Analysis and simulation of The Lady in the Lake problem

Yuntao Wu
dept. Electrical and Computer Engineering
University of Toronto
winstonyt.wu@mail.utoronto.ca **UTORid: wuyuntao**

***Abstract*—This project focuses on the lady in the lake problem. Firstly, the solution to a simplified version of the problem (dog and duck problem) is discussed as motivation. Next, the Lady in the lake problem is formulated as a two-person deterministic zero-sum differential game and the solution to it is presented. Then player actions are added to the motivation problem to formulate the inverse version of the Lady in the lake problem. The pursuer is restricted within the circle and the evader is restricted to the circumference of the circle. The evader can go either counter-clockwise or clockwise, and the pursuer can also move in any directions. This relaxes the restrictions of the original problem and gives a more interesting result in simulation, but also makes it harder to solve the problem analytically. Reinforcement learning is used to simulate the problems. Details of implementation related to the problem formulation are included.**

***Index Terms*—game theory, pursuit-evasion problem, reinforcement learning**

I. Introduction

The pursuit - evasion game is a type of differential game that can help formulate frameworks to analyze solutions to various problems such as collision avoidance [4], missile guidance [5] and UAV controls [6]. Within the set of pursuit - evasion games, there is a type called the "The Lady in the lake" problem [1]. A simplified version of it can be studied using nonlinear dynamics [2]. And it can be traced back to 18th century [7].

A. Motivation Problem

The dog and duck circular pursuit problem presented as an exercise in [2] is a simplified version of The Lady in the lake problem. The problem is formulated as follows:

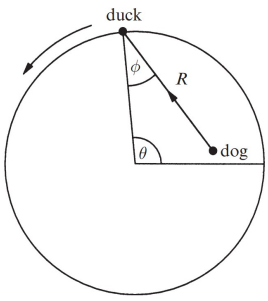

Fig. 1. Dog and duck circular pursuit problem.

A dog at the center of a circular pond sees a duck swimming along the edge as in Fig. 1. The dog chases the duck by always swimming straight toward it. In other words, the dog's velocity vector always lies along the line connecting it to the duck. Meanwhile, the duck takes evasive action by swimming around the circumference as fast as it can, always moving counterclockwise. Assuming the pond has unit radius and both animals swim at constant speed with $v_{dog} = kv_{duck}$. What does the dog do in the long run when $k = 1$ and $k = \frac{1}{2}$?

The derivation for the differential equations capturing the system behavior is omitted as it is not the focus of this project. It can be done using simple geometry. The final differential equations are:

$$\begin{cases} \frac{dR}{d\theta} = \sin\phi - k \\ \frac{d\phi}{d\theta} = \frac{\cos\phi}{R} - 1 \end{cases} \tag{1}$$

with $R \in [0, 2]$, $\phi \in [0, \frac{\pi}{2}]$.

We can use the phase plane to analyze the long term behavior.

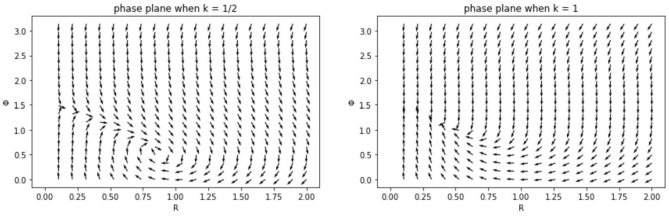

Fig. 2. Phase planes of the system when $k = \frac{1}{2}$ and $k = 1$.

When $k = \frac{1}{2}$, as in the left plot of Fig. 2, if the dog start at $R = 1$, it will be attracted to a fixed point $\phi = \frac{\pi}{6}$, $R = \frac{\sqrt{3}}{2}$, where $\frac{dR}{d\theta} = \frac{d\phi}{d\theta} = 0$. This means that the dog will eventually have a distance $\frac{\sqrt{3}}{2}$ to the duck and never catch it.

When $k = 1$, as in the right plot of Fig. 2, if the dog starts at $R = 1$, it will still never be able to reach the $(0, 0)$ point. This means that the dog still cannot catch the duck.

B. Project Questions

As shown in I-A, we can know the long term behavior of a simple pursuit - evasion problem by analyzing the differential equations that describes the system behavior. However, there

are several questions that are not captured in this simple problem.

- For what ks can the dog reach the duck as close as possible? (R cannot be zero in the analysis, so there is no definite number to define closest.)
- What will happen if the speeds of the dog and duck are not constant?
- What will happen if the dog doesn't swim straight to the duck?

These questions can potentially be solved using game theory formulation and reinforcement learning.

II. Game Theoretical Formulation

A. Basics

The study of nonlinear continuous time dynamics mostly focuses on the long term behavior of an autonomous system

$$\dot{\mathbf{x}} = f(\mathbf{x}, a), \mathbf{x}(0) = \mathbf{x}_0 \tag{2}$$

with $x \in F^n$, F a field, a a parameter, $\mathbf{x}_0 \in F^n$ the initial condition. It analyzes how the trajectories and limiting behavior $\lim_{t\to\infty} \mathbf{x}(t)$ change w.r.t. a and and $\mathbf{x}_0$. [2]

The pursuit - evasion games going to be considered in this project is two-person deterministic zero-sum differential games with variable terminal time. It can be described by

$$\dot{\mathbf{x}} = f(t, \mathbf{x}, u^1(t), u^2(t)), \mathbf{x}(0) = \mathbf{x}_0, \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^n$, $u^i \in \mathbb{R}^m$, $i = 1, 2$. u^1 is the strategy chosen by the pursuer $\mathbf{P}$. u^2 is the strategy chosen by the Evader $\mathbf{E}$. The final time T is defined by

$$T = \inf\left\{t \in \mathbb{R}^+, (\mathbf{x}(t), t) \in \Lambda\right\}, \tag{4}$$

where Λ is the target set with boundary $\partial\Lambda$ described by $l(t, x) = 0$. The single objective function is

$$J\left(u^1, u^2\right) = \int_0^T g\left(t, \mathbf{x}, u^1, u^2\right) dt + q\left(T, \mathbf{x}(T)\right). \tag{5}$$

Compared to the dynamics problem formulation in (2), the system of pursuit - evasion game (3) is non-autonomous. It depends on time t and time-dependent player strategies u^1, u^2. This means that we can formulate and analyze a more complicated problem. Also, it does not focus on the long term behavior as $t \to \infty$. Rather, it tries to optimize the objective function at the final time $T < \infty$, so we are able to find an optimal strategy by analyzing the objective function.

