AE 5222 – Optimal Control of Dynamical Systems

Homework Submission Cover Page and Statement of Academic Honesty

I, John O'Neill, submit the solution to Homework Problem 4.

My signature below affirms that all of the writing in this submission is my own work. Any reference material that I used to prepare this submission, including text or video resources, but excluding the lecture notes and videos provided on the Canvas site for this course, is properly cited.

To prepare this submission:

■ I verbally collaborated with the following individuals (excluding *Piazza* discussions):

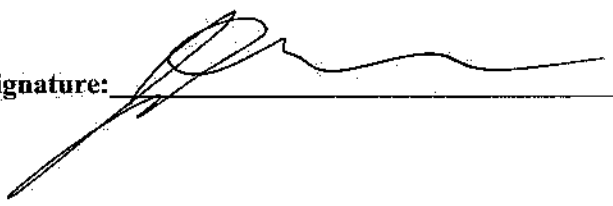
Currently enrolled in AE 5222: Evan Kelly

Not currently enrolled in AE 5222:

☐ I did not verbally collaborate with any other individual.

This submission reflects my individual effort and my own understanding of the course content.

I have read and I understand WPI's Academic Honesty Policy, and my conduct in preparing this submission has been in accordance with this Policy.

Signature:  **Date:** 04/28/2019

Jack O'Neill
AE 5222
Homework 11

Method

The first step to solving this problem was to determine the differential equation describing the heading angle Ψ. To do this I needed to go through the process of determining the Hamiltonian equation, then using the necessary conditions of the system to solve for the equation for heading angle. The Hamiltonian equation is shown below:

$$H(...) = \left[c_{off} + \sum_{n=1}^{25} exp(-\tfrac{1}{2v_n}((x-\bar{x})^2 + (y-\bar{y})^2)) \right] + p_1 V cos(\psi) + p_2 V sin(\psi)$$

After defining the Hamiltonian I could then define necessary conditions in order to determine the differential equation describing the heading angle.

$$\frac{d}{dt}p_1^*(t) = -\frac{\partial H}{\partial x} = \sum_{n=1}^{25} \frac{\theta_n(x-\bar{x})}{\sqrt{2\pi v_n}\cdot v_n} exp(-\frac{1}{2v_n}((x-\bar{x})^2 + (y-\bar{y})^2))$$

$$\frac{d}{dt}p_2^*(t) = -\frac{\partial H}{\partial y} = \sum_{n=1}^{25} \frac{\theta_n(y-\bar{y})}{\sqrt{2\pi v_n}\cdot v_n} exp(-\frac{1}{2v_n}((x-\bar{x})^2 + (y-\bar{y})^2))$$

$$p_1^*(t) = -\frac{c_{off} + \sum_{n=1}^{25} exp(-\frac{1}{2v_n}((x-\bar{x})^2+(y-\bar{y})^2))}{V cos(\psi) + V\ tan(\psi)\ sin(\psi)}$$

$$p_2^*(t) = -tan(\psi)\frac{c_{off} + \sum_{n=1}^{25} exp(-\frac{1}{2v_n}((x-\bar{x})^2+(y-\bar{y})^2))}{V cos(\psi) + V\ tan(\psi)\ sin(\psi)}$$

Using these equations I could then solve for the differential equation for the optimal control of Ψ:

$$\frac{d}{dt}\psi^*(t) = V\frac{\frac{\partial c}{\partial y}cos(\psi^*(t)) - \frac{\partial c}{\partial x}sin(\psi^*(t))}{c_{off} + \sum_{n=1}^{25} exp(-\frac{1}{2v_n}((x-\bar{x})^2 + (y-\bar{y})^2))}$$

Where $\frac{\partial c}{\partial x}$ and $\frac{\partial c}{\partial y}$ are defined as:

$$\frac{\partial c}{\partial x} = \sum_{n=1}^{25} \frac{\theta_n(x-\bar{x})}{\sqrt{2\pi v_n}\cdot {v_n}^3} exp(-\frac{1}{2v_n}((x-\bar{x})^2 + (y-\bar{y})^2))$$

$$\frac{\partial c}{\partial y} = \sum_{n=1}^{25} \frac{\theta_n(y-\bar{y})}{\sqrt{2\pi v_n}\cdot {v_n}^3} exp(-\frac{1}{2v_n}((x-\bar{x})^2 + (y-\bar{y})^2))$$

Now that all three state equations are known, the “last step” in determining the optimal control input of this system is to determine the initial heading angle Ψ_0.

