AEM 566 Project 1

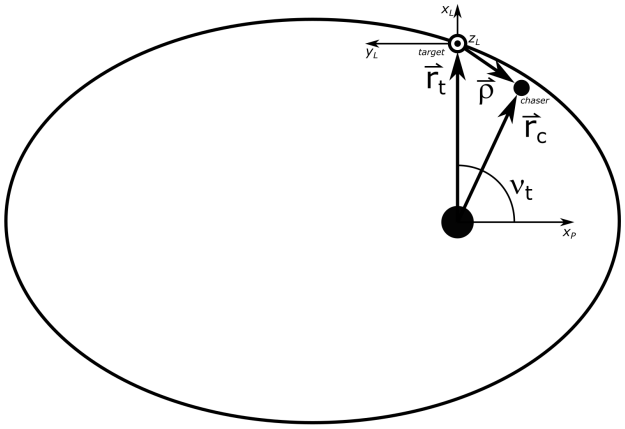
Linear-Quadratic Regulator Design

Learning Objective

This project is intended to introduce the use of the different versions of the linear-quadratic regulator (LQR) for the control of a MIMO LTI system.

Dynamical System

In many instances for spacecraft dynamics and control, one is interested in modeling the dynamics between multiple spacecraft, e.g., a **rendezvous and proximity operation (RPO)**. To that end, consider the following simplified model of two satellites operating in proximity, i.e. a simplified **three-body problem**, involving a **chaser spacecraft** and a **target spacecraft** (subscript t) on an elliptical orbit. One can represent this relative motion in the **Hill frame (HF)** for the target spacecraft as shown in the following figure.



Thus, one can represent the target position relative to the celestial body in the target's HF axes as

$$\vec{r}_t = \begin{bmatrix} r_t \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

and the relative position of the chaser in the target's HF axes as

$$\vec{\rho} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2}$$

Then, noting

$$\vec{r}_c = \vec{r}_t + \vec{\rho} \tag{3}$$

and using Newton's Law of Gravitation for the relative motion of the chaser spacecraft relative to the target spacecraft, one has the equations of motion

$$\ddot{\vec{\rho}} + 2\vec{\omega}_{t,P/I} \times \dot{\vec{\rho}} + \dot{\vec{\omega}} \times \vec{\rho} + \vec{\omega}_{t,P/I} \times [\vec{\omega}_{t,P/I} \times \vec{\rho}] = \frac{\mu m_c}{\|\vec{r}_t + \vec{\rho}\|_2^3} (\vec{r}_t + \vec{\rho}) - \frac{\mu m_t}{r_t^3} \vec{r}_t \tag{4}$$

which is a nonlinear due to the gravity dependence on $\|r_c\|_2^{-3}$ and time-varying due to the $\vec{r}_t$ dependence.

If one assumes a circular orbit for the target spacecraft and by linearizing about the resulting constant $\vec{r}_t$, one obtains the LTI **Clohessy-Wiltshire (CW) equations** for artificial satellites, also known as the **Hill equations** for natural satellites, in the target's HF as

$$\begin{aligned} \ddot{x} &= 2n_t\dot{y} + 3n_t^2 x + u_x \\ \ddot{y} &= -2n_t\dot{x} + u_y \\ \ddot{z} &= -n_t^2 z + u_z \end{aligned} \tag{5}$$

where $n_t = \sqrt{\mu/r_t^3}$ and $\vec{u} = [u_x\ u_y\ u_z]^T$ is the acceleration input, e.g., the mass-normalized thrust force, for the chaser satellite. Then, one has the continuous-time LTI dynamics equation

$$\dot{\vec{x}}(t) = A\vec{x}(t) + B\vec{u}(t) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n_t^2 & 0 & 0 & 0 & 2n_t & 0 \\ 0 & 0 & 0 & -2n_t & 0 & 0 \\ 0 & 0 & -n_t^2 & 0 & 0 & 0 \end{bmatrix} \vec{x} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{u} \tag{6}$$

where

$$\vec{x} = \begin{bmatrix} \vec{\rho} \\ \dot{\vec{\rho}} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{7}$$