Consider the minimax value of the cost function, i.e. the value function, when started from $(t, \mathbf{x})$:

$$V(t, \mathbf{x}) = \min_{u^1}\max_{u^2}\left\{\int_0^T g\left(t, \mathbf{x}, u^1, u^2\right) dt + q\left(T, \mathbf{x}(T)\right)\right\}. \tag{6}$$

It satisfies the Issacs equation if V is continuously differentiable in t and $\mathbf{x}$:

$$-\frac{\partial V}{\partial t} = \min_{u^1}\max_{u^2}\left\{\frac{\partial V}{\partial \mathbf{x}} f(t, \mathbf{x}, u^1, u^2) + g(t, \mathbf{x}, u^1, u^2)\right\}. \tag{7}$$

Let $H = \frac{\partial V}{\partial \mathbf{x}} f(t, \mathbf{x}, u^1, u^2) + g(t, \mathbf{x}, u^1, u^2)$. $\frac{d}{dt}\left(\frac{\partial V}{\partial \mathbf{x}}\right) = -\frac{\partial}{\partial \mathbf{x}} H$ with $\frac{\partial V(T)}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} q\left(T, \mathbf{x}^*(T)\right)$ along $l(t, \mathbf{x}) = 0$ is the costate function which can help us find the optimal strategies for the game.

B. The Lady in the Lake problem

The problem statement can be summarized as follows:

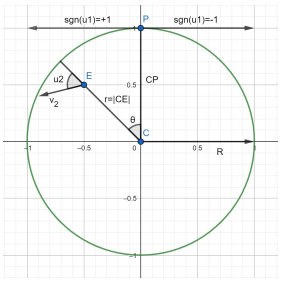

Fig. 3. The Lady in the Lake problem.

$\mathbf{E}$ is swimming at a maximum speed v_2 in a circular pond with radius R, trying to reach the shore without being caught by $\mathbf{P}$. $\mathbf{P}$ runs along the perimeter with maximum speed $v_1 = 1$, trying to intercept $\mathbf{E}$, i.e. catch $\mathbf{E}$, when $\mathbf{E}$ reaches the shore. Both can change direction instantaneously. $\mathbf{E}$'s goal is to maximize the angle $\theta = \angle PCE$ w.r.t. the center C of the pond, while $\mathbf{P}$ tries to minimize θ. Assume $v_2 < 1$ in the analysis.

Let u^1 be the strategy of $\mathbf{P}$, $\left|u^1\right| \leq 1$, indicating the direction and speed $\mathbf{P}$ goes along the perimeter, $r = |CE|$ be the distance $\mathbf{E}$ is from the center of the pond, and u^2 be the strategy of $\mathbf{E}$, indicating its direction w.r.t. the vector $\overrightarrow{CE}$.

Here the positive direction for u^1 and θ is counter-clockwise instead of clockwise in the book. This change better suits my convention and gives easier setup for consistency in simulation without affecting the numerical analysis.

The system can be described by

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{\theta} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{v_2 \sin u^2}{r} - \frac{u^1}{R} \\ v_2 \cos u^2 \end{pmatrix}. \tag{8}$$

The objective function is

$$\begin{gathered} J = |\theta(T)|, \\ \text{with } T = \inf\{t : r(t) = R\}, |\theta(t)| \leq \pi, \forall t \in [0, T]. \end{gathered} \tag{9}$$

With the given set up, we can see that $g \equiv 0$, $q\left(T, \mathbf{x}(T)\right) = |\theta(T)|$. The value function has no t dependence, so $\frac{\partial V}{\partial t} = 0$. Also, $V(T, \mathbf{x}) = |\theta(T)|$. Then the Issacs equation becomes

$$
\begin{aligned}
0 &= \min_{u^1}\max_{u^2}\left\{\frac{\partial V}{\partial \mathbf{x}} f(t,\mathbf{x},u^1,u^2)\right\} \\
&= \min_{u^1}\max_{u^2}\left\{\frac{\partial V}{\partial \theta}\dot{\theta}+\frac{\partial V}{\partial r}\dot{r}\right\} \\
&= \min_{u^1}\max_{u^2}\left\{\frac{\partial V}{\partial \theta}\left(\frac{v_2\sin u^2}{r}-\frac{u^1}{R}\right)+\frac{\partial V}{\partial r}v_2\cos u^2\right\}.
\end{aligned} \tag{10}
$$

To find the optimal strategy u^{1*} and u^{2*}, we firstly reorganize the terms. RHS of (10) then becomes

$$
\min_{u^1}\max_{u^2}\left\{-\frac{u^1}{R}\frac{\partial V}{\partial \theta}+v_2\left(\frac{1}{r}\frac{\partial V}{\partial \theta}\sin u^2+\frac{\partial V}{\partial r}\cos u^2\right)\right\}. \tag{11}
$$

The minimax operation is separable. We get

$$
\min_{u^1}\left\{-\frac{u^1}{R}\frac{\partial V}{\partial \theta}\right\}+v_2\max_{u^2}\left\{\frac{1}{r}\frac{\partial V}{\partial \theta}\sin u^2+\frac{\partial V}{\partial r}\cos u^2\right\}. \tag{12}
$$

It is easy to get that $u^{1*} = \operatorname{sgn}\left(\frac{\partial V}{\partial \theta}\right)$, with $\min_{u^1}\left\{-\frac{u^1}{R}\frac{\partial V}{\partial \theta}\right\} = -\frac{1}{R}\left|\frac{\partial V}{\partial \theta}\right|$. For u^{2*}, observe that the value we try to maximize is a dot product between vectors $a_1 = \left(\frac{1}{r}\frac{\partial V}{\partial \theta}, \frac{\partial V}{\partial r}\right)$ and $a_2 = \left(\sin u^2, \cos u^2\right)$. The maximum value is achieved when $a_1 \parallel a_2$.

