexterous drone acrobatics [29], a robot balls juggling task [6], or any parallel task domain verification problem.

⁵With metric distance rules, we will have to formulate the breaking apart of value functions that encode a consensus heading problem in order to resolve the extrema of multiple payoffs; which is typically what we want to mitigate against during real-world autonomous tasks.

1.3.4 Framework for Separated Payoffs. Suppose that a murmuration's global heading is given and that each agent i within each flock, F_j , ($j = \{1, \dots, n_f\}$) in the murmuration has a constant linear velocity, v^i . An agent's orientation is its control input, given by the average of its own orientation and that of its neighbors. What constitutes an agent's neighbors are computed based on empirical findings and studies from the lateral vision of birds and fishes [2, 23, 27] that provide insights into their anisotropic kinematic density and structure. Importantly, starlings' lateral visual axes and their lack of a rear sector reinforces their lack of nearest neighbors in the front-rear direction. As such, this enables them to maintain a tight density and robust heading during formation and flight.

Each agent within flock F_j interacts with a fixed number of neighbors, n_c , within a fixed topological range, r_c . The range is chosen as the difference between the numerical labels of agents in a flock. This is consistent with findings in collective swarm behaviors in that it reinforces *group cohesion* [2]. Since in starlings behavior, flying performance is often spurred by a predator, we emulate this by introducing an external disturbance on the zeroth-index agent within the flock of interest (this aids compactness of the zero levelset of flock \mathcal{S}_i as the theory of HJ Reachability recommends [37]). However, we are interested in *robust group cohesion*. Ergo, we let a pursuer, \mathbf{P} , with a worst-possible disturbance attack the flock. Here, flocks constitute an evading player, \mathbf{E} .

The delineation of an agent's nearest neighbors is given in Algorithm 1. On lines 3 and 7 of Algorithm 1, cohesion is reinforced by leveraging the observations above. While the neighbor updates for an agent involve an $O(n^2)$ algorithm in Algorithm 1, we are merely dealing with 6 – 7 agents at a time in a local flock – making the computational cost measly.

1.3.5 Global Isotropy via Local Anisotropy. Isotropy of motion fields is a natural characteristic in Starlings motion. The global isotropy of murmurations where group cohesion is maintained in highly uncertain environments with limited or noisy information is often stimulated by Peregrine Falcon attacks. Local birds maintain structural anisotropy via nearest neighbor rules, and a collection of multiple local groups in the entire collection results in the global isotropy that is observed. However, structural anisotropy is not merely an effect of a preferential velocity in animal flocking kinematics but rather an explicit effect of the anisotropic interaction character itself: agents choose a mutual position on the state space in order to maximize the sensitivity to changes in heading and speed of neighbors; the neighbors' anisotropy is optimized via vision-based collision avoidance characteristically unrelated to the eye's structure [2].

To reinforce robust group cohesion in local flocks, we let a pursuer \mathbf{P}_j play attack an evading agent \mathbf{E}_j in a flock F_j so that one agent within F_j is always in relative coordinates with \mathbf{P}^j . By averaging the heading of individual agents' orientations with its neighbors (cf. (25)), a flock can achieve fast response to danger when a pursuer is nearby. In this specialized case, \mathbf{E} and \mathbf{P} 's speeds and maximum turn radii are equal: if both players start the game with the same initial velocity and orientation, the relative equations of motion show that \mathbf{E} can mimic \mathbf{P} 's strategy by forever keeping the starting radial separation. As such, the *barrier* is closed and *the central theme in this game of kind is to determine the surface of the boundary* [33]. We defer a thorough analysis of the nature of the surface to a future work. Owing to the high-dimensionality of the state space, we cannot resolve this barrier analytically, hence we resort to our HJ PDE numerical approximation.

For agent i within a flock with index j in a murmuration, the equations of motion under attack from a predator p in relative coordinates is

$$\begin{bmatrix} \dot{\mathbf{x}}_1^{(i)j}(t) \\ \dot{\mathbf{x}}_2^{(i)j}(t) \\ \dot{\mathbf{x}}_3^{(i)j}(t) \end{bmatrix} = \begin{bmatrix} -v_e^{(i)j}(t) + v_p^{(j)} \cos \mathbf{x}_3^{(i)j}(t) + \langle w_e^{(i)j} \rangle_r \mathbf{x}_2^{(i)j}(t) \\ v_p^{(i)j}(t) \sin \mathbf{x}_3^{(i)j}(t) - \langle w_e^{(i)j} \rangle_r \mathbf{x}_1^{(i)j}(t) \\ w_p^{(j)}(t) - \langle w_e^{(i)j} \rangle_r \end{bmatrix} \quad \text{for } i = 1, \dots, n_a \quad (26)$$

Algorithm 1 Nearest Neighbors For Agents in a Flock.

```

1: Given a set of agents  $\mathbf{a} = \{a_1, a_2, \dots, a_{n_a} \mid [a] = n_a\}$   $\triangleright n_a$  agents in a flock  $F_k$ .
2: function UpdateNeighbor( $n$ )
3:   for  $i$  in  $1, \dots, n$  do  $\triangleright$  Look to the right and update neighbors.
4:     for  $j$  in  $i + 1, \dots, n$  do
5:       COMPARE_NEIGHBOR( $a[i], a[j]$ )
6:     end for
7:     for  $j$  in  $i - 1$  down to  $0$  do  $\triangleright$  Look to the left and update neighbors.
8:       COMPARE_NEIGHBOR( $a[i], a[j]$ )
9:     end for
10:  end for
11:  for each  $a_i \in F_k, i = 1, \dots, n_a$  do  $\triangleright$  Recursively update agents' headings.
12:    Update headings according to (25).
13:  end for
14: end function


---


1: function Compare_Neighbor( $a_1, a_2$ )  $\triangleright (a_1, a_2)$ : distinct instances of AGENT.
2:   if  $|a_1.\text{label} - a_2.\text{label}| < a_1.r_c^1$   $\triangleright r_c^n$ : agent  $n$ 's capture radius,  $r_c$ .
3:      $a_1.\text{UPDATE\_NEIGHBORS}(a_2)$  then
4:       end if
5: end function


---


1: procedure Agent( $a_i, \text{Neighbors}=\{\}$ )  $\triangleright$  Neighbors: Set of neighbors of this agent.
2:    $\triangleright$  Agent  $a_i$  with attributes  $\text{label} \in \mathbb{N}$ ,  $\text{avoid}$  and  $\text{capture}$  radii,  $r_a, r_c$ .
3:   function UPDATE_NEIGHBORS( $\text{neigh}$ )
4:     if  $\text{length}(\text{neigh}) > 1$  then  $\triangleright$  Multiple neighbors.
5:       for each neighbor of  $\text{neigh}$  do
6:         UPDATE_NEIGHBORS( $\text{neighbor}$ )  $\triangleright$  Recursive updates.
7:       end for
8:     end if
9:     Add  $\text{neigh}$  to  $\text{Neighbors}$ 
10:  end function
11: end procedure