To determine the initial heading angle for the optimal control I solved the three ODE’s for the initial coordinates of (-1,-1) and swept the initial heading angle from 0 to 2π. I generated a circle of starting radius 0.2 centered about the desired final position (1,1). For any path that traveled through this circle I designated that particular initial heading angle in a list called “new_repeat_list” shown in the code below:

```
71 -  for y = 1:length(states_out) % Test whether the trajectory crosses check radius
72 -      if (norm([states_out(y,1) states_out(y,2)] - [1 1]) < check_radius)...
73                && (~ismember(current_psi_0,new_repeat_list))
74 -          new_repeat_list = [new_repeat_list current_psi_0];
75
76 -          path_cost = 0;
77 -          for i = ceil(linspace(1,length(states_out(:,1)),50))
78 -              for j = ceil(linspace(1,length(states_out(:,2)),50))
79 -                  path_cost = path_cost + c(states_out(i,1),states_out(j,2));
80 -              end
81 -          end
82 -          cost_list       = [cost_list path_cost];
83 -          break;
84 -      end
85 -  end
```

Lines 76 through 82 show the process of gathering path cost data. For any path which crosses through the circle the code will also read 50 points along the path and add up the costs at each point. Doing so provides allows me to compare how well each of the candidate initial headings perform relative to the other candidates.

Once the code finishes the sweep it divides the circle's radius by 2, and defines a new range of initial heading angles to sweep. There were a number of initial heading angles which resulted in paths crossing the circle, so I manually checked which of them resulted in the smallest cost. I then redefined the initial heading span to focus on the angle which provides the most optimal path. This means that I restarted the program with the newly defined initial heading span, with the first iteration of this process shown below:

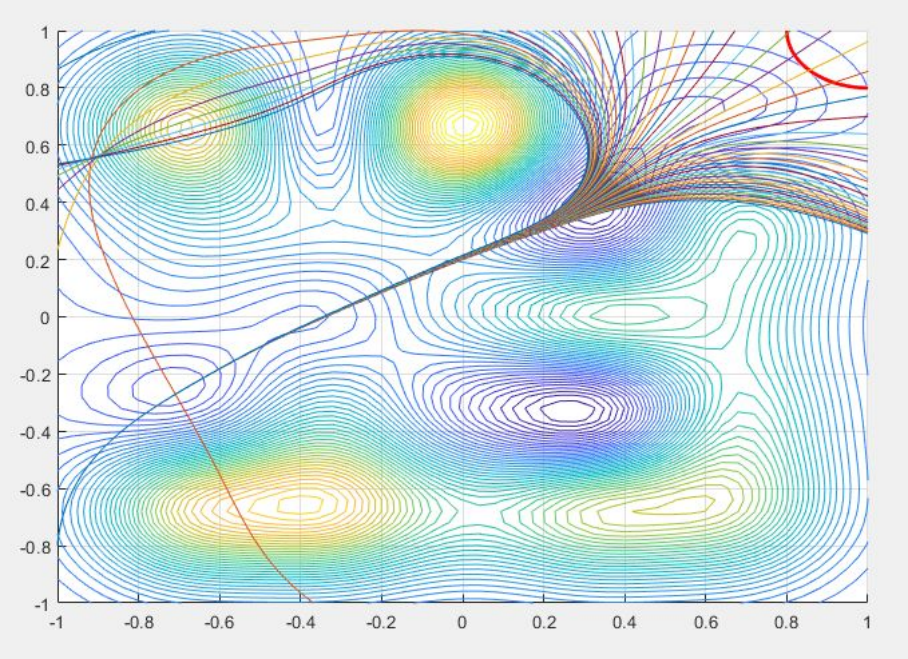


Figure 1: Iteration 1

It is apparent that a number of initial heading angles will cross the circle. Now that this iteration is complete, the script defines a new radius of 0.1 and redefines the initial heading span to only include the headings which passed through the circle. It will also increase the resolution between each initial heading angle of the span. The second iteration of this process is shown below:

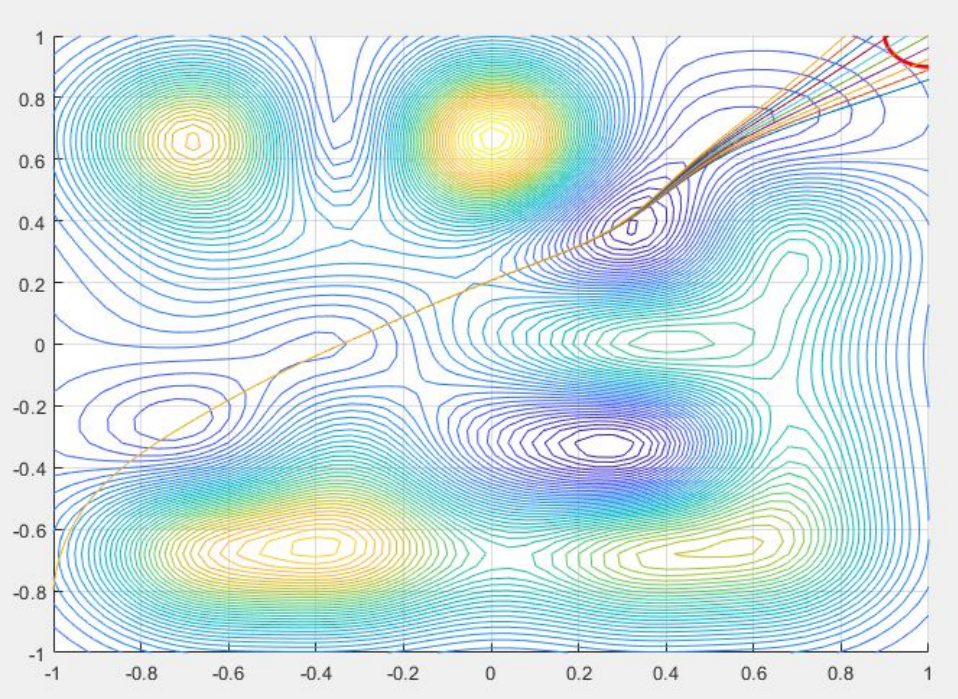


Figure 2: Iteration 2

At around this point I ran into an issue where the code would not detect the paths passing through the circle even when they could visually be seen passing through it. After a closer look at the code I realized that the ODE solver sampled distances greater than that of the diameter of the circle, so even if the path passed through it the actual sample points may not have been inside the circle. This issue was solved by forcing the step size of the ODE solver to always be less than the radius of the circle. The code for this is shown below:

```
t          = t0; % initialize t
step_size = check_radius/3;
n_steps    = tf/step_size;
tspan      = t0:step_size:tf;             % Total time span
currentStates = [pos_0 current_psi_0]'; % State array used inside

for k = 1:n_steps
    t1 = step_size*(k-1);
    t2 = step_size*k;
    temp_tspan = t1:(t2-t1)/nip:t2;
    [tNew,tempStates] = ode45(@(t,y) homework_11_ode...
        (y,V,psi_dot),...
        temp_tspan,currentStates);
    t(k) = t2;
    currentStates = tempStates(nip+1,1:3)';
    states_out(k,:) = currentStates';
end
```

The following figure shows this in action during the third iteration, when the circle radius becomes 0.05. The points are individually shown by squares:

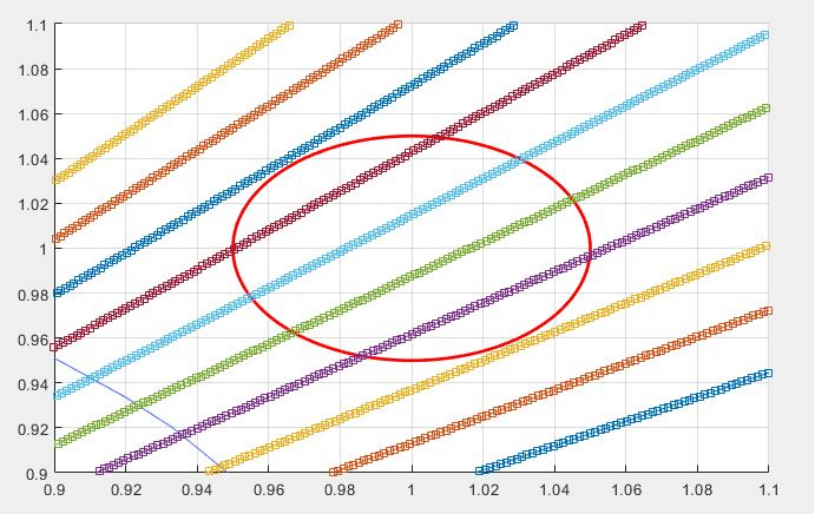


Figure 3: Solution to step size problem visualized in the third iteration

After solving this issue, the script could then run many iterations to gain an increasingly precise number for the initial heading angle. In my code I opted to break out of the iterative cycle after six iterations. This gave me a sufficiently precise value for the initial heading angle.

