Dynamics between a Fox and a Rabbit

10906976

November 2023

The project focuses on the dynamics between a fox and a rabbit, by solving differential equations that model their positions at different times. The fox starts chasing from the origin, while the rabbit tries to escape from its predator and moves towards its burrow.

1 Question 1

1.1 Polar Coordinate System

Let's introduce polar coordinates (R and ϕ) [1]. Let O be the origin, the x-axis the horizontal line, and the y-axis the direction of the vector $\overrightarrow{OG}$. The rabbit runs at a constant speed of $s_r = 12m/s$ on a circle with a radius of $R = 800$. Assume the rabbit's position at time t is $P_r(r_1(t), r_2(t))$. Therefore, $\phi R = s_r t$, where ϕ is the angle between $P_r O$ and the y-axis.

Clearly,

$$r_1(t) = -Rsin\phi = -Rsin\frac{s_r t}{R}, r_2(t) = Rcos\phi = Rcos\frac{s_r t}{R} \tag{1}$$

The time it takes for the rabbit to run from the initial point (0,800) to the burrow ($800(-sin(\pi/3)$, $cos(\pi/3))$) is $t_f = \phi_{burrow} R/s_r = \pi/3 * 800/12 = 200\pi/9$.

1.2 Analysis of the Fox's and Rabbit's Motion

We can separate the fox's motion into two parts. Firstly, the fox moves from O to G. In the second part, we can further divide the fox's motion into two cases. After the fox passes through G, if the rabbit is in sight, the fox heads straight towards the rabbit. Otherwise, the fox goes straight to A and then directly to the rabbit once it is in sight. In the constant speed condition, there are four stages of the fox's motion, with time nodes t1, t2, t3 and t4 (as shown in Figure 1). The motion of the fox and the rabbit before and at t_1 can be analysed using mathematical methods, as demonstrated in sections 1.2.1 to 1.2.3. However, to study their motion after t_1, it is better to employ code.

1.2.1 The Fox's Motion from O to G

The fox runs at a constant speed of $s_f = 17m/s$. The time it takes for the fox to travel from O to G is $t_1 = |OG|/s_f = 300/17$.

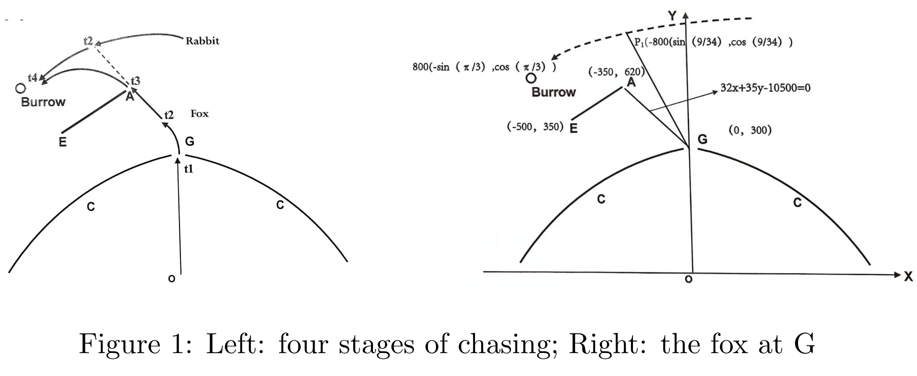

Figure 1: Left: four stages of chasing; Right: the fox at G

1.2.2 The Rabbit's Position at Time t_1

Assume the rabbit's position at time t_1 is P_1 (as shown in Figure 1). Firstly, let's calculate ϕ_1, the ϕ for the point P_1. Clearly, $\phi_1 = s_r t_1/R = 12 \times 300/17/800 = \frac{9}{34}$. Then we can express the polar coordinates of P_1 as $(-Rsin(\phi_1), Rcos(\phi_1)) = (-800sin(\frac{9}{34}), 800cos(\frac{9}{34}))$ using equation (1).