```

where n_a is the number of agents within a flock, $(\mathbf{x}_1^{(i)j}(t), \mathbf{x}_2^{(i)j}(t)) \in \mathbb{R}^2$, and we have $\mathbf{x}_3^{(i)j}(t) \in [-\pi, +\pi)$. We have multiplied the dynamics by -1 so that the extremal's resolution evolves backwards in time. Read $\mathbf{x}_1^{(i)j}(t)$: the first component of the state of agent i for flock j at time t . In absolute coordinates, the equation of motion for *free agents* is

$$\begin{bmatrix} \dot{\mathbf{x}}_1^{(i)j}(t) \\ \dot{\mathbf{x}}_2^{(i)j}(t) \\ \dot{\mathbf{x}}_3^{(i)j}(t) \end{bmatrix} = \begin{bmatrix} v_e^{(i)j}(t) \cos \mathbf{x}_3^{(i)j}(t) \\ v_e^{(i)j}(t) \sin \mathbf{x}_3^{(i)j}(t) \\ \langle w_e^{(i)j}(t) \rangle_r \end{bmatrix}. \quad (27)$$

1.3.6 Flock Motion from Aggregated Value Functions. We introduce the union operator i.e. \cup below as an aggregation symbol since the respective payoffs of each agent in a flock may be implicitly or explicitly constructed. In resolving the zero-level sets of HJ value functions, it is typical to represent the payoff's surface as the isocontour of some function (usually a signed distance

function). In these instances, we shall aggregate the payoff of agents 1 and 2, for example, as

$$\cup \{\phi_1(\mathbf{x}, t), \phi_2(\mathbf{x}, t)\} \equiv \phi_1(\mathbf{x}, t) \cup \phi_2(\mathbf{x}, t) = \min(\phi_1(\mathbf{x}, t), \phi_2(\mathbf{x}, t)). \quad (28)$$

Standard assumptions about the existence of a flock's *value* applies. And by an extension of Hamilton's principle of least action, the terminal motion of a flock coincide with the extremal of the payoff functional i.e. ,

$$\mathbf{v}(\mathbf{x}, t) = \inf_{\beta^{(1)} \in \mathcal{B}^{(1)}} \sup_{\mathbf{u}^{(1)} \in \mathcal{U}^{(1)}} \cup \left[g^{(1)}(\mathbf{x}(T)) \right] \cup \dots \cup \inf_{\beta^{(n_f)} \in \mathcal{B}^{(n_f)}} \sup_{\mathbf{u}^{(n_f)} \in \mathcal{U}^{(n_f)}} \left[g^{(n_f)}(\mathbf{x}(T)) \right]$$

where n_f is the total number of distinct flocks in a murmuration. The resolution of this equation admits a viscosity solution to the following variational terminal HJI PDE [37]

$$\cup_{j=1}^{n_f} \left[\cup_{i=1}^{n_a} \left(\frac{\partial \mathbf{v}_i}{\partial t}(\mathbf{x}, t) + \min \left[0, \mathbf{H}^{(i)}(\mathbf{x}^{(i)}, \mathbf{v}_x(\mathbf{x}, t)) \right] \right) \right] = 0. \quad (29)$$

with Hamiltonian,

$$\mathbf{H}^{(i)}(t; \mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{v}^{(i)}, p^{(i)}) = \max_{\mathbf{u}^{(i)} \in \mathcal{U}^{(i)}} \min_{\mathbf{v}^{(i)} \in \mathcal{V}^{(i)}} \langle f^{(i)}(t; \mathbf{x}, \mathbf{u}^{(i)}, \mathbf{v}^{(i)}), p^{(i)} \rangle. \quad (30)$$

In swarms' collective motion, when e.g. a Peregrine Falcon attacks, immediate nearest agents change direction almost instantaneously. And because of the interdependence of the orientations of individual agents with respect to one another, all other agents respond instantaneously. Thus, we only simulate a single attack against a flock within the murmuration to realize robust cohesion. Throughout the game, we assume that the roles of \mathbf{P} and \mathbf{E} do not change, so that when capture can occur, a necessary condition to be satisfied by the saddle-point controls of the players is the Hamiltonian, $\mathbf{H}^i(\mathbf{x}, p)$.

THEOREM 1. *For a flock, F_j , the Hamiltonian is the total energy given by a summation of the exerted energy by each agent i so that we can write the main equation or total Hamiltonian of a murmuration as*

$$\mathbf{H}(\mathbf{x}, p) = \max_{w_e^{(k)j} \in [\underline{w}_e^j, \bar{w}_e^j]} \min_{w_p^{(k)j} \in [\underline{w}_p^j, \bar{w}_p^j]} \cup_{j=1}^{n_f} \left[H_a^{(k)j}(\mathbf{x}, p) \cup \left(\cup_{i=1}^{n_a-1} H_f^{(i)j}(\mathbf{x}, p) \right) \right] \quad (31)$$

$$\begin{aligned} &\triangleq \cup_{j=1}^{n_f} \left(\cup_{i=1}^{n_a-1} \left[p_1^{(i)j} v^{(i)j} \cos \mathbf{x}_3 + p_2^{(i)j} v^{(i)j} \sin \mathbf{x}_3 + p_3^{(i)j} \langle w_e^{(i)j} \rangle_r \right] \right. \\ &\quad \cup \left[p_1^{(k)j} \left(v^{(k)j} - v^{(k)j} \cos \mathbf{x}_3^{(k)j} \right) - p_2^{(k)j} v^{(k)j} \sin \mathbf{x}_3^{(k)j} - \underline{w}_p^j |p_3^{(k)j}| \right. \\ &\quad \left. \left. + \bar{w}_e^j \left| p_2^{(k)j} \mathbf{x}_1^{(k)j} - p_1^{(k)j} \mathbf{x}_2^{(k)j} + p_3^{(k)j} \right| \right] \right). \end{aligned} \quad (32)$$

where $H_a^{(k)j}(\mathbf{x}, p)$ is the Hamiltonian of the individual under attack by a pursuing agent, \mathbf{P} , and $H_f^{(i)j}(\mathbf{x}, p)$ are the respective Hamiltonians of the free agents, $i = \{1, \dots, n_f\}$, within an evading flock, and not under the direct influence of capture or attack by \mathbf{P} . We denote by $w_e^{(i)j}$ the heading of an evader i within a flock j and $w_p^{(j)}$ the heading of a pursuer aimed at flock j ; $\underline{w}_e^{(k)j}$ is the orientation that corresponds to the orientation of the agent with minimum turn radius among all the neighbors of agent k , inclusive of agent k at time t ; similarly, $\bar{w}_e^{(k)j}$ is the maximum orientation among all of the orientation of agent k 's neighbors.