Given that $V(T,\mathbf{x}) = V(\theta, r) = |\theta(T)|$, and the costate function $\frac{d}{dt}\left(\frac{\partial V}{\partial \theta}\right) = -\frac{\partial(11)}{\partial \theta} = 0$, $u^{1*} = \operatorname{sgn}\left(\frac{\partial V}{\partial \theta}\right) = \operatorname{sgn}(\theta(T))$. This makes the angle θ smaller.

Since $a_1 = \left(\frac{1}{r}\frac{\partial V}{\partial \theta}, \frac{\partial V}{\partial r}\right)$ and $a_2 = \left(\sin u^2, \cos u^2\right)$ are parallel, the following equations hold for any $k \in \mathbb{R}$.

$$
\frac{1}{r}\frac{\partial V}{\partial \theta} = k\sin u^{2*}, \frac{\partial V}{\partial r} = k\cos u^{2*}. \tag{13}
$$

Substitute u^{1*} and (13) into (12)=0, we get $k = \frac{1}{Rv_2}$, and thus $\sin u^{2*} = \frac{Rv_2}{r}\operatorname{sgn}(\theta(T))$. This equation only holds for $r \geq Rv_2$. For $r < Rv_2$, consider the angular velocities for the two players. For $\mathbf{P}$, $\omega_P = \frac{v_1}{R} \leq \frac{1}{R}$. For $\mathbf{E}$, she can always swim tangential to the circle she is at with $\omega_E = \frac{v_2}{r} > \frac{1}{R} \geq \omega_P$, so she can always make θ to be the optimal value π when reaching the decision boundary $r = Rv_2$ from inside and we only need to consider the case as $r > Rv_2$ where $\omega_P > \omega_E$.

The outcome of the game is $|\theta(T)| = \pi + \arccos v_2 - \frac{1}{v_2}\sqrt{1-v_2^2}$. The lady can escape if $|\theta(T)| > 0$, which gives $v_2 > 0.21723$ [1].

C. Inverse Problem

Now, let's consider the inverse problem where the situation of $\mathbf{P}$ and $\mathbf{E}$ are switched. This is similar to the motivation problem, but has less restrictions.

$\mathbf{E}$ is swimming at maximum speed $v_1 = 1$ along the perimeter, trying not to be caught by $\mathbf{P}$ as long as possible. $\mathbf{P}$ starts at the center of the circle, swimming at constant speed v_2, trying to catch $\mathbf{E}$ as soon as possible. Both can change direction instantaneously, so we can reduce the probability that $\mathbf{P}$ goes to the boundary of the bond directly and move in the opposite direction of $\mathbf{E}$ which gives a trivial problem. Similar to the motivation problem in I-A, $\mathbf{E}$ wants to maximize the distance $r = |PE|$ while $\mathbf{P}$ wants to minimize it.

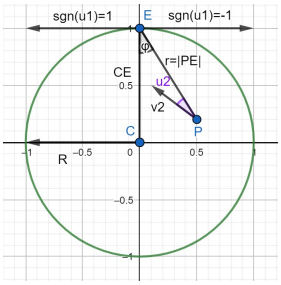

Fig. 4. The inverse problem.

Let u^1 be the strategy of $\mathbf{E}$, $|u^1| \leq 1$, indicating the direction and speed $\mathbf{E}$ goes along the perimeter, u^2 be the strategy of $\mathbf{P}$, indicating its direction w.r.t. the vector $\overrightarrow{PE}$, $\phi = \angle PEC$.

The system can be described by:

$$
\dot{\mathbf{x}} = \begin{pmatrix}\dot{\phi}\\ \dot{r}\end{pmatrix} = \begin{pmatrix}\left(\frac{R\cos\phi}{r}-1\right)u^1 - \frac{v_2\sin u^2}{r}\\ u^1\sin\phi - v_2\cos u^2\end{pmatrix}. \tag{14}
$$

The objective function is

$$
\begin{aligned}
&J = r(T),\\
&\text{with } T = \min\left\{\inf\left\{t : r(t) = 0\right\}, 15 \text{ sec}\right\}.
\end{aligned} \tag{15}
$$

The 15 sec restriction on the time is added, because the game may not end, especially when u_2 keeps zero and we reduce to the simplified version similar to I-A. This is to help terminate the simulation.

The Issacs equation for this problem is:

$$
\begin{aligned}
0 = \min_{u^2}\max_{u^1}\Bigg\{&\frac{\partial V}{\partial \phi}\left(\left(\frac{R\cos\phi}{r}-1\right)u^1 - \frac{v_2\sin u^2}{r}\right)\\
&+\frac{\partial V}{\partial r}\left(u^1\sin\phi - v_2\cos u^2\right)\Bigg\}.
\end{aligned} \tag{16}
$$

This Issacs equation is still separable,

$$
\begin{aligned}
0 =& \max_{u^1}\left\{u^1\left[\frac{\partial V}{\partial \phi}\left(\frac{R\cos\phi}{r}-1\right)+\frac{\partial V}{\partial r}\sin\phi\right]\right\}\\
&+ v_2\min_{u^2}\left\{-\frac{1}{r}\frac{\partial V}{\partial \phi}\sin u^2 - \frac{\partial V}{\partial r}\cos u^2\right\}.
\end{aligned} \tag{17}
$$

This gives the optimal strategies $u^{1*} = \operatorname{sgn}\left[\frac{\partial V}{\partial \phi}\left(\frac{R\cos\phi}{r}-1\right)+\frac{\partial V}{\partial r}\sin\phi\right]$, u^{2*} such that $\left(\sin u^2, \cos u^2\right) \parallel \left(-\frac{1}{r}\frac{\partial V}{\partial \phi}, -\frac{\partial V}{\partial r}\right)$, but with reverse sign.

