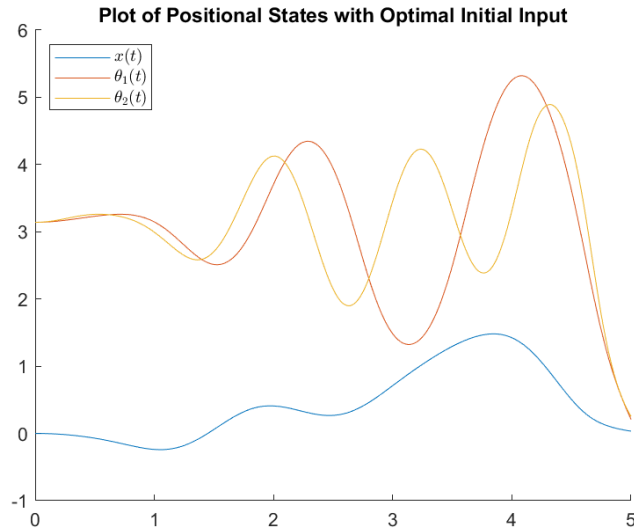


MAE 200 Final project

Matthew Stringer

Step 1: Computing $u(t)$ on $t \in [0, T]$

For step 1, we must use adjoint-based optimization in order to compute the optimal input u_k . To do this, we use the provided code from Numerical Renaissance to calculate a more optimal input given a previous "guess" input. In order to get a satisfactory input, 2 things were done: First, an initial optimal input was generated given no input at all. Second, several more inputs were generated based on the previous computed optimal input. This results in a more optimal input. While generating inputs, I found that it was easier to generate satisfactory system responses when given a larger time horizon of 5 seconds rather than the default 3 second time horizon. This program is contained in `Dual_Pendulum_Input.m`



Step 2

Step 3

Step 4: State Estimation based on α -horizon

I began with constructing my model of my system based on the linearized model around equation 22.34 of Numerical Renaissance.

Since E is invertible around $\vec{q} = \vec{0}$, we can solve for the A and B matrices from the standard form,

$$\dot{q} = Aq + Bu,$$

by inverting the E matrix.

After inputting these matrices into matlab, we are left with the following system:

```
sys =  
A =  
      x1      x2      x3      x4      x5      x6  
x1      0      0      0      1      0      0  
x2      0      0      0      0      1      0  
x3      0      0      0      0      0      1  
1
```

x4	0	0.491	0.982	0	0	0
x5	0	5.175	0.9428	0	0	0
x6	0	0.9428	20.7	0	0	0

B =

	u1
x1	0
x2	0
x3	0
x4	0.1044
x5	0.1002
x6	0.2004

C =

	x1	x2	x3	x4	x5	x6
y1	1	0	0	0	0	0
y2	0	1	0	0	0	0
y3	0	0	1	0	0	0

D =

	u1
y1	0
y2	0
y3	0

Continuous-time state-space model.

Setting our Q matrix to 1, and our R matrix to 1, we create a Kalman filter using the `kalman` function. This results in an L matrix

L =

0.4575	0.3394	0.3830
0.3394	4.5172	0.2583
0.3830	0.2583	9.0800
0.2356	0.9622	1.9510
0.8249	10.2934	1.8422
1.7895	1.7995	41.3303

Step 5: Optimal Control

Using the linearized model described in Step 5, it is possible to utilize MATLAB's `lqr` function. Since we primarily care about the positional states of our system, we set their weights to be twice as large as their derivatives.

```
Q = diag([1 1 1 0.5 0.5 0.5]);
```

Since we only have one input, we can set its weight to 1.

```
R = 1;
```