PROOF. The proof to this theorem is given in Appendix A. □

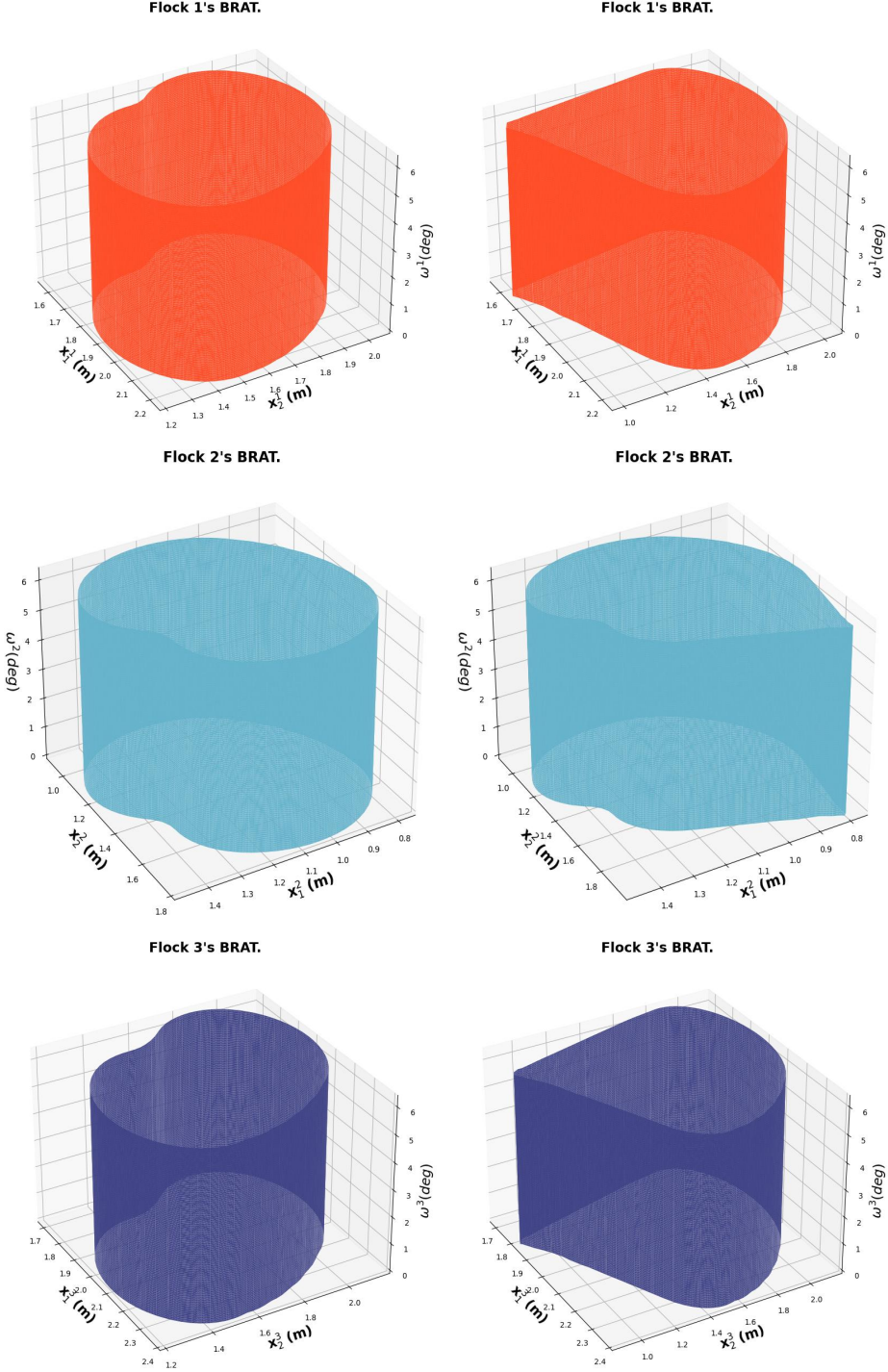


Fig. 8. Left column: Initial zero-level set for various flocks at different initial conditions. Right column: Evading flock's interface under a pursuer's attack after specific Lax-Friedrichs' integration. (Metric reach radius= $0.2m$, Avoid Radius= $0.2m$).

COROLLARY 2. *For the special case where the linear speeds of the evading agents and pursuer are equal i.e. $v_e^{(i)j}(t) = v_p(t) = +1m/s$, we have the Hamiltonian as*

$$\begin{aligned} \mathbf{H}(\mathbf{x}, p) = & \cup_{j=1}^{n_f} \left(\cup_{i=1}^{n_a-1} \left[p_1^{(i)j} \cos \mathbf{x}_3 + p_2^{(i)j} \sin \mathbf{x}_3 + p_3^{(i)j} \langle w_e^{(i)j} \rangle_r \right] \right. \\ & \cup \left[p_1^{(k)j} \left(1 - \cos \mathbf{x}_3^{(k)j} \right) - p_2^{(k)j} \sin \mathbf{x}_3^{(k)j} - \underline{w}_p^j |p_3^{(k)j}| \right. \\ & \left. \left. + \bar{w}_e^j \left[p_2^{(k)j} \mathbf{x}_1^{(k)j} - p_1^{(k)j} \mathbf{x}_2^{(k)j} + p_3^{(k)j} \right] \right] \right). \end{aligned} \quad (33)$$