The costate functions are $\frac{d}{dt}\left(\frac{\partial V}{\partial \phi}\right) = u^1 V_\phi \frac{R\sin\phi}{r} - u^1 V_r\cos\phi$, $\frac{d}{dt}\left(\frac{\partial V}{\partial r}\right) = V_\phi\left(\frac{R\cos\phi}{r^2}u^1 - \frac{v_2\sin u^2}{r^2}\right)$ with $V(T,\mathbf{x}) = r(T)$. This time the costate functions don't give any useful information to simplify the player strategies, thus the problem is not solvable at least for me. Only some simple analysis are done.

u^{1*} is the sign of a dot product between $a_1 = \left(\frac{\partial V}{\partial \phi}, \frac{\partial V}{\partial r}\right)$ and $a_2 = \left(\frac{R\cos\phi}{r}-1, \sin\phi\right)$. Observe that a_2 is the $\left(\dot{\phi}, \dot{r}\right)$ of

the simplified system with $\mathbf{P}$'s velocity along $\overrightarrow{PE}$ being 0. To make this happen, $\mathbf{E}$ should go to the direction that makes the angle ϕ larger. Then, $\mathbf{P}$ will try to make ϕ smaller by making u^2 larger, and larger portion of the v_2 will be projected onto the direction perpendicular to $\overrightarrow{PE}$. By this way $\mathbf{E}$ can also make r larger, i.e. moving away from $\mathbf{P}$.

For u^{2*}, following the same calculation as in II-B, we get $\cos u^{2*} = -\frac{v_2}{S}\frac{\partial V}{\partial r}$ where $S = \frac{\partial V}{\partial \phi}\left(\frac{R\cos\phi}{r} - 1\right) + \frac{\partial V}{\partial r}\sin\phi$ which doesn't tell much as compared to u^{1*}.

III. Reinforcement Learning Simulation

A. Environment Setup

Unity ML-Agents toolkit is used to setup the simulation. It provides different state-of-the-art reinforcement learning algorithms based on PyTorch and allows user to train and simulate intelligent agents in 3D environment [3].

Fig. 5 shows the setup of training environment. Each scene contains 6 training regions so that 6 sets of agents train in parallel. This speeds up and also stabilizes the training process. In each training region, spheres are used to represent the agents. Red spheres represent the pursuers $\mathbf{P}$ and yellow spheres represent the evaders $\mathbf{E}$. The $\mathbf{E}$ escapes when $\|x_{\mathbf{E}}\| \geq R$. Due to the non-negligible size of the spheres, the condition for $\mathbf{P}$ catching $\mathbf{E}$ is relaxed to $\|\mathbf{x_P} - \mathbf{x_E}\| < 3\rho_{\text{sphere}}$, where $\rho_{\text{sphere}} = 0.5$ is the standard sphere radius in Unity. The region enclosed by this distance is shown as the small sphere gizmos around the red spheres. The agents are only allowed to move on their corresponding platforms within a circular region $\{\mathbf{x} : \|\mathbf{x}\| \leq R\}$ as represented by the large sphere gizmos. $R = 10$ in all training and inference stages.

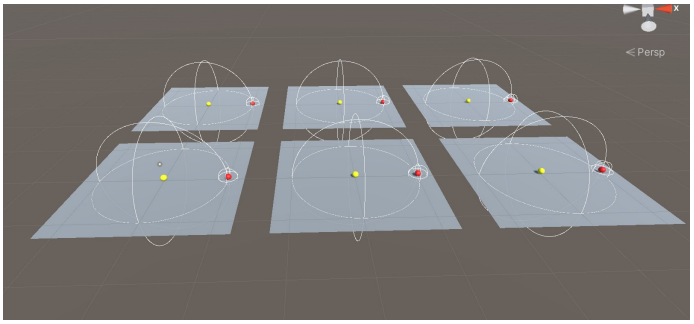

Fig. 5. Unity environment setup. The figure shows the case where evader starts from the origin and the pursuer tries to catch the evader on the perimeter. The inverse problem has the same environment setup. The differences are listed in III-B and III-C.

Proximal Policy Gradient is used for agent learning. It is an actor-critic method with an objective function that allows multiple epoches of minibatch updates as compared to standard policy gradient method [8]. The algorithm is able to beat other policy gradient methods such that A2C and TRPO in most continuous control environments.

The batch size is 512 and the buffer size is 5,120. The learning rate starts at 3×10^{-4} at decrease linearly. All agents in a team train for a maximum of 10^6 steps altogether. The neural network is a simple 3 layer MLP with 512 neurons in each layer. The discount factor for the reward is $\gamma = 0.99$.

Since the two agents $\mathbf{P}$ and $\mathbf{E}$ are non-cooperative and have inverse reward signals, self-play is also used. $\mathbf{P}$ and $\mathbf{E}$ are trained in alternation. Snapshots of each agent are taken at every 20,000 steps. Training teams are switched every 100,000 steps. The opponents' policies are changed every 2,000 steps. The chance of playing against the opponent's latest policy is 50%. At other times, the opponent's policy is taken from a past snapshot.

B. Original Problem

This section focus on the setup and simulation results specific to the original problem discussed in II-B. The termination condition is either $\mathbf{E}$ escapes or $\mathbf{P}$ catches $\mathbf{E}$ according to the conditions defined in III-A.