1.2.3 Analysis of whether the Fox can See the Rabbit at Time t_1

As shown in Figure 1, if the line segment P_1G and line segment AE intersect, the fox cannot see the rabbit. Otherwise, the fox can see the rabbit. Now, let's formulate the equation for the line AG. The slope is determined by $\frac{620-300}{-350-0} = -\frac{32}{35}$, thus the equation for this line takes the form

$$32x + 35y + b = 0 \tag{2}$$

By substituting the coordinates of either A or G into equation (2), we can determine that $b = -10500$. Hence, the equation for the line AG is expressed as:

$$32x + 35y - 10500 = 0 \tag{3}$$

By substituting the coordinates of P_1 into equation (3), we observe that the left-hand side of the equation (3) is greater than zero. Thus, P_1 lies above the line AG. Therefore, the line segments P_1G and AE do not intersect, affirming that the fox can see the rabbit when the fox is at G. Consequently, the fox goes on a straight path to the rabbit.

1.2.4 The Fox's Motion after it Passes G

Assume the fox's position coordinates at time t ($t \geq t_1$) are $P_z(z_1(t), z_2(t))$. Then, the ODE representing the fox's motion to be solved should be:

$$\begin{cases} \dfrac{dz_1}{dt} = s_f \cdot \dfrac{r_1(t) - z_1(t)}{\sqrt{(r_1(t) - z_1(t))^2 + (r_2(t) - z_2(t))^2}} \\ \dfrac{dz_2}{dt} = s_f \cdot \dfrac{r_2(t) - z_2(t)}{\sqrt{(r_1(t) - z_1(t))^2 + (r_2(t) - z_2(t))^2}} \end{cases} \tag{4}$$

where $(r_1(t) - z_1(t), r_2(t) - z_2(t))$ represents the direction of the velocity vector of the fox at time t. Then, we determine if

$$(r_1(t) - z_1(t))^2 + (r_2(t) - z_2(t))^2 \leq 0.1 \tag{5}$$

to see if the fox can catch the rabbit. The initial coordinates of P_z are $(z_1(t_1), z_2(t_2)) = (0{,}300)$. By implementing the code, we obtain t_2, the time when the rabbit starts to be blocked by AE after the fox passes G, which is 39.3125. The fox goes on a straight path to point A after t_2, and the time when the fox is at A is $t_3 = 46.4100$. Then, the fox goes directly from A to the rabbit. Finally, we discover that the fox is unable to catch the rabbit, and the total chasing time (depicted as t_4 in Figure 1) is 69.8132. The fox's final position is (-677.8279, 422.3653), having covered a total distance of 1.1868e+03. (Refer to Line 113 to 120 in the appendix for details.)

1.3 Programming

Two functions are introduced for this analysis. The first, named "rpos", calculates the real-time position of the rabbit. The second, named "cantsee", is based on the Mapping Toolbox of MATLAB. It takes the positions of both the rabbit and the fox as inputs and returns true if the fox's view of the rabbit is blocked. (Refer to section 4.1 for details.)

The programming is based on a traversal method, involving the following steps:

1. Initialization: inputs the initial conditions (radius, speeds, $|OG|$, t_1, r(rabbit's position)).

2. ODE Function and Solution: defines the ODE function based on the equation (4) or (9) and uses ode45 to solve the ODE and obtain the fox's position over time.
3. Checking for Rabbit Catch:
   - Checks if the fox catches the rabbit using the condition (see equation (5)).
   - Breaks the loop if the fox can't see the rabbit.
4. Determination of t_2: Determines t_2 when the fox can't see the rabbit after passing G.
5. ODE Solution after Passing A:
   - Determines t_3 and the new time span after the fox passes A.
   - Solves the ODE for the new time span.
6. Checking for Rabbit Catch after Passing A: checks if the fox catches the rabbit after passing A and displays relevant information if the rabbit can't be caught.
7. Plotting: plots the rabbit's and the fox's paths.