We adopt the essentially non-oscillatory Lax-Friedrichs scheme of [42] in resolving (33). Denote by (x, y, z) a generic point in \mathbb{R}^3 so that given mesh sizes $\Delta x, \Delta y, \Delta z, \Delta t > 0$, letters u, v, w will represent functions on the x, y, z lattice $\Delta = \{(x_i, y_j, z_k) : i, j, k \in \mathbb{Z}\}$. We define the numerical monotone flux, $\hat{\mathbf{H}}^{(i)j}(\cdot)$, of $\mathbf{H}_j^{(i)}(\cdot)$ as

$$\begin{aligned} \hat{\mathbf{H}}^{(i)j}(u^+, u^-, v^+, v^-, w^+, w^-) = & \mathbf{H}^{(i)j} \left(\frac{u^+ + u^-}{2}, \frac{v^+ + v^-}{2}, \frac{w^+ + w^-}{2} \right) \\ & - \frac{1}{2} \left[\alpha_x^{(i)j} (u^+ - u^-) + \alpha_y^{(i)j} (v^+ - v^-) + \alpha_z^{(i)j} (w^+ - w^-) \right] \end{aligned} \quad (34)$$

where

$$\alpha_x^{(i)j} = \max_{\substack{a \leq u \leq b \\ c \leq v \leq d \\ e \leq w \leq f}} |\mathbf{H}_u^{(i)j}(\cdot)|, \alpha_y^{(i)j} = \max_{\substack{a \leq u \leq b \\ c \leq v \leq d \\ e \leq w \leq f}} |\mathbf{H}_v^{(i)j}(\cdot)|, \alpha_z^{(i)j} = \max_{\substack{a \leq u \leq b \\ c \leq v \leq d \\ e \leq w \leq f}} |\mathbf{H}_w^{(i)j}(\cdot)| \quad (35)$$

are the dissipation coefficients, controlling the level of numerical viscosity in order to realize a stable solution that is physically realistic [12]. Here, the subscripts of \mathbf{H} are the partial derivatives w.r.t the grid dimension variable, and the flux, $\hat{\mathbf{H}}(\cdot)$ is monotone for $a \leq u^\pm \leq b, c \leq v^\pm \leq d, e \leq w^\pm \leq f$. We adopt the total variation diminishing Runge-Kutta scheme of [41, 45] in efficiently calculating essentially non-oscillating upwinding finite difference gradients of $\mathbf{H}(\cdot)$.

The computed safe sets are as shown in Figure 8. Note that the symmetry between non-consecutive flock labels e.g. flock 1 and flock 3's RCBRAT is because the we multiplied the initial position of a flock's state by -1 .

1.4 Dubins' Game of Two Identical Vehicles

This example was originally proposed by Merz [33] as an iteration upon Isaacs [26]'s homicidal chauffeur game, whereupon a pursuit-evasion game between two players with similar speeds and minimum turn radii, is thoroughly analyzed. In Mitchell [36], this problem was established as a benchmark for testing the solubility of capturable set of states (the backward reachable tube) in Merz's classical pursuit-evasion game. In this example, we solve the problem with our LevelSetPy toolbox and establish that the approximated barrier surface to the two-player game conforms with standard results.

The game is that of two cars sharing similar Dubins dynamics [16]: \mathbf{P} and \mathbf{E} both have a positive minimum turn radii, w , and constant speeds v – with motion restricted to a plane as we have for the rocket launch differential game above. In relative coordinates, the diagrammatic structure of the motion is as depicted in Fig. 9. Choosing the Cartesian coordinate for motion representation, the state vector of the game with \mathbf{E} at the origin can be characterized by its x_1, x_2 position relative to \mathbf{P} and the angle θ between the two vehicles. Capture occurs when the distance $\|\mathbf{PE}\|_2$ between the pursuer and the evader becomes less than a specified radius.

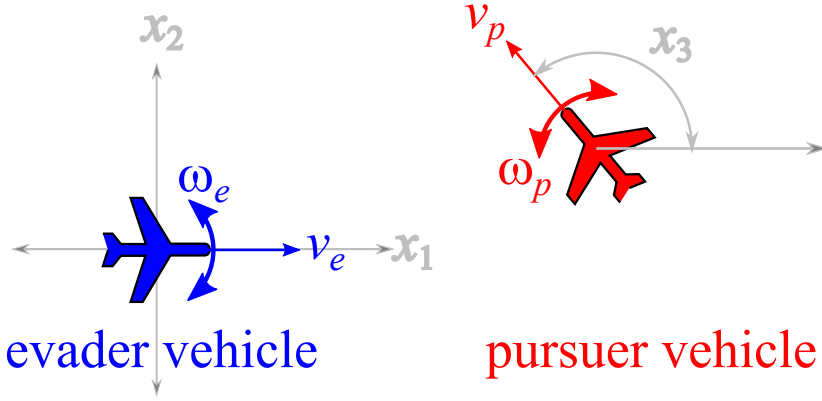


Fig. 9. Two Dubins' vehicles in relative Cartesian coordinates. Reprinted from Mitchell [36].

The relative equations of motion, going by Fig. 9, is

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -v_e + v_p \cos x_3 + w_e x_2 \\ v_p \sin x_3 - w_e x_1 \\ w_p - w_e \end{pmatrix}. \quad (36)$$

We adopt specialization to a case where the two vehicles only possess a unit velocity and unit maximum turn rates. Here, as Merz notes, if the initial velocities are parallel such as $x_3 = 0$, then the equations of relative motion imply that E can be separated from P forever by the initial radial separation if it replicates P 's strategy. Whence, the barrier surface is closed and we are presented with Isaacs [26]'s game of kind where we must determine the nature of the surface. This terminal surface possesses a closed-form solution and we refer readers to the treatment by Merz [33]. In this example, our chief concern is to judge the efficacy of our toolbox with respect to the analytical solution of the barrier surface.

The the backward reachable tube that consists of the paths taken by the trajectories of either player is defined as in the rockets pursuit evasion game so that we have

$$\Phi(0, x) = \{x \in \mathcal{X} | x_1^2 + x_2^2 \leq r^2\}, \quad (37)$$

where again r is the capture radius. The target set is a cylinder as Φ above excludes the heading, x_3 . It is represented as shown in Fig. 10.

For a detailed treatment of the barrier surface, we refer readers to a proper analysis as elucidated in [36]. Here, we focus on the construction of the BUP. The set of states that constitute the useable part and its boundary are respectively a function of the implicit surface function representation $\Phi : [-T, 0] \times \mathcal{X} \rightarrow \mathbb{R}$ so that for a $t \in [0, T]$, where $T > 0$ is

$$\mathcal{T} = \{x \in \mathcal{X} | \Phi(0, x) \leq 0\} \quad (38)$$

$$R([-t, 0], \mathcal{T}) = \{x \in \mathcal{X} | \Phi(t, x) \leq 0\}, \quad (39)$$

When $t > 0$, the implicit surface representation is the following HJI PDE

$$\frac{\partial}{\partial t} \Phi(t, x) + \min(0, H(x, \nabla_x \Phi(t, x))) = 0. \quad (40)$$

It is easy to verify that the Hamiltonian is

$$H(x, p) = p_1(v_e - v_p \cos x_3) - p_2(v_p \sin x_3) - w|p_1 x_2 - p_2 x_1 - p_3| + w|p_3|. \quad (41)$$