Results

My calculated initial heading angle is equal to $\mathbf{\Psi_0 = 92.86190}$ **degrees**. The path at this initial heading angle looks like this:

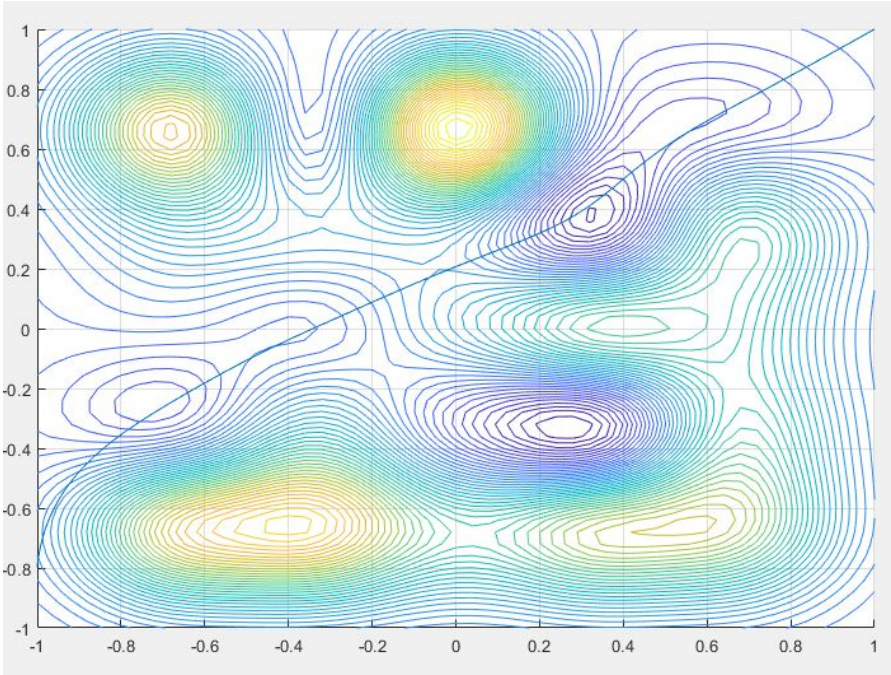


Figure 4: Optimal path

It reaches the final position at t=29.65695 seconds. The heading angle plotted over time is shown below:

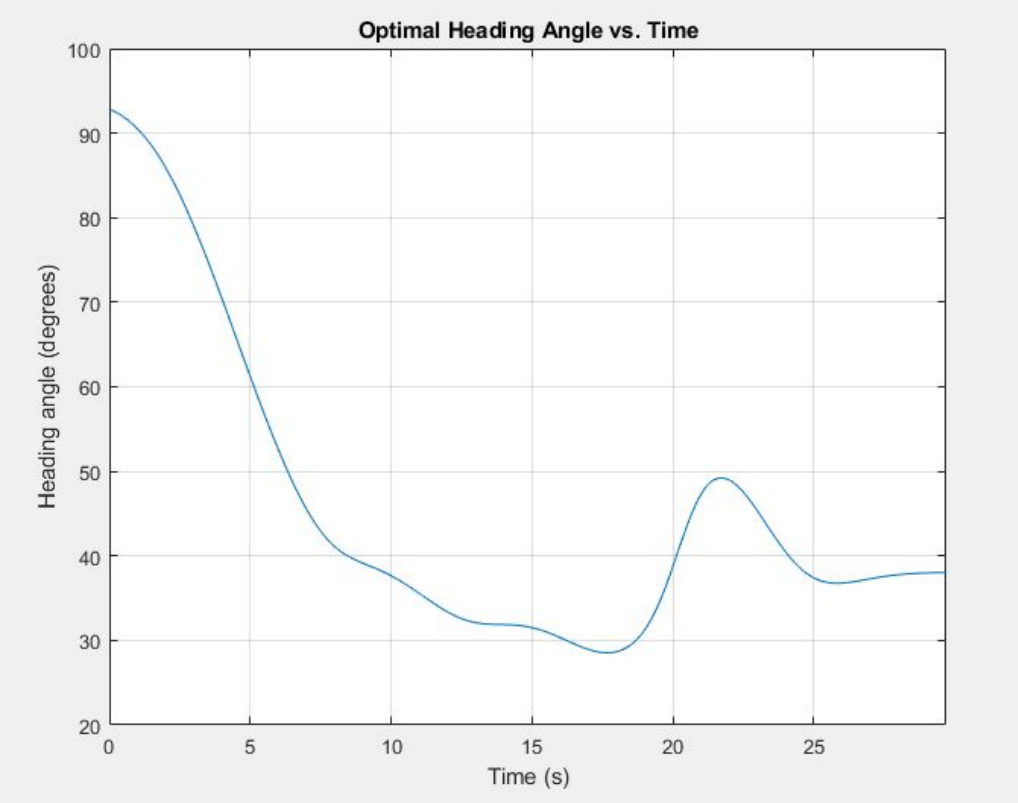


Figure 5: Heading angle plotted over time

Discussion

This unique problem proved to be quite the challenge, albeit a rewarding one. I believe my final calculated initial heading angle is correct since I am confident in the differential equation derived for the heading angle. The process to actually find the initial heading angle which provides the optimal path seemed to work out and was able to give me a precise value for the initial heading angle so I am happy with this result.