Figure 2 illustrates the flow chart depicting the code's execution:

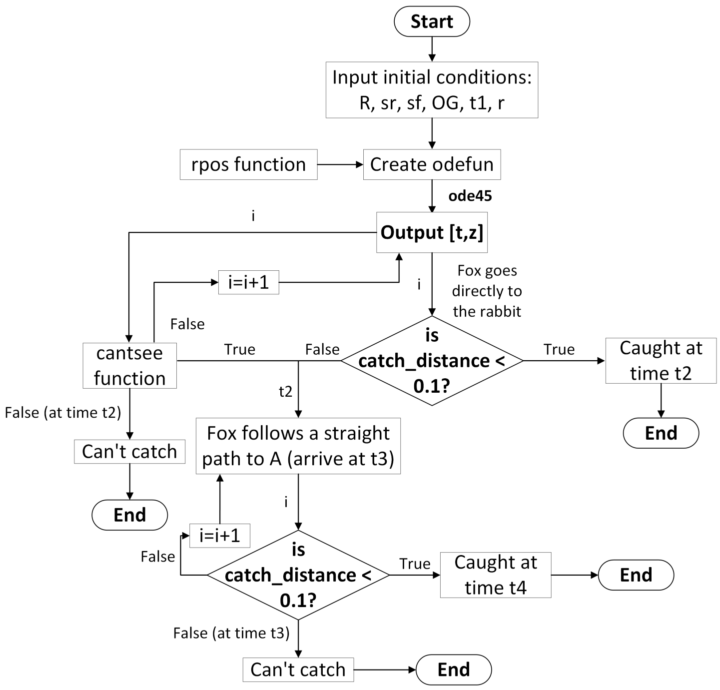

Figure 2: Flow chart depicting the code's execution

2 Question 2

The fox's and the rabbit's speeds diminish in time, given by $s_f(t) = s_{f0}e^{-\mu_f d_f(t)}, s_r(t) = s_{r0}e^{-\mu_r d_r(t)}$, where $s_{f0} = 17$m/s and $s_{r0} = 12$m/s are the initial speeds, $\mu_f = 0.0002m^{-1}$ and $\mu_r = 0.0008m^{-1}$ are the rates of the diminishing speeds, and $d_f(t)$ and $d_r(t)$ are the distance they have travelled up to time t.

For the rabbit,

$$\frac{d}{dt}(d_r(t)) = s_r(t) = s_{r0}e^{-\mu_r d_r(t)} \tag{6}$$

By integrating both sides of equation (6), we get $\int_0^{d_r(t)} e^{\mu_r d_r(t)} d(d_r(t)) = \int_0^t s_{r0} dt$. Solving the integral, we obtain $\frac{1}{\mu_r}(e^{\mu_r d_r(t)} - 1) = s_{r0}t$. Hence,

$$d_r(t) = \frac{1}{\mu_r} log(\mu_r s_{r0} t + 1) \tag{7}$$

Therefore, in the diminishing speed condition, the time it takes for the rabbit to run from the initial point to the burrow is $T_f = \frac{1}{\mu_r s_{r0}}(e^{\mu_r \frac{\pi}{3} R} - 1)$. By substituting equation (7) to equation (6), the rabbit's speed at time t is $s_r(t) = s_{r0} e^{-\mu_r \frac{1}{\mu_r} log(\mu_r s_{r0} t+1)} = \frac{s_{r0}}{\mu_r s_{r0} t+1} = \frac{12}{0.0096t+1}$. Additionally, we can obtain the polar coordinates (denoted as $P_R(R_1(t), R_2(t))$) of the rabbit's position at time t, where

$$R_1(t) = -Rsin(\frac{1}{\mu_r R} log(\mu_r s_{r0} t + 1)), R_2(t) = Rcos(\frac{1}{\mu_r R} log(\mu_r s_{r0} t + 1)) \tag{8}$$