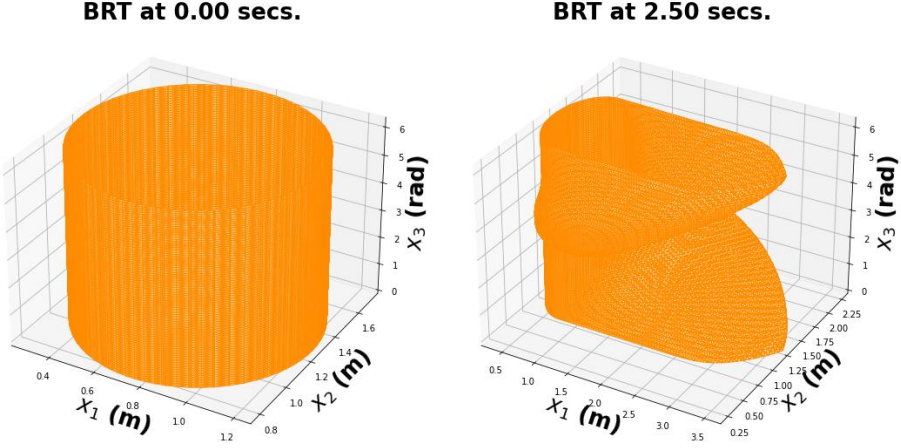


Fig. 10. The target set (left) and the boundary of the useable part of the state space after the differential game between P and E .

Since we are concerned with the special case that the linear and angular speeds are equal, we set $v_e = v_p = w \triangleq +1$ in the foregoing so that the Hamiltonian, in the final analysis is

$$H(x, p) = p_1(1 - \cos x_3) - p_2(\sin x_3) - |p_1x_2 - p_2x_1 - p_3| + |p_3|. \quad (42)$$

As before, we set up the differential game as in Listing 4

```
1 finite_diff_data = {"innerFunc": termLaxFriedrichs,
2 "innerData": {"grid": g, "hamFunc": dubins_rel.ham,
3 "partialFunc": dubins_rel.dissipation,
4 "dissFunc": artificialDissipationGLF,
5 "CoStateCalc": upwindFirstENO2},
6 "positive": True} // direction of approx. growth
```

Listing 4. HJ ENO2 computational scheme for the rockets.

The BRTs at various time steps for the approximation of the differential game is shown in Fig. 11. Compared to the standardized benchmark of the analytical solution [33] to the differential game problem and the approximated solutions [36, 37], our results jibe.

1.5 Computational Time Comparison with LevelSet Toolbox

In this subsection, we will compare the solution for recovering the zero level set of the systems presented in the previous examples against Mitchell [35]’s `LevelSet Toolbox` in MATLAB®. In all, we compare the efficacy of running various computational problems using our library on a CPU – running with Numpy and its fast arithmetic libraries – versus on a CPU with MATLAB® – as originally written in Mitchell [35]’s library. In addition, we compare the efficacy of running these computational problems on a single GPU.

For the CPU tests, we run the computation on an Intel Core™i9-10885H 16 cores-processor with a 2.4GHz clock frequency, and 62.4GB memory. We employed an NVIDIA Quadro RTX 4000 with 8.192 GiB memory running on a mobile workstation with the CPU specifications mentioned erstwhile in all of our GPU library accelerations.

Table 1 depicts the time it takes to process a full global optimization and the average time for the Lax-Friedrichs internal computational optimization algorithm for the reachable sets/tubes and time-to-reach sets for the examples we have considered. The column `Avg. local` depicts the

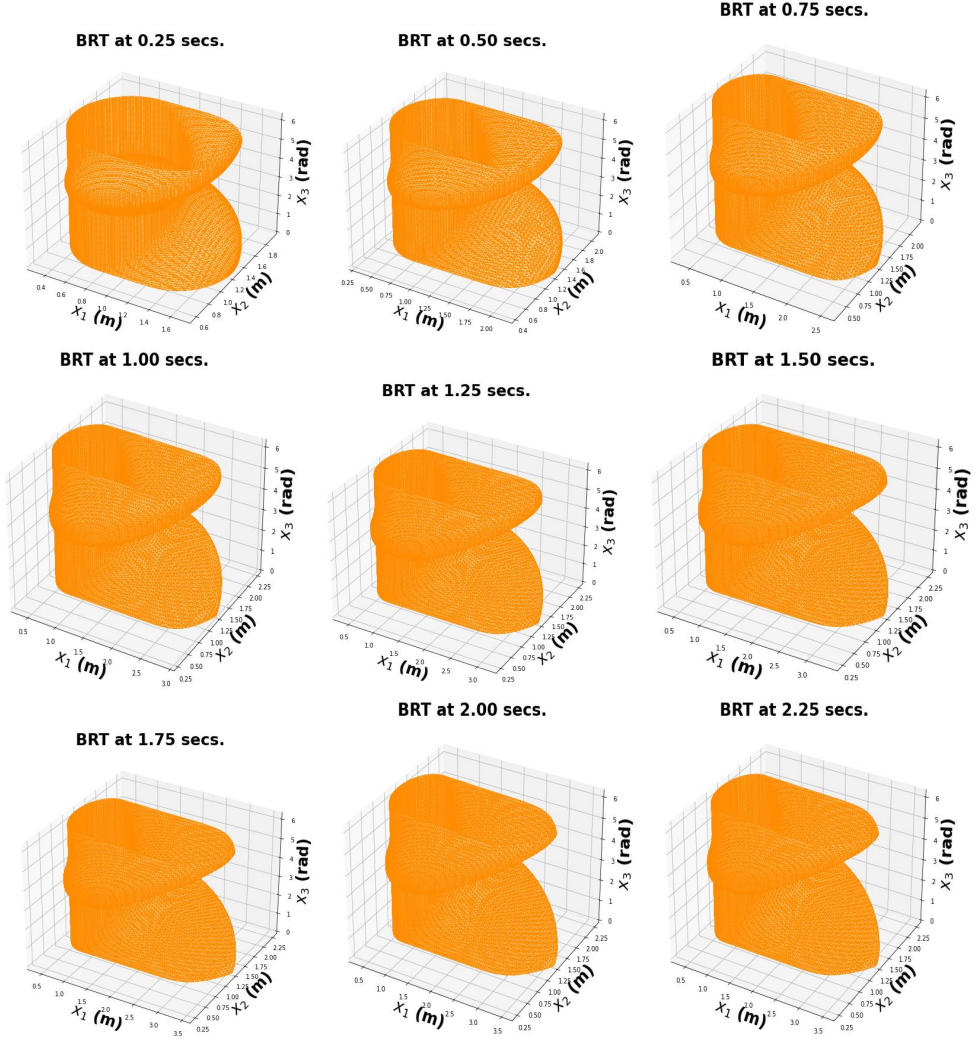


Fig. 11. BRTs for the differential game until termination time (2.25 secs).

average time the employed method of lines for resolving the HJ PDE takes per optimization step. The column `Global` denotes the average time it takes to compute the numerical solution to the HJ PDE. Each time query field represents an average over 20 experiments. We compare results of running the algorithm on a GPU, and CPU (both in `Numpy` and `MATLAB`). In all our evaluations, we aggressively free up GPU memory between and during computations in order to make GPU data streaming and memory computations more efficient.

