Robot Learning and Sensorimotor Control Course Assignment

Siyuan Wu s2032669

March 2020

1 Part 1

1.1

$$\mathbf{x} = \begin{bmatrix} x \\ z \\ \theta \\ \dot{x} \\ \dot{z} \\ \dot{\theta} \end{bmatrix}$$

$$\mathbf{u} = [u_1, u_2]^\intercal$$

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\sin\theta}{m} & \frac{\sin\theta}{m} \\ \frac{\cos\theta}{m} & \frac{\cos\theta}{m} \\ -\frac{L}{I_{yy}} & \frac{L}{I_{yy}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ 0 \\ -g \\ -\dot{\theta}^2 \end{bmatrix}$$

so we get $\mathbf{A}$

$$\mathbf{A} = \frac{\partial f(x_0, u_0)}{\partial x} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{u_1+u_2}{m}\cos\theta & 0 & 0 & 0 \\ 0 & 0 & -\frac{u_1+u_2}{m}\sin\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2\dot{\theta} \end{bmatrix}$$

and $\mathbf{B}$

$$\mathbf{B} = \frac{\partial f(x_0, u_0)}{\partial u} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\sin\theta}{m} & \frac{\sin\theta}{m} \\ \frac{\cos\theta}{m} & \frac{\cos\theta}{m} \\ -\frac{L}{I_yy} & \frac{L}{I_yy} \end{bmatrix}$$

Around the fixed point $x_0 = [0, 0, 0, 0, 0, 0]^\intercal$, $u_0 = [mg/2, mg/2]^\intercal$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & g & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

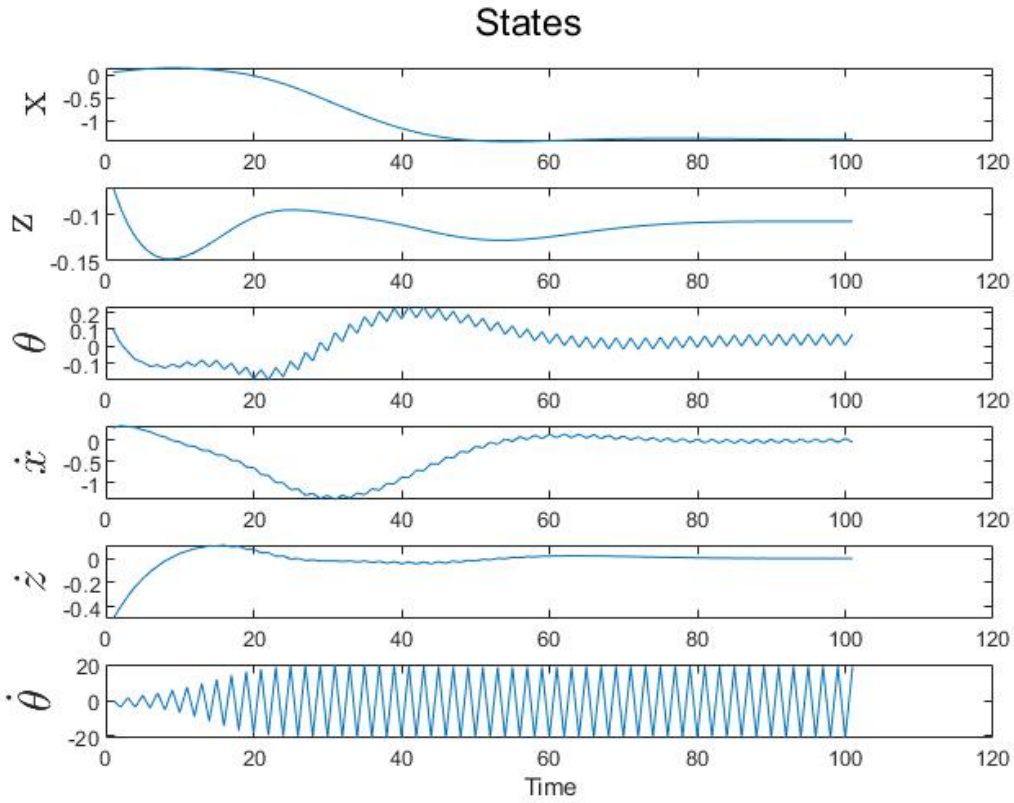
$$\mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & \frac{1}{m} \\ -\frac{L}{I_{yy}} & \frac{L}{I_{yy}} \end{bmatrix}$$