This can also be discretized at $t = k\Delta t$ as the discrete-time LTI dynamics equation

$$\begin{aligned} \vec{x}[k+1] =& F\vec{x}[k] + G\vec{u}[k] \\ =& \begin{bmatrix} 4 - 3\cos(n_t\Delta t) & 0 & 0 & n_t^{-1}\sin(n_t\Delta t) & 2n_t^{-1}(1 - \cos(n_t\Delta t)) & 0 \\ 6(\sin(n_t\Delta t) - n_t\Delta t) & 1 & 0 & -2n_t^{-1}(1 - \cos(n_t\Delta t)) & n_t^{-1}(4\sin(n_t\Delta t) - n_t\Delta t) & 0 \\ 0 & 0 & \cos(n_t\Delta t) & 0 & 0 & n_t^{-1}\sin(n_t\Delta t) \\ 3n_t\sin(n_t\Delta t) & 0 & 0 & \cos(n_t\Delta t) & 2\sin(n_t\Delta t) & 0 \\ -6n_t(1 - \cos(n_t\Delta t)) & 0 & 0 & -2\sin(n_t\Delta t) & 4\cos(n_t\Delta t) - 3 & 0 \\ 0 & 0 & -n_t\sin(n_t\Delta t) & 0 & 0 & \cos(n_t\Delta t) \end{bmatrix} \vec{x}[k] \\ &+ \begin{bmatrix} n_t^{-2}(1 - \cos(n_t\Delta t)) & 2n_t^{-2}(n_t\Delta t - \sin(n_t\Delta t)) & 0 \\ -2n_t^{-2}(n_t\Delta t - \sin(n_t\Delta t)) & 4n_t^{-2}(1 - \cos(n_t\Delta t)) - \frac{3}{2}\Delta t^2 & 0 \\ 0 & 0 & n_t^{-2}(1 - \cos(n_t\Delta t)) \\ n_t^{-1}\sin(n_t\Delta t) & 2n_t^{-1}(1 - \cos(n_t\Delta t)) & 0 \\ -2n_t^{-1}(1 - \cos(n_t\Delta t)) & 4n_t^{-1}\sin(n_t\Delta t) - 3\Delta t & 0 \\ 0 & 0 & n_t^{-1}\sin(n_t\Delta t) \end{bmatrix} \vec{u}[k] \end{aligned} \tag{8}$$

Thus, for a rendezvous operation, one must approach the origin of the HF, i.e., the target, at zero relative velocity. Thus, the optimal control problem (OCP) can be cast as an LQR where the state should be regulated to $\vec{0}$ and the energy used in $\vec{u}$, e.g., a low-thrust satellite. Namely, for continuous-time, one has the control problem

$$\begin{aligned} \vec{u}^{\text{opt}}(t) = \underset{u(t)\ \forall t\in[0,\ t_f]}{\text{argmin}} \quad & \mathcal{J} = \int_0^{t_f} \vec{x}^T(t) Q \vec{x}(t) + \vec{u}^T(t) R \vec{u}(t) dt \\ \text{subject to:} \quad & \dot{\vec{x}}(t) = A\vec{x}(t) + B\vec{u}(t) \\ \text{initial condition:} \quad & \vec{x}(0) = \vec{x}_0 \end{aligned} \tag{9}$$

and for discrete-time, one has the control problem

$$\begin{aligned} \vec{u}^{\text{opt}}[k] = \underset{\vec{u}[k]\ \text{for}\ k=0,\ldots,N-1}{\text{argmin}} \quad & \mathcal{J} = \sum_{k=0}^{N-1} \vec{x}[k]^T Q \vec{x}[k] + \vec{u}[k]^T R \vec{u}[k] \\ \text{subject to:} \quad & \vec{x}[k+1] = Fx[k] + Gu[k] \\ \text{initial condition:} \quad & \vec{x}[k] = \vec{x}_0 \end{aligned} \tag{10}$$

The solution for the continuous-time OCP is given by the differential Riccati equation

$$\dot{P} = -PA - A^T P + PBR^{-1}B^T P - Q \tag{11}$$

which describes the dynamics of the continuous-time Riccati matrix, $P(t)$, and can be solved using the boundary condition on the $P(t_f) = 0$. The unconstrained finite-horizon continuous-time LQR has the form

$$\vec{u}^{\text{opt}}(t) = -K(t)\vec{x}(t) \tag{12}$$

where

$$K(t) = R^{-1}B^T P(t) \tag{13}$$

which notably results in closed-loop state dynamics represented by

$$\dot{\vec{x}}(t) = (A - BK(t))\ \vec{x}(t) \tag{14}$$

or

$$\dot{\vec{x}}(t) = \left(A - BR^{-1}B^T P(t)\right)\ \vec{x}(t) \tag{15}$$

The solution for the discrete-time OCP is given by the dynamic Riccati equation

$$P[k-1] = F^T P[k] F + Q - \left(F^T P[k] G\right)\left(G^T P[k] G + R\right)^{-1}\left(G^T P[k] F\right) \tag{16}$$