Similarly, for the fox, the fox's speed at time t is $s_f(t) = \frac{s_{f0}}{\mu_f s_{f0} t+1} = \frac{17}{0.0034t+1}$, and the fox's distance up to time t is $d_f(t) = \frac{1}{\mu_f} log(\mu_f s_{f0} t + 1)$. The time it takes for the fox to travel from O to G is $t_1 = \frac{1}{\mu_f s_{f0}}(e^{\mu_f |OG|} - 1) = \frac{1}{0.0034}(e^0.06 - 1) \approx 18.1872$. By substituting t_1 to equation (8), the rabbit's position at time t_1 should be approximately (-75.7148, 796.409). Using the same analysis method as in Figure 1, because the rabbit's position in this scenario is to the right of that in the constant speed condition, the fox can undoubtedly see the rabbit when the fox is at G. After that, the fox goes on a straight path to the rabbit, and the new ODE to be solved should be

$$\begin{cases} \frac{dz_1}{dt} = \frac{s_{f0}}{\mu_f s_{f0} t + 1} \cdot \frac{r_1(t) - z_1(t)}{\sqrt{(r_1(t) - z_1(t))^2 + (r_2(t) - z_2(t))^2}} \\ \frac{dz_2}{dt} = \frac{s_{f0}}{\mu_f s_{f0} t + 1} \cdot \frac{r_2(t) - z_2(t)}{\sqrt{(r_1(t) - z_1(t))^2 + (r_2(t) - z_2(t))^2}} \end{cases} \tag{9}$$

Under the diminishing speed condition, by implementing the code, we find that the fox can always see the rabbit after the fox passes G. Finally, the rabbit is caught at time 70.0186 at the position (-575.7344, 555.4546). The total distance travelled by the fox is 1.0677e+03. (Refer to Line 202 to Line 209 in the appendix for details.) Figure 3 shows the paths of both the rabbit and the fox under constant speed (on the left) and diminishing speed (on the right) conditions:

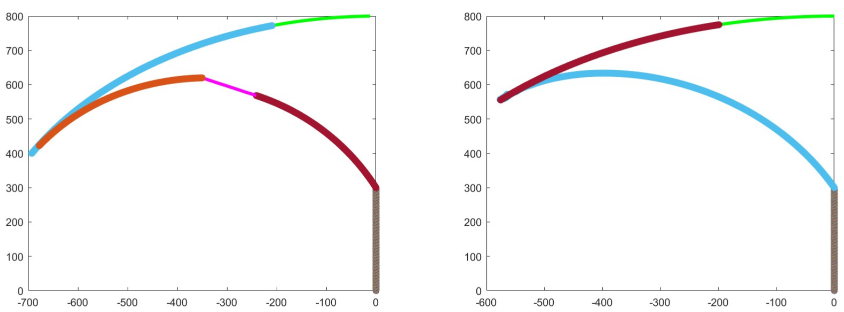

Figure 3: The rabbit's and fox's paths under constant speed (left) and diminishing speed (right) conditions

3 References

[1] StackExchange, mathematics questions, "Fun calculus problem I can't seem to solve", last accessed 3 November 2023. [Online]. Available:
`https://math.stackexchange.com/questions/40139/fun-calculus-problem-i-cant-seem-to-solve`

4 Appendix: MATLAB Code

4.1 Two Functions

```
1 %t=time; axis: 1->x-axis, 2->y-axis; stage=question number (1 or 2)
```

```
function res = rpos(t,axis,stage)
R = 800;
s0 = 12;
if stage == 1
    if axis == 1
        res = -R*sin(s0*t/R);
    else
        res = R*cos(s0*t/R);
    end
else
    mu_r = 0.0008;
    if axis == 1
        res = -R*sin(1/mu_r/R*log(mu_r*s0*t+1));
    else
        res = R*cos(1/mu_r/R*log(mu_r*s0*t+1));
    end
end
%Inputs consist of the positions of the rabbit and the fox
function res = cantsee(r_x,r_y,f_x,f_y)
a_x = -350;
a_y = 620;
e_x = -500;
e_y = 350;
line1 = [r_x,r_y;f_x,f_y];
line2 = [a_x,a_y;e_x,e_y];
%Return the intersection points of two polylines
[p_x,p_y] = polyxpoly(line1(:,1),line1(:,2),line2(:,1),line2(:,2));
%If line1 and line2 intersect, the function 'cantsee' returns true
if isempty(p_x)
    res = false;
else
    res = true;
end
```