1.2

The optimal gain matrix $\mathbf{K}$ for the infinite horizon LQR is

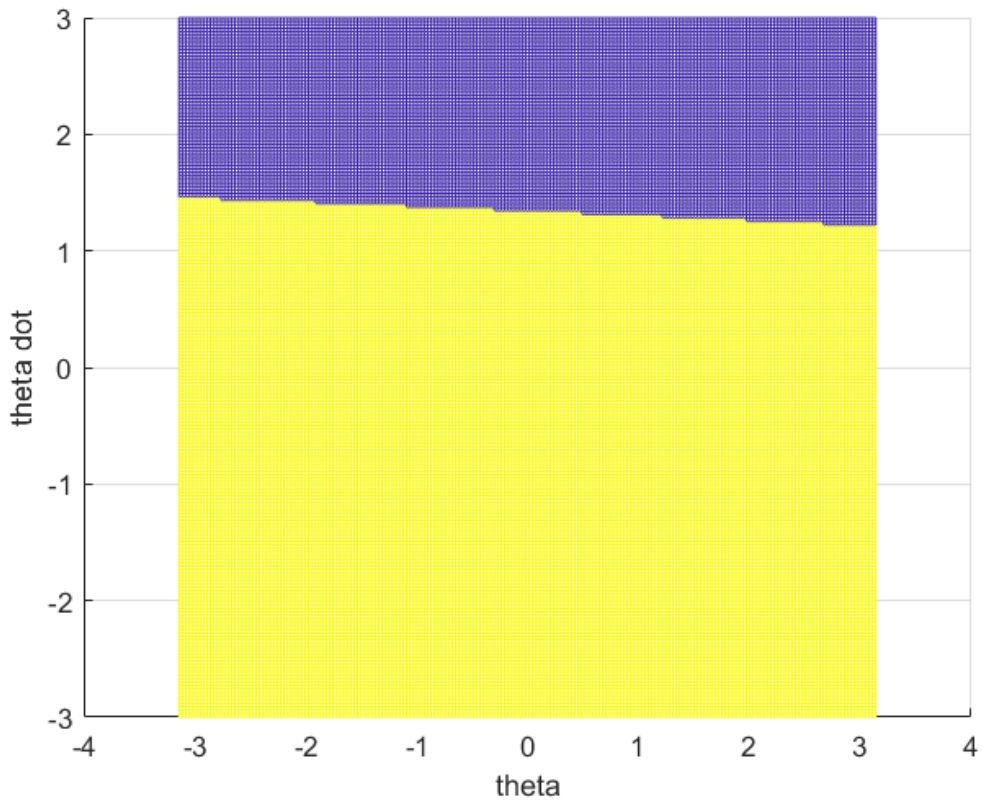
$$\mathbf{K} = \begin{bmatrix} -2.2361 & 2.2361 & -5.8266 & -1.7773 & 1.6541 & -0.8562 \\ 2.2361 & 2.2361 & 5.8266 & 1.7773 & 1.6541 & 0.8562 \end{bmatrix}$$

1.3



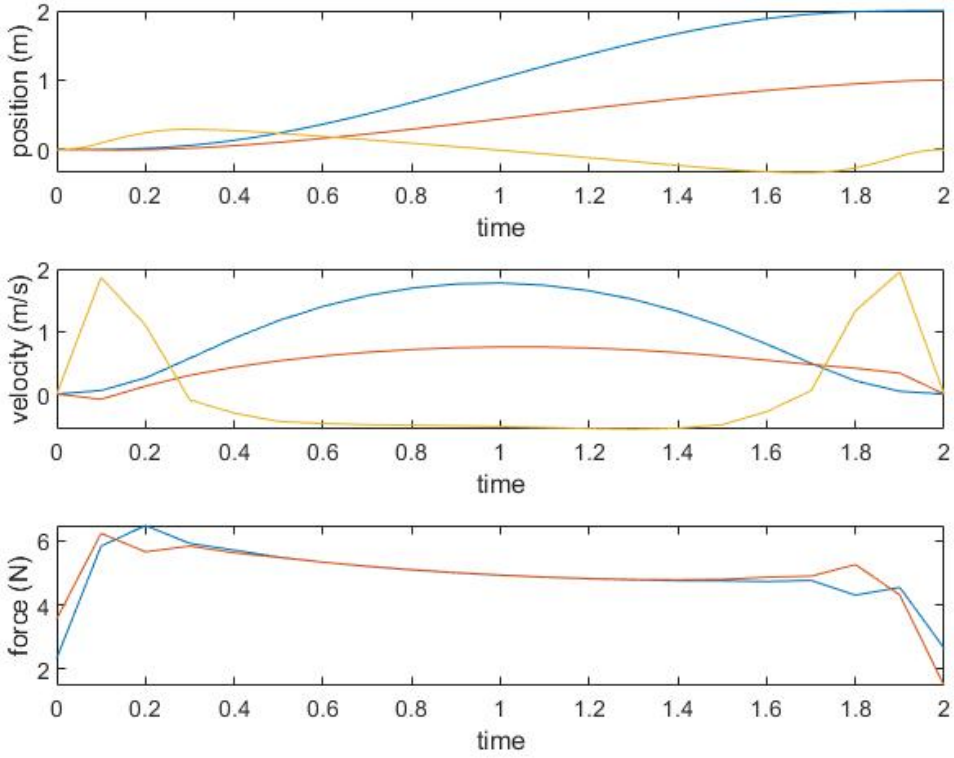
1.4

The yellow part in the following graph is the estimate of region of attraction around the fix point.