Across the GPU experiments for the examples presented, we see that computation is significantly faster across all categories save the low-dimensional double integral plant experiment. We attribute this to the little amount of data points used in the overapproximated stacked levelsets. For the Air3D game of two vehicles on a plane problem and the two rockets differential game problem, the average local time for computing the solutions to the stagewise HJ PDE's using the method of lines for Air3D is a gain of $\sim 76\%$; the global time is a gain of 76.09%. Similarly, we notice substantial

Table 1. Time to Resolve HJ PDE 's.

Expt \ Lib	levelsetpy GPU Time (secs)		levelsetpy CPU Time (secs)		MATLAB CPU (secs)	
	Global	Avg. local	Global	Avg. local	Global	Avg. local
Rockets	11.5153 ± 0.038	1.1833	107.84 ± 0.42	10.4023	138.50	13.850
Doub. Integ.	14.7657 ± 0.2643	1.5441	3.4535 ± 0.34	0.4317	5.23	0.65375
Air 3D	30.8654 ± 0.1351	3.0881	129.1165 ± 0.13	12.6373	134.77	16.8462
Starlings	8.6889 ± 0.8323	0.42853	15.2693 ± 0.167	7.4387	N/A	N/A

computational gains for the two rockets differential game problem: 89% faster global optimization time and 88.62% average local computational time compared to our CPU implementations in Numpy. For this rockets game problem, compared against Mitchell [35] library, we notice a speedup of almost 92% in global optimization using our GPU-calibrated library versus an 89.32% gain using our CPU-calibrated library.

Notice the exception with the `Double Integrator` experiment, however: local and global computations take a little longer compared to deployments on the CPU – both on our Harris et al. [22]’s implementation and using Mitchell [35]’s native `MATLAB@toolbox`. We attribute this to the little size of the arrays of interest in this problem. The entire target set of the double integral plant exists on a two-dimensional grid whose analytic and approximate time-to-reach-the-origin computational time involves little computational gain in passing data onto the GPU. Nevertheless, we still see noticeable gains in using our CPU implementation as opposed to Mitchell [35]’s native `MATLAB@toolbox`.

On a CPU, owing to efficient arrays arithmetic native to Harris et al. [22]’s Numpy library, the average time to compute the zero levelsets per optimization step for the `odeCFLx` functions is faster with our Numpy implementation compared against Mitchell [35] `LevelSets MATLAB@ Toolbox` library computations across all experiments. The inefficiencies of `MATLAB@`’s array processing routine in the longer time to resolve stagewise BRTs and the effective time to finish the overall HJ PDE resolution per experiment manifests in all of our experimental categories. For CPU processing of HJ PDE ’s, it is reasonable, based on these presented data to expect that users would find our library far more useful for everyday computations in matters relating to the numerical resolution of HJ PDE ’s.

In all, there is conclusive evidence that our implementations are faster, extensible to modern libraries, and scalable for modern complex system design and verification problems that arise.

REFERENCES

- [1] Michael Athans and Peter L. Falb. 2013. *Optimal Control: An Introduction to the Theory and its Applications*. Courier Corporation.
- [2] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, and V. Zdravkovic. 2008. interaction Ruling Animal Collective Behavior Depends On Topological Rather Than Metric Distance: Evidence From A Field Study. *Proceedings of the National Academy of Sciences* 105, 4 (2008), 1232–1237. <https://doi.org/10.1073/pnas.0711437105> arXiv:<https://www.pnas.org/content/105/4/1232.full.pdf>
- [3] Tamer Basar and Geert Jan Olsder. 1999. *Dynamic Noncooperative Game Theory*. Vol. 23. Society for Industrial and Applied Mathematics.
- [4] Richard Bellman. 1957. *Dynamic programming*. Princeton University Press.
- [5] Sanjay P Bhat and Dennis S Bernstein. 1998. Continuous finite-time stabilization of the translational and rotational double integrators. *IEEE Transactions on automatic control* 43, 5 (1998), 678–682.
- [6] Burridge, R R and Rizzi, A A and Koditschek, D E. 1999. *Sequential Composition of Dynamically Dexterous Robot Behaviors*. Technical Report 6. 534–555 pages.
- [7] P Cardaliaguet. 1997. Nonsmooth Semipermeable Barriers, Isaacs’ Equation, And Application to a Differential Game with One Target and Two Players. *Applied Mathematics and Optimization* 36, 2 (1997), 125–146.
- [8] Andrea Cavagna, Alessio Cimarelli, Irene Giardina, Giorgio Parisi, Raffaele Santagati, Fabio Stefanini, and Massimiliano Viale. 2010. Scale-Free Correlations In Starling Flocks. *Proceedings of the National Academy of Sciences* 107, 26 (2010), 11865–11870.
- [9] M. G. Crandall, L. C. Evans, and P. L. Lions. 1984. Some Properties of Viscosity Solutions of Hamilton-Jacobi Equations. *Trans. Amer. Math. Soc.* 282, 2 (1984), 487.
- [10] Michael G Crandall and Pierre-Louis Lions. 1983. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American mathematical society* 277, 1 (1983), 1–42.
- [11] Michael G Crandall and P-L Lions. 1984. Two Approximations of Solutions of Hamilton-Jacobi Equations. *Mathematics of Computation* 43, 167 (1984), 1–19.
- [12] Michael G Crandall and Andrew Majda. 1980. Monotone Difference Approximations For Scalar Conservation Laws. *Math. Comp.* 34, 149 (1980), 1–21.
- [13] Defense Acquisition University. 2023. Validation. <https://www.dau.edu/tools/se-brainbook/Pages/Technical%20Processes/validation.aspx> Accessed April 5, 2023.
- [14] Defense Acquisition University. 2023. Verification. <https://www.dau.edu/tools/se-brainbook/Pages/Technical%20Processes/verification.aspx> Accessed April 5, 2023.
- [15] Stuart E Dreyfus. 1966. *Control Problems With Linear Dynamics, Quadratic Criterion, and Linear Terminal Constraints*. Technical Report. Rand Corp. Santa Monica Calif.
- [16] Lester E Dubins. 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics* 79, 3 (1957), 497–516.
- [17] L.C. Evans and Panagiotis E. Souganidis. 1984. Differential Games And Representation Formulas For Solutions Of Hamilton-Jacobi-Isaacs Equations. *Indiana Univ. Math. J* 33, 5 (1984), 773–797.
- [18] L.C. Evans and Panagiotis E. Souganidis. 1984. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. *Indiana Univ. Math. J* 33, 5 (1984), 773–797.
- [19] Lawrence C Evans. 2022. *Partial Differential Equations*. Vol. 19. American Mathematical Society.
- [20] Melanie Haiken. 2021. Starling murmurations are dazzling, ubiquitous, and puzzling. <https://tinyurl.com/4973byey> Accessed April 5, 2023.
- [21] Melanie Haiken. 2021. These birds flock in mesmerizing swarms of thousands—but why is still a mystery. <https://tinyurl.com/4973byey> Accessed April 5, 2023.
- [22] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with NumPy. *Nature* 585, 7825 (2020), 357–362.
- [23] Dirk Helbing, Illés Farkas, and Tamas Vicsek. 2000. Simulating dynamical features of escape panic. *Nature* 407, 6803 (2000), 487–490.
- [24] Kerianne L. Hobbs, Mark L. Mote, Matthew C.L. Abate, Samuel D. Coogan, and Eric M. Feron. 2023. Runtime Assurance for Safety-Critical Systems. *IEEE Control Systems Magazine* 43 (2023), 28–65. Issue 2.
- [25] Eberhard Hopf. 1950. The Partial Differential Equation $u_t + uu_x = \mu_{xx}^*$. (1950).
- [26] R Isaacs. 1999. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Kreiger, Huntigton, NY.
- [27] Ali Jadbabaie, Jie Lin, and A Stephen Morse. 2003. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on automatic control* 48, 6 (2003), 988–1001.