4.2 Question 1

```
R = 800;
s_r = 12;
s_f = 17;
OG = 300;
t_1 = OG/s_f;
tspan0 = 1:0.0001:t_1;
r0 = [-R*sin(s_r*tspan0/R); R*cos(s_r*tspan0/R)];
plot(r0(1,:),r0(2,:),'green',LineWidth=3);
hold on
x = 0;
y = linspace(0,300,10000);
plot(x,y,'-o');
TF = R*pi/3/12;
tspan = t_1:0.0001:TF;%Discretise the time
%The real-time position of the rabbit
r = [-R*sin(s_r*tspan/R); R*cos(s_r*tspan/R)];
plot(r(1,:),r(2,:),'-o');
%Define the ODE function as specified in equation (4)
```

```
odefun = @(t,z) [(s_f*(rpos(t,1,1)-z(1))/sqrt((rpos(t,1,1)-z(1))^2
+(rpos(t,2,1)-z(2))^2));(s_f*(rpos(t,2,1)-z(2))/sqrt((rpos(t,1,1)-z(1))^2
+(rpos(t,2,1)-z(2))^2))];
%Use ode45 to return an array of solutions (z: the fox's position) and a
%column vector of evaluation points (t: time)
[t,z] = ode45(odefun,tspan,[0 300]);
%Determine if the fox can catch the rabbit
for i = 1:size(t,1)
    catch_distance = sqrt((r(1,i) - z(i,1))^2+(r(2,i) - z(i,2))^2);
    if catch_distance < 0.1
        disp("Caught at time t2");
        disp("Time:");
        disp([t_1 t(i)]);
        disp("Place:");
        disp([r(1,i) r(2,i)]);
        return;
    end
    if cantsee(r(1,i),r(2,i),z(i,1),z(i,2))
        break;
    end
end
t_2 = t(i);
plot(z(1:i,1),z(1:i,2),'-o');
if ~cantsee(r(1,i),r(2,i),z(i,1),z(i,2))
    disp("Can't catch it...");
    disp("Time:");
    disp([t_1 t_2]);
    disp("Fox's position:");
    disp([z(i,1) z(i,2)]);
    return;
end
t_3 = t_2 + sqrt((350+z(i,1))^2 + (620-z(i,2))^2)/s_f;
tspan2 = t_3:0.0001:TF;
r2 = [-R*sin(s_r*tspan2/R); R*cos(s_r*tspan2/R)];
[t2,z2] = ode45(odefun,tspan2,[-350 620]);
z_x = [z(i,1) z2(1,1)];
z_y = [z(i,2) z2(1,2)];
plot(z_x,z_y,LineWidth=3,Color='magenta');
plot(z2(:,1),z2(:,2),'-o');
hold off
%Determine if the fox can catch the rabbit after the fox passes A
for i = 1:size(t2,1)
    catch_distance = sqrt((r2(1,i) - z2(i,1))^2+(r2(2,i) - z2(i,2))^2);
    if catch_distance < 0.1
        disp("Caught at time t4");
        disp("Time:");
        disp([t_1 t_2 t_3 t2(i)]);
        disp("Place:");
        disp([r2(1,i) r2(2,i)]);
        return;
    end
end
fox_total_distance = s_f*t2(i);
```

```
disp("Can't catch it...");
disp("Time:");
disp([t_1 t_2 t_3 TF]);
disp("Fox's position:");
disp([z2(i,1) z2(i,2)]);
disp("Fox's distance:");
disp(fox_total_distance);
>> const
Can't catch it...
Time:
   17.6471   39.3125   46.4100   69.8132
Fox's position:
  -677.8279  422.3653
Fox's distance:
   1.1868e+03
```