Pursuers are initialized at random spots on the perimeter of the large circle so the agents they can potentially learn better strategies. Evaders are initialized at the center of the circles.

The values considered in evaluation include instantaneous θ, average terminal θ when the evader is caught or the evader escaped, average θ when the evader cross the decision boundary $r = Rv_2$ for the first time and the number of times the evader is caught/escaped.

Reinforcement learning setup for P:

- *Observation:* The current positions of both $\mathbf{P}$ and $\mathbf{E}$ w.r.t. the center of the circle, the distance $\|\mathbf{PE}\|$ and the normalized dot product $\frac{\overrightarrow{CE}\cdot\overrightarrow{CP}}{\|CE\|\|CP\|}$. $\mathbf{s_P} = \left(x_{\mathbf{P}}, z_{\mathbf{P}}, x_{\mathbf{E}}, z_{\mathbf{E}}, \|\mathbf{PE}\|, \frac{\overrightarrow{CE}\cdot\overrightarrow{CP}}{\|CE\|\|CP\|}\right)$. y values are the unchanged height, thus not considered.
- *Action:* Continuous action $a_{\mathbf{P}} \in [-1, 1]$ representing the speed and direction $\mathbf{P}$ goes along the perimeter. Parametric equation $\mathbf{x} = (R\cos(\theta_{\mathbf{P}} + a_{\mathbf{P}}dt), R\sin(\theta_{\mathbf{P}} + a_{\mathbf{P}}dt))$ is used to represent the path, where $\theta_{\mathbf{P}}$ is the angle $\overrightarrow{CP}$ makes with the positive x-axis. So the action here is the instantaneous angular velocity. The actual speed of $\mathbf{P}$ is $v_{\mathbf{P}} = a_{\mathbf{P}}R = 10a_{\mathbf{P}}$ with maximum speed $v_{\mathbf{P}\,\max} = 10$. The maximum speed assigned to the evader is adjusted accordingly.
- *Reward:* For each step taken, the normalized dot product $\frac{\overrightarrow{CE}\cdot\overrightarrow{CP}}{\|CE\|\|CP\|}$ is calculated and added as positive reward, because $\mathbf{P}$ wants to minimize the angle $\angle PCE$ which is equivalent to maximizing the dot product. At the end of each round, $\mathbf{P}$ gets +1000 reward if successfully catches $\mathbf{E}$, and gets -1000 if $\mathbf{E}$ escaped. The final reward is set to this scale to account for the number of steps (usually several hundreds) it may take and the reward at each step.

Reinforcement learning setup for E:

- *Observation:* The current positions of both $\mathbf{E}$ and $\mathbf{P}$ with respect to the center of the circle, the distance $\|\mathbf{PE}\|$ and the normalized dot product $\frac{\overrightarrow{CE}\cdot\overrightarrow{CP}}{\|CE\|\|CP\|}$. $\mathbf{s_E} = \left(x_{\mathbf{E}}, z_{\mathbf{E}}, x_{\mathbf{P}}, z_{\mathbf{P}}, \|\mathbf{PE}\|, \frac{\overrightarrow{CE}\cdot\overrightarrow{CP}}{\|CE\|\|CP\|}\right)$.

- *Action:* Continuous action $a_{\mathbf{E}} \in [-1, 1]$ representing the direction of $\mathbf{E}$. The velocity vector is $\mathbf{v_E} = (v_2 \cos(\theta_{\mathbf{E}} + a_{\mathbf{E}}\pi), v_2 \sin(\theta_{\mathbf{E}} + a_{\mathbf{E}}\pi))$, where $\theta_{\mathbf{E}}$ is the angle $\overrightarrow{CE}$ makes with the positive x-axis.
- *Reward:* For each step taken, the normalized dot product $\frac{\overrightarrow{CE} \cdot \overrightarrow{CP}}{\|CE\|\|CP\|}$ is calculated and added as negative reward, because $\mathbf{E}$ wants to maximize the angle $\angle PCE$ which is equivalent to minimizing the dot product. At the end of each round, $\mathbf{E}$ gets +1000 reward if successfully escaped, and gets -1000 if caught.

Results when $v_{\mathbf{E}} < 0.21 v_{\mathbf{P}}$:

As discussed in II-B, $v_2 > 0.21723$ is a threshold for $\mathbf{E}$ to successfully escape if $\mathbf{P}$'s max speed is 1. Now that $\mathbf{P}$'s speed has been scaled to 10 times the original value, the speed of $\mathbf{E}$ is chosen to be 1 to simulate the situation. The decision boundary is then the circle $\{\mathbf{x} : \|\mathbf{x}\| = 1\}$. Fig. 6 shows the training statistics. $\mathbf{P}$'s rewards become consistently positive (around 1800 at the end) and $\mathbf{E}$'s rewards become negative (around -1800 at the end) within 200,000 steps and their sum is approximately 0. This is inline with the analytical result, since for this velocity configuration, $\mathbf{P}$ should always catch $\mathbf{E}$ with optimal strategy. The values exceeding ± 1000 are also consistent with episode length and the angle rewards added to the agents. The slight mismatch in the number might be caused by the asynchronous angle calculation by the two agents, which is a limitation of using a game engine for simulation.

Then inference is run on the trained models. When $\mathbf{E}$ is within the decision boundary, it can indeed make θ as large as possible, but $\mathbf{P}$ minimizes θ as soon as $\mathbf{E}$ goes across the decision boundary. Fig. 7 shows the results at 102 episodes. $\mathbf{E}$ is still possible to escape, potentially because the model is not yet trained to optimum, but the success rate is less than 10%. $\theta \approx 90°$ at decision boundary, which means that $\mathbf{E}$ is trying to maximize θ around the decision boundary. However, $\mathbf{P}$ can quickly make θ close to 0 after $\mathbf{E}$ crosses the decision boundary. The average termination angle is close to 0. This means that $\mathbf{P}$ is always close enough to $\mathbf{E}$ and able to catch $\mathbf{E}$. The failing cases are usually when $\mathbf{E}$ tangentially enters the catching region of $\mathbf{P}$ and hits the escape boundary simultaneously. In this case, $\mathbf{P}$ may think that it is close enough to $\mathbf{E}$ and stops, while it could just be some noisy data caused by the floating point rounding.