- [28] Guang-Shan Jiang and Danping Peng. 2000. Weighted ENO schemes for Hamilton–Jacobi equations. *SIAM Journal on Scientific computing* 21, 6 (2000), 2126–2143.
- [29] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. 2020. Deep Drone Acrobatics. *arXiv preprint arXiv:2006.05768* (2020).
- [30] S.N. Kruzkov. 1970. First Order Quasilinear Equations In Several Independent Variables. *Mathematics of the USSR-Sbornik* (1970).
- [31] Pierre-Louis Lions. 1982. *Generalized solutions of Hamilton-Jacobi equations*. Vol. 69. London Pitman.
- [32] John Lygeros. 2004. On reachability and minimum cost optimal control. *Automatica* 40, 6 (2004), 917–927.
- [33] AW Merz. 1972. The game of two identical cars. *Journal of Optimization Theory and Applications* 9, 5 (1972), 324–343.
- [34] Ian Mitchell. 2001. Games of two identical vehicles. *Dept. Aeronautics and Astronautics, Stanford Univ.* July (2001), 1–29.
- [35] Ian Mitchell. 2004. A Toolbox of Level Set Methods, version 1.0. *The University of British Columbia, UBC CS TR-2004-09* (July 2004), 1–94.
- [36] Ian Mitchell. 2020. A Robust Controlled Backward Reach Tube with (Almost) Analytic Solution for Two Dubins Cars. *EPiC Series in Computing* 74 (2020), 242–258.
- [37] Ian M. Mitchell, Alexandre M. Bayen, and Claire J. Tomlin. 2005. A Time-Dependent Hamilton-Jacobi Formulation of Reachable Sets for Continuous Dynamic Games. *IEEE Trans. Automat. Control* 50, 7 (2005), 947–957.
- [38] Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, and Crissman Loomis. 2017. CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*.
- [39] S Osher and R Fedkiw. 2004. Level Set Methods and Dynamic Implicit Surfaces. *Applied Mechanics Reviews* 57, 3 (2004), B15–B15.
- [40] Stanley Osher and James A. Sethian. 1988. Fronts Propagating with Curvature-Dependent Speed: Algorithms based on Hamilton-Jacobi Formulations. *Journal of Computational Physics* 79, 1 (1988), 12–49.
- [41] Stanley Osher and Chi-Wang Shu. 1988. *Efficient Implementation of Essentially Non-oscillatory Shock-capturing Schemes*. Technical Report 2. Hampton, Virginia. 439–471 pages.
- [42] Stanley Osher and Chi-Wang Shu. 1991. High-Order Essentially Nonoscillatory Schemes for Hamilton-Jacobi Equations. *SIAM Journal of Numerical Analysis* 28, 4 (1991), 907–922.
- [43] James A Sethian. 1996. A Fast Marching Level Set Method For Monotonically Advancing Fronts. *Proceedings of the National Academy of Sciences* 93, 4 (1996), 1591–1595.
- [44] James A. Sethian. 2000. Level Set Methods And Fast Marching Methods: Evolving Interfaces In Computational Geometry, Fluid Mechanics, Computer Vision, And Materials Science. *Robotica* 18, 1 (2000), 89–92.
- [45] Chi-Wang Shu and Stanley Osher. 1989. Efficient Implementation of Essentially Non-oscillatory Shock-capturing Schemes, II. *Journal of computational physics* 83, 1 (1989), 32–78.
- [46] W Murray Wonham. 1985. Linear Multivariable Control: A Geometric Approach. *Applications of Mathematics* 10 (1985).

A HAMILTONIAN OF A MURMURATION.

In this appendix, we provide a derivation for the overall Hamiltonian of a flock as elucidated in Theorem 1.

PROOF OF THEOREM 1. We write the free agents’ Hamiltonians in absolute coordinates and that of the agent under attack in relative coordinates with the pursuer. Henceforth, we drop the templated time arguments for ease of readability. The overall flock’s Hamiltonian is

$$\bigcup_{i=1}^{n_a-1} H_f^{(i)_j}(\mathbf{x}, p) = \bigcup_{i=1}^{n_a-1} \begin{bmatrix} p_1^{(i)_j} & p_2^{(i)_j} & p_3^{(i)_j} \end{bmatrix} \begin{bmatrix} v^{(i)_j} \cos \mathbf{x}_3 \\ v^{(i)_j} \sin \mathbf{x}_3 \\ \langle w_e^{(i)_j} \rangle_r \end{bmatrix}. \quad (43)$$

It follows that

$$\bigcup_{i=1}^{n_a-1} H_f^{(i)_j}(\mathbf{x}, p) = \bigcup_{i=1}^{n_a-1} \left[p_1^{(i)_j} v^{(i)_j} \cos \mathbf{x}_3 + p_2^{(i)_j} v^{(i)_j} \sin \mathbf{x}_3 + p_3^{(i)_j} \langle w_e^{(i)_j} \rangle_r \right]. \quad (44)$$