which is a backwards recursion formula to solve for the discrete-time Riccati matrix $P[k]$ starting from $P[N] = 0$. The unconstrained finite-horizon discrete-time LQR here has the form

$$\vec{u}^{\text{opt}} = -K[k]\vec{x}[k] \quad \forall\ k \tag{17}$$

where

$$K[k] = \left(G^T P[k+1] G + R\right)^{-1} G^T P[k+1] F \tag{18}$$

which notably results in closed-loop state dynamics represented by

$$\vec{x}[k+1] = (F - GK[k])\, \vec{x}[k] \tag{19}$$

or

$$\vec{x}[k+1] = \left(F - G\left(G^T P[k+1] G + R\right)^{-1} \left(G^T P[k+1] F\right)\right) \vec{x}[k] \tag{20}$$

For this project, consider the Earth as the celestial body ($\mu = 3.986004418 \times 10^{14}$ m^3/s^2), the target orbital radius as $r_t = 6,783,000$ m, the initial condition as

$$\vec{x} = \begin{bmatrix} 1000 \\ 1000 \\ 1000 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{21}$$

and the following three cases for the weight matrices.

Case 1 is

$$Q = I_{6\times6} \qquad R = I_{3\times3} \tag{22}$$

Case 2 is

$$Q = I_{6\times6} \qquad R = 100 I_{3\times3} \tag{23}$$

and case 3 is

$$Q = I_{6\times6} \qquad R = 10000 I_{3\times3} \tag{24}$$

Project Assignment and Deliverables

Do: the following tasks in MATLAB or Python.

1. Compute the eigenvalues of the state matrix for the continuous-time LTI system and comment on the stability of the system.
2. Compute the controllability matrix for the continuous-time LTI system.
3. Design a finite-horizon LQR for the continuous-time LTI system with $t_f = 16200$ s.
    - Use an ODE solver to numerically solve the Riccati differential equation;
        - E.g., MATLAB's `ode45`; or
        - E.g., `scipy.integrate.ode`;
    - Use an ODE solver to numerically solve the closed-loop state dynamics equation with interpolating your time-varying Riccati matrix solution;
    - Output three plots of u_x, u_y, and u_z for all three cases;
    - Output a trajectory plot of x vs. y for all three cases; and
    - Output a trajectory plot of z for all three cases.
4. Design an infinite-horizon LQR for the continuous-time LTI system.
    - Use a continuous-time algebraic Riccati equation (CARE) solver;
        - E.g., MATLAB's `icare`; or
        - E.g., `scipy.linalg.solve_continuous_are`;
    - Use an ODE solver to numerically solve the closed-loop state dynamics equation with your time-invariant Riccati matrix solution;
    - Output three plots of u_x, u_y, and u_z for all three cases;
    - Output a trajectory plot of x vs. y for all three cases; and
    - Output a trajectory plot of z for all three cases.
5. Design a finite-horizon LQR for the discrete-time LTI system approximation with $\Delta t = 1$ s and $N = 1500$.
    - Solve the differential Riccati equation recursively;
    - Recursively solve the closed-loop state dynamics equation with your time-varying Riccati matrix solution;
    - Output three plots of u_x, u_y, and u_z for all three cases;
    - Output a trajectory plot of x vs. y for all three cases; and
    - Output a trajectory plot of z for all three cases.
6. Design an infinite-horizon LQR for the discrete-time LTI system approximation with $\Delta t = 1$ s.
    - Use a discrete-time algebraic Riccati equation (DARE) solver;
        - E.g., MATLAB's `idare`; or
        - E.g., `scipy.linalg.solve_discrete_are`.
    - Recursively solve the closed-loop state dynamics equation with your time-invariant Riccati matrix solution;
    - Output three plots of u_x, u_y, and u_z for all three cases;
    - Output a trajectory plot of x vs. y for all three cases; and
    - Output a trajectory plot of z for all three cases.
7. Comment on all designs and the differences between them.

Deliver: in the Blackboard assignment, all files to run your MATLAB or Python script(s). There is no need to zip your files.