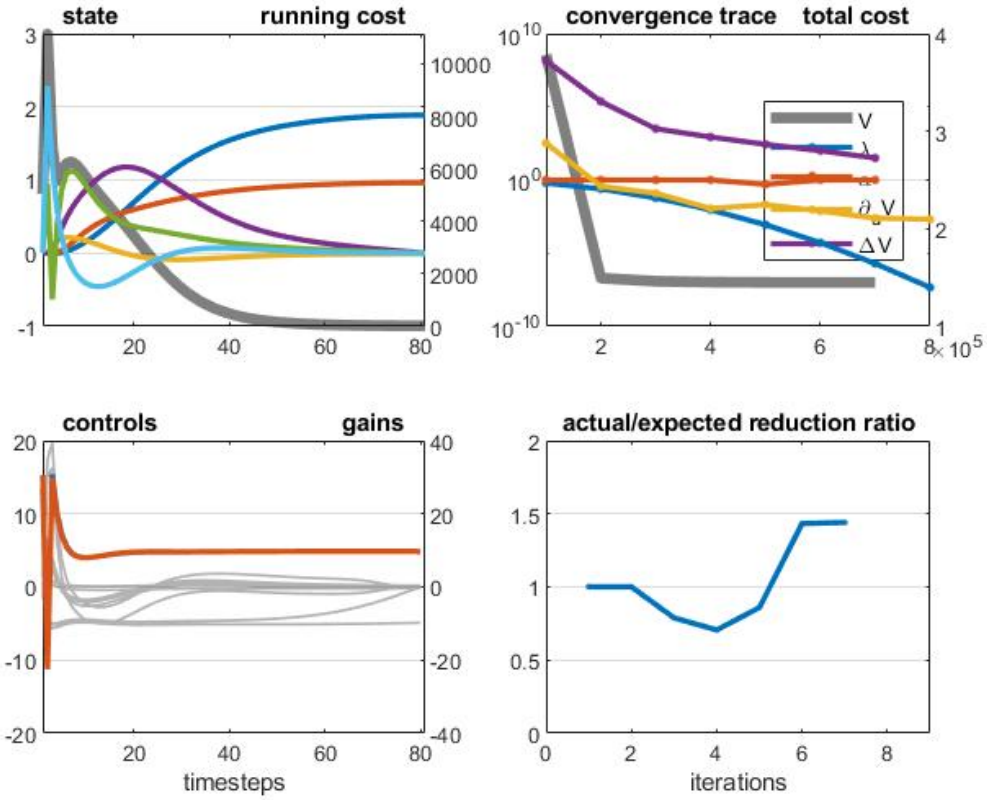
2 Part 2

Plot state and input trajectory results by SQP



3 Part 3

Plot state and input trajectory results by DDP



4 Part 4

4.1

For infinite horizon LQR, it can only plan trajectory for linear system. For states around the fixed points, the system can be approximated as a linear system, so LQR can robust control them. For trajectory planned with SQP and DDP, the states in these two problems are not around the fixed points, so it cannot be planned using infinite horizon LQR.

4.2

No. Because both of these two methods use approximations to apply the dynamics for computing the optimal trajectory. In SQP, forward dynamics are encoded using trapezoidal integration to the non-linear equalities. In DDP, the dynamics are used in the integral form $y = x + dy = x + h\dot{x}$

4.3

4.3.1

In SQP, the equations of motion are defined as the non-linear equalities in the non-linear programming problem. The solver has to satisfy all the constraints, including all the equalities and inequalities. So the solver is enforced to obey the equations of motion.

In DDP, the forwards pass can compute the new state using the equations of motion. The solver is enforced to obey the equations of motion by using them to compute the optimal trajectories in the forwards pass.

4.3.2

In SQP, the initial and desired final states are defined as the non-linear equalities of the non-linear problem. In Q2 program, it's in the function

`dynamicConstraintQuadrotorSystem.m`

In DDP, the initial states are defined as the initial value of the forwards and backwards pass. The desired final states are defined in the cost function, for which the final states of the optimal trajectory is x_d in $\sum_{k=0}^{t_f} u_k^\intercal R u_k + [x_{tf} - x_d]^\intercal Q_f [x_{tf} - x_d]$.

4.3.3

The computational cost in SQP much higher than DDP. In SQP, the dimension of non-linear equalities is nm, in which n is the dimension of the problem and m is the number of time steps. For each step, SQP need to compute the 1st-order and 2nd-order partial derivatives of the cost function, and the 1st-order partial derivative of the equality constraints. This step has large computational cost. So the computation complexity is $O(mn + n^2)$

But for DDP, each step the solver only need to compute the 2nd-order partial derivatives of u of the Q function, which has the dimension of n, and forwards pass the u to get next step. So the computation complexity is $O(n)$.

In this way, the SQP algorithm is computational expensive.