Equation (32) can be re-written as

$$\mathbf{H}_a^{(k)j}(\mathbf{x}, p) = - \left(\max_{w_e^{(k)j} \in [\underline{w}_e^j, \bar{w}_e^j]} \min_{w_p^{(k)j} \in [\underline{w}_p^j, \bar{w}_p^j]} \begin{bmatrix} p_1^{(k)j}(t) & p_2^{(k)j}(t) & p_3^{(k)j}(t) \\ -v_e^{(k)j}(t) + v_p^{(j)} \cos \mathbf{x}_3^{(k)j}(t) + \langle w_e^{(k)j} \rangle_r(t) \mathbf{x}_2^{(k)j}(t) \\ v_p^j(t) \sin \mathbf{x}_3^{(k)j}(t) - \langle w_e^{(k)j} \rangle_r(t) \mathbf{x}_1^{(k)j}(t) \\ w_p^j(t) - \langle w_e^{(k)j} \rangle_r(t) \end{bmatrix} \right), \quad (45)$$

where $p_l^{(k)j}(t) \mid_{l=1,2,3}$ are the adjoint or co-state vectors [33]. For the pursuer, its minimum and maximum turn rates are fixed so that we have \underline{w}_p^j as the minimum turn bound of the pursuing vehicle, and \bar{w}_p^j is the maximum turn bound of the pursuing vehicle. Rewriting (44), we find that

$$\begin{aligned} \mathbf{H}_a^{(k)j}(\mathbf{x}, p) &= - \left(\max_{w_e^{(k)j} \in [\underline{w}_e^j, \bar{w}_e^j]} \min_{w_p^{(k)j} \in [\underline{w}_p^j, \bar{w}_p^j]} \begin{bmatrix} -p_1^{(k)j} v_e^{(k)j} + p_1^{(k)j} v_p^j \cos \mathbf{x}_3^{(k)j} \\ + p_1^{(k)j} \langle w_e^{(k)j} \rangle_r \mathbf{x}_2^{(k)j} + p_2^{(k)j} v_p^j \sin \mathbf{x}_3^{(k)j} - p_2^{(k)j} \langle w_e^{(k)j} \rangle_r \mathbf{x}_1^{(i)j} + p_3^{(k)j} (w_p^j - \langle w_e^{(k)j} \rangle_r) \end{bmatrix} \right), \\ &= p_1^{(k)j} (v_e^{(k)j} - v_p^j \cos \mathbf{x}_3^{(k)j}) - p_2^{(k)j} v_p^j \sin \mathbf{x}_3^{(k)j} \\ &+ \left(\max_{\langle w_e^{(k)j} \rangle_r \in [\underline{w}_e^j, \bar{w}_e^j]} \min_{w_p^j \in [\underline{w}_p^j, \bar{w}_p^j]} \begin{bmatrix} \langle w_e^{(k)j} \rangle_r (p_2^{(k)j} \mathbf{x}_1^{(k)j} - p_1^{(k)j} \mathbf{x}_2^{(k)j} + p_3^{(k)j}) - p_3^{(k)j} w_p^j \end{bmatrix} \right). \end{aligned} \quad (46)$$

It follows that we have from (46) that

$$\begin{aligned} \mathbf{H}_a^{(k)j}(\mathbf{x}, p) &= p_1^{(k)j} (v_e^{(k)j} - v_p^j \cos \mathbf{x}_3^{(k)j}) - p_2^{(k)j} v_p^j \sin \mathbf{x}_3^{(k)j} - \underline{w}_p^j |p_3^{(k)j}| \\ &+ \bar{w}_e^j \left| p_2^{(k)j} \mathbf{x}_1^{(k)j} - p_1^{(k)j} \mathbf{x}_2^{(k)j} + p_3^{(k)j} \right| \end{aligned} \quad (47)$$

and that

$$\mathbf{H}_f^{(i)j}(\mathbf{x}, p) = \left[p_1^{(i)j} v^{(i)j} \cos \mathbf{x}_3 + p_3^{(i)j} v^{(i)j} \sin \mathbf{x}_3 + p_3^{(i)j} \langle w_e^{(i)j} \rangle_r \right]. \quad (48)$$

A fortiori the main equation (32) becomes

$$\begin{aligned} \mathbf{H}(\mathbf{x}, p) &= \bigcup_{j=1}^{n_f} \left(\bigcup_{i=1}^{n_a-1} \begin{bmatrix} p_1^{(i)j} v^{(i)j} \cos \mathbf{x}_3 + p_2^{(i)j} v^{(i)j} \sin \mathbf{x}_3 + p_3^{(i)j} \langle w_e^{(i)j} \rangle_r \\ \cup \left[p_1^{(k)j} (v^{(k)j} - v^{(k)j} \cos \mathbf{x}_3^{(k)j}) - p_2^{(k)j} v^{(k)j} \sin \mathbf{x}_3^{(k)j} - \underline{w}_p^j |p_3^{(k)j}| \right. \right. \\ \left. \left. + \bar{w}_e^j \left| p_2^{(k)j} \mathbf{x}_1^{(k)j} - p_1^{(k)j} \mathbf{x}_2^{(k)j} + p_3^{(k)j} \right| \right] \right) \end{bmatrix} \right). \end{aligned} \quad (49)$$

For the special case where the linear speeds of the evading agents and pursuer are equal i.e. $v_e^{(i)j}(t) = v_p(t) = +1m/s$, we have a murmuration's Hamiltonian as

$$\begin{aligned} \mathbf{H}(\mathbf{x}, p) &= \bigcup_{j=1}^{n_f} \left(\bigcup_{i=1}^{n_a-1} \begin{bmatrix} p_1^{(i)j} \cos \mathbf{x}_3 + p_2^{(i)j} \sin \mathbf{x}_3 + p_3^{(i)j} \langle w_e^{(i)j} \rangle_r \\ \cup \left[p_1^{(k)j} (1 - \cos \mathbf{x}_3^{(k)j}) - p_2^{(k)j} \sin \mathbf{x}_3^{(k)j} - \underline{w}_p^j |p_3^{(k)j}| \right. \right. \\ \left. \left. + \bar{w}_e^j \left| p_2^{(k)j} \mathbf{x}_1^{(k)j} - p_1^{(k)j} \mathbf{x}_2^{(k)j} + p_3^{(k)j} \right| \right] \right) \end{bmatrix} \right). \end{aligned} \quad (50)$$

□

Received 30 April 2024