Results when $v_{\mathbf{E}} > 0.21 v_{\mathbf{P}}$:

The max speed of $\mathbf{E}$ is chosen to be 4 to simulate the situation. The decision boundary is then the circle $\{\mathbf{x} : \|\mathbf{x}\| = 4\}$. Fig. 8 shows the training statistics. It converges much better and also verifies the result that when $v_{\mathbf{E}} > 0.21 v_{\mathbf{P}}$, the evader is able to escape easily. The reward for $\mathbf{E}$ is positive (around 1100) and the reward for $\mathbf{P}$ is negative (around -1100).

This time, in inference stage, $\mathbf{E}$ can always escape as in Fig. 9. $\mathbf{E}$ manages to make $\theta \approx 150°$ on average when reaching the decision boundary and the terminal $\theta \approx 100°$.

C. Inverse Problem

This section focus on the setup and simulation results specific to the original problem discussed in II-C. The termination

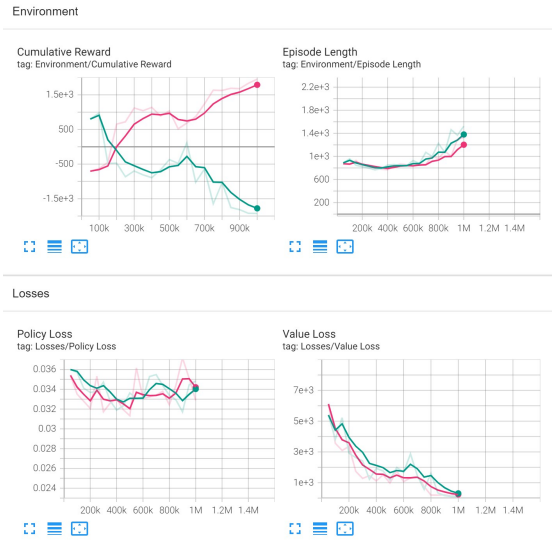

Fig. 6. Original problem. Cumulative Reward, episode length and policy & value losses in training w.r.t. steps for $v_{\mathbf{E}} = 1$. Red is pursuer. Blue is evader.

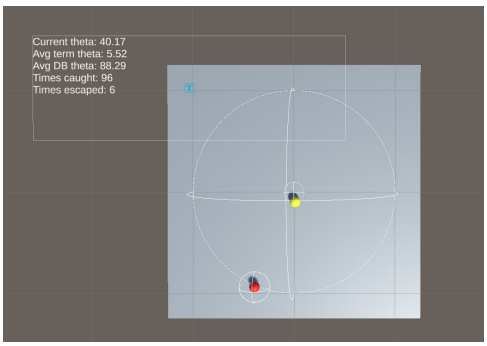

Fig. 7. Original problem. Inference sample output for $v_{\mathbf{E}} = 1$.

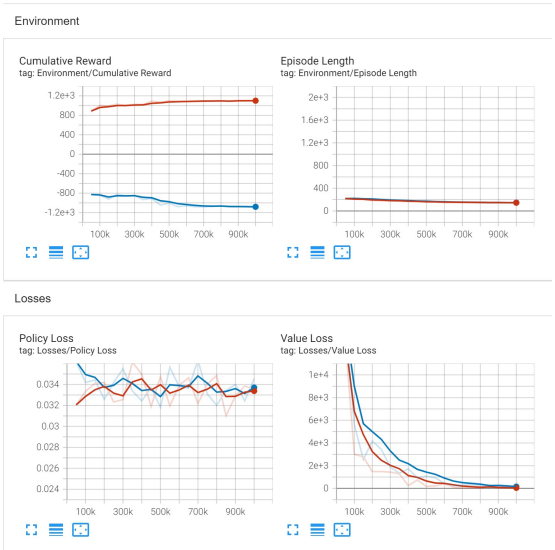

Fig. 8. Original problem. Cumulative Reward, episode length and policy & value losses in training w.r.t. steps for $v_{\mathbf{E}} = 4$. Blue is pursuer. Red is evader.

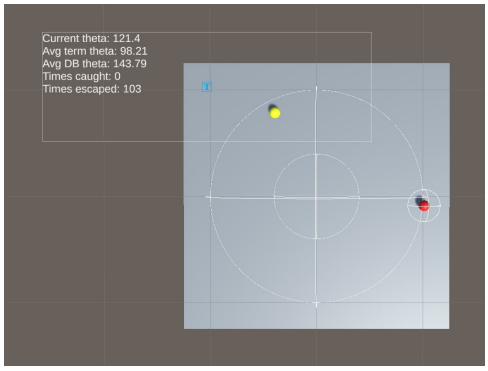

Fig. 9. Original problem. Inference sample output for $v_{\mathbf{E}} = 4$.

condition is either $\mathbf{P}$ catches $\mathbf{E}$ according to the conditions defined in III-A or 15 seconds has passed.

Pursuers are initialized at the center of the circles. If the pursuer goes out of the circle, its position is reset. Evaders are initialized at random spots on the perimeter of the large circle so the agents they can potentially learn better strategies. Since the threshold speed for catching is not known, a maximum speed is given to the pursuer, and it should learn to adjust the speed to catch the evader quickly.

The values considered in evaluation include instantaneous speed and direction of $\mathbf{P}$, distance $\|\mathbf{PE}\|$, angle ϕ, the number of times the evader is caught/survives.