4.3 Question 2

```
R = 800;
s_r0 = 12;
s_f0 = 17;
mu_r = 0.0008;
mu_f = 0.0002;
OG = 300;
t_1 = 1/(mu_f*s_f0)*(exp(mu_f*OG)-1);
tspan0 = (0:0.0001:t_1)';
r0 = zeros(2,size(tspan0,1));
for i = 1:size(tspan0,1)
    r0(1,i) = rpos(tspan0(i),1,2);
    r0(2,i) = rpos(tspan0(i),2,2);
end
plot(r0(1,:),r0(2,:),'green',Linewidth=3);
hold on
x = 0;
y = linspace(0,300,10000);
plot(x,y,'-o');
TF = (exp(mu_r*R*pi/3)-1)/mu_r/s_r0;
tspan = (t_1:0.0001:TF);
r = [-R*sin(1/mu_r/R*log(mu_r*s_r0*tspan+1)); R*cos(1/mu_r/R
*log(mu_r*s_r0*tspan+1))];
odefun = @(t,z) [(s_f0/(mu_f*s_f0*t+1)*(rpos(t,1,2)-z(1))/sqrt((rpos(t,1,2)
-z(1))^2+(rpos(t,2,2)-z(2))^2));(s_f0/(mu_f*s_f0*t+1)*(rpos(t,2,2)-z(2))
/sqrt((rpos(t,1,2)-z(1))^2+(rpos(t,2,2)-z(2))^2))];
[t,z] = ode45(odefun,tspan,[0 300]);
for i = 1:size(t,1)
    catch_distance = sqrt((r(1,i) - z(i,1))^2+(r(2,i) - z(i,2))^2);
    if catch_distance < 0.1
        disp("Caught at time t2");
        disp("Time:");
        disp([t_1 t(i)]);
        disp("Place:");
        disp([r(1,i) r(2,i)]);
        plot(z(1:i,1),z(1:i,2),'-o');
        plot(r(1,(1:i)),r(2,(1:i)),'-o');
```

```
        fox_total_distance = 1/mu_f*log(mu_f*s_f0*t(i)+1);
        disp("Fox's distance:");
        disp(fox_total_distance);
        return;
    end
    if cantsee(r(1,i),r(2,i),z(i,1),z(i,2))
        break;
    end
end
t_2 = t(i);
if ~cantsee(r(1,i),r(2,i),z(i,1),z(i,2))
    disp("Can't catch it...");
    disp("Time:");
    disp([t_1 t_2]);
    disp("Fox's position:");
    disp([z(i,1) z(i,2)]);
    return;
end
t_3 = t_2 + 1/(mu_f*s_f0)*(exp(mu_f*sqrt((350+z(mid,1))^2
+ (620-z(mid,2))^2))-1);
tspan2 = t_3:0.0001:TF;
r2 = [-R*sin(1/mu_r/R*log(mu_r*s_r0*tspan2+1)); R*cos(1/mu_r/R
*log(mu_r*s_r0*tspan2+1))];
[t2,z2] = ode45(odefun,tspan2,[-350 620]);
z_x = [z(i,1) z2(1,1)];
z_y = [z(i,2) z2(1,2)];
plot(z_x,z_y,LineWidth=3,Color='magenta');
plot(z2(:,1),z2(:,2),'-o');
hold off
for i = 1:size(t,1)
    catch_distance = sqrt((r2(1,i) - z2(i,1))^2+(r2(2,i) - z2(i,2))^2);
    if catch_distance < 0.1
        disp("Caught at time t4");
        disp("Time:");
        disp([t_1 t_2 t_3 t2(i)]);
        disp("Place:");
        disp([r2(1,i) r2(2,i)]);
        return;
    end
end
disp("Can't catch it...");
disp("Time:");
disp([t_1 t_2 t_3 TF]);
disp("Fox's position:");
disp([z2(i,1) z2(i,2)]);
>> change
Caught at time t2
Time:
   18.1872   70.0186
Place:
 -575.7344  555.4546
Fox's distance:
  1.0677e+03
```