Reinforcement learning setup for P:

- *Observation:* Same as the original problem. The current positions of both $\mathbf{P}$ and $\mathbf{E}$ with respect to the center of the circle, the distance $\|\mathbf{PE}\|$ and the normalized dot product $\frac{\overrightarrow{CE}\cdot\overrightarrow{CP}}{\|CE\|\|CP\|}$. $\mathbf{s_P} = \left(x_{\mathbf{P}}, z_{\mathbf{P}}, x_{\mathbf{E}}, z_{\mathbf{E}}, \|\mathbf{PE}\|, \frac{\overrightarrow{CE}\cdot\overrightarrow{CP}}{\|CE\|\|CP\|}\right)$.
- *Action:* Two continuous actions $a_{\mathbf{P}} = (a_{\mathbf{P}1}, a_{\mathbf{P}2}) \in [0.5, 1.0] \times [-1, 1]$ representing the speed fraction and direction of $\mathbf{P}$. The velocity vector is $\mathbf{v_P} = (v_2 a_{\mathbf{P}1} \cos(\theta + a_{\mathbf{P}2}\pi), v_2 a_{\mathbf{P}1} \sin(\theta + a_{\mathbf{P}2}\pi))$, where θ is the angle $\overrightarrow{PE}$ makes with the positive x-axis. $a_{\mathbf{P}1}$ is chosen to be in the range $[0.5, 1.0]$ to make sure that $\mathbf{P}$ at least moves so that it can learn a bit faster.
- *Reward:* For each step taken, distance $\|\mathbf{PE}\|$ is calculated and $0.5 - \frac{\|\mathbf{PE}\|}{2R}$ is added as reward. When the distance is too large ($> R$), the reward is negative, so $\mathbf{P}$ should try to get as close to $\mathbf{E}$ as possible. At the end of each round, $\mathbf{P}$ gets +1000 reward if successfully catches $\mathbf{E}$ and gets -1000 if 15 seconds has passed (timed-out).

Reinforcement learning setup for E:

- *Observation:* Same as the original problem. The current positions of both $\mathbf{E}$ and $\mathbf{P}$ with respect to the center of the circle, the distance $\|\mathbf{PE}\|$ and the normalized dot product $\frac{\overrightarrow{CE}\cdot\overrightarrow{CP}}{\|CE\|\|CP\|}$. $\mathbf{s_E} = \left(x_{\mathbf{E}}, z_{\mathbf{E}}, x_{\mathbf{P}}, z_{\mathbf{P}}, \|\mathbf{PE}\|, \frac{\overrightarrow{CE}\cdot\overrightarrow{CP}}{\|CE\|\|CP\|}\right)$.
- *Action:* Continuous action $a_{\mathbf{P}} \in [-1, 1]$ representing the speed and direction $\mathbf{E}$ goes along the perimeter. The actual speed of $\mathbf{E}$ is $v_{\mathbf{E}} = a_{\mathbf{E}} R = 10 a_{\mathbf{E}}$ with maximum speed $v_{\mathbf{E}\,\max} = 10$.
- *Reward:* For each step taken, $\frac{\|\mathbf{PE}\|}{2R} - 0.5$ is added as reward. When the distance is too small ($< R$), the reward is negative, so $\mathbf{E}$ should try to get as far from $\mathbf{P}$ as possible. At the end of each round, $\mathbf{E}$ gets +1000 if survives for 15 seconds and gets -1000 if gets caught.

Results when $v_{\mathbf{P}} = 0.8 v_{\mathbf{E}}$:

This means that $\mathbf{P}$ can have speed $4 \sim 8$. Under this case, the game is in favor of $\mathbf{P}$. As in Fig. 10, the reward for $\mathbf{P}$ is always positive. There are a few times for the first 500,000 steps where $\mathbf{E}$ successfully survives, but the probability is low.

Fig. 10. Cumulative Reward, episode length and policy & value losses in training w.r.t. steps for $v_{\mathbf{P}} = 8$. Red is pursuer. Blue is evader.

In inference as shown in Fig. 11, we can see more clearly on how the agents behave. $\mathbf{P}$ is always able to catch $\mathbf{E}$ for the 106 episodes. $\mathbf{P}$ achieves this by maintaining the speed at maximum, and making a positive angle from $\overrightarrow{PE}$, trying to move into the front of $\mathbf{E}$. This behavior makes sense, as this gives a higher chance of successful catch than moving directly towards $\mathbf{E}$. However, $\mathbf{E}$ may dodge a bit at the beginning, causing $\mathbf{P}$ to start in the wrong direction and need to chase up $\mathbf{E}$ from the back. However, since the speed is large and the condition for catching is relaxed, $\mathbf{P}$ can catch $\mathbf{E}$ easily.

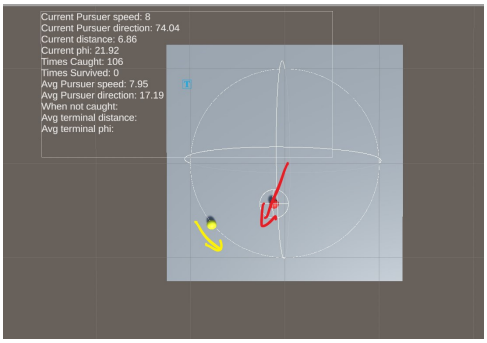

Fig. 11. Inference sample output for $v_{\mathbf{P}} = 0.8 v_{\mathbf{E}}$.

Results when $v_{\mathbf{P}} = 0.5v_{\mathbf{E}}$:

Since this is the inverse of the original problem, there should also be a threshold. When $v_{\mathbf{P}} > v_{th}$, $\mathbf{P}$ can catch $\mathbf{E}$. And when $v_{\mathbf{P}} < v_{th}$, $\mathbf{P}$ cannot catch $\mathbf{E}$. In the $v_{\mathbf{P}} = 0.8v_{\mathbf{E}}$ result, $\mathbf{P}$ is able to catch $\mathbf{E}$, so $0.8 > v_{th}$. It's also interesting to test $v_{\mathbf{P}} < v_{th}$ situation to see what will happen when $\mathbf{P}$ cannot catch $\mathbf{E}$ in 15 seconds. Also, we don't want the speed to be too slow which means $\mathbf{P}$ will behave passively and does not try to catch $\mathbf{E}$. However, the agents don't learn too well under these settings. $v_{\mathbf{P}} = 0.5v_{\mathbf{E}}$ seems to behave the best among the tests, although the training is still not able to converge within 10^6 steps as shown in Fig. 12, but there seems to be a even distribution of winning.

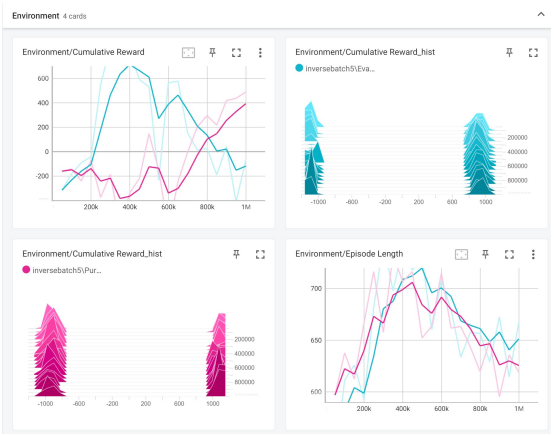

Fig. 12. Cumulative Reward, episode length and policy & value losses in training w.r.t. steps for $v_{\mathbf{P}} = 5$. Red is pursuer. Green is evader.

In inference as shown in Fig. 13, $\mathbf{E}$ has a higher chance to survive. However, $\mathbf{P}$ doesn't perform well in this case, it doesn't try to maximize the speed, only moving at 50% of its maximum speed. $\mathbf{P}$ tries to move into the front of $\mathbf{E}$ as in the previous case, but it is not able to due to its lower speed and is forced to follow behind $\mathbf{E}$. Also, due to the worse behavior of $\mathbf{P}$, $\mathbf{E}$ doesn't learn too well either, it tends to move in the clockwise direction from any starting point, moving as fast as it can to avoid being caught, unless $\mathbf{P}$ approaches it from counter-clockwise direction.

IV. Conclusion

In this project, the "The Lady in the Lake" type pursuit - evasion problem is analyzed and simulated. Game theory provides a toolset for analyzing the dynamic problem. We are able to solve more complicated version of the problem with non-constant controls by the players using Game Theory as compared to analyzing the differential equation itself. The simulation using reinforcement learning also verifies the results got from the numerical analysis.

However, there are limitations in the simulation. Firstly, the size of the agents are non-negligible. We have to enlarge the environment and modify the termination condition to have a better simulation. Collision detection could have been applied for the termination condition. However, it doesn't work well in

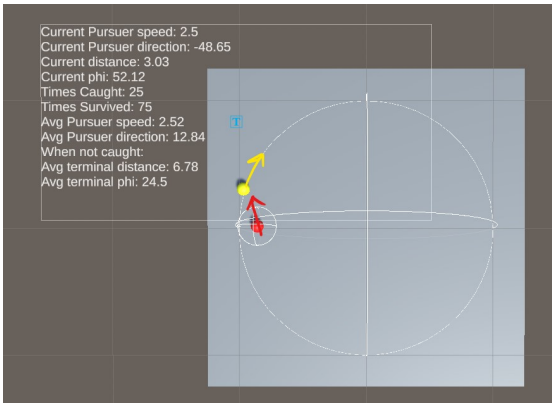

Fig. 13. Inference sample output for $v_{\mathbf{P}} = 0.5v_{\mathbf{E}}$.

my training. The pursuer may not learn the optimal solution, so I only used the distance detection. Also, the potential asynchronous calculation of angles causes the game to be not exactly zero-sum. With some additional testings, it seems that the agents can learn better if the actions are speeds (v_x, v_z) directly instead of velocity + angle. However, it will deviate from the problem formulation and give a non-constant maximum speed.

Regarding the project itself, there are also many points to improve. I was not able to calculate the outcome of the inverse game, because there was little resource to understand how the authors got the outcome $\theta(T)$ for the original game. Secondly, the inverse problem is unsolvable analytically with my setup. There might be a way to describe the system such that the problem is analytically solvable. In the analysis and simulation, there are no information advantages in either party. $\mathbf{P}$ and $\mathbf{E}$ share the same knowledge of the situation. It would be interesting to analyze and simulate the game where one part has an information advantage. Also, the speed of $\mathbf{E}$ in the original problem can be made non-constant as well, similar to what I have done in the inverse problem simulation.

References

[1] T. Basar and G. J. Olsder. Dynamic noncooperative game theory. SIAM Series Classics in Applied Mathematics, 2nd edition, 1999.

[2] Strogatz, Steven. Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering. CRC Press, 2018.

[3] Juliani, A., Berges, V., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D. (2020). Unity: A General Platform for Intelligent Agents. arXiv preprint arXiv:1809.02627. https://github.com/Unity-Technologies/ml-agents.

[4] T. L. VINCENT and W. Y. PENG, Ship collision avoidance. Workshop on differential games, Naval Academy, Annapolis, MO, 1973.

[5] Salimi, Mehdi and Ibragimov, Gafurjan and Ismail, Fudziah. (2010). An application of pursuit-evasion games in a missile guidance system. Advances in Differential Equations and Control Processes. 6.

[6] Parras, J., Zazo, S., del Val, J. et al. Pursuit-evasion games: a tractable framework for antijamming games in aerial attacks. J Wireless Com Network 2017, 69 (2017). https://doi.org/10.1186/s13638-017-0857-8

[7] A. S. Hathaway. Problems and solutions 2801. The American Mathematical Monthly Vol. 28, No. 6/7 (Jun. - Jul., 1921), pp. 91-97, 281-282. https://doi.org/10.2307/2973352